SEVENTH FRAMEWORK PROGRAMME	**Project Acronym: Giraff+** **Project Title: Combining social interaction and long term monitoring for promoting independent living** **Grant agreement no.: 288173** **Starting date: 1st January 2012** **Ending date: 31st December 2014**

D4.1 The Interaction and Visualization Service and Personalization Module – Alpha Release

WP related to the Deliverable:	4
Nature:	P
Dissemination Level :	PU
Version:	V0.1
Author(s):	Amedeo Cesta (CNR-ISTC), Gabriella Cortellessa (CNR-ISTC), Luca Coraci (CNR-ISTC), Riccardo De Benedictis (CNR-ISTC), Andrea Orlandini (CNR-ISTC), Filippo Palumbo (CNR-ISTI), Ales Stimec (XLAB)
Project Participant(s) Contributing:	CNR-ISTC, XLAB, CNR-ISTI
Contractual Date of Delivery:	20130630
Actual Date of Delivery:	20130726

Document History

Version	Date	Type of editing	Editorial
0.1	17/04/13	Table of Contents	CNR-ISTC
0.5	07/07/13	Significant content inserted	CNR-ISTC, CNR-ISTC
0.6	15/07/13	Further content added	CNR-ISTC, XLAB
0.7	19/07/13	First complete refinement	CNR-ISTC
0.8	23/07/13	Second complete refinement	CNR-ISTC (+comments)

Deliverable Summary

This deliverable contains material that supports the delivery of the Alpha Release of the GiraffPlus DVPIS, the Data Visualization, Personalization and Interaction Service. The DVPIS is the module dedicated to the interaction of multiple users with the GIRAFFPLUS system. This software release contains the basic data structures and active capabilities for managing data visualization and interaction with the different actors involved in the GIRAFFPLUS use. The report describes also the basic data structures for performing customized user classification and the consequent dynamic model update.

Table of Contents

1 Introduction 6

1.1 Scope of the document 7

1.2 Deliverable structure 7

2 Interaction and Visualization Service and Personalization Module 8

2.1 Different users of the DVPIS 8

2.2 Front-end Services overview 9

2.3 Back-end Services overview 10

2.3.1 Basics on timelines 10

2.3.2 Timelines and user modeling 11

2.3.3 The reminder 12

2.3.4 The generation of personalized reports 13

2.3.5 Integration of the services through the middleware 13

3 Main components of the DVPIS Alpha Release 15

3.1 DVPIS@Office 16

3.1.1 Module design 16

3.1.2 Implementation details 17

3.1.3 An example of use 20

3.2 DVPIS@Home 26

3.2.1 Module design 26

3.2.2 Implementation details 31

3.2.3 An example of use 34

3.3 The Install & Maintenance Service 36

3.3.1 Module design 36

3.3.2 Implementation details 36

3.3.3 An example of use 37

4 Conclusions 41

5 Bibliography 42

Appendix A 43

List of Figures

Figure 1 - The DVPIS schematic architecture 6
Figure 2 - Different users of the Interaction and Visualization Services. 9
Figure 3 - Main components of the Interaction and Visualization Services 10
Figure 4 - Reminder service message type class diagram. 14
Figure 5 - Outside vs. Inside services through DVPIS. 15
Figure 6 - The current DVPIS in operation. 16
Figure 7 - Description of starting flow. 18
Figure 8 - Bundle activation and login phase activity diagram. 18
Figure 9 - Service activation activity diagram. 19
Figure 10 - Data loading activity diagram. 20
Figure 11 - DVPIS@Office main folder. 20
Figure 12 - Login Dialog. 21
Figure 13 - General Layout. 21
Figure 14 - Menu bar, detailed description. 22
Figure 15 - List item: details. 22
Figure 16 - Choosing a different layout. 23
Figure 17 - Content Area. 23
Figure 18 - Physiological data Panel. 24
Figure 19 - Blood Pressure Chart. 25
Figure 20 - Activity Panel. 25
Figure 21 - Activity Panel button bar. 26
Figure 22 - DVPIS@Home design concept. 30
Figure 23 - The DVPIS@Home and the hPlugins. 30
Figure 24 - Consistency Manager sequence diagram. 32
Figure 25 - DVPIS@Home layout. 34
Figure 26 - hPlugin personalization example. 35
Figure 27 - Showing data on the DVPIS@Home. 36
Figure 28 - Login screen of the GiraffPlus EngineerUI WebApp. 38
Figure 29 - Basic view of the GiraffPlus EngineerUI. Menu on the left side manages stored configuration entities. 38
Figure 30 - The list of all monitored homes configured in the GiraffPlus LTS. 39
Figure 31 - Part of the input form presented to the GiraffPlus profesional when configuring a new installation. 39
Figure 32 - The configuration of a Giraff robot can be attached to the configuration of the monitored home. 40
Figure 33 - A screenshot of the GiraffPlus EngineerUI as seen on the Apple's iPhone. 40
Figure 34 - DVPIS@Office main folder. 43
Figure 35 - Login Dialog. 44
Figure 36 - Available Demo Data Viewers. 44
Figure 37 - Reminder Wizard. 45
Figure 38 - Starting the DVPIS@Home simulator. 46
Figure 39 - The DVPIS@Home simulator. 46
Figure 40 - Sending activity data to the DVPIS@Home. 47
Figure 41 - Setting a reminder for the Primary User. 47
Figure 42 - The DVPIS@Home displays a reminder. 48

List of Tables

Table 1 - Implementation of main guidelines. 28
Table 2 - DTD Model for the configuration file. 31
Table 3 - System requirements..... 43

1 Introduction

The GIRAFFPLUS WP4 is a research and development work package dedicated to the design and development of services that allow the GIRAFFPLUS system to serve specific classes of users. According to the project DOW the contribution of WP4 is expected into two main services:

- The **Interaction and Visualization Service** (IVS), whose objective is to produce a well-organized set of functionalities for allowing different users to connect with the GIRAFFPLUS environment;
- The **Personalization Service** (PerS), whose objective is to contribute with new functionalities that continuously guarantee fine-grained personalization to the healthcare professionals, to the informal caregivers and to the specific elderly at home.

During the architectural design phase of GIRAFFPLUS (see the description in D1.3) the user interaction services are allocated to the *Data Visualization, Personalization and Interaction Service* (DVPIS). Additionally the DVPIS has been conceptualized as in the Figure 1, where the Interaction and Visualization Service (IVS) is the part directly in contact with different users (as a consequence it is conceived as the *front-end* module) and a second part that has a proactive role of "preparing content" for the users (hence these functionalities are better conceived as the DVPIS *back-end*). A distinction in the back-end discussed in D1.3 concerns the Personalization Services, more dedicated to modeling use features over time, and User Oriented Services, more dedicated to specific data handling, from the project internal services (e.g., the Report generation) and to support specific functionalities requested by user requirement analysis (e.g., the Reminding service).

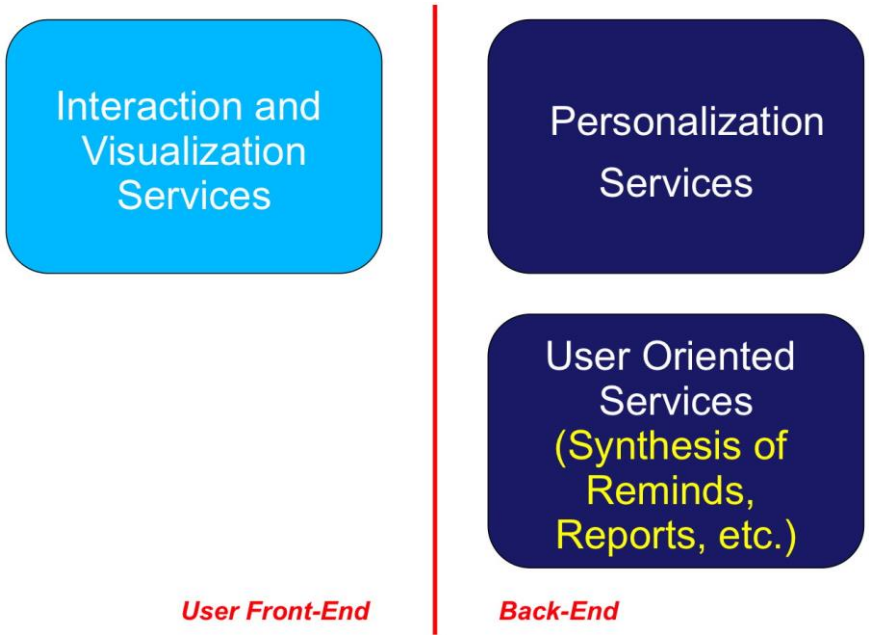


Figure 1 - The DVPIS schematic architecture

Aim of this deliverable is to describe the components of the DVPIS to reflect the current status of the integrated demo in Alpha Release. As described in the DOW, the active goal for M18 is to have a first instance of the DVPIS endowed with a consolidated IVS whose functionalities are

robust enough to support the start of the GiraffPlus test sites. As a consequence the report elaborates with more details the work concerning the IVS. Such part is subdivided into the three modules called DVPIS@Office, DVPIS@Home and Install and Maintenance Service. As described below each of the modules implements dedicated services for classes of users (Secondary and Primary Users and System Engineers). The report describes also the work connected to the DVPIS back-end, elaborating on both data structures for the Personalization Service (PerS) and, shortly, on the current work for the Reminder Service and the Personalized Report Service.

1.1 Scope of the document

The general scope of this deliverable is a brief description of the software component identified in the GIRAFFPLUS system as the DVPIS.

In particular, we describe two different components of the system: one for being used "outside the home" by secondary users (we have called it DVPIS@Office), and another dedicated to the primary user (hence elderly people at home, called DVPIS@Home). A third module (Install & Maintenance Service) has been developed in the project with the aim to facilitate technicians and engineers during the installation of a new test site (hence when new assisted homes are created). Because this particular issue turns out to be very relevant in GIRAFFPLUS we have included this service in the complete set of project user services.

It is worth reminding that both the Interaction and Visualization Service and the Personalization service, are based on the results achieved in task T4.1, and T4.2 where the various services have been designed.

1.2 Deliverable structure

This document is mainly subdivided into two sections. Section 2 introduces the general idea behind the Interaction and Visualization Service and the Personalization Module. Additionally it gives attention to the description of the underlying technology exploited for realizing the DVPIS back-end part. Section 3 describes more in detail the users services currently provided by the DVPIS. It is subdivided in 3 subparts corresponding to the three main modules of the Interaction and Visualization Service we have mentioned before:

- Sub-section 3.1 introduces the Visualization @Office, dedicated to the secondary users and, in general, to those who are "outside the home";
- Sub-section 3.2 introduces the Visualization @Home, dedicated to the primary users;
- Sub-section 3.3 introduces the Install & Maintenance Service, dedicated to engineers and technicians involved in installation and maintenance of the system.

Some conclusions (Section 4) on the outcomes of the work are then reported together with a brief description of future development directions.

2 Interaction and Visualization Service and Personalization Module

This section of the deliverable touches upon general issues concerning the DVPIS as a whole while the following section specifically describes the current release of the Interaction and Visualization Service (IVS) which is delivered together with this report.

2.1 Different users of the DVPIS

GIRAFFPLUS the influence of end-user in the different choices is pervasive. As a matter of fact, the active dialogue with end-users is essential to ensure that project services reflect and respond to their real needs and expectations.
To facilitate the readers, before entering the specificity of the DVPIS technical development, we report here a summary of the different end-users who receive specific attention from the GIRAFFPLUS system (for a complete analysis see GIRAFFPLUS D1.1). This clarifies the rationale behind the different DVPIS parts.

Figure 2 presents a basic schema of the needed interactions to be supported. We see mainly three classes of individuals:

1. **Primary end-users** are the people around which the GIRAFFPLUS system/services are created. In fact, the elderly play the main role of recipients of the innovative technology aimed at prolonging their stay independent at home, i.e., the intelligent environment, but above all as “experts” of their own everyday lives.
2. **Secondary end-users** are persons directly being in contact with a primary end-user. This group benefits from the GIRAFFPLUS system directly when using its services (mainly remotely) and indirectly when the care needs of primary end-users are reduced. Usually, different secondary users may have different expectations from the system functionalities. For this reason, in order to further investigate this aspect, we will subdivide secondary end-users into two different sub-groups:
    a. **Healthcare Professionals**. An individual healthcare provider who may be a healthcare professional in medicine, nursing, or a field allied to health.
    b. **Caregivers**. An informal caregiver is a close relative or a friend who takes care of the primary end-user or in general has regular contact with him/her. A formal caregiver is a person trained to take care of the elderly especially with a social emphasis (rather than a medical one) on the type of support. Municipalities or social health cooperatives usually provide and train the formal caregivers.
3. **GiraffPlus Engineers** are the professionals who are involved in the installation, configuration and maintenance of the GIRAFFPLUS system. We decided to introduce explicitly this class of users to evaluate the “relative simplicity” of the technology we are creating in the project.

The different kinds of secondary end-users focus their attention on different aspects of the system functionalities. For example, most often professionals focus on normative needs that can be generalized to a population, while caregivers are more driven from what is meaningful and

supportive for the elder people they take care of. The coexistence of these different views and perspectives highlights the need for personalized services and flexible solutions the system should be able to provide. Also, allowing primary users to access the information on their own health condition potentially enable them to better manage their health and lifestyle.

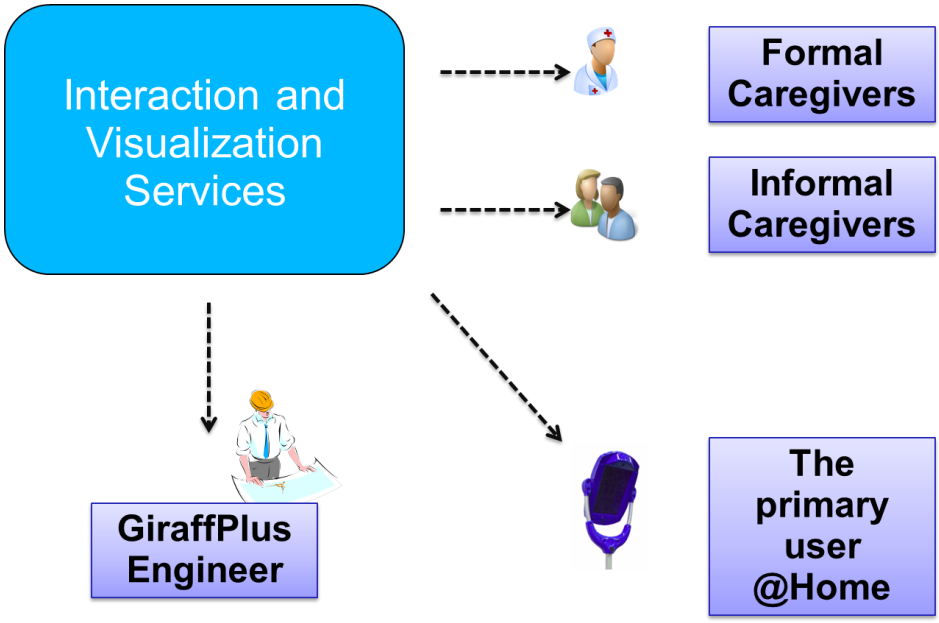


Figure 2 - Different users of the Interaction and Visualization Services.

2.2 Front-end Services overview

The DVPIS as a whole, and the Interaction and Visualization Service (IVS) in particular, have been designed to provide an entry point to the different users of the GIRAFFPLUS system. The key idea is to continuously guarantee the use of the system by means of interaction modalities that are both easy to use and adequate for the different classes of people.

