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**

D4.2 The Interaction and Visualization Service and Personalization Module – Beta Release

WP related to the Deliverable:	4
Nature:	P
Dissemination Level:	PU
Version:	Final
Author(s):	Giulio Bernardi (CNR-ISTC), Amedeo Cesta (CNR-ISTC), Luca Coraci (CNR-ISTC), Gabriella Cortellessa (CNR-ISTC), Riccardo De Benedictis (CNR-ISTC), Andrea Orlandini (CNR-ISTC), Ales Stimec (XLAB)
Project Participant(s) Contributing:	CNR-ISTC, XLAB
Contractual Date of Delivery:	20140630
Actual Date of Delivery:	20140715

Document History

Version	Date	Type of editing	Editorial
0.1	09/06/14	Table of Contents	CNR-ISTC
0.2	19/06/14	First Draft	CNR-ISTC, XLAB
0.3	02/07/14	First Complete Draft	CNR-ISTC
0.4	08/07/14	Second Complete Draft	CNR-ISTC
0.9	14/07/14	Third Draft (w. consortium comments)	CNR-ISTC
Final	15/07/14	Final Version of the Document	CNR-ISTC

Deliverable Summary

The aim of this deliverable is to describe the components of the DVPIS (Data Visualization, Personalization and Interaction Service) to reflect the current status of the integrated demo in the Beta Release. The deliverable is a natural continuation of D4.1 that introduces some of the concepts also addressed here. It is worth observing that the additional year of work since D4.1 contributed to gain maturity and strength to all the parts of the workpackage work. The deliverables is mainly distributed in 5 sections, the first one being introductory, while the subsequent four sections are dedicated to key aspects of the tool, namely the services for the secondary users, for the primary users, for the engineers who are responsible for the installation of the DVPIS in the real homes and a final one that describes the current personalisation services.

1 Table of Contents

List of Figures

List of Tables

2 Introduction

The GIRAFFPLUS WP4 is the 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 to allow different users the access to 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).

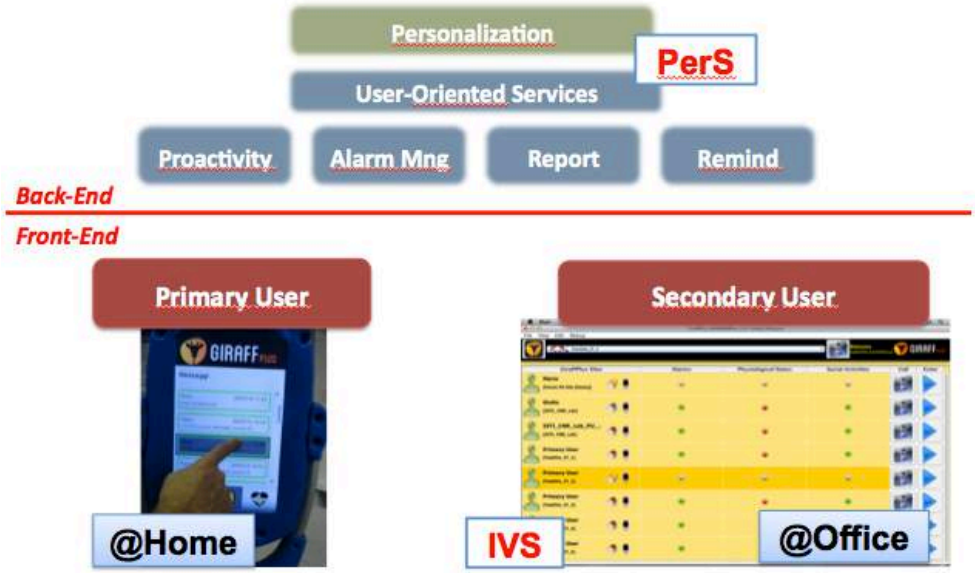


Figure 1 The WP4 set of services based on the DVPIS

The overall idea is sketched in Figure 1 where the key aspect is that the DVPIS service covers the complete issue of bringing data in the right form to each category of users. The DVPIS has been realized to manage interaction with the different actors of this AAL scenario. In particular, two different instances of the DVPIS have been designed, implemented and provided with this release:

one devoted to the secondary users (DVPIS@Office), and another dedicated to the primary users (DVPIS@Home).

The figure illustrates the general idea of the DVPIS. The DVPIS backend is responsible for preparing "personalized information to the users" and to offer support for different types of services like reminders, reports and alarms (and also the proactive suggestions and warnings described for the first time in Section 6 of the present deliverable). The front-end part is responsible for presenting the information and services to the different categories of users. To this purpose the project has developed two modules:

- @Office devoted to the secondary users and
- @Home dedicated to the primary users.

The rest of the deliverable will show the evolution of these two modules during this year of work since D4.1. Additionally the deliverable describes in Section 5 the evolution of the Install and Maintenance module that is not included in the general picture above but has been introduced in D4.1 to describe the work done to support test site deployment with a set of engineering tools.

2.1 Scope of the document

The first WP4 deliverable (D4.1), released at M19, described the Alpha Release of the DVPIS Modules. The present deliverable describes the situation at M30, in particular focusing on the novelties with respect to the Alpha Release contained in the Beta Release of the DVPIS, which is delivered together with the document. Such DVPIS release is a significantly more mature version of the system that includes both services modified by input from intensive field test (i.e., the use test sites and a special evaluation session described in D6.2 and also mentioned in Section 3.2 of this deliverable), and also completely new services not present in the previous release.

2.2 Deliverable structure

Aim of this deliverable is to describe the components of the DVPIS to reflect the current status of the integrated demo in Beta Release. The deliverable is a natural continuation of the D4.1 that introduces some of the concepts also addressed here. It is worth observing that the additional year of work contributed to both the enhancement of the single subcomponents of the DVPIS module (i.e., the DVPIS@Office and the DVPIS@Home) and to the improvement and strengthening of the overall integrated module. The new version of the module is the results of a cyclic *evaluation-development-deployment* approach that exploited the feedback obtained from both the use in real contexts (test sites) and a further experimentation with representative of users.

The rest of this deliverable is subdivided in 4 parts:

- Section 3 describes the @Office evolution. During the reported period we have decided to redesign completely the tool following the results of the evaluation hence this section describes the initial version and the new one in order to maintain consistency with respect to D4.1 without requiring going back reading the previous report.
- Section 4 describes the implementation of the @Home which is currently deployed on the Giraff robot.
- Section 5 describes the new support services that have been added to the Intall&Maintain module.
- Section 6 offers a report of the current personalization services as implemented in the DVPIS back-end.

2.3 Deviations with respect to the plan

The WP is in-line with expectation. Furthermore one aspect that is worth mentioning is that the integration effort in GiraffPlus turned out to be very demanding and this contributed to accumulate some delay on some specific work of the DVPIS module. Specifically, the DVPIS tool was influenced by the integration effort since several services should be visible through the DVPIS. In some case development of new features of the DVPIS has been delayed to serve the needs of the integrated system. Also the prioritization of the various part of the work has always preferred the support to the tools directly used in the test sites. As a result, in our self-assessment of the WP results we should mention that the current results we have on personalization are a bit late with respect to the original plan due to the effort dedicated to the building of a robust integrated system. In this respect the results described in Section 6 are currently subject to further refinements and improvements that we are pursuing by the end of the project.

3 The DVPIS@Office

The DVPIS@Office is a software environment dedicated to allow access to the GiraffPlus system by secondary users. In particular it contains services for both formal and informal caregivers that can inspect the data gathered in the house, and also have the possibility to use the Giraff Pilot to communicate with the house.