The Interaction and Visualization Service has been subdivided into three main modules (see Figure 3) to serve the three categories of users identified in Figure 2 In. this way the DVPIS is able to offer a first level of personalized services to different kind of users. In particular:

- The *DVPIS@Office* module is dedicated to those healthcare professionals and informal caregivers are responsible for taking care of the health of elderly people. In this way, these GIRAFFPLUS users are supported by means of a flexible and efficient monitoring tool while taking care of old persons (relatives/patients).
- The *DVPIS@Home* module is dedicated to primary users. By providing access to information on their own health condition, the GIRAFFPLUS system enables them to better manage their health and lifestyle (end user empowerment).
- The *Install & Maintenance Service* is addressed to technicians and engineers whose task is to physically install the system into the house of an old person, configure the system for a new home and maintaining over time the efficiency of the system.

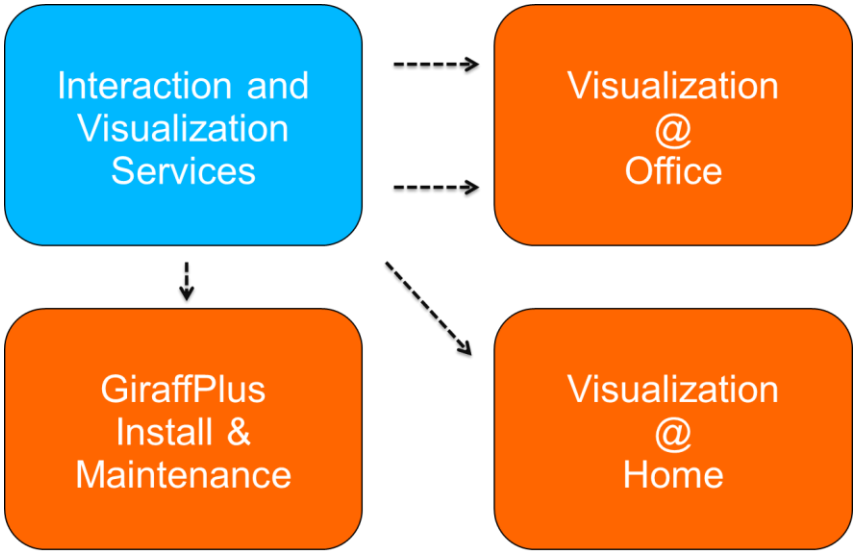


Figure 3 - Main components of the Interaction and Visualization Services

2.3 Back-end Services overview

The Interaction and Visualization Service is enriched by the Personalization Service that allows the personalization of the content by generating a continuous loop around two main data structure: (a) a *User Knowledge Base* that initially contains data about classes of potential users (describing a sort of User Stereotypes); (b) a *Person Dynamic Model* which uses a representation based on timelines, and further relying on temporal constraint networks, for modeling and updating key features that should be maintained over time to enable more complex personalized services. By interacting with the involved users during the use of the system, the personalization service will contain an increasingly accurate model of the target primary user at home, thus, dynamically and automatically improving the GIRAFFPLUS experience in time.

2.3.1 Basics on timelines

In GIRAFFPLUS we use timeline-based reasoning (Muscettola, 1994; Jonsson, Morris, Muscettola, Rajan, & Smith, 2000; Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A., 2009) to support the back-end services of the DVPIS. A timeline is a data structure to support personalization services. This section introduces the basic terminology for understanding timeline uses.

In generic terms, a timeline is a temporal function over a continuous domain. Possible values for a timeline are generically called *tokens* and are represented through predicates holding over a time interval characterized by a starting and an ending time. In addition to the temporal arguments, these predicates can have further arguments that allow a better characterization of values in time. Each argument has, in general, a set of allowed values that can be reduced through the use of constraints.

The set of initial allowed values depends on the instance of the argument type that characterizes the argument itself and is defined at the planning domain level. Examples of variable types are temporal points – ranging from 0 to +inf –, angles – ranging from 0 to 360 –, alphabet letters – assuming one of {*a, b, c, …*}, etc. Although new argument types can be easily defined, two main typologies have emerged as important to model salient features of the personalization services:

- **Numeric variable types** – to represent numeric concepts. An instance of a numeric variable type (e.g. a time point, an angle, etc.) is characterized by a lower bound value *lb* and an upper bound value *ub*. Creating an instance of a numeric variable type results in the creation of a numeric argument x_i that, without any other enforced constraint, can assume any value inside the allowed domain *[lb, ub]*.
- **Enumeration variable types** – to represent enumerative concepts. An instance of an enumeration variable type (e.g. letter) is characterized by a set of allowed values {room_*a, room_b, room_c, …*}. Creating an instance of an enumeration variable type results in the creation of an enumeration argument x_i that, without any other enforced constraint, can assume any value inside the allowed values {room_*a, room_b, room_c, …*}.

Different tokens can be linked each other by means of **relations** representing constraints imposed over involved token arguments. In general, relations can represent any logical combination of linear constraints among token arguments. According to the number of involved tokens, relations can be divided into *unary*, *binary*, and *n-ary*.
A further basic ingredient in timeline modeling is the so-called "*Compatibility*". Compatibilities are the way to express planning domain/causal rules in the current internal representation. Any given valid plan should be consistent with respect to the set of such specified rules.

Compatibilities are defined through a logic implication *reference* → *requirement* where reference is the token value that demands compatibility application while requirement is the "consequence" of the presence of the reference value in the plan. Making use of a recursive definition, a requirement can be a value, on the same timeline or on another timeline, a relation among the reference value and the target values, a conjunction of requirements or a disjunction of requirements. Being relations, in the most general case, linear constraints, compatibilities provide great expressiveness, which enables GIRAFFPLUS users to represent quite complex behaviours.

Inside the GIRAFFPLUS system, the concept of compatibility has been exploited both in the reminder service and in the generation of personalized reports. In the following, we will describe in a higher detail, both of these services and we will show an example of use.

2.3.2 Timelines and user modeling

The timeline-based technology is widely used in the GIRAFFPLUS project. Within the DVPIS it is exploited to represent the models of the users following an approach similar to the one described in (Cortellessa, De Benedictis, & Pagani, June, 2013). A *user model* is a set of relevant personal features that may influence human behaviours and can be used to perform personalization. The selection of the factors that are relevant to build the user model is currently on-going. At present, the variables to be monitored are under study before a final selection. Just as an example, consider that we are taking into account variables like personality traits, perceived stress and anxiety. In broad terms, the personalization module has been designed in order to be easily extendible with further variables in order to address more complex scenarios.

The first macro subdivision of these behavioural variables is between (a) *static* features that do not change in time, being mainly related to individual personality and (b) *dynamic*, which can be, on the contrary, related to both the context and time -- hence, they may vary during the period of

use in a test site. Both dynamic and static variables are used to create the initial user profile, while the dynamic ones are also used to update the model in time.

The user model has also been enriched by additional information that regards the interaction with the system in terms of time of usage, frequency of usage, inactivity time, amount of information shared with the other users and so on. This additional info contributes to gather useful feedback on the primary user's interaction with the system that can be presented to the secondary user to support the analysis phase.

The wide variety of possible combinations of behavioural parameters, together with the difficulty of making an accurate assessment, have suggested the use of machine learning techniques to classify primary users. Given the discrete nature of the input data and a small amount of training data, we have chosen to use a C4.5 algorithm (Quinlan, 1996) to perform primary user classification. Machine learning techniques, also, enabled us to customize the classification of primary users to the secondary user specific style. Different users could use over time different rules and the machine learning techniques could consequently allow learning a "specific secondary user style", so that it could be automatically adopted during the test site.

2.3.3 The reminder

The need for integrating a reminding service has emerged during the phase of User Requirement Analysis (see D1.1). The idea underlying the reminding service is to asynchronously deliver messages to users in order to remember them of important tasks as, for example, to take medicines and/or to perform some physical activity during a rehabilitation phase. By leveraging the power of timeline-based technology, we have enriched this simple idea in order to achieve an active interaction with primary users by providing stimuli properly customized to the single user specific needs. A possible use of this technology is, for example, during a rehabilitation phase where each primary user constitutes a basic case and possible evolutions can be reduced to a discrete finite set.

Given the temporal nature of the problem, we have chosen to use timelines as the underlying structure for the reminding services. In particular, we have chosen to use tokens to represent single reminders. By adding temporal constraints among these tokens we can easily place reminds in time. An execution module will then activate the graphical user interface for displaying reminds at proper time.

The reminding system allows four kinds of temporal constraints:

- *At-time*: placing the reminder after a specific time.
- *At-date*: placing the reminder at a specific date. In case the specified date has already passed then the reminder is displayed immediately.
- *At-time-periodic*: placing the reminder after a specific time and periodically re-proposed at a specified rate.
- *At-date-periodic*: placing the reminder at a specific date and periodically re-proposed at a specified rate.

We have used timelines both for representing reminders and for performing classification of primary users. The classification of users allows us to automatically perform different activities according to the class that the user is associated to. This is the leading idea that has been used to personalize interfaces. In particular, the content of the reminder is customized according to the current classification of primary users.

Together with the reminders, the reminder system allows the users to develop "questions" to other users (e.g., a secondary user asking to a primary user for a confirmation of having taken some particular medicines). Thanks to the concept of compatibility, it is possible to automatically generate further reminders and/or questions, temporally and causally linked to the specific answers that the target primary user gives. In other words, the system can automatically generate new questions to the user in order to further investigate some strange and/or unexpected behaviour and/or to accompany the primary user during a rehabilitation phase.

2.3.4 The generation of personalized reports

The main task of the reporting service is to perform long term data analysis in order to observe, for example, trends of different users. Deviations from common behaviours constitute typically a sort of "alarm bell" that could be exploited to detect the emergence of new possible pathologies.

Given the need to periodically generate reports and to personalize compiled reports according to the users who actually need to read them, we can share the main technological background that has been used for the reminding service also for the generation of personalized reports. While the reminder service is mainly aimed at primary users, reports are mainly intended for secondary users. Therefore, in this case, the personalization environment should take into account secondary user's model rather than primary user's one.

2.3.5 Integration of the services through the middleware

As described in D2.1 and D2.2, the aim of the middleware is to provide a publish/subscribe mechanism for accessing the context information about the physical environment and physiological data. The information is exposed as different topics: topics for discovery and description of devices and services that form the service bus and topics for publishing and retrieving data from devices and services that form the context bus. Once a resource has been announced on the service bus a generic service can search for it filtering on the descriptor fields and use it. The topics used for announce and discovery of devices and service, the so-called service bus, has this format:

```
<<location>>\serviceBus\<<serviceID>>
```

where `location` identifies the room in the assisted persons apartment, `serviceBus` is the keyword to identify the topic as a service bus topic and `serviceID` is the unique identifier of the service. The message of this topic is a JSON descriptor file.

The reminder service and each reminder client announce themselves on the topic:

```
<<location>>\serviceBus\reminder
```

and describe themselves by means of an id, a description, a type (i.e. service), a set of resources, and a set of exposed methods (for publishing new reminds and new questions). Similarly, the reporting service and each reporting client announce themselves on the topic:

`<<location>>\serviceBus\reporting`

The middleware is in charge of dispatching information about the state of the resources among services by means of a context bus. The topics used for gathering data from devices and service, the so-called context bus, have this format:

`<<location>>\contextBus\<<serviceID>>\<<subtreefield>>`

where `location` identifies the room, `contextBus` is the keyword to identify the topic as a context bus topic, `serviceID` is the unique identifier of the service and the `subtreefield` identifies all the resources of that service that can be monitored. For each resource there will be a dedicated context bus sub-topic. The message of these topics is a string value. In particular, the reminder service publishes data on the topic:

`<<location>>\contextBus\reminder`

The string value representing the content of the message is, again, a JSON file having a structure described in Figure 4.

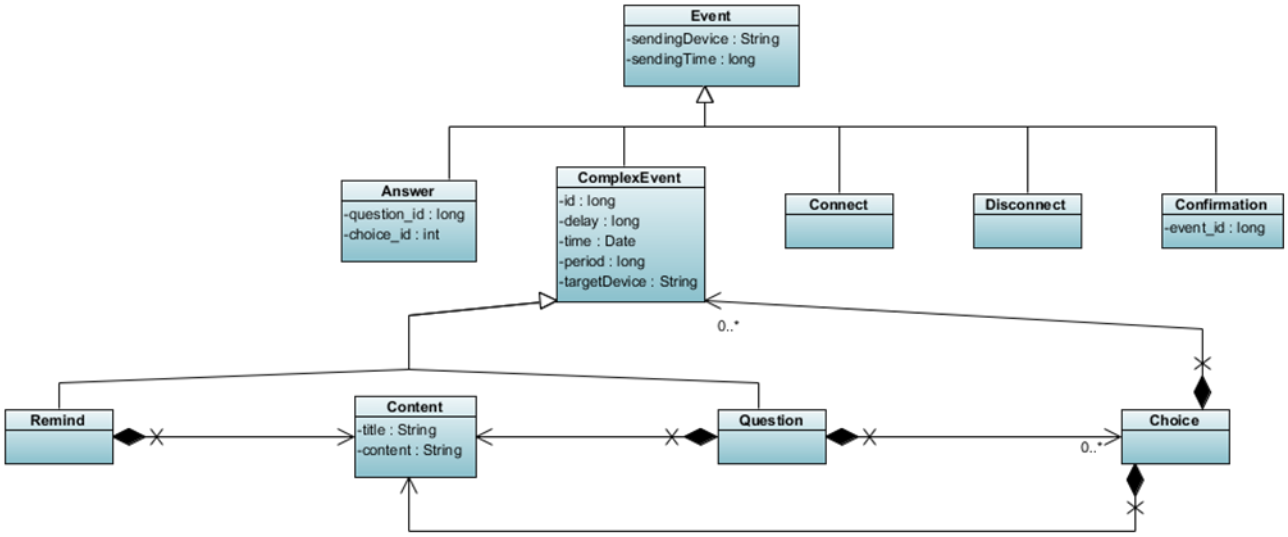


Figure 4 - Reminder service message type class diagram.

3 Main components of the DVPIS Alpha Release

To understand the complete scenario of use in which the DVPIS services are situated we insert Figure 5 taken from (Cesta, et al., July 2013). Different aspects are worth being underscored: first of all the double directionality pursued in the GIRAFFPLUS service model:

- *Internal vs. External Data Flow*: there is a basic direction of data gathering from the house toward the external world (red arrows in the figure). Notice that we further distinguish in:
  (a) long-term data storage for subsequent analysis and on-demand visualization (red arrow in upper part of the figure),
  (b) direct real-time fast connections managed directly through the middleware for alarms (red arrow in lower part);
- *Bidirectional External/Internal communication*: this channel allows supporting the social interaction services. Initially, the basic services of the Giraff telepresence robot have been created, subsequently the system has been empowered with additional services for increasing support to the primary users (blue arrow in the figure).

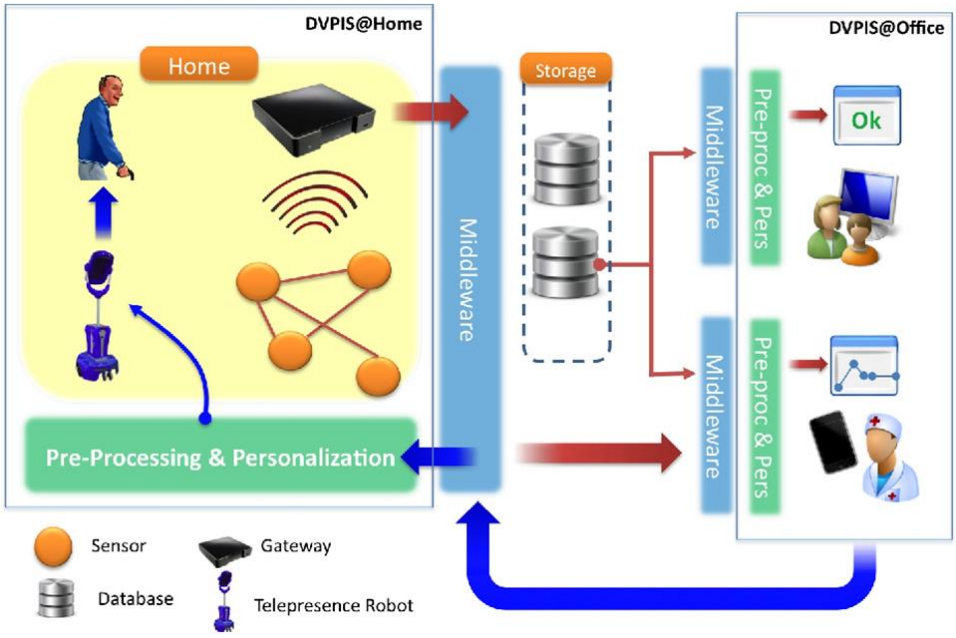


Figure 5 - Outside vs. Inside services through DVPIS.