3.1 The DVPIS 1.0: Summary of the deployed functionalities

In this section a brief summary of the main structure of the DVPIS 1.0 module is provided. This summary is useful to understand how the DVPIS 2.0 (also referred to as DVPIS Beta version) has been improved and changed to respond to new requirements coming after a period of usage and in response to users' feedback. This first version of the system allows a generic secondary user to follow a list of homes (and consequently primary users), which have the GiraffPlus system installed in their apartment (see Figure 2 A).

Figure 2 Home Panel and Environmental Data Visualization in the DVPIS@Office v1.0

The list of followed primary users has also an indication on the status of the house (connected or disconnected, the house icon that is coloured or grey respectively) and the Giraff Robot presence (present or absent, Giraff robot icon). A central panel provides a summary view of the latest news related to the test sites. For each of the followed primary users, the secondary users can observe the environmental sensors data in form of timelines (see Figure 2 B);

Figure 3 - Physiological Data and Report Visualization in DVPIS@Office v1.0

the physiological data specific for the patient (Figure 3 C); and the daily/weekly or monthly report for the main activities in order to observe possible deviations from routines (Figure 3 D).

Finally, the secondary user can observe the set of "high level activities" (i.e., activities like, *cooking*, *watching TV*, *sleeping*) inferred by the Context Recognition Module by reasoning upon data coming from sensors (Figure 4) module.

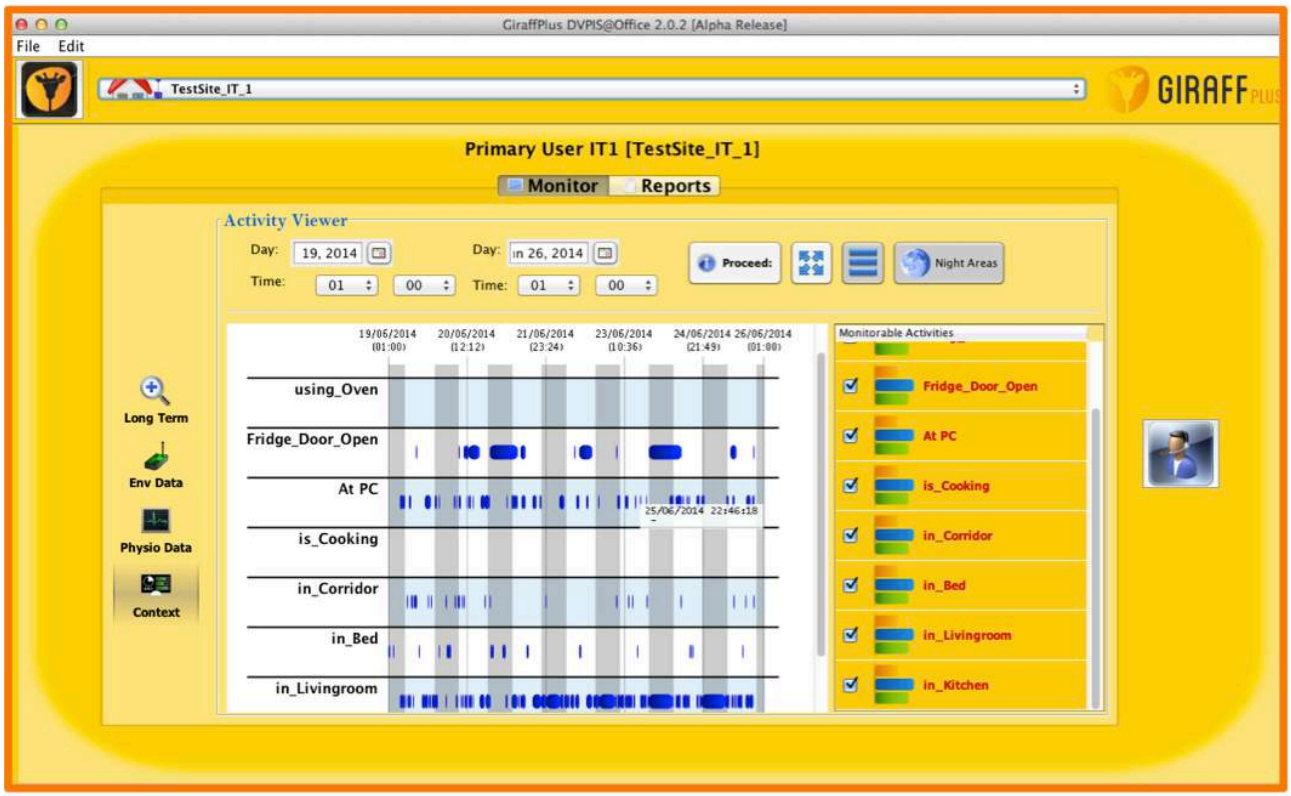


Figure 4 - DVPIS@Office1.0 - Context Recognition Visualization Panel

This view provides an abstraction of low-level data that translates the information coming from the sensors into high-level activities the older person is performing in the house.

All these panels and views allow secondary users to visualise the result of temporal queries that extend on data of the day, of last month or of a personalized time interval. The context recognition ensurs a long-term analysis of gathered data which differs from pure sensor reading representation. For example the module allows the synthesis of a new activity by monitoring the temporal combination of more than one sensor value.

The DVPIS1.0 has been indeed deployed into test sites at M18 and is used since the second year of the project to monitor real houses. This allowed us to get feedback from secondary user of test sites on the visualization services provided.

3.2 From user feedback to new system requirements

In addition to the secondary users of test sites who provided their feedback in terms of reports on problems encountered and suggestions for improvements or additional services, the DVPIS@Office1.0 has also been tested during a specific evaluation session in Rome organized in collaboration between CNR-ISTC and ASL-Roma A. The evaluation aimed to assess the current version of the system and specifically the Data Visualization, Personalization and Interaction Service (DVPIS) fostering a participatory design approach for the system development. The specific results of this evaluation session are described in D6.2 Intermediate Evaluation Report. In this section we describe how the feedback obtained by the secondary users has been translated into technical requirements for the system modification with specific reference to the DVPIS module. To this purpose we carefully analysed the results of the focus groups, and reports from users of test sites and produced a list of detailed user requirements. It is worth saying that these new requirements are strictly related to the module produced by the WP4 work package while other improvements have been done that involve different work packages.

For each requirement the following information is provided:

Serial/Ref: an identifier of the User Requirement

Capability Descriptor: a brief textual description of the User Requirement

Requirement Statement: a more detailed description of the User Requirement

Justification References: A short reference to the motivations for the User Requirement and specifically the source that inspired it.

In particular, the origin of each UR is specified through the specific source (Focus Groups, test sites) from which the UR has been addressed.

Validation criteria: a statement suggesting how the User Requirement could be checked

Priority: the level of importance of the User Requirement in the range of **K**ey, **D**esirable, **O**ptional

Serial/ Ref	Capability Descriptor	Requirement Statement	Justification References	Validation Criteria	Priority
1.1.1 1. Visualization					
1.1.2 1.a. Direct access to relevant information					
1.a.1	*Immediate contact with information*	*GiraffPlus shall be able to provide an immediate and easy to access visualization of relevant information*	*Focus Group*	*GiraffPlus provides the capability to immediately access the information considered as more relevant*	***K***
1.a.2	*Summary information in the first GiraffPlus page*	*GiraffPlus shall be able to visualize a summary page where the relevant information is provided*	*Focus group*	*GiraffPlus starts with a page that summarizes the relevant information and allows for more detailed views in case it is needed*	***D***
1.a.3	*Immediate access to long term over a period (daily, weekly report)*	*GiraffPlus shall allow to access easily long term data*	*Focus group* *Test Sites*	*GiraffPlus provides the capability to ask for periodic report that show long term data*	***K***
1.b. Improvement of existing visualization features					

1.b.1	*Distinction between real time and long term data*	*GiraffPlus shall make the distinction between real time and long term visualization more evident*	*Focus group* *Test Sites*	*GiraffPlus provides alternative and distinct views for real time and long term data*	***D***
1.b.2	*Icon to recognize alarms and dangerous situations*	*GiraffPlus should emphasize alarms or dangerous situation through an easy visible icon*	*Focus group*	*GiraffPlus visualizes alarms and dangerous situation through recognizable icons in the home panel visible at a first glance.*	***K***
1.b.3	*Names and icons on the list of assisted person*	*GiraffPlus should add a picture or a name of the person in the list of assisted persons*	*Focus group*	*Giraffplus visualizes the list of assisted persons of a given primary user adding a picture and/or the name of the persons instead of the identification number*	***D***
1.c. Additional features of the DVPIS					
1.c.1	*Facilitating contact among the persons belonging to the network of caregivers*	*GiraffPlus shall allow messaging among people that assist a given home and from each of these people and the user at home*	*Focus Group* *Test sites*	*GiraffPlus provides an additional environment where persons involved in the care of the older adult can communicate, leave message and discuss about both the status of the persons and the management of alarms*	***K***

1.c.2	*Shared view of data between PU and SU*	*GiraffPlus shall provide a share environment from which primary and secondary users can discuss about personal data*	*Focus Group*	*GiraffPlus provides a means to show to primary users his/her personal psychological data so as to be able to discuss about the health condition with their secondary users*	***K***
1.c.3	*Alarms in real time*	*GiraffPlus shall be able to manage alarms in real time informing secondary users*	*Focus Group* *Test sites*	*GiraffPlus provides a real time alerting system that allows a secondary user to receive an alarm message in case of environmental/physiological alarm occurred in the home of the old person*	***K***
1.c.4	*Data accessible through smartphones*	*GiraffPlus shall be able to provide a secondary users to access data through a device alternative to the PC (e.g., a smartphone)*	*Focus Group* *Test sites*	*GiraffPlus provides a simplified view of the data coming from the older user's house that is accessible through a mobile device (tablet, smartphone)*	***D***
1.c.5	*Indication of presence or absence of the person in the house*	*GiraffPlus shall be able to show to secondary users the presence/absence of the assisted person in the house*	*Focus Group* *Test sites*	*GiraffPlus provides a message on the DVPIS that gives an indication of the presence/absence of the assisted person in the house*	***D***
1.d. Personalization					

1.d.1	*Personalized report should be provided*	*GiraffPlus shall be able to provide customizable report according to the needs of the users*	*Focus Group* *Test sites*	*GiraffPlus provides the capability to customize the report according to the specific needs of users*	***K***
1.d.2	*Alert of changes of habit*	*GiraffPlus shall provide warnings to secondary users in response to detection of changes in elderly person's habit in ADL/IADL inside the house*	*Focus Group* *Test sites*	*GiraffPlus provides the capability to inform secondary users about changes of habits of the assisted older person, e.g., the elderly person wakes up too late with respect to usual standard and/or he/she spends more time than usual in personal cleaning*	***D***
1.d.3	*Personalize the type of information sent to different secondary users*	*GiraffPlus shall be able to differentiate the type of information visible to secondary users according to different roles*	*Focus Group* *Test sites*	*GiraffPlus provides the capability of selecting the type of information visible by different typology of secondary users*	***D***

Table 1 - New user requirements

3.3 Designing a new version of the DVPIS@Office

The requirements listed in the table reported in the previous section inspired a new design of the DVPIS@Office that led to the current layout of the new release. In the following subsection the new version of the DVPIS@Office is briefly summarised together with the link to the specific user requirements.

3.3.1 New Home Panel

The feedback obtained during the evaluation session has been extremely useful for re-designing the module according to concrete requests from representative secondary users and for solving

many problems that were spotted. The first change of the DVPIS@Office2.0 is related to the Home Panel. Figure 4 shows a new layout of the home page of the DVPIS@Office implemented following the Improvement suggestions reported above.

Figure 5 - New layout of the Home Page of the DVPIS@Office

Specifically the list of followed user contains now brief and immediate information on the status of the assisted person in terms of three main indicators: **Alarms**, **Physiological** and **Social Activities**. For each of this dimension an immediate feedback is given with a judgment on the level of each indicator: *green = good*; *yellow = warning*; *red = risk*. In this way a secondary user who is following several assisted persons can easily judge if he/she needs to urgently intervene on some specific situations and in general he/she can modulate and prioritize the visit to the different patients thanks to an immediate feedback without the need to entering into the details of each home.

For each primary user the status of the house is provided and the possibility to both call the person trough the telepresence robot and visit the details of the house is still provided.

The new version of the home panel responds to user requirements:

- *1.a.1 Immediate contact with information;*
- *1.a.2 Summary information in the first GiraffPlus page;*
- *1.b.2 Icons to recognize alarms and dangerous situations;*

This version will be now evaluated in the test sites where the DVPIS2.0 will be installed.

3.3.2 Refactoring of panels

Other requirements coming from the DVPIS@Office evaluation have been translated in a refactoring of existing panels and addition of new features and views.

More specifically, in response to requirement *1.b.1 Distinction between real time and long term monitoring*, that was mainly due to a confusion created by the timeline representation of information coming from sensors, a new alternative view of the real time data has been implemented.

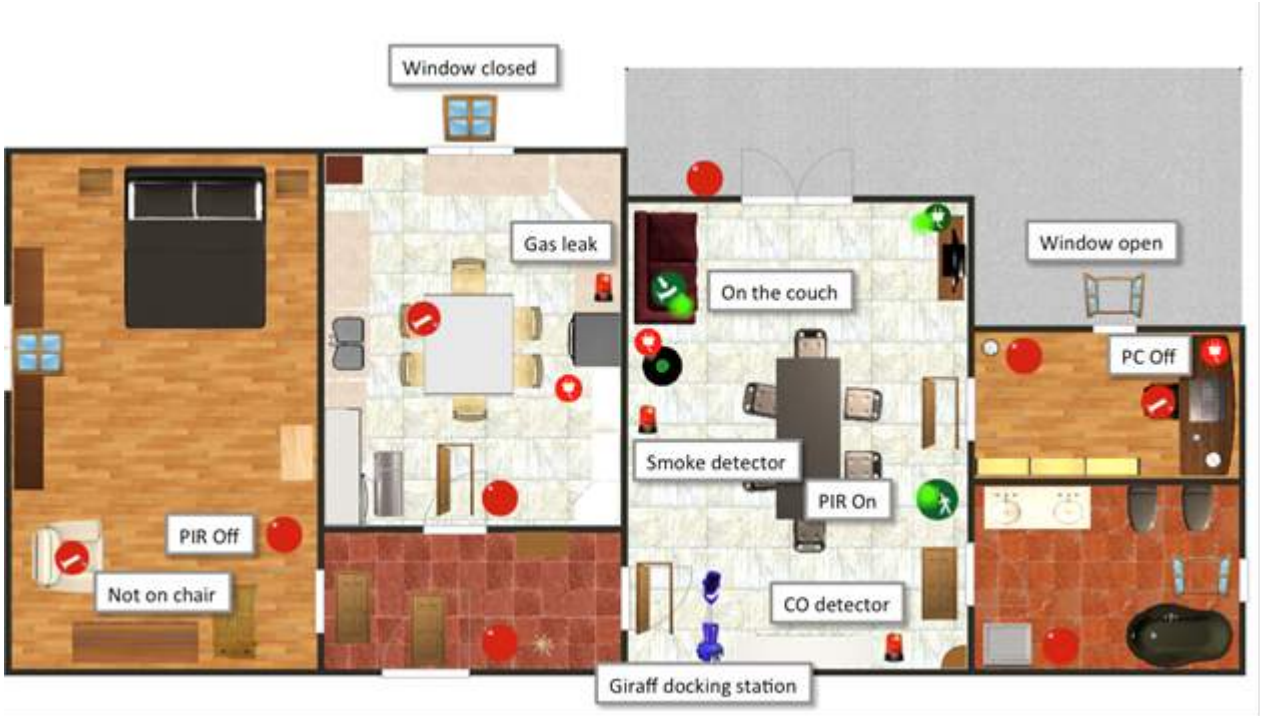


Figure 6 - New visualization of real time data based on a topological map

Figure 6 shows the alternative panel that has been added in the DVPIS@Office2.0 release. It shows a sketched map of the primary user's house and icons of sensors available in the house. Each type of sensor is represented by a different icon that can be red or green (e.g., an icon indicating a PIR is green if the sensors is detecting movements, is red otherwise; an icon indicating pressure on a chair/bed is green if someone is sitting on a chair/bed, it is red otherwise; an icon indicating an electric device usage sensor is green if the device is on, red otherwise and so on). Other icons represent the status of window or doors that can be open or closed (see Figure 6). This new vision of the data coming from the sensors provides a more informative and easy access to the situation of the house in real time and can be inspected in alternative to the previous timeline view. In response to requirement *1.a.3 Immediate access to long term over a period (daily, weekly report)*, the secondary user can switch to the Long term panel and inspect data over long period of time