The Figure 5 also identifies the DVPIS modules that have been realized to manage interaction with the different actors in the Ambient Assisted Living (AAL) scenario. As said before in the figure we sketch the two instances of the DVPIS one for use “outside the house”, the DVPIS@Office, and another dedicated to the primary user, the DVPIS@Home. Each of these DVPIS modules are composed of a front-end which is a visualizer and interaction manager and a back-end that deals with pre-processing of data and personalization of services.

Figure 6 - The current DVPIS in operation.

3.1 DVPIS@Office

The DVPIS@Office module, dedicated to healthcare professionals and informal caregivers, strongly relies on querying the permanent data storage service (see D1.3). Additionally, the access to the Giraff telepresence robot, via its usual client software interface, is possible within the same comprehensive layout. At present we have designed an interaction front-end that runs on a personal computer but the technological choices allow us to easily develop also App-style versions for other mobile platforms like tablets or smartphones by suitably porting the current software. Different services are provided for formal and informal caregivers. The personalization for a doctor or a social worker takes into account the fact that these workers may connect to multiple patients at home. Hence there is an environment that manages different houses and helps the user to maintain information on the different cases he/she follows.

The formal caregiver may access data and study their evolution over time. The visualization environment allows a combination of queries that are able to generate different graphical views. Figure 6-B presents a test example of the current DVPIS@Office for a doctor observing a week of blood pressure data. Figure 6-A shows a secondary user inspecting a printed report received through the GIRAFFPLUS system and at the same time has decided to query the long-term data storage service for observing temporal trends (see the PC screen). The personalization for an informal caregiver (the current target is a relative of the person at home) is far straightforward. Such a person needs a more synthetic information and just report about warning over an interval of time. At present it is possible to ask for daily reports that contain a summary of the information of the day. We have also produced a scenario similar to the one in Figure 6-A tailored for informal caregivers, with reports that contain a different perspective on the information delivered.

3.1.1 Module design

As sketched in Figure 5 the DVPIS@Office provides functionalities toward the secondary users supporting static personalization among classes of users (in particular focusing on health professionals and informal caregivers).

The module has been designed for providing different functionalities customized to the specific user that is involved in a certain interaction. Besides providing basic features like system authentication and account configuration, the module allows the initialization of primary users, telepresence calls through the Giraff robot, and emergency calls. Through the DVPIS@Office,

secondary users have complete access to both current and past activities of the selected primary user; additionally they can examine both current sensor data and historical ones. Finally, DVPIS@Office allows secondary user to set or edit reminders, to display periodic reports and, in case, display warnings and alarms as a consequence of perceived dangerous situations.

It is worth underscoring how the Giraff robot pilot has been embedded into the DVPIS@Office and customized in a homogeneous look-and-feel together with the other functionalities. More innovative uses of the pilot will be explored in specific research activities of the project (their description is outside the scope of this report).

The DVPIS@Office requires high modularity, hence it fits well with the OSGi technology that allows, among other things, an easy integration with the GIRAFFPLUS middleware.

3.1.2 Implementation details

The DVPIS@Office is presented as a standard OSGi bundle[1] (OSGi Alliance). In order to correctly start the DVPIS@Office a list of OSGi-services must be provided to the OSGi container.

Currently, when the DVPIS is started it will look up for implementations of several services:

- **StorageAPI**: this service exposes the API to the basic CRUD operations over the long-term storage entities used by DVPIS@Office. Actually, the application does not directly look up for this service but this is done by another OSGi component called "**giraffplus.storage.wrappers**" that is required by the DVPIS@Office as a regular dependency. The runner engine is responsible to run this component before the DVPIS@Office one. When started, the giraffplus.storage.wrappers module will search into the Felix framework, an implementation of the StorageAPI service described by a shared interface. Once the implementation has been retrieved, the component can provide to the DVPIS@Office a higher-level layer of the API providing some additional features that improve the efficiency of data-retrieval process and provides a large set of specific error-handling messages especially useful both in debug phase and in the user interaction control. Hence, the DVPIS@Office does not directly use the StorageAPI, but the wrapped version offered by the giraffplus.storage.wrappers bundle.
- **Middleware**: all the data not directly coming from the long-term storage will be delivered through the Middleware service. The implementation of this service will be retrieved in the initialization step and the success of this step is mandatory for a full operative system.
- **Other Services**: for each instance of the DVPIS@Office a set of additional services is supposed to be available. After the login and initialization phase the application is supposed to load all additional services. Currently the DVPIS@Office will look up for available personalization services: the Reminder Service and the Report Generation Service. The OSGi system provides a standard way for defining, searching and using a service and this methodology is fully followed by the DVPIS@Office and all related services.

[1] The OSGi container choosen for the GIRAFFPLUS Project is Felix (Apache Felix)

3.1.2.1 Description of starting flow

The overall flow of the DVPIS@Office application can be represented as shown in the figure below (Figure 7).

.

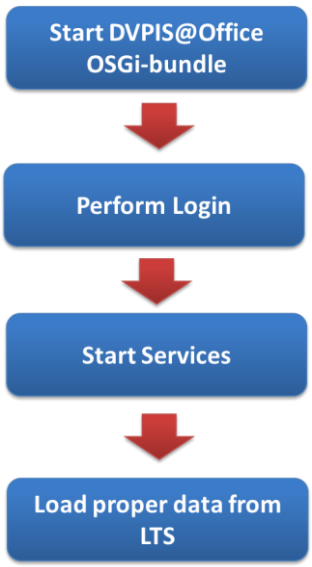


Figure 7 - Description of starting flow.

3.1.2.2 Bundle activation and login phase

The first two steps are described in the activity diagram shown in Figure 8.

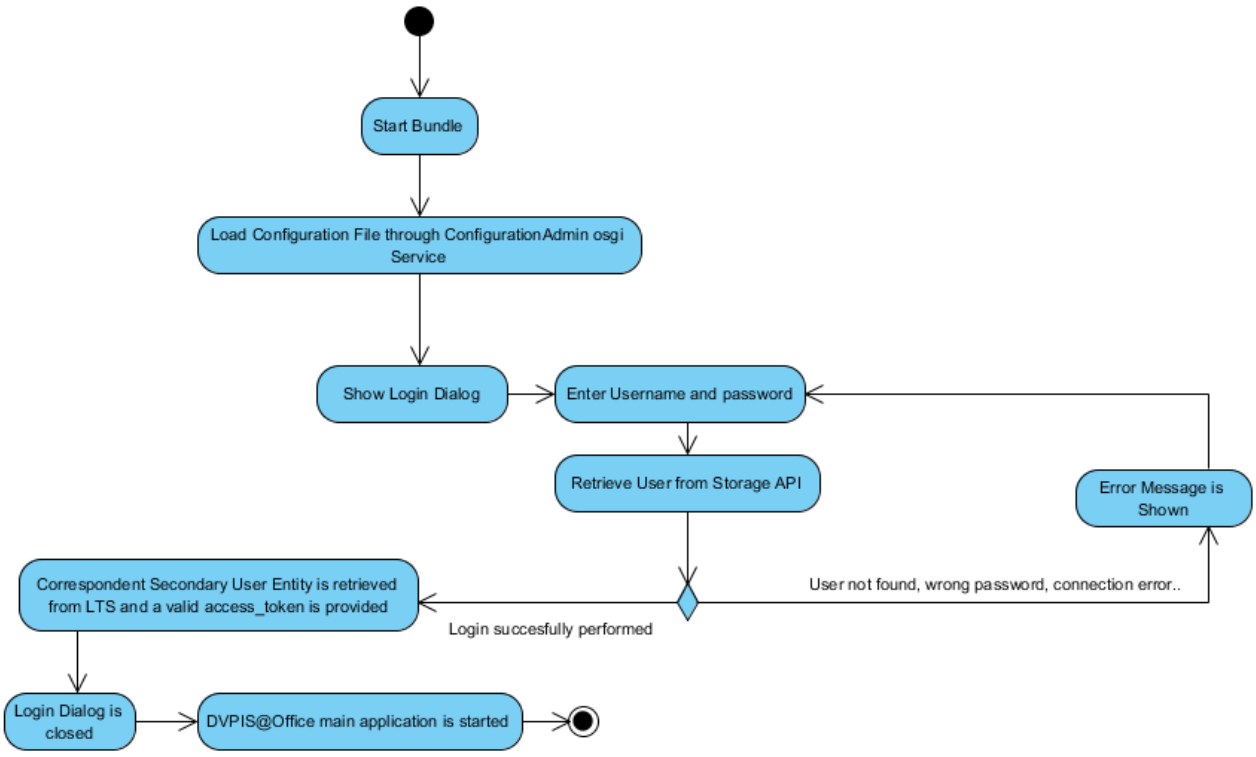


Figure 8 - Bundle activation and login phase activity diagram.

Essentially, when the DVPIS@Office bundle is started, an initial configuration is retrieved by the system. This configuration will be used for gathering system dependent parameters. After that, a login dialog will be shown to the Secondary User. Once the login has been performed and a valid User entity is provided the login dialog will be closed and the application will go through the next step: Start Services. The authentication process will be performed via *giraffplus.storage.wrappers* which must be installed and started before the DVPIS@Office.

3.1.2.3 Starting Services

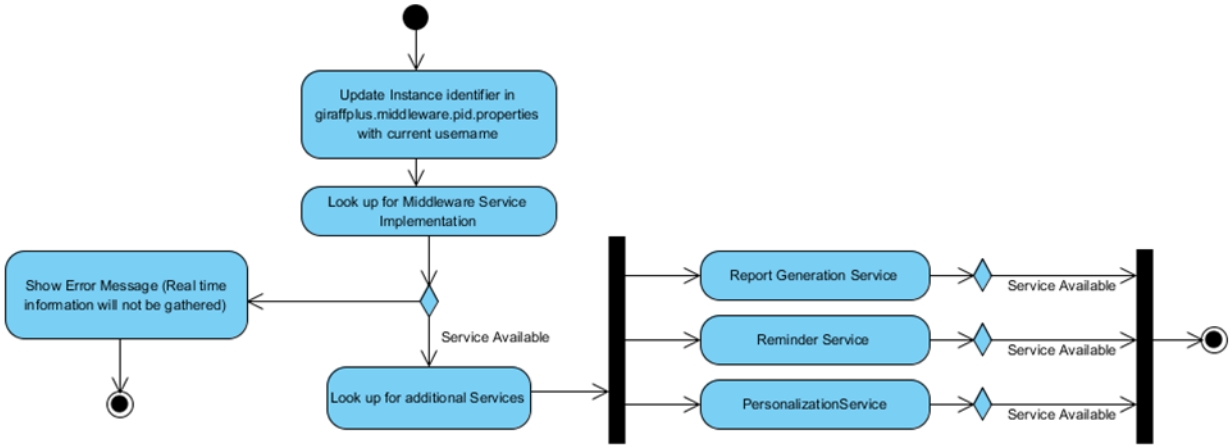


Figure 9 - Service activation activity diagram.

Once login has been correctly performed an instance of the "User Entity" that logged into the DVPIS@Office is retrieved from long-term storage. The unique (property held by long term storage construction) field "username" is used as global identifier for the DVPIS@Office instance that is totally related to the logged user. This field will be stored in a shared properties file named:

`giraffplus.middleware.pid.properties`

This file is placed in `/rundir/confadmin/services/` in the delivered alpha distribution.

Services like Reminder, for example, need to know which user is logged in order to properly deliver and get messages. This property is crucial to the correct definition of communication infrastructures used by the Middleware as the topic definition.

The first service that is searched into the OSGi framework is an instance of the Middleware. By adopting the *publish-subscribe* protocol the middleware ensures the delivery of data, also in almost real time. If the DVPIS@Office is unable to start the middleware for any reason, any of the data coming from the services engines do not reach that instance because the Middleware is the communication system on which every service relies. Nevertheless, the connection to the long term storage is working and historical data can still be queried and accessed. In this case a warning message is shown by the application starting procedure.

The next step is now gathering all other service references. If implementations of wanted services are found, that services are available to the system, otherwise not. By OSGi framework construction, it is possible that a service is also available after this step. In such a case, the DVPIS@Office is notified about the activation of a new service and this became available.

3.1.2.4 Load Houses and Primary User descriptions

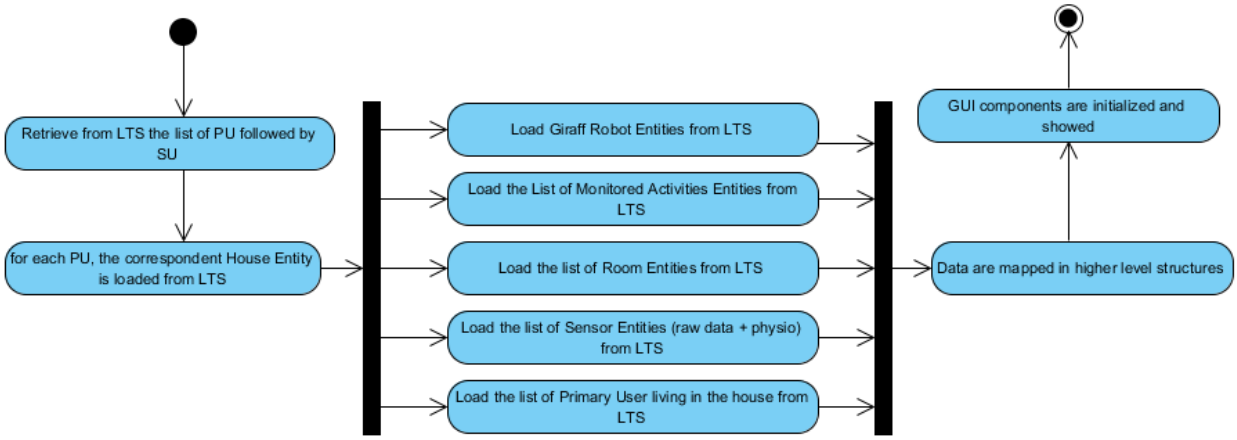


Figure 10 - Data loading activity diagram.

Finally, the DVPIS@Office gathers all data needed to construct the information set and these are displayed to the secondary user. In this phase most of the data are collected and mapped in internal data structures with the aim to speed up the access to data. In this step a large set of information is retrieved from the long term storage as described in the picture above. After that, the main GUI is displayed and all graphical components are initialized with proper data.

3.1.3 An example of use

3.1.3.1 Unpacking and starting the software

The DVPIS@Office alpha release can be downloaded as described in Appendix A.

Once the download is complete, uncompress the .rar file and double click on the *dvpis@office* icon highlighted in Figure 11.

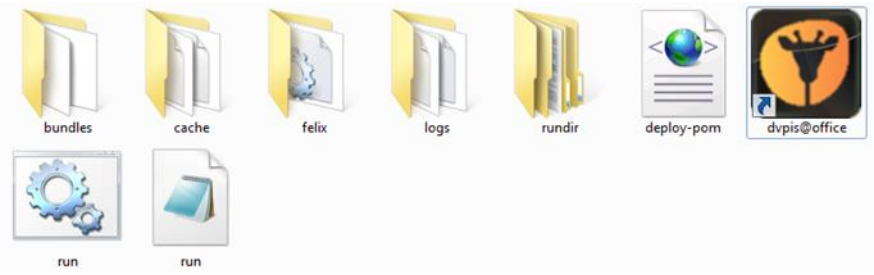


Figure 11 - DVPIS@Office main folder.