through both the long term data panel and the report panel. More specifically the long-term panel allows inspecting the data coming from environmental sensors (Figure 7), the physiological sensors (Figure 8), the Context Recognition module to see the activities (Figure 9) and the report service (Figure 10). All these panels can be inspected on a daily, monthly or personalized time interval base.

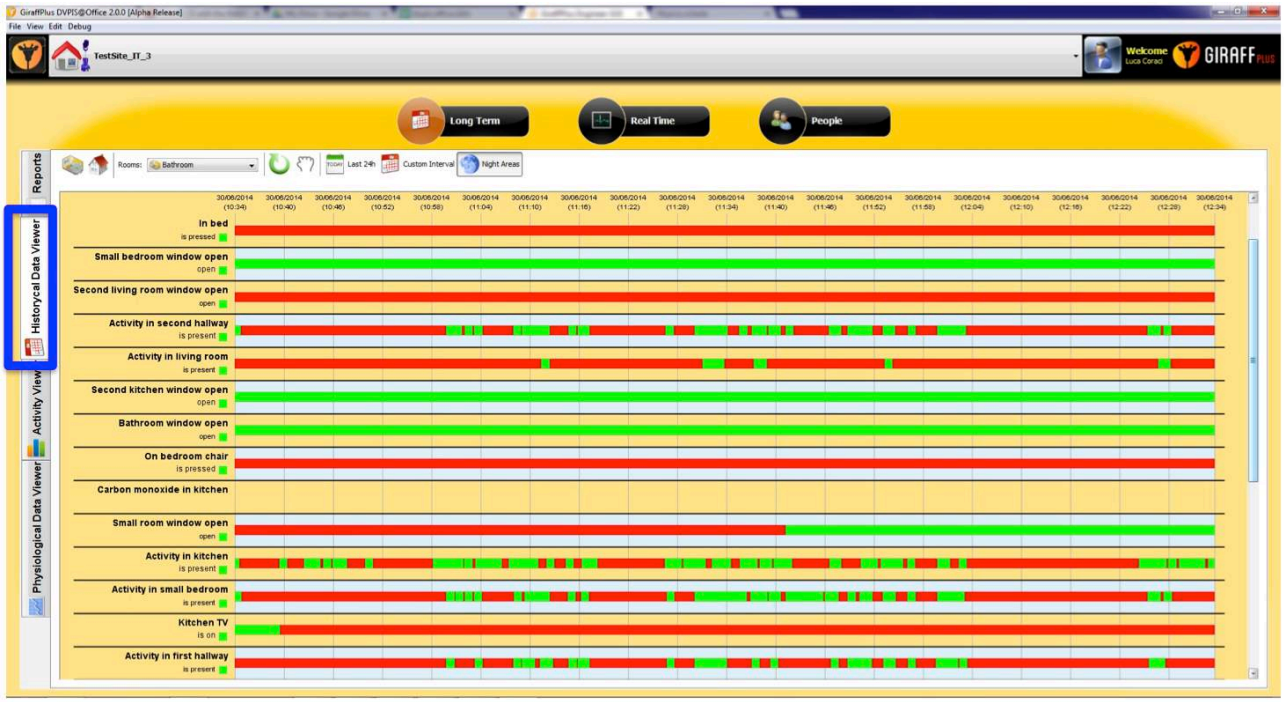


Figure 7 - Long-term panels showing environmental data

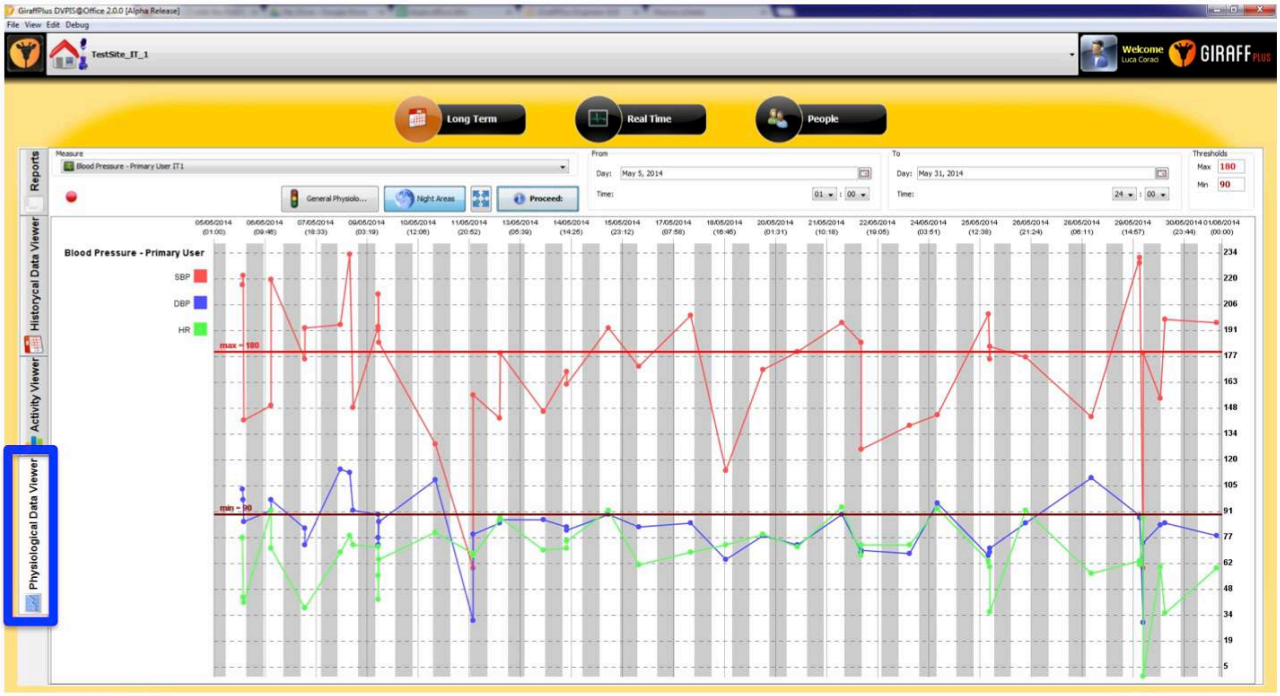


Figure 8 - Long-term panels showing Physiological data

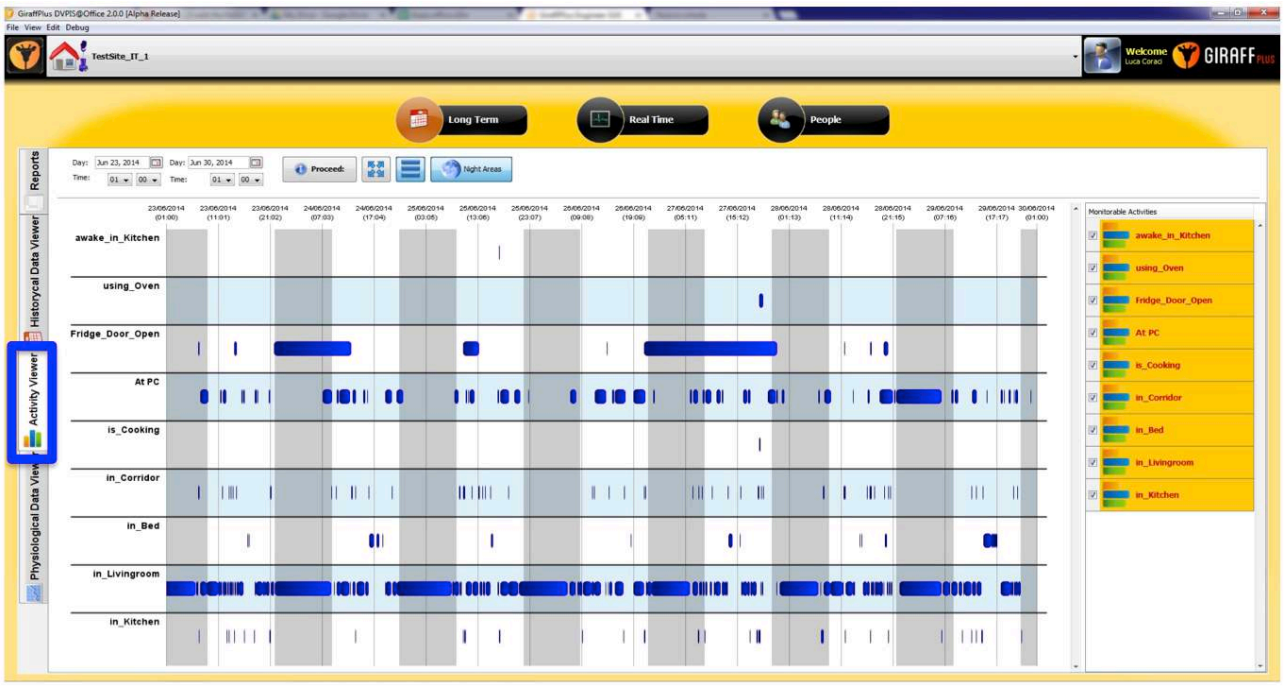


Figure 9 - Long-term panel showing activities recognized by the context recognition module

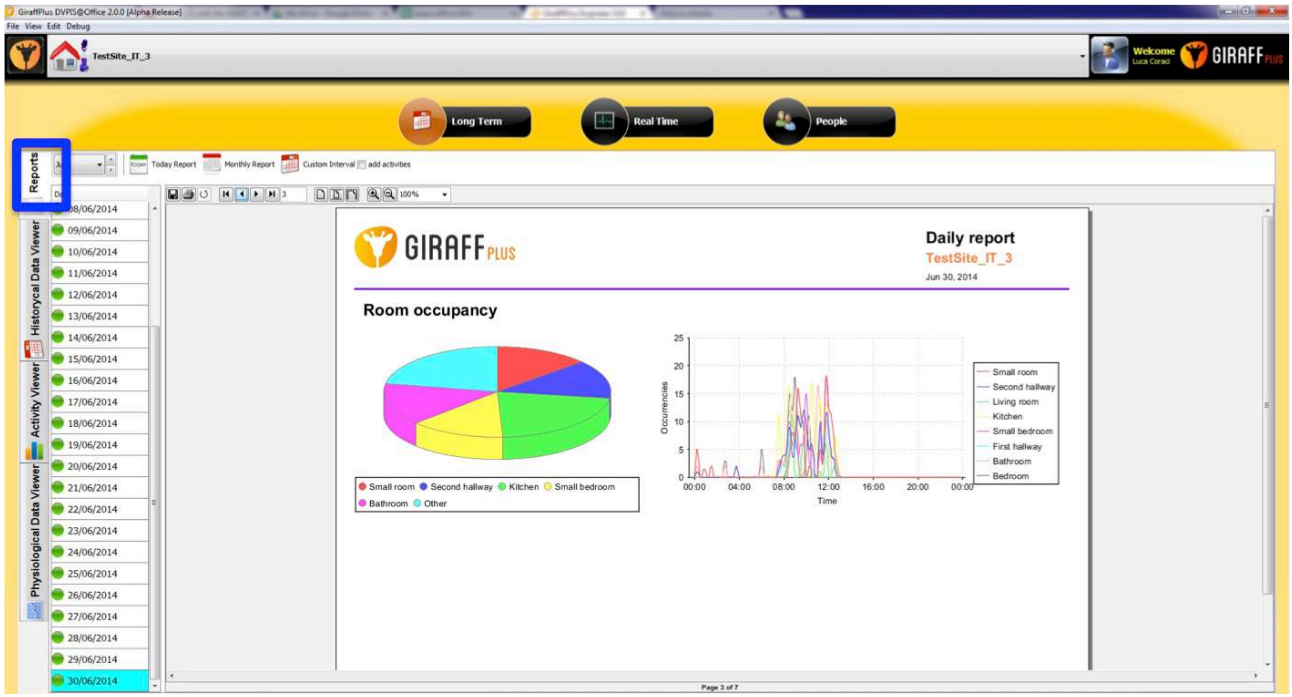


Figure 10 - Long-term panel showing the reports

3.3.3 Additional features

Another relevant improvement in the current version of the DVPIS@Office is the implementation of a new panel dedicated to foster the discussion among the network of persons related to a primary user. Specifically, to respond to requirement 1.c.1 *Facilitating contact among the persons belonging to the network of caregivers*, a new environment has been added where the different actors involved in the care of a primary user can exchange information and opinions so as to maximize the overall care for the old person at home.

More specifically there are two communication channels implemented in the new version of the DVPIS@Office:

1. the communication between each secondary user and the primary user, and
2. the communication among the group of secondary users following the same old person.

Figure 11 shows the first communication channel between the secondary user and the old person at home. Through this environment it is possible to send messages, questions and/or set reminders to the primary users at home that will be delivered through the Giraff telepresence robot as we will show in the next section.