A login dialog-box will appear (Figure 12). In order to show the functionality of the software, a default secondary user account has been created and some primary user, houses definition data and some bunch of data have been embedded with the distribution through a configuration file.

Figure 12 - Login Dialog.

Insert the following credentials and press to the Login button to access to the application:

- **Username**: antonio
- **Password**: #2abam3#

3.1.3.2 General Layout

The general layout of the DVPIS@Office is shown in the Figure 13,

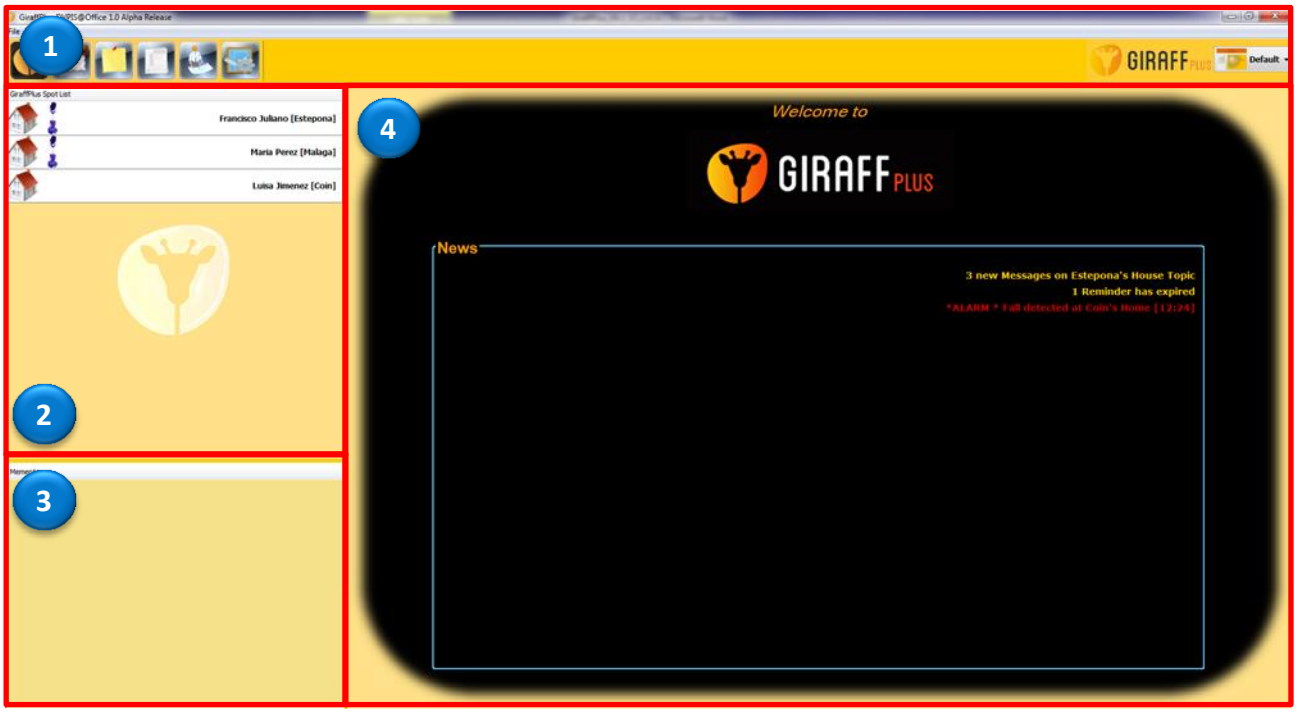


Figure 13 - General Layout.

This is composed of four basic main elements numbered in the above figure:

1. **Menu bar with general commands**: this area is the access point of the main services offered by the DVPIS@Office. A fixed area of the menu bar is reserved to the services which are designed to be always available. DVPIS@Office is projected to be extendible and in future releases it will be possible to edit the main bar adding entry points to additional services both by direct customization from the user or from the Personalization Service input.

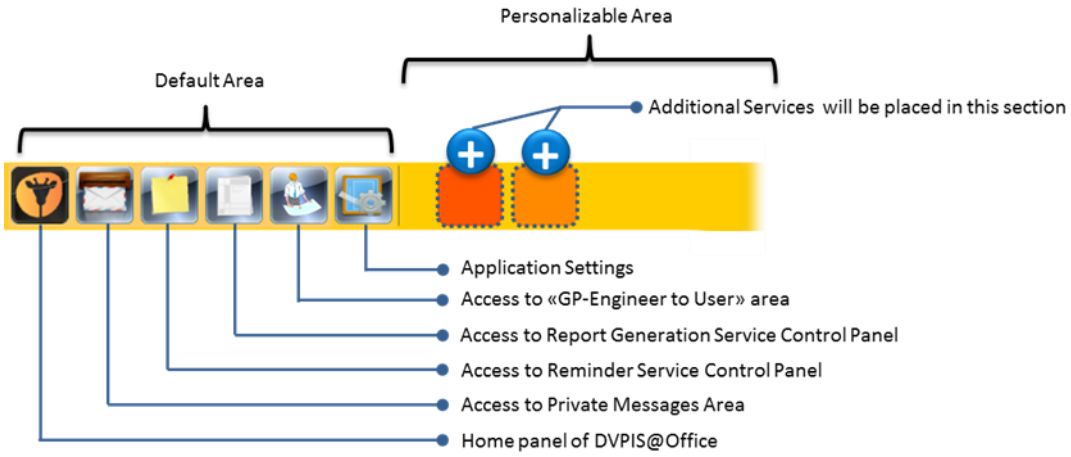


Figure 14 - Menu bar, detailed description.

Figure 14 illustrates in details the different functionalities of each button in the main menu bar. The picture shows how the main bar is split into two sections. For each additional service deployed together with a particular instance of DVPIS@Office a new button will be placed in the second section of the main menu bar.

2. **List of Primary users followed**: here the primary users currently under authorized access by the logged secondary user are listed. By Clicking on an item of such list the Main control Area will be updated and populated with the related data. Further information is delivered by different icon as displayed in Figure 15.

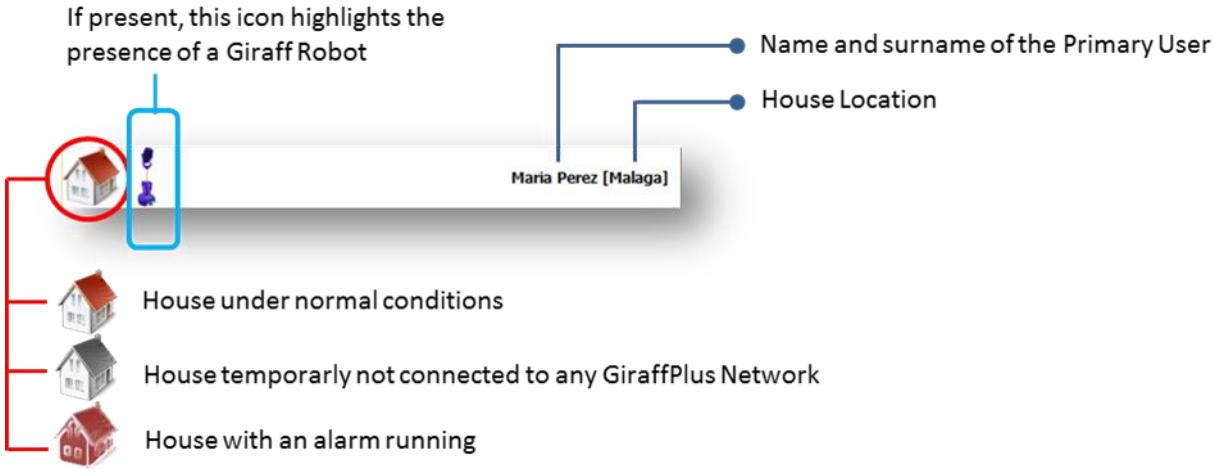


Figure 15 - List item: details.

3. **Container area for real time messages**: this panel will list all incoming real time messages delivered during an active session. Messages can spring from different services. Each item

will have a short description about the type of the delivered message and by clicking on it, a detailed dialog with all the information needed will be displayed.

4. **Main Control Area (e.g., where monitored data are visualized)**: this is the place where all detailed information about a test site will be displayed. The sub-components of this area will be filled from data of the selected item on the primary-user list. Hereinafter, a more detailed description of this component will be provided. Since this area is the more important one, it will be possible to enlarge this panel to full screen with a different main layout setting already available and accessible by the selection of the second item in the combo-box at the right-top corner as shown in the Figure 16.

Figure 16 - Choosing a different layout.

3.1.3.3 An example of data monitoring

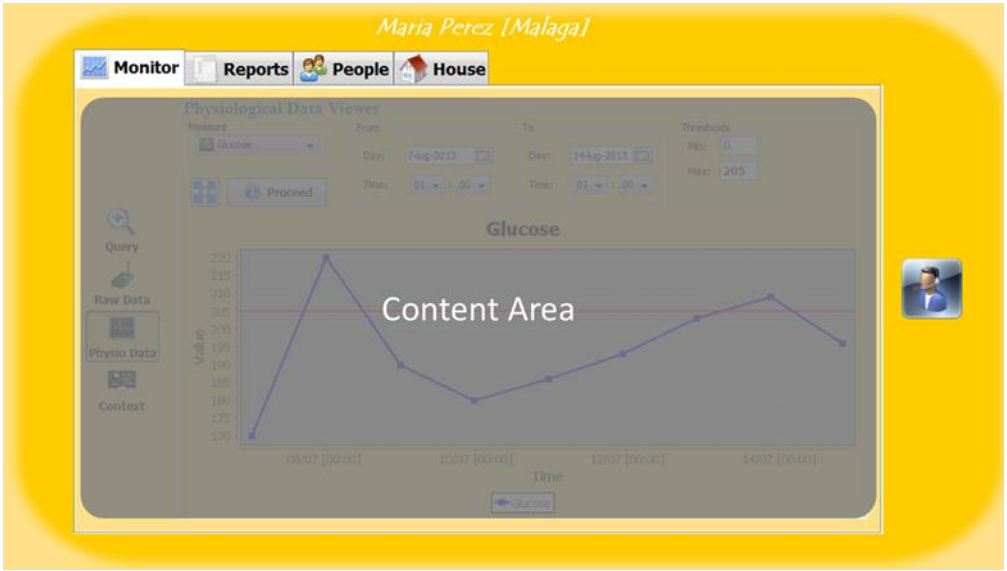


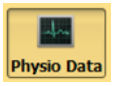
Figure 17 - Content Area.

The software delivered allows having an idea about how the data are shown to secondary users. The main content area shows 4 different tabs that aim to group the information in logical topics. These 4 different areas are so characterized:

- **Monitor**: through this panel the user can access most of the direct data coming from the monitored house. The data are then grouped in sub areas according to their meaning.
- **Reports**: all the reports generated by the system will be listed here. In addition here is an entry point to customize the report generation process.

- **People**: the persons related to each primary user are considered to belong to a network that is all around a specific primary user. From here it is possible to interact with them ,both start and follow discussions, etc.
- **House**: information about the house, locations are supposed to be shown here (e.g. the map of the house with the placement of the sensors)

It is possible to have a preview of how the data will look by clicking on “Physio Data” button.



The panel that appears is initialized with data from the previous week (for this distribution real data is not available). Through this panel shown in Figure 18 it is possible to query the LTS (Long Term Storage system) for visualizing stored data. In order to perform a specific time windowed query, two time intervals must be specified in the proper field (From / To) and then by clicking on “Proceed” button the new data will be retrieved and the chart will be updated.

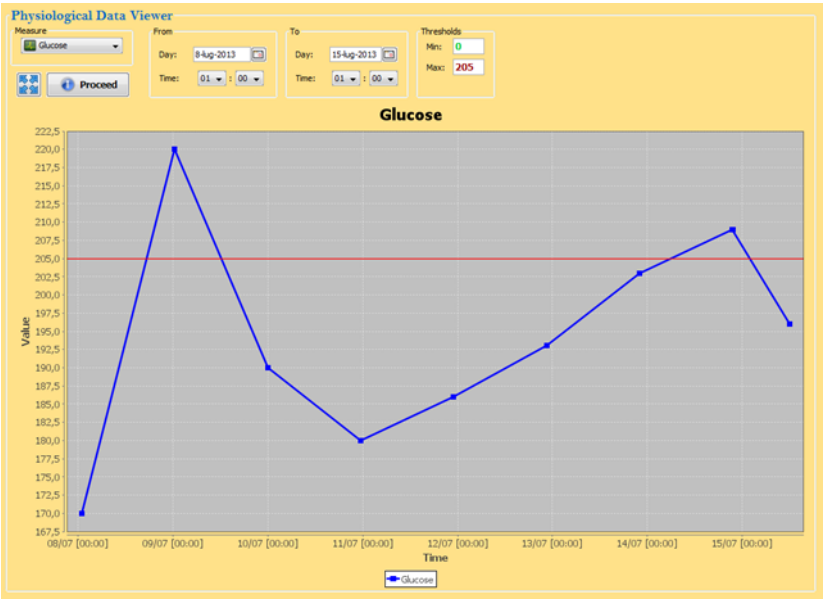


Figure 18 - Physiological data Panel.

Is it also possible to enter values for drawing directly on the chart two horizontal lines acting as thresholds, which could be very helpful when the user is visualizing a large amount of data. The example shown in the release contains 3 physiological measures: Glucose, Weight and Blood Pressure. By Switching for example to Blood Pressure (Clicking on the ComboBox at left-top corner), another chart will be shown (see Figure 19). The blood pressure measure contains three values for each assessment. This aspect is rendered in the chart by simply showing three different lines that describe the recent blood pressure assessments.

Figure 19 - Blood Pressure Chart.

The delivered software release also allows to have a short preview of how the activities will be displayed. By clicking to the "Context" button the main monitor Tab displays a view of the Context Recognition output module (see Figure 20).

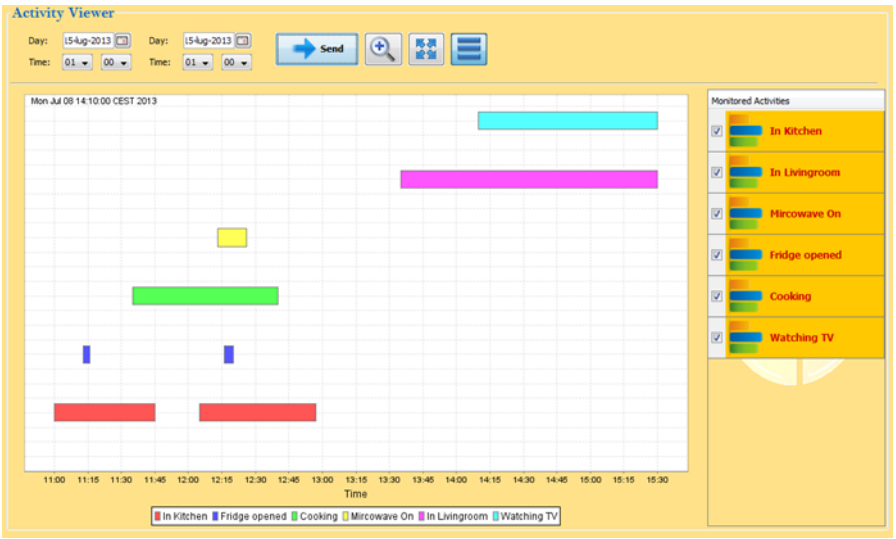


Figure 20 - Activity Panel.

The example shows few hours of activities temporally located at lunch time. On the right side a list of possible monitored activity is displayed. Selection or deselection of such items on that list will allow displaying or hiding the correspondent activity in the chart. The little button bar on top of this panel gives access to some additional functionality described below:

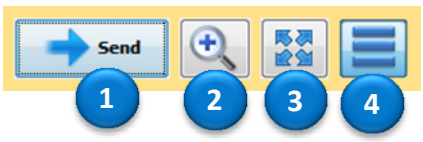


Figure 21 - Activity Panel button bar.

1. The Send button allows the secondary user to share data with other users included the primary user himself.
2. This button allows to query the Context Recognition module with the following input:
    a. the list of selected activities available on the right list (see Figure 20)
    b. two temporal intervals gathered from the Date field exposed in the Activity Panel
3. This provides a full screen view of the activity-chart
4. Here it is possible to Hide/Display the list of monitored activity on the right side.

3.2 DVPIS@Home

In a preliminary phase of GIRAFFPLUS we have considered alternative media to deliver information in the house. In order not to introduce too many technological items in the house we are now pursuing the use of the telepresence robot. The Giraff robot is an integral part of the project comprehensive design and has a computer on-board for its basic operations; we have decided to exploit it to run the local additional software services. The partner company that manufactures the robot has synthesized a touch screen version specifically for the project. The DVPIS@Home is designed to take advantage of this new functionality; somehow the telepresence robot resembles a "tablet on wheels" from the point of view of old people at home. The front end (a) allows to access the standard telepresence services of the robot (b) allows the home users to read simplified reports concerning his/her own health status (c) enables the delivery of asynchronous messages to the user. This last functionality is currently integrated for obtaining a reminding service (a message example is shown in Figure 6-D).

3.2.1 Module design

Many aspects have affected the design of this particular module. A first general issue must be taken into account: the system is meant to be installed and, just after that, is supposed to be fully operative. This is rather obvious but is important here because our system should limit as much as possible direct human interventions on the platform. The predominant user seniority aspect, leads us to consider dynamic systems that allows changing one or more parts of it which, over time, could have become obsolete or simply inadequate to a new state which the primary user may have acquired. Furthermore, the robustness must accompany the characteristics of AAL products like this. It is unthinkable that, for instance by turning off the robot, maybe while performing an update, it is required the intervention of a technician at home in order to fix a failed update process or to restore an older software version. To address these issues we provided the DVPIS@Home with some software-extendible characteristic. The OSGi framework, adopted as main software environment in the overall GIRAFFPLUS project, perfectly suits to our case.

The OSGi framework offers a set of software infrastructures that implements the Extender Pattern providing to a software element the property to be dynamically extendible. Another, maybe obvious but very strongly affecting issue, is the particular typology of the user: the elderly. Designing a user interface for an elderly user is, in fact, very much different and harder than designing it for a not-elderly person. Each user, in general, can be categorized into a specific group and some personalization pattern could be applied. This reasoning not simply perfectly suits with

the elderly users, which, due to a generically compromised health status, tends to change habits and perception faculties level over time and in some case also quite quickly. The overall design cannot avoid of taking into account the high need of personalization of the output stream of information that can vary consistently from a user to another, and on the same user can also vary over time. At this point is possible to wrap up the main properties that the DVPIS@Home should hold considering our precise contest:

- **Personalization**: defined as the capacity of the system to react to Personalization primitives as input coming from the Personalization Module.
- **Software Dynamic Extendibility**: defined as the capacity of the software to be extended (or reduced) with other modules able to communicate each other's. Some sub component has also the possibility to be dynamically updated without affect the overall performance and main basic features.
- **Robustness**: due to our particular type of user, the deployed software on the robot platform must be robust and remotely controllable as much as possible in order to prevent direct human software maintenance operations in place. The possibility to recover a desired state of the deployed system should be a satisfied property.

3.2.1.1 General Guidelines

Many studies in literature (Zajicek, 2001; Newell & Gregor, 2000; Karavidas, Lim, & Katsikas, 2005; Sam, Othman, & Nordin, 2005) exist regarding the difficulties encountered by the process of a user interface design dedicated to such a particular typology of user. As an example, Newell and Gregor (Newell & Gregor, 2000) define a methodology from which we get inspiration for the DVPIS@Home design process. This is called: User Sensitive Inclusive Design and aims to fill the gap left by the usual User Centric Design pattern when the end user is "critical" like in our case. User Sensitive Inclusive Design enforces a lot the personalization need. The word "sensitive" is crucial and means the need to build a dynamic system that includes a large set of specifications.

Starting from these considerations we have defined some high-level guidelines that has led all the DVPIS@Home design process. Each of the further point will find a counterpart in the implementation and actual design of the @Home application:

- **Variation of the Stimuli**: keeping using the same shape and output modality will not help to keep active and stressed the learning capabilities of a user. Considering that for instance the visual capabilities of an elderly user are often compromised, reading a long text could result very tiring and sometimes also impossible. Usually changing the output stream (e.g., audio message instead of textual) is a way to address this issue.
- **Familiarity**: even if this point is in contrast with the previous one, it is necessary to consider them both. Once a user has become familiar with a certain output-form, he will be much more prepared to deal with similar forms rather than experiment with new ones. Finally having a system that keeps maintaining the same shape and same way to access main functionalities, helps very much the user on its learning.
- **Simple Mental Models**: elderly users have often many difficulties to understand the patterns underlying a normal flow even if it is simple. Nested menu or tortuous paths needed for accessing a feature can discourage the user from using a graphical interface.

- **Fear of the Unknown (computer anxiety)**: many people but even more amongst the elderly have the fear of break a technological system when using it. This fear is simply based on the lack of experience of using technology. Unfortunately the effect is that a user with this fear prefers not to use a technological device rather than risking to break it. A good user interface should not let the user have a doubt about if he is doing an unsafe operation. Instruction should always be very clear. Usually, users like those tends to use help or to ask to some expert how to proceeds.
- **Reduction of Memory Overload**: showing too much information over a unique screen-frame can be counter-productive, especially when users are elderly or stressed. It would be preferable to display all the essentials information and that all the data needed to complete a procedure is immediately reachable and repeatedly accessible. A menu with many options could confuse a user instead of providing help even if providing a better set of available choices.

Table 1 describes how the above points have been considered and addressed in the current development process.

Guide Line	How is implemented
Variation of the Stimuli	DVPIS@Home will be able to produce an audio message for each textual message displayed on the screen. Personalization module will be responsible to set when this will occurs.
Familiarity	Each message displayed on the DVPIS@Home will have the same frame and same basic structure (e.g. the button to close a dialog will have same position and same icon). Also the main button providing basic functionality will have always the same layout, shape and position.
Simple Mental Models	No nested menu will be allowed by default. Most of the dialogs or popups displayed to the user will just have few buttons that will be mostly used to provide feedbacks for the system or for secondary users. The overall flow will be based on a general Action-Reaction pattern. Conditional flows will be allowed only if really needed.
Fear of the Unknown	DVPIS@Home offers to the users an **Audio-Tooltip** service. The application can produce a short audio message describing the functionality of each single button that has a textual description attached.
Reduction of Memory Overload	DVPIS@Home will provide only two main layers: one for the main frame and one for the incoming dialogs/popups. In case more than one popup/dialog is needed to be shown at the same time, the DVPIS@Home will handle the control of them and will show them one per time.

Table 1 - Implementation of main guidelines.

3.2.1.2 Interacting with existing Telepresence Functionalities

The primary role of the Giraff robot is to provide telepresence functionalities. This priority has been taken into account during the design phase of the DVPIS@Home and the resulted system will aim at not interfering with telepresence functionalities, but rather, to integrate and merge them with AAL services. This integration aims to raise the rank of a regular telepresence robot such as the Giraff to a Social Assistive Robot, i.e., an element of the class of robots in AAL environment able to provide assistance via social channels.

Even if the integration of the Giraff Telepresence software with the DVPIS@Home is still under work we can describe the general idea of this integration and how this will look. First, the telepresence software is meant to automatically start together with the machine and its GUI will occupy the screen size in full screen mode that is always active. However, it will be the DVPIS@Home to have this property, and the full control of the screen will be released to the telepresence software only during calls to get it back when call has terminated. So, the integration strategy has been to slightly modify the telepresence software providing some high level API that allows other applications to listen for events like incoming calls, ended calls and some more. Furthermore, the main panel containing the GUI elements has been exposed giving to other applications the possibility to embed this component in another GUI. In this way the DVPIS@Home will be able to embed and control the main events of the telepresence features of the Giraff Robot. The general design choice adopted here is that during a call no one of the incoming popups or dialogs will be displayed on the screen if the incoming source of such events is not the caller. This will help to keep the interface clear in respect of the guidelines previously described. Also, still some work must be performed in order to integrate both the components into the middleware infrastructure.

3.2.1.3 Software Extention Dynamics

The way that we have in mind to characterize the Personalization Modality in the DVPIS@Home can be split into two macro layers. The DVPIS@Home software may be thought as a main core element that can manage a variable set of different satellite sub-elements. Therefore, each instance of the DVPIS@Home can have a different set of those sub-elements and this difference will cause a first, macro level of Personalization. Moreover, each sub-element will expose the functionalities allowing the central Personalization Engine to adapt and modify their aspects and contents. Although some functionality will be mandatory to be implemented, the sub-elements will be able to expose or not most of the other personalization primitives.

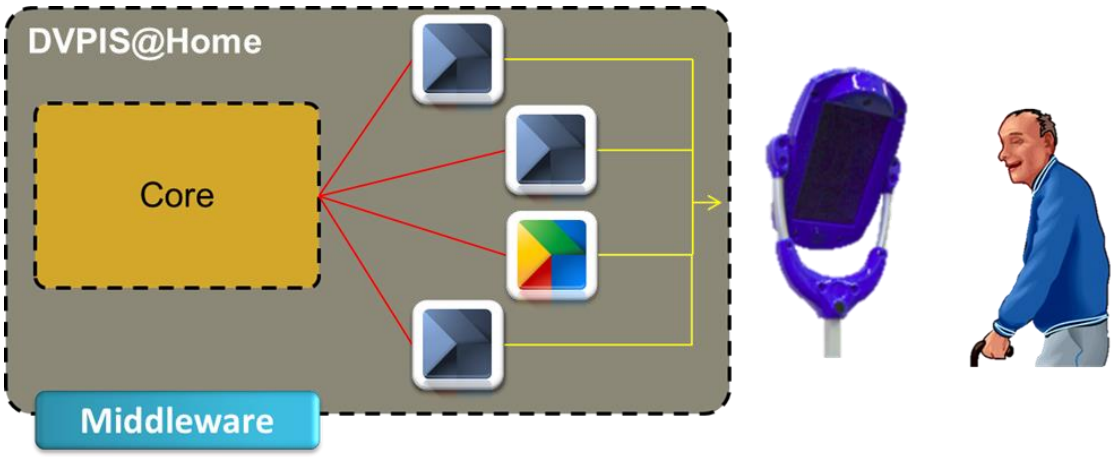


Figure 22 - DVPIS@Home design concept.

Each sub-element will be named, henceforth, **hPlugin** [Home Plugin]. Figure 22 shows a high level description of how the DVPIS@Home is designed. As anticipated above, the Core element will be responsible to:

- Dispatch proper Personalization directives to all hPlugins;
- Manage the screen as a shared resource between the Telepresence Software and the DVPIS@Home Services;
- Manage the consistency from the actual instance to a reference model defined remotely.

The last point is about the potentiality that we want to give to the Secondary User to define, remotely, the set of *hPlugin* installed and running on a particular DVPIS@Home instance. Each *hPlugin* will provide one (or more) AAL Service(s) that a secondary user may use during a GiraffPlus installation. To this purpose, a secondary user will be also able to perform some operations over time about the set of *hPlugin* available on a robot platform.

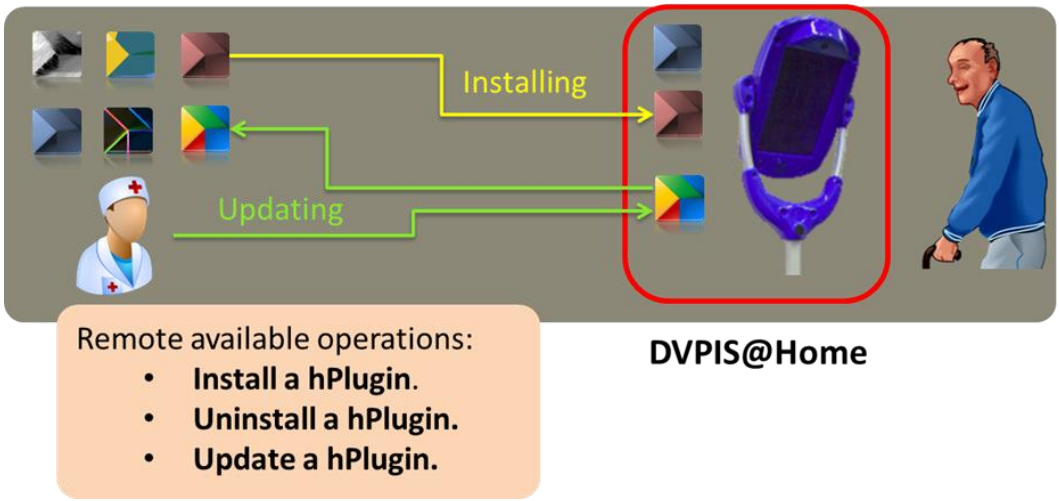


Figure 23 - The DVPIS@Home and the hPlugins.

Similarly to what happens with smartphones, the DVPIS@Home acts like a sandbox that will run single applications separating their flows from the main flow that will be totally managed by the

DVPIS@Home's core. Even if it is not implemented yet, a panel will be inserted in the DVPIS@Office application where the secondary user will have the possibility to visualize a complete view of the whole set of available hPlugins and she may decide whether installing new ones or deleting/updating others in each DVPIS@Home instance that he can control. This mechanism is visualized in Figure 23.

3.2.2 Implementation details

Also here, the OSGi framework has been considered as the more suitable choice to develop our designed model. As we want to exploit the benefit of having a Middleware (deployed as OSGi Bundle), the DVPIS@Home is presented as an OSGi-Bundle and the Felix is chosen as OSGi-Container. This section will illustrate more in details how two main entities constitute the final DVPIS@Home application (**the DVPIS@Home - Core** and the **DVPIS@Home - hPlugin)**.

3.2.2.1 DVPIS@Home – Core

As stated previously, the core has many important responsibilities among which an important one is to maintain the consistency between the actual local instance and a desired configuration, expressed remotely. This configuration is actually designed by writing a set of properties in an xml file. These properties, essentially define which exact *hPlugin* (with version) must be installed. A set of other properties will tells to the DVPIS@Home which *hPlugin* must be removed or updated providing a dynamic capability to change over time the macro state of each DVPIS@Home instance. The format of this configuration file is still under development and a final shape will be available after a long test phase. Many other parameters could be added or removed.

The current DTD model for the configuration file is shown in Table 2 and hosted at: http://pst.istc.cnr.it/giraffplus/hdvpis/hdvpis-settings.dtd

```
<?xml version='1.0' encoding='UTF-8'?>

<!ELEMENT setting
(user,logger,installedPlugins*,pluginsToUpdate*,pluginsToDelete*,pluginsToInstall*)>

<!ELEMENT user (#PCDATA)>
<!ELEMENT logger (#PCDATA)>
<!ATTLIST logger
    update CDATA #IMPLIED
  >

<!ELEMENT installedPlugins (plugin)*>
<!ELEMENT plugin (name,path)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT pluginsToUpdate (plugin)*>
<!ELEMENT pluginsToDelete (plugin)*>
<!ELEMENT pluginsToInstall (plugin)*>
```

Table 2 - DTD Model for the configuration file.