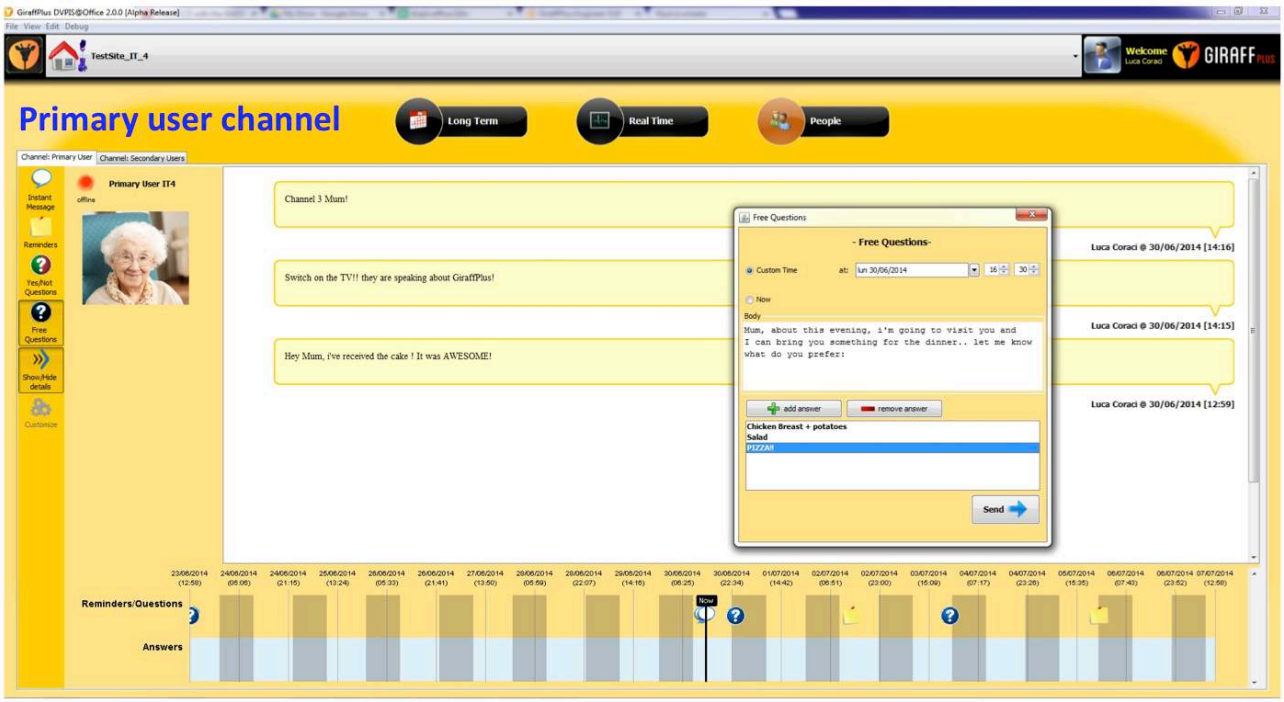


Figure 11 - New Panel to create a dialog between the secondary and primary user

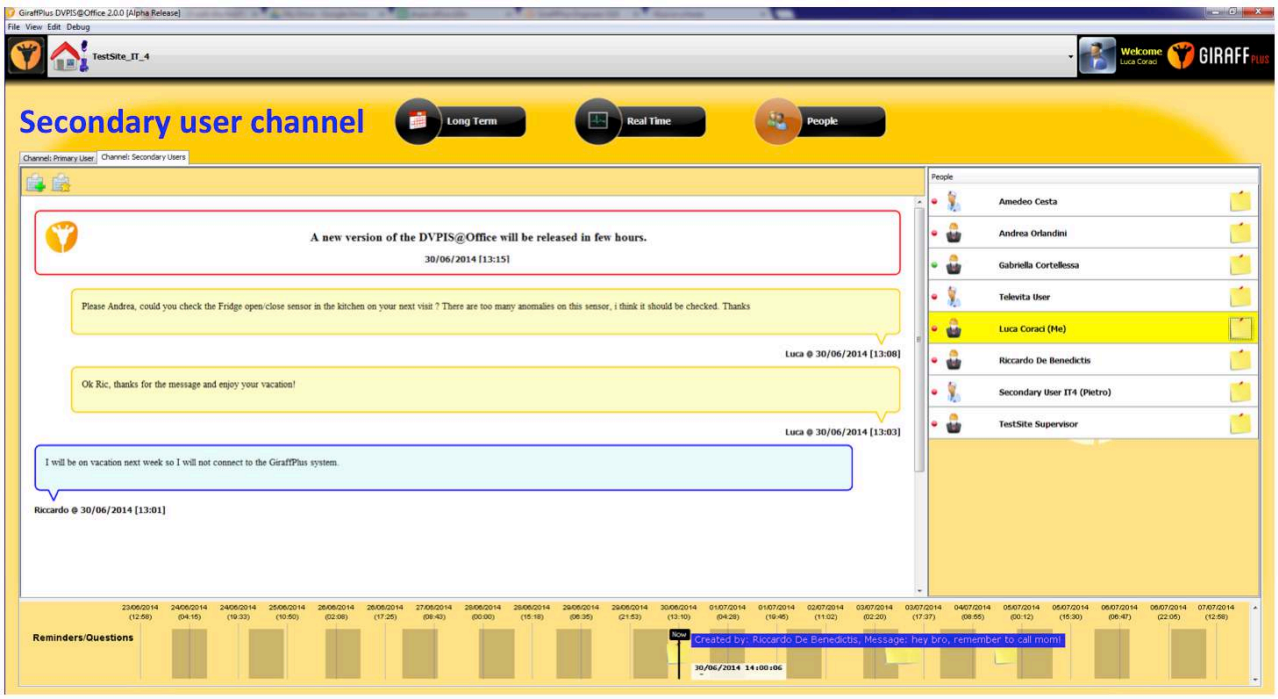


Figure 12 - Panel to create a dialog among secondary users belonging to the network of persons of a primary user

Figure 12 shows the second environment where the network of secondary users can share opinions and comments. Specifically, this environment represents a dialogue space to allow the social networking of people who assist the same primary user.

Through this environment it is possible to exchange messages among a group of people or in a peer-to-peer modality, send reminders and messages. It is worth underscoring how the Primary User Channel together with the @Home environment provide a completely new communication channel inside the GiraffPlus system that allows additional communication capability with respect to the Giraff robot call. In this way we can either create and deliver sms-equivalent, simple remind, etc.

3.3.4 Status of new user requirements

In addition to the user requirements managed and dealt with in the previous section, other user requirements and their respective solutions are described in the next sections. More specifically Requirement *1.c.2 Shared view of data between PU and SU* is dealt in Section 3, while Requirements *1.c.3 Alarms in real time*; *1.d.1 Personalised Report should be provided*; *1.d.3 Personalise the type of information sent to different secondary users* are dealt in Section 6.

To summarize, Figure 12 shows the status of the implementation of the new user requirements with this release of DVPIS2.0[1].

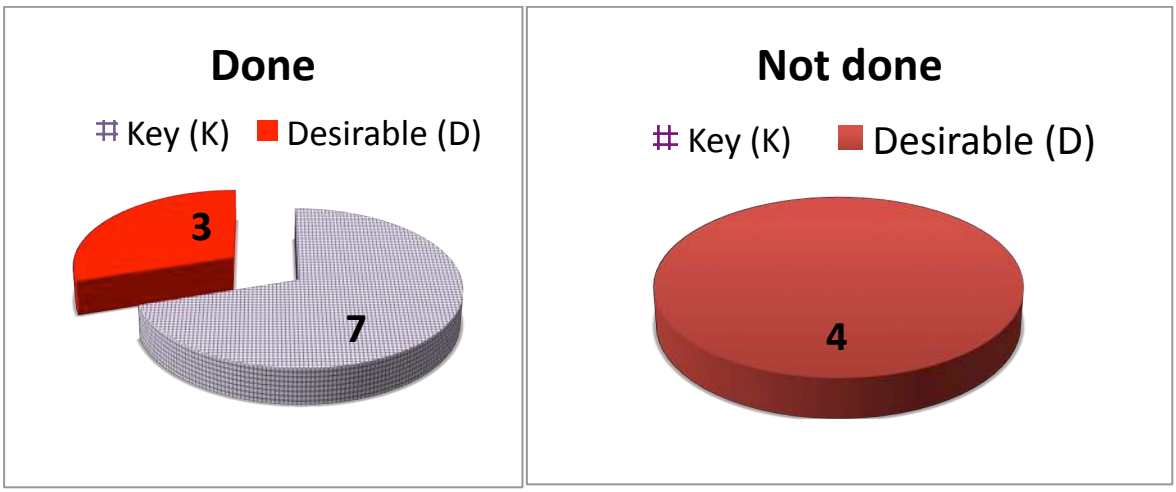


Figure 13 - Summary of user requirements dealt with the DVPIS2.0

[1] The optional category is not reported in Figure 13 since there where no optional requirements

It is possible to see that only 4 requirements with Desirable priority have not been included in this release, which are: Requirement *1.b.3 Names and icons in the list of assisted person*; *1.c.4 Data accessible through smartphone*; *1.c.5 Indication of the presence/absence of the person in the house*; *1.d.2 Alert of changes of habit*. These requirements will be considered for the final release of the software.