To provide to the DVPIS@Home the Consistency property, it is necessary to grant it with a **logging** mechanism in a complete DBMS transaction management fashion. In fact, we must prevent and manage situations where, for instance, the system crashes during an hPlugin installation/update procedure which could imply in theory, some sensible download time. Therefore, each of such operation will be wrapped in two main statements:

- *begin operation*
- *end operation*

These statements will be reflected into a log file. The responsibility for writing it will be of a new core-component called **LogManager**. Another core-component called **ConsistencyManager** (CM) will translate instructions expressed into the configuration file into real procedures that actually updates, installs and deletes hPlugins. This component also updates the remote configuration file always after the LogManager completes the writings. The CM will update the remote file sending messages through the Middleware on a special topic. At this point the consistency could also be broken if the connection is lost after the end-operation statement is entered into the log file and before the message is propagated into the middleware. This problem is solved with the *2-phase commit protocol* that in our case we implement by simply adding an additional log statement:

- *begin operation*
- *end operation*
- ***end propagation***

When the server side, hosting the setting file, receive the message (sent by the DVPIS@Home) to update the configuration file, it updates the file and sends back on a certain topic an ACK-Message that will cause a writing of "end propagation" message on the DVPIS@Home. At this point we can guarantee that both DVPIS@Home's configuration and the remote one are exactly the same. The full process is represented in detail in the Figure 24.

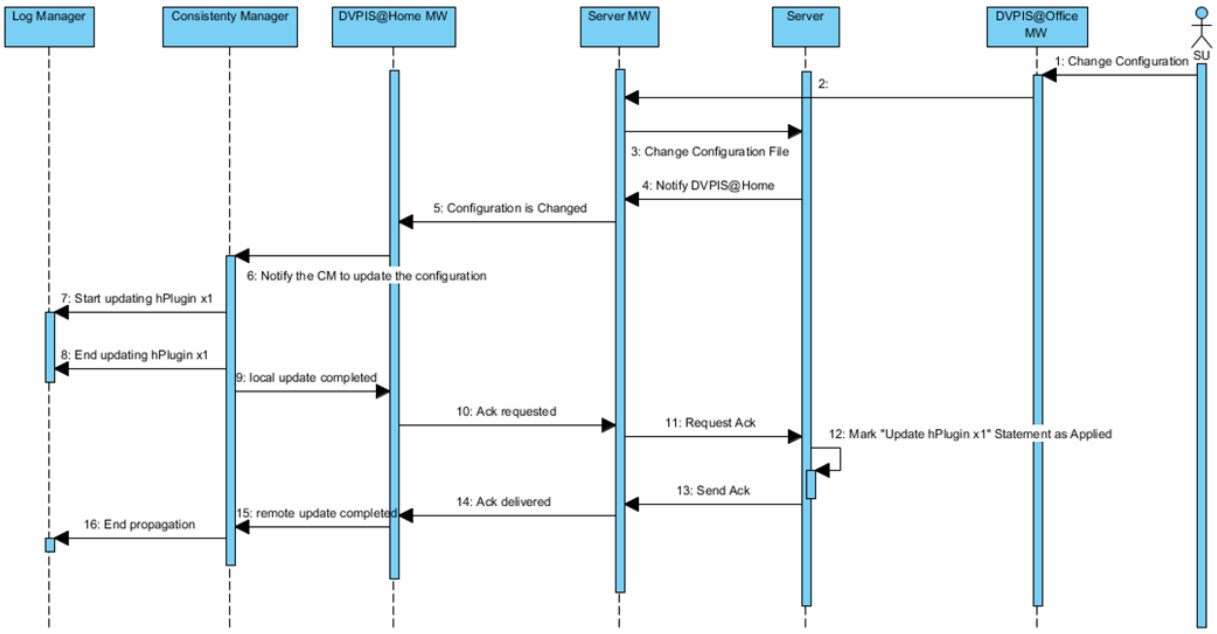


Figure 24 - Consistency Manager sequence diagram.

This process can be performed when the system starts but also dynamically in background during a normal session of the DVPIS@Home since the OSGi framework provides all the features to be able to do this. The main task of the CM is therefore to understand when this procedure (update the DVPIS@Home configuration) has failed and to consequently restore the system to a consistent state.

3.2.2.2 DVPIS@Home – hPlugin

The other main role of the Core component is to dispatch the personalization primitives to the hPlugins and to manage the screen as shared resources. In order to do this we must provide, first, a more strict definition of an hPlugin:

An hPlugin is an OSGi Bundle, which offers implementations of ***HDVPISPlugin Interface*** *and* ***HDVPISPersonalizable Interface*** *as* ***OSGi Services****.*

The previously cited interfaces expose a set of methods strictly necessary to fulfill the core's goals. The DVPIS@Home core will search in the OSGi environment all implementation of HDVPISPlugin Interface in order to list all available hPlugin. Through the implementation of this interface by each hPlugin, the core is able to share the screen amongst all running hPlugins.

Implementing the HDVPISPersonalizable Interface a Bundle declares to be able to react to some default personalization property (e.g., setting the maximum text font-size, maximum text length, switching to dark colours theme and so on). When a particular personalization directive needs to be applied to the DVPIS@Home, the Core will search all hPlugins that implements that particular one exposing a desired method and invoking that method. A little example will clarify this statement:

The robot is active in the house and some hPlugin is active and showing some messagse on the screen. Let us suppose that the primary users switch off the light. The Personalization engine, that is also listening on sensor data, is getting data from a sensor able to measure the luminosity in the room where the robot is placed. The Personalization engine decides that the environment is too dark; therefore triggers the signal to switch to a dark colored theme for that instance of DVPIS@Home active on that house. When the DVPIS@Home is notified with this information, his core will look for all active hPlugins that offer the Personalization Service and will call the "switchToDarkTheme" primitive on them.

The example here cited, wants to illustrate a simple loop that the DVPIS@Home is able to close offering a dynamic set of AAL services plus the overall extendibility property of the software.

Actually, other typologies of services are being designed, in order to enhance and enlarge the possible set of services that a single hPlugin can offer. Moreover, this architecture based on the service-oriented pattern with the adoption of the OSGi technology allows the hPlugins also to directly interact with the Middleware. This leads us to consider the opportunity to let to the hPlugins the possibility to provide another class of information about the health status of the Primary User. This kind of information could have a much larger range of quality, from a mnemonic status to a vision quality assessments and so on, giving to the Personalization engine and to the Secondary User a set of data that both the sensors and Context Recognition module cannot provide.

Even if all the advanced services are provided via hPlugins, some default services will already be provided by the Core. Such services will be very simple and consist of offering a view of some Primary User data and a simple way to communicate with the other GP-Users via textual or audio-messages. These concepts are still under design phase.

3.2.3 An example of use

Once the robot is started, the DVPIS@Home is started too and as soon as the loading phase is terminated, the DVPIS@Home main frame is shown in full screen mode, as shown in Figure 25.

Figure 25 - DVPIS@Home layout.

Applying the reduction of memory overload design option, the chosen layout is simple: a common frame hosts in its content area the different specific proper information according to the main functionality selected. To select a main functionality, it is possible to press the associated buttons as shown in the picture.

At the moment four buttons allows the user to access the following four different areas:

- **Call Button**: to switch to the telepresence functionalities. It is the default position when the DVPIS@Home is started. During a call it is possible to switch to the other functionality and then to come back to see the output stream of the call clicking again on this button.
- **My Info**: contains all sensible information about mainly physiological measurements taken inside the GiraffPlus system that the primary user wants to follow and periodically check and maybe share with doctors or relatives.
- **Messenger**: to access a social area where the primary user can list all the incoming messages that have been received and stored into the system. In the future it will also enable the primary user to send message (textual/vocal) to other users of the system when he cannot contact them by the main telepresence channel.
- **Apps**: it shows all the installed hPlugins. They appear as a grid of icons. By clicking on them the correspondent application will start in a modern smart phone fashion. The content area will be then occupied with the main frame of that active hPlugin. By re-clicking on the Apps button, the current active application is closed and the list of installed hPlugins will be shown again.

Some preliminary tests have been performed by implementing the Reminder Service Client for DVPIS@Home as an hPlugin providing some Personalization capabilities. Figure 26 shows an example of this test. We propagated the personalization primitive to switch to dark colors. Correspondingly the Reminder hPlugin implemented it showing a blue theme that better suits with the darker environment.

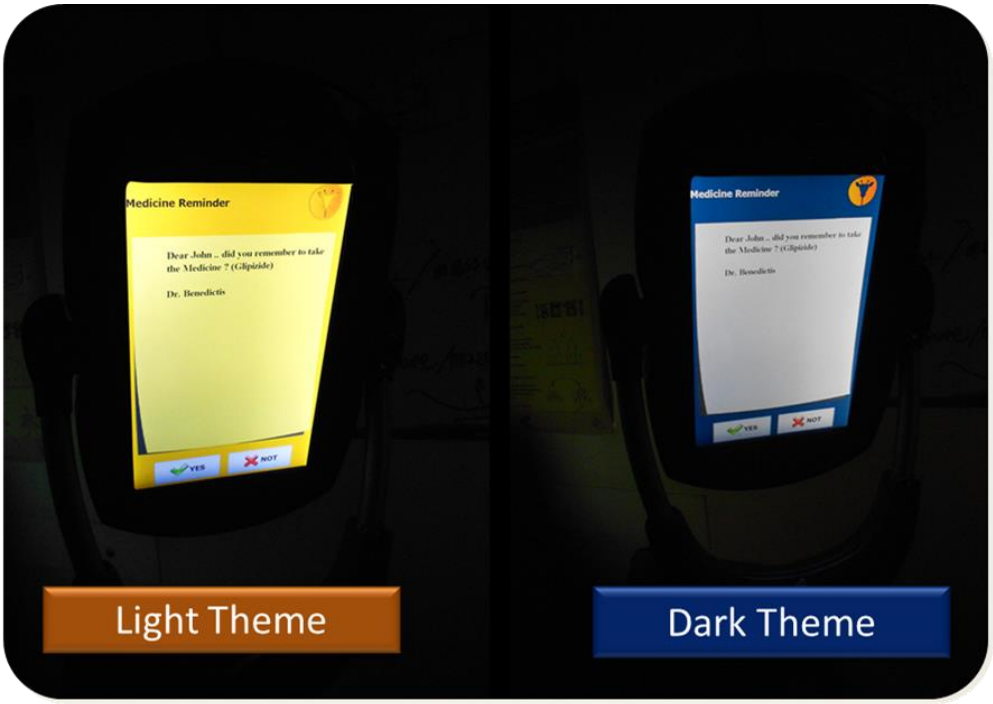


Figure 26 - hPlugin personalization example.

Another feature that we have been testing is to provide the secondary user with the functionality of sending a particular set of data to the primary user, as a base-material on which to start a conversation. Figure 27 shows a possible implementation of it. In the overall DVPIS@Home design this feature may be integrated as a default service that the application exposes.

Figure 27 - Showing data on the DVPIS@Home.

3.3 The Install & Maintenance Service

The problem of facilitating the installation and maintenance of the complete home software is quite relevant in GiraffPlus because we enter the house of fragile people and we have to minimize the impact. For this reason we are also investing in a project EngineerUI also called the Install & Maintenance Service. Such an environment was created to enable GiraffPlus professionals to easily configure and maintain the configuration of each monitored home in the GiraffPlus Long Term Storage (LTS). For this purpose a separate module has been designed taking advantage of the existing GiraffPlus LTS web service and presenting an easy-to-use interface that enables users to make changes directly in the GiraffPlus LTS.

3.3.1 Module design

To make the GiraffPlus EngineerUI as usable as possible we designed it as a HTML5 web application capable of running on most modern web browsers enabling GiraffPlus professionals to edit the configuration of individual monitored home from a variety of devices: from laptops to tablet computers (running either iOS, Android, Windows or any other OS that contains an HTML5 browser).

3.3.2 Implementation details

The GiraffPlus EngineerUI is implemented in two parts: the RESTful web service that handles requests generated by the GUI and handles secure communication with the GiraffPlus LTS web service, and the web application running in the users' web browser.

3.3.2.1 RESTful web service

The RESTful web service was implemented using the Play Framework (http://www.playframework.com/), which enables development of web services in Scala and Java and provides predictable and minimal resource consumption for highly-scalable applications.

This web service decouples the GiraffPlus EngineerUI from the GiraffPlus LTS web service and provides additional logic checks. It takes care of secure communication in two ways. First, the Apache web server was set up as a proxy to the RESTful web service implemented in the Play Framework to provide secure communication via SSL from the user's browser to the web service. Second, the Play Framework uses provided certificates to secure the communication to the GiraffPlus LTS web service.

By taking advantage of features provided by the Play Framework, we were able to implement all REST calls in an asynchronous fashion, making the implemented web service highly-scalable.

3.3.2.2 The GiraffPlus EngineerUI Web Application

The GiraffPlus EngineerUI was implemented in JavaScript using the Marionette.js framework (http://marionettejs.com/), an extension of the Backbone.js framework (http://backbonejs.org/). Both support the MVC architecture (model-view-controller) of web applications and offer a variety of functions that enable developers to write efficient and reliable code.

In addition we used Bootstrap (http://twitter.github.io/bootstrap/), which was developed by Twitter and is a powerful front-end framework for faster and easier web development.

3.3.3 An example of use

The GiraffPlus EngineerUI first presents the GiraffPlus engineer with the login screen shown in Figure 28. After the username and password are checked and the system checks that the user trying to access the system is indeed a GiraffPlus authorized engineer, the system presents the screen shown in Figure 29 in which the engineer can choose to edit configuration of any entity stored in the GiraffPlus LTS system. Figure 30 shows an example where the engineer is presented with a list of all monitored homes configured in the GiraffPlus LTS system. To add a configuration of a new monitored home, the engineer is presented with an input form show in Figure 31, where the engineer is asked to provide all information regarding the new home. When asked to enter the configuration of a new home, the engineer needs to add configurations of many other entities, such as sensors, Giraff robots, rooms and locations at the home, context recognition components running at that particular home, etc. All these input forms are shown embedded in the original "home add form" as show in Figure 32, where an example is show where the engineer is asked to provide details about the Giraff robot installed at that particular home. Figure 33 shows a screenshot how the GiraffPlus EngineerUI works on a smartphone, which has a very small screen and thus limited usability, but as the screenshots show, the web application still works, which may be useful for small changes that an engineer may have to make during the physical installation of the system at a new home – e.g. changing exact location of sensor installation.

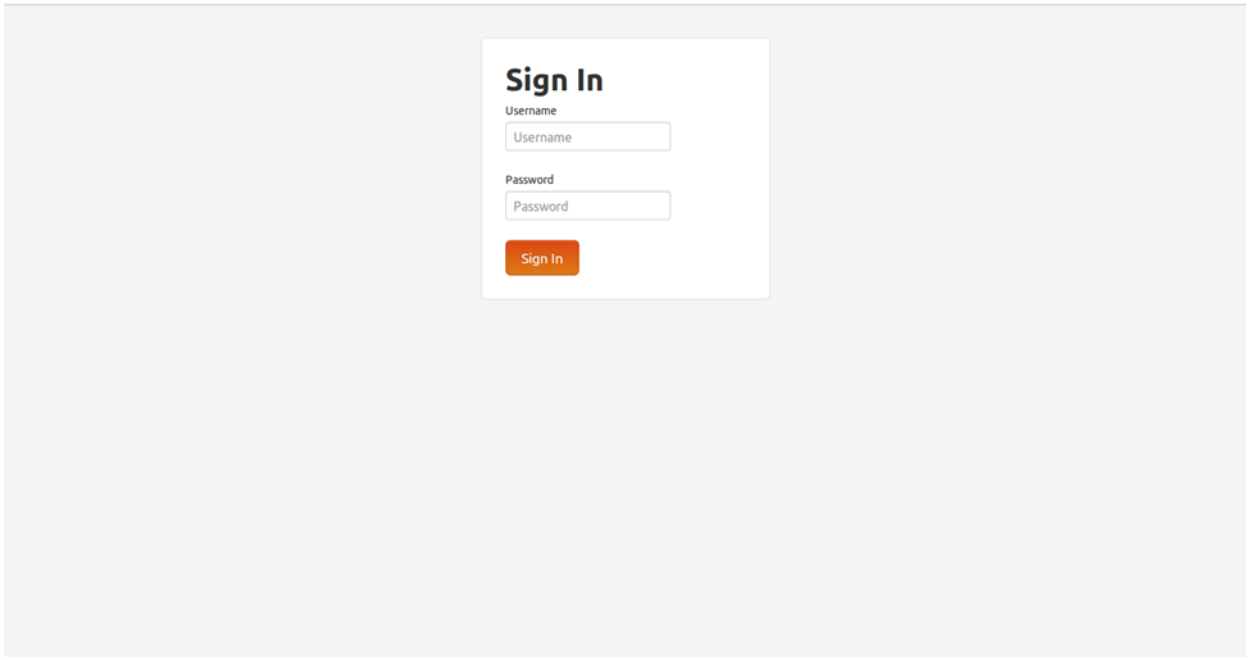


Figure 28 - Login screen of the GiraffPlus EngineerUI WebApp.