3.4 Implementation Details of new features

In this section we provide some additional information on the technical implementation of some of the new features implemented with this release. More specifically we provide some implementation details of the new environments available and also describe some additional enhancements we made that do not directly derive from the user feedback but still represent improvements of the software.

3.4.1 Real time map: Implementation notes

As mentioned above, the aim of this view is to give to the user a real time representation of the current situation on the selected home. The choice has been to use a realistic layout of the house as the background of an active view upon the real time data flow. The main features we decided to provide to the Secondary User can be summarized as follows:

1. Spatial view of sensors' positions in the house
2. Real time reaction of displayed sensors according to real time data
3. Intuitive and immediate sensor type identification
4. Possibility to reflect any real sensor position/type change into the House View
5. Internationalization of language

The design and implementation of these five points entailed several interactions with different modules such as the Long Term Storage (LTS) and the Middleware (MW) services. A procedure has been defined to assure the best setup for this component.

This real time map view is mainly composed by a main static component, which is obviously the layout of the house, and several distinct dynamic components, which are specifically:

- Sensor Icons
- Room labels

These two objects are considered dynamic due to their variable position in the real houses that must be reflected in this view. In addition, the main panel is engineered into 3 levels, each of which representing a distinct aspect. Figure 14 describes this concept:

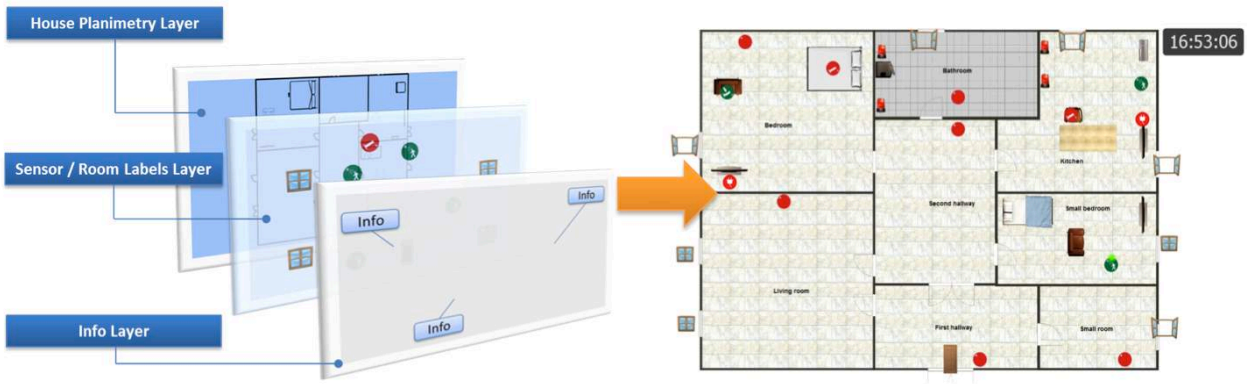


Figure 14 - Three layers to implement the components of the real time map

- **House Layout Layer**: aims at rendering the house layout. This is the only static sub component. It is only possible to replace the house layout file (image) at runtime in any moment by editing the house definition through the Engineer GUI (see later in the document). Nevertheless, the layout image will fill all the space the container will offer to this layer. Any resize or movement of the image is currently not allowed.
- **Sensor/Room Labels Layer**: aims at showing the real-time sensors data (when they are triggered by the middleware) by representing proper icons placed where the real sensors have been deployed into the house. Icons reflect the type of the represented sensor as well as the current data value or status of the sensor. Please notice that only environmental sensors are displayed by this View.
- **Info Layer**: aims at providing additional information about the house, sensors or any transversal useful info.

Position representation

The position on the map of each item that populates the second layer has an important meaning in term of visual impact and consistency with the real house and moreover also a semantic value in some case (e.g., room labels). Another feature is related to the capability of this view to be resizable in order to keep the distances amongst sensors, always at the same proportional quotes, according to the current size of the main container frame. The solution has been to store into the LTS a proportional values instead of fixed coordinates, which follows the following simple formula e.g. for the x coordinate:

$$x = \frac{current_x \cdot 100}{width}$$

For instance, this means that a sensor placed exactly at the center of the screen will have a value of 0.5 for its *X* and *Y* value. When the main container component is resized, the real *x* value on the screen will be recalculated considering the base stored value. Since each sensor can have its own

location (described by the LocationEntity entity) we decided to store the position values on those entities. Therefore in order to have each sensor with its own position it is important to create different locations for any sensor or it will be impossible to have two sensors in two different positions.

The room labels deserve a specific note: when running the DVPIS@Office, if the application notice no room labels in a certain site it will automatically create room labels by reading the room list and creating an item for each room. As anticipated above, all items of the second layer need to be set on the correct position to have a proper value that allows a "good visual representation". Hence each item has the capability to be manually dragged and dropped anytime. However, only GP-engineer users are allowed to use this feature (see Appendix A for further details). When an item does not have a pre-set position it will be displayed in the left-top corner indicating to the user that the position is missing. From the programing point of view a custom extension of the java JLabel class (javax.swing.JLabel) has been used as main container of all items of the second layer. JLabel naturally handles icons, mouse events, text displaying, transparencies and many other useful aspects.

Icons. At the moment, only sensor labels are showing icons. This icon shows at the same time:

- **Sensor Type**: sensors can be of different types since they can get different measures such as pressure, movements etc. Hence, the icon must display without ambiguity the type of the sensor that it represents.
- **Current value**: all deployed sensors produce Boolean values (all but the PIR and Alarm sensors, which will be treated separately). The icons used are different if the current value is true or false. It is important to note that the current value mostly means the last value.

Table 2 shows the current icon set used in the last DVPIS@Office version:

Value/Sensor Type	**Device**	**Pressure**	**Door Contact**	**Fridge**	**Wardrobe**	**Window**
False						
True						

Table 2 - Icons used to display sensors type and status in the real time map

Alarms and PIR sensor have to be treated separately because these sensors only produce true values. The PIR sensor follow this behaviour: they produce a true value as soon as they detect a movement from a no-movement situation, but if in the range of the sensor the movements keep to be detected the sensor only produce more value every a certain amount of seconds, typically 10-20. For this reason when a PIR sensor produce a new value, the sensor's icon will switch to an icon that indicates the presence of some movement around the sensor and this icon will persist for a fixed amount of time of 30 seconds. In addition, to show the temporal distance from the time when the data has been produced and the actual time, a blinking green light has been added close to the sensor's icon. The blinking rate will decrease in time, from a very fast blinking (when the value has been produced) to a very slow blinking next to the end of the 30 second time window of this event. If during this time interval another new PIR value occurs the thread that handles the blinking function is reset and the blinking is restarted with full speed. The aim of this behaviour is to indicate to the user "how much movement" is being registered by the frequency of the blinking light. Figure 15 and Figure 16 clarify this behaviour.