Figure 29 - Basic view of the GiraffPlus EngineerUI. Menu on the left side manages stored configuration entities..

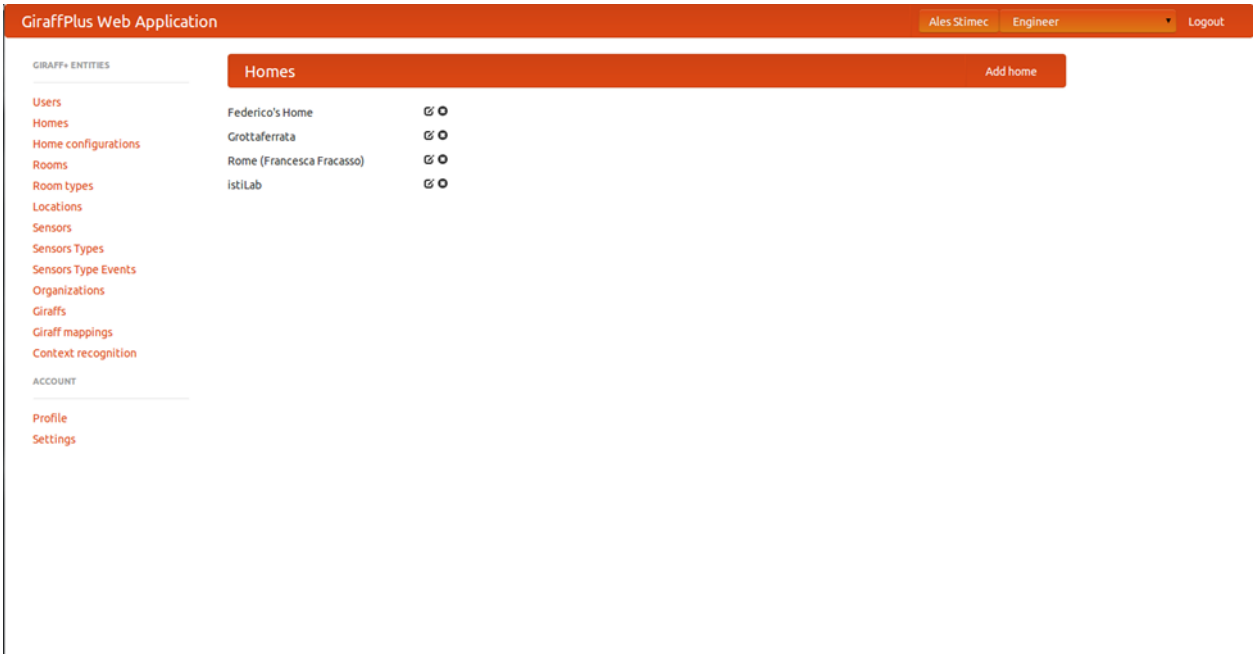


Figure 30 - The list of all monitored homes configured in the GiraffPlus LTS.

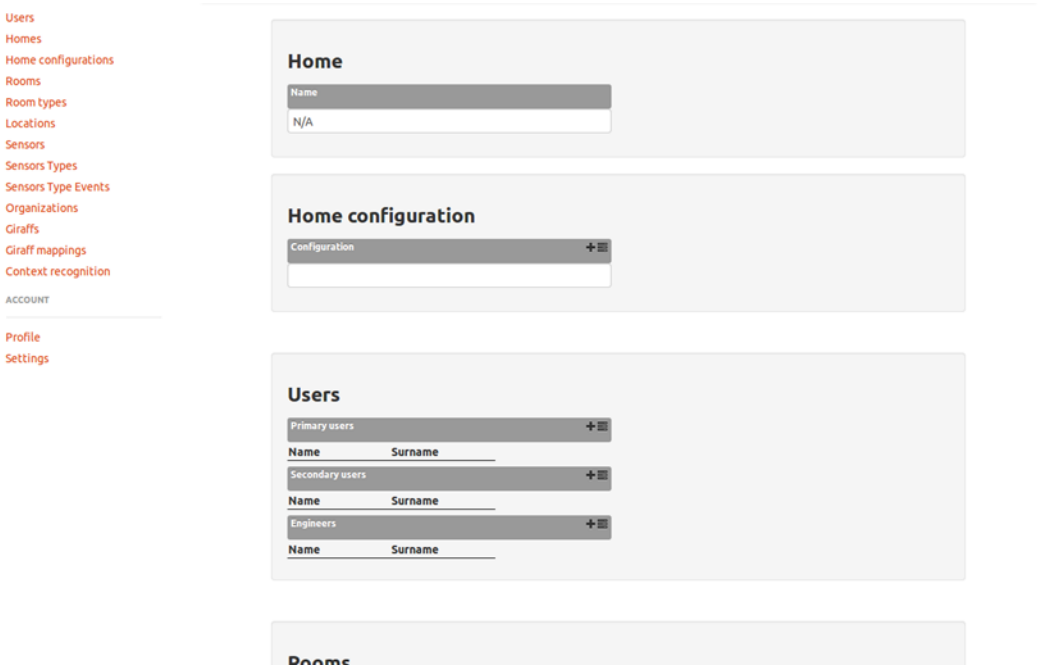


Figure 31 - Part of the input form presented to the GiraffPlus profesional when configuring a new installation.

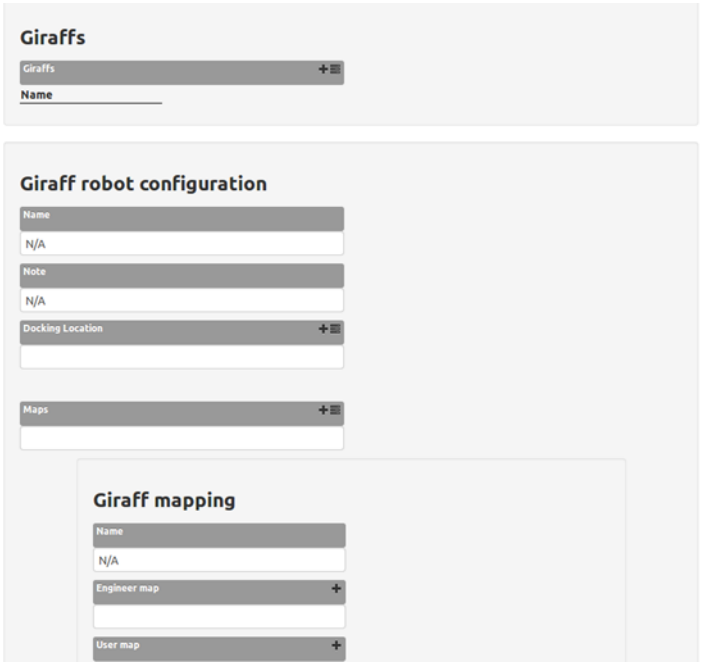


Figure 32 - The configuration of a Giraff robot can be attached to the configuration of the monitored home.

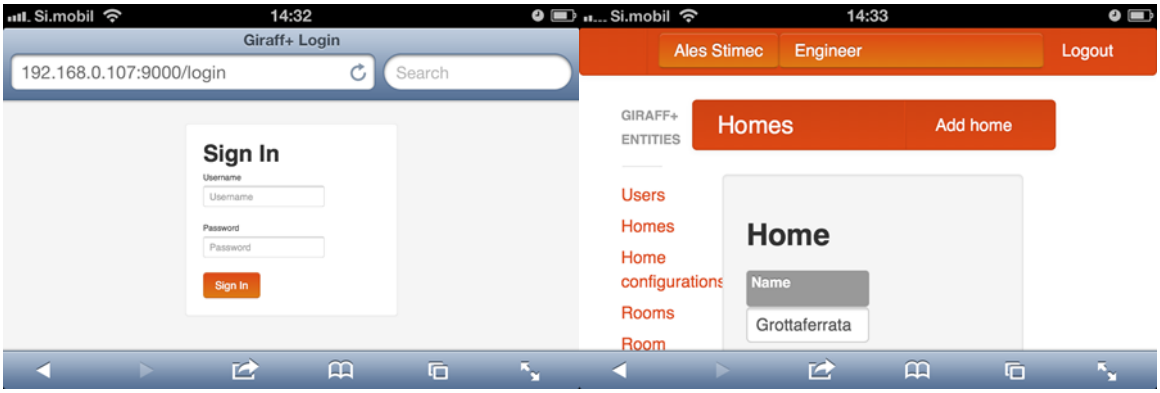


Figure 33 - A screenshot of the GiraffPlus EngineerUI as seen on the Apple's iPhone.

4 Conclusions

The GIRAFFPLUS infrastructure guarantees two very important aspects: (a) the long-term storage of sensor data; (b) the possibility of real-time connection for information sharing and immediate alarm intervention. The work on user-oriented services has started with an analysis of the different types of people who can take advantage of the GIRAFFPLUS services. The basic subdivision between users concerns the old person in the house (*Primary*) and the different people outside the house (*Secondary*). The secondary users can be subdivided in various groups: within GIRAFFPLUS we are particularly interested in services for: (1) *formal caregivers* (e.g., doctors, social workers); (2) *informal caregivers* (e.g., relatives).

One interesting distinction among the provided services concerns the expected time constants: (a) long-term data analysis, for example, to observe trends for creating regular reports for different users, (b) short-term reactive services, like in the case of alarms, to detect emergencies (e.g., fall detection sensors, emergency call buttons, etc.) (c) continuous asynchronous dialogues (e.g., in the case of social network channels, reminding services, etc.) (d) synchronous communication channels for conversations through the telepresence robots.

The GIRAFFPLUS project is currently in its second year. Since the first focus group with potential home users, the issue of considering a dialogue Inside/Outside has been a very important one. For this reason we are endowing the DVPIS with the capability of synchronous communications fully supported by the middleware. Using this functionality we have created a shared information space that allows a dialogue between Primary and Secondary users sharing some information on the screen. The current demo of this functionality is shown in the combination of Figure 6-B and -C. It is possible to see that the same information (the weekly blood pressure samples) are in front of the doctor, in her office, and of the old person, at home, and they can share also the audio channel for talking. In this way, we have set up an environment for flexible end-to-end information delivery on top of an AAL service.

Before closing two things are to be underscored again: (a) the current software release particularly focuses on user interaction aspects. The current personalization is rather static. Research is ongoing for a more flexible use of the personalization back-end in the DVPIS; (b) there is one interactive aspect that is not completely addressed here and concerns the capability of the primary users to be constantly informed of the changes that concern his/her continuous monitoring. This aspect will be specifically studied during the evaluation with real users and specific solutions will be designed and integrated in the processes described in this report.

5 Bibliography

Apache Felix. (n.d.). Retrieved from Apache Felix: http://felix.apache.org

Cesta, A., Coraci, L., Cortellessa, G., De Benedictis, R., Orlandini, A., Palumbo, F., et al. (July 2013). Steps Toward End-to-End Personalized AAL Services. *8th Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'13).* Athens, Greece.

Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A. (2009). Developing an End-to-End Planning Application from a Timeline Representation Framework. *Proc. of the 21th Conf. on Innovative Applications of Artificial Intelligence.*

Cortellessa, G., De Benedictis, R., & Pagani, M. (June, 2013). Timeline-based Planning for Engaging Training Experience. *23rd International Conference on Automated Planning and Scheduling.* Rome, Italy.

Jonsson, A., Morris, P., Muscettola, N., Rajan, K., & Smith, B. (2000). Planning in Interplanetary Space: Theory and Practice. *AIPS-00. Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling.*

Karavidas, M., Lim, N. K., & Katsikas, S. L. (2005). The effects of computers on older adult users. *Computers in Human Behavior , 21*, 697--711.

Muscettola, N. (1994). HSTS: Integrating Planning and Scheduling. *Intelligent Scheduling.* Morgan Kauffmann.

Newell, A. F., & Gregor, P. (2000). "User sensitive inclusive design" - in search of a new paradigm. *Proceedings on the 2000 conference on Universal Usability.* Arlington, Virginia, USA.

OSGi Alliance. (n.d.). Retrieved from OSGi Alliance: http://www.osgi.org

Quinlan, J. R. (1996). Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research , 4*, 77-90.

Sam, H. K., Othman, A. E., & Nordin, Z. S. (2005). Computer Self-Efficacy, Computer Anxiety, and Attitudes toward the Internet: A Study among Undergraduates in Unimas. *Educational Technology & Society , 8*, 205-219.

Zajicek, M. (2001). Supporting older adults at the interface. In C. Stephanidis (Ed.), *HCI* (pp. 454-458). New Orleans, USA: Lawrence Erlbaum.

Appendix A

This report is supposed to support the delivery of the DVPIS Alpha Release. A demo version of the software can be downloaded from the following link:

http://pst.istc.cnr.it/giraffplus/demo-releases/dvpis_alpha_20130724.rar

In case the reader is interested to a complete demonstration, please contact the project coordinator.

In order to properly run the demo is important to satisfy the following requirements:

Mandatory	
Requirement	**Downloadable at**
Java Runtime Environment Version 7+ (WINDOWS Users)	http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html
Java Development Kit, Version 7+ (MAC OS Users)	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
Recommended	
24-inch size Monitor Screen *(smaller size could have layout problems while running dvpis@home simulator)*	
1920x1080 Screen resolution *(smaller size could have layout problems while running dvpis@home simulator, and the decorated version will be disabled)*	

Table 3 - System requirements.

Once the download is complete, unzip the .rar file and double click on the dvpis@office icon highlighted in Figure 34.

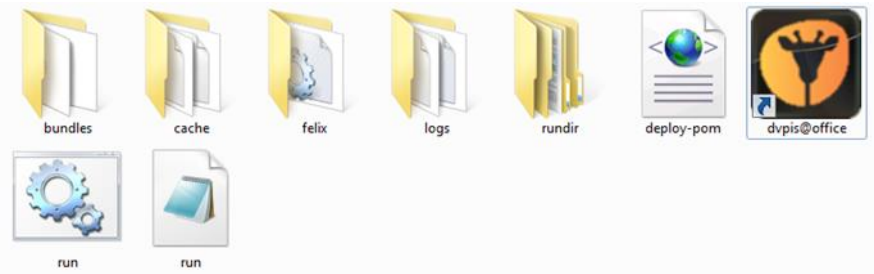


Figure 34 - DVPIS@Office main folder.

A login dialog-box will appear (Figure 35). In order to show the functionality of the software, a default secondary user account has been created and some primary user, houses definition data and some bunch of data have been embedded with the distribution through a configuration file.

Figure 35 - Login Dialog.

Insert the following credentials and press the Login button to access the application:

- **Username**: antonio
- **Password**: #2abam3#

The DVPIS@Office will now start and the main frame will be presented with the layout described in Figure 13. Since the test sites are still not active and we don't have real data available, we attached to the demo release a bunch of data in order to show main implemented functionalities. The provided account credentials belong to a dummy secondary user who is following the three primary users listed in the left area (see section 3.1.3.2 for further details).

By selecting a primary user, the content area will be filled with some data (data are the same for each user). Two views will be available:

- Physiological Data
- Activity Data

These views are accessible by clicking on the buttons highlighted in the picture below:

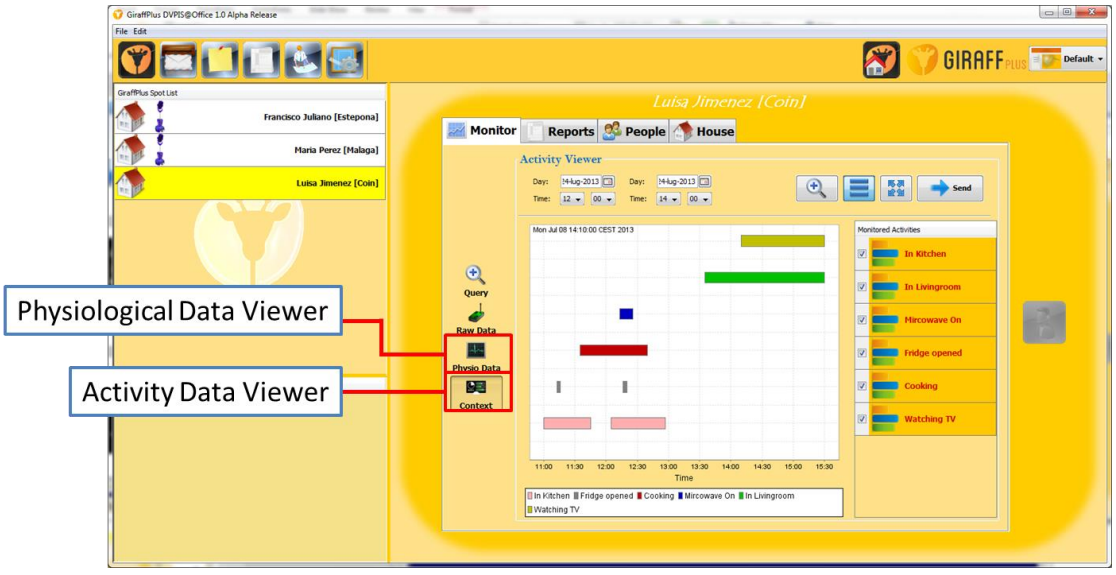


Figure 36 - Available Demo Data Viewers.

A preliminary version of the Reminder Service has been integrated into the DVPIS@Office. To access the Wizard that will allow you to define a simple reminder, please click on the Reminder Button and follow the steps here described and indicated in Figure 37:

1. Press the Reminder Button.
2. Select “My Self” option. This will create a Reminder that will be delivered to the logged in secondary user.
3. Click Next.
4. Edit the content of the reminder. To simplify the procedure in this version the wizard will automatically create a reminder that will be shown 8 seconds after the wizard completion.
5. Click Finish to activate the reminder.

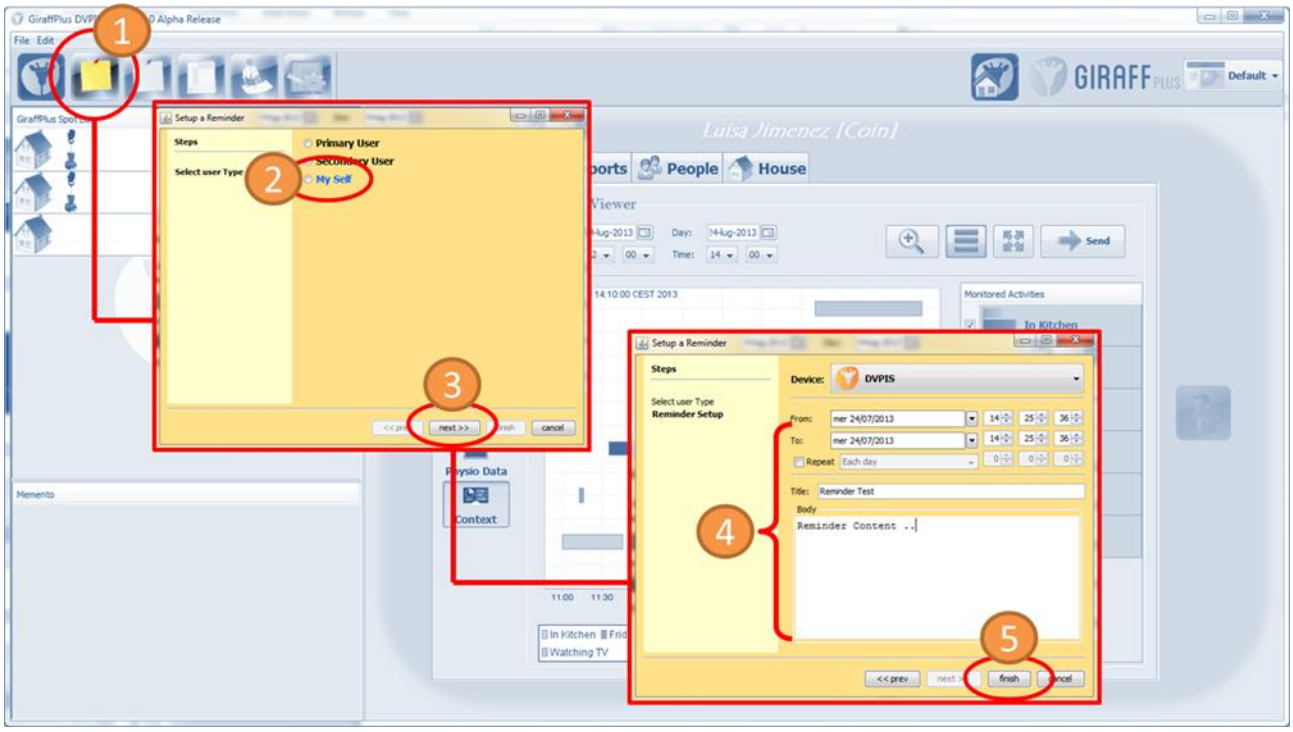


Figure 37 - Reminder Wizard.

When the Reminder will be triggered a popup will be displayed with the content of the reminder.

In order to have a preview of the layout and basic functionalities of the DVPIS@Home it is possible to start a simple DVPIS@Home simulator (see Figure 39) by clicking the button highlighted in the Figure 38.

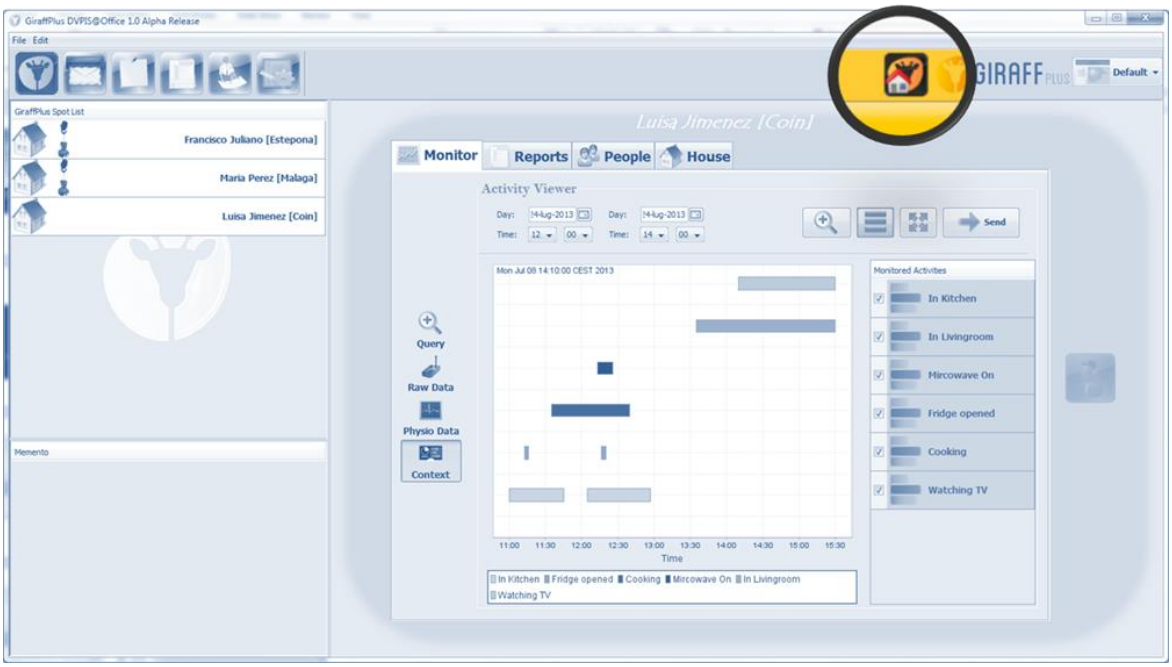


Figure 38 - Starting the DVPIS@Home simulator.

Figure 39 - The DVPIS@Home simulator.

Hence, in the second frame on the left of Figure 39, is being simulated a very preliminary version of the DVPIS@Home, mainly focused at showing the general layout and some basic functionality. For instance in the example delivered the primary user can have a view of his own physiological data.

Once the DVPIS@Home simulator has been activated it is possible to test the Send data functionality by the Activity Data Viewer in the DVPIS@Office. By pressing the Send button, as shown in Figure 21, the data will be sent at the DVPIS@Home instance and the DVPIS@Home will look like in Figure 40:

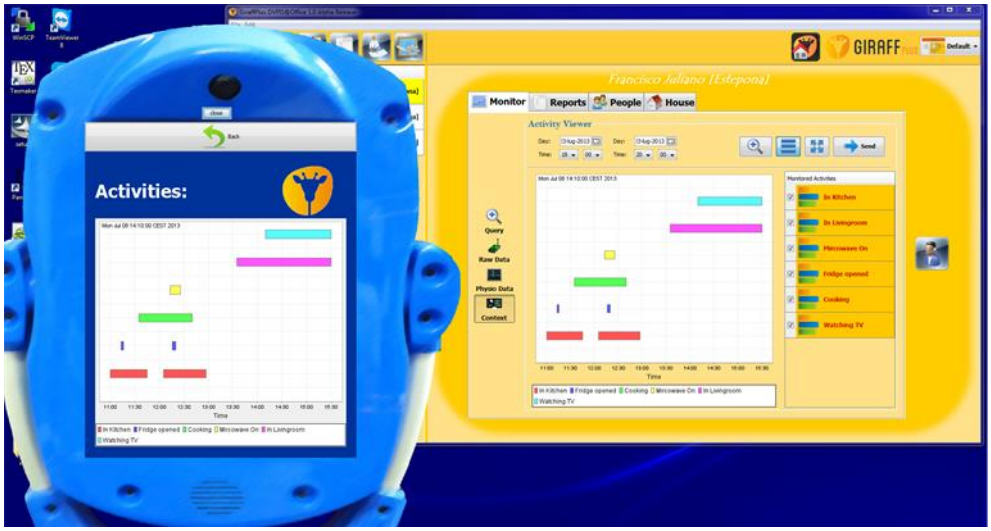


Figure 40 - Sending activity data to the DVPIS@Home.

While having active the DVPIS@Home Simulator, it is also possible to set a reminder targeted to a primary user by following the steps indicated in Figure 41.

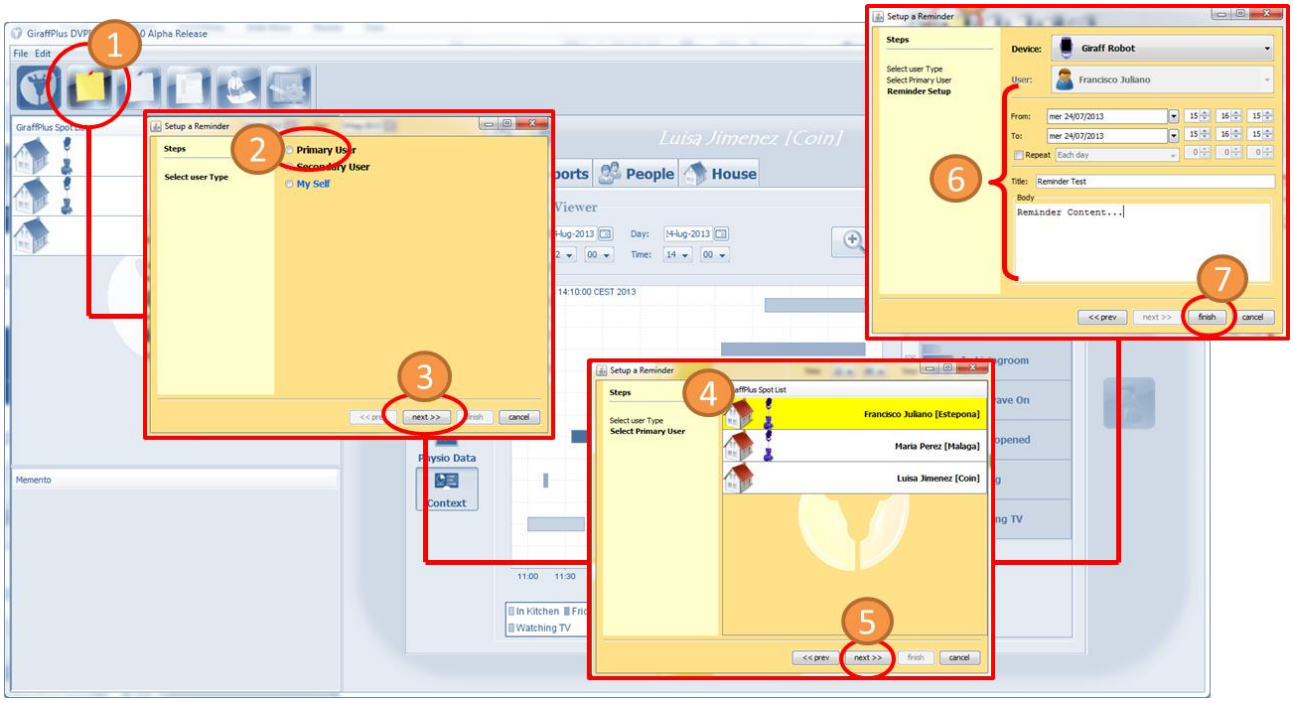


Figure 41 - Setting a reminder for the Primary User.

After a fixed time of 8 seconds (just for demo purposes), the reminder will be shown into the DVPIS@Home Simulator, see Figure 42.

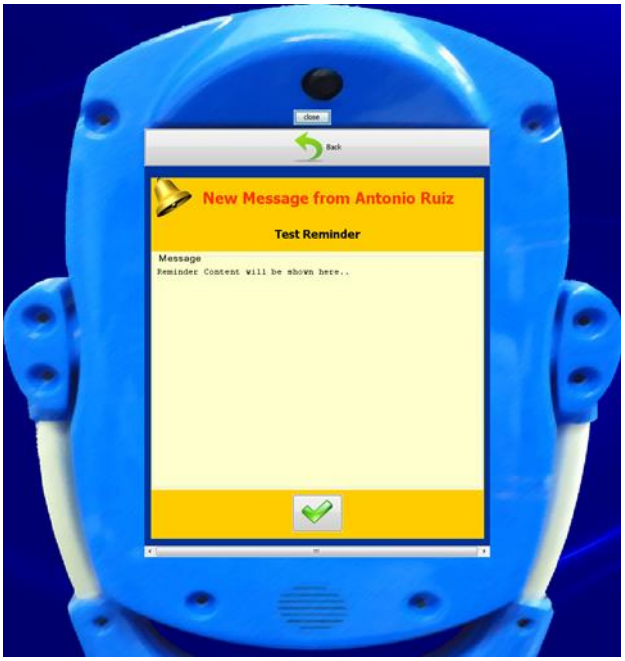


Figure 42 - The DVPIS@Home displays a reminder.