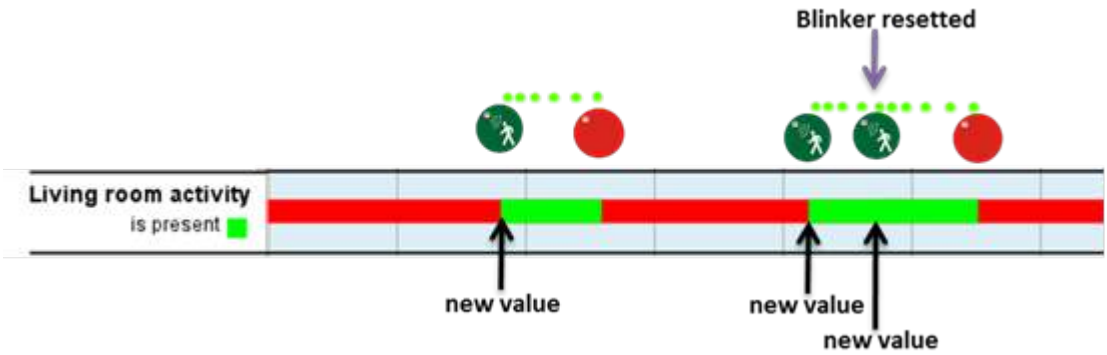


Figure 15 - Graphical representation of the threads handling the blinking of the PIR sensors

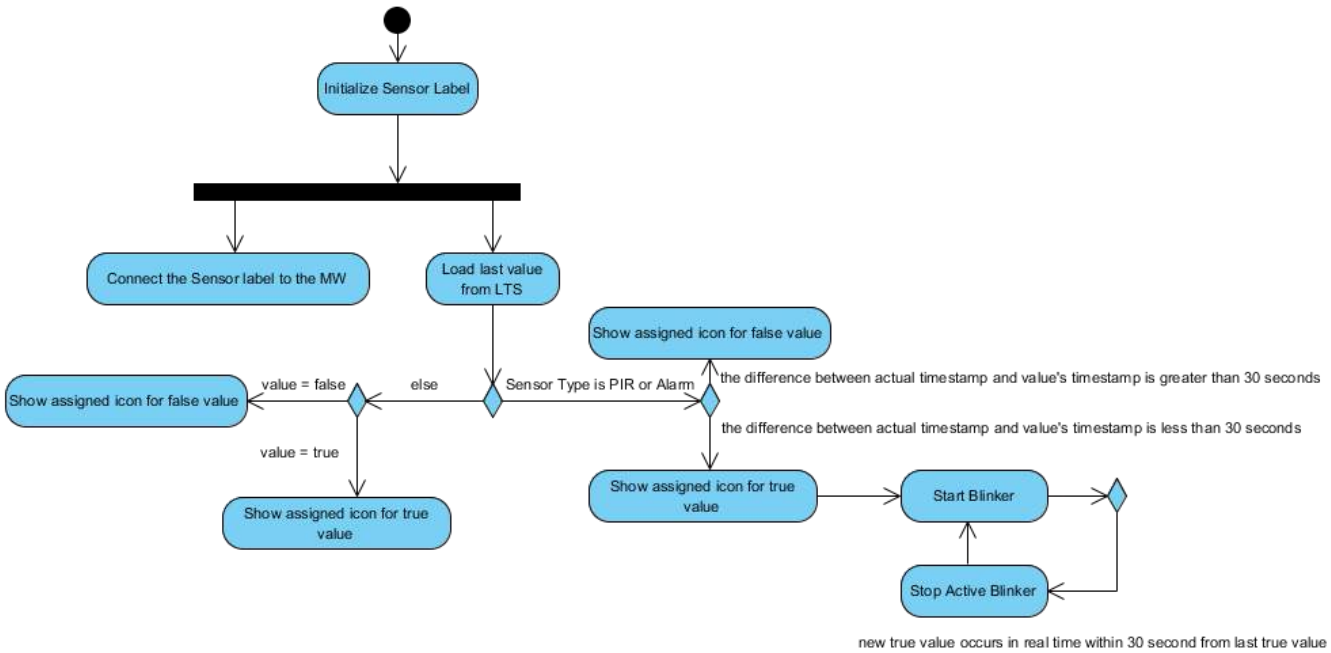


Figure 16 - Activity flow diagram of the threads handling the blinking of the PIR sensors

As shown in the activity diagram above, similarly to the PIR sensor, the alarm sensor's output is a single true value when the alarm itself is triggered. The current behaviour adopted is to draw proper icons around the alarm sensor for the time of 30 seconds starting from the time when the alarm has been sent. Currently alarms, in addition to the notification sent through emails to secondary users, are displayed in the DVPIS@Office using the icons shown in the table below.

CO2	Smoke	Fall	Flood
			

Table 3 - Icons used to display alarms in the DVPIS@Office

The third and last layer at the moment is only exploited by an active clock showing the current time. The clock component can be moved, just dragging it, at any time to to avoid that is covers any interesting part of the underneath map.

The plan is to use this layer to add more info components to enhance both quantity and quality of information and details about the current situation on the house, especially when an alarm is launched.

3.4.2 The *People* panel: Implementation Notes

As mentioned above the people panel is the frontend of a set of services that allow closing a communication loop around the user and putting in touch the @Office with the @Home. Before presenting the design and implementation details of this new environment we first clarify some main concepts which are the core of this component.

Network Concept. The initial design of this component was based on the idea of building a network with the house as center and users as connected nodes with the ability to freely communicate with each other in various forms. After some internal tests we noticed that the original concept needed some kind of enhancement or in some case, a sort of restriction of potentiality. Even if the underlying infrastructures that models all the necessary data remained more or less unchanged, we needed to develop additional layers to support the needs emerged from the internal tests.

The main issue coming from these internal tests resulted in a new organization of the environment that is deployed with this release of the software.

The final choice for this new environment has been to consider the house and the Primary User as the natural center of this kind of network. Figure 17 shows the evolution of the choice. The initial model entails that secondary users are simply connected to the Primary User (left part of the figure).

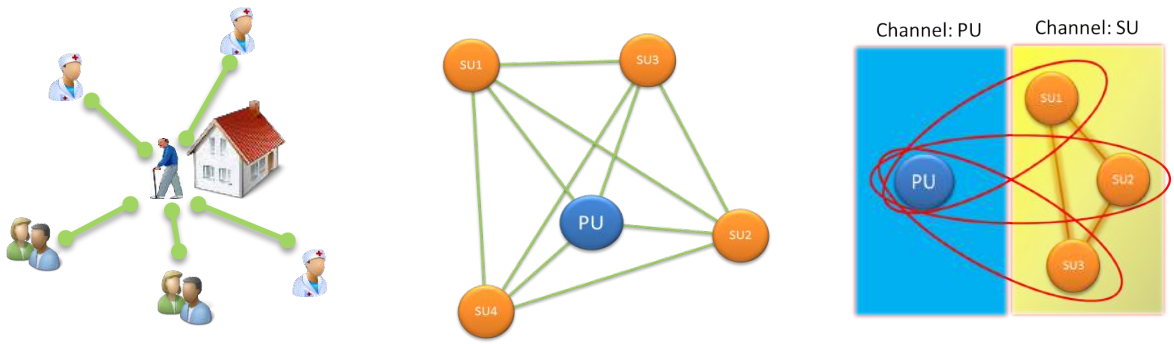


Figure 17 - Different Models for the communication among users in the People Component

This allowed us to establish a one-to-one communication channels between PU and SU. However we needed to enhance the model with one-to-one communications between pairs of SUs and finally also to add a channel where the SU could communicate amongst them. The evolved model, shown in the central part of Figure 17 resembles a “clique structure”. This model has also been implemented and tested in an internal prototype version. After these internal tests we finally opted for the actual model, that has 2 main channels of communication and that introduces a distinction between the **PU interaction sphere** and the **SU interaction sphere**. This final model is implemented by splitting the prototypal single People panel into 2 separate tabs, each one

representing one channel of communication. This distinction allowed us to provide to the user a clearer access to the set of features and functionalities of each channel removing some misleading components. The distinction of the channels relies on the purpose and the centric element of each channel.

Channel 1: Communication with Primary User. The first communication channel is completely dedicated to the social relationship between the Secondary user who is using the DVPIS@Office and the Primary User. This relation is intended to be private with respect to other secondary users so that the messages, as well as the reminder or questions exchanged by these two actors will remains private and hence invisible to other secondary users. We already cited the importance to avoid messages overflow or to be annoying with the primary user, so in this channel it is possible to monitor the quantity of total messages exchanged. The secondary user will only be able to read the messages he created by himself. Accordingly, the main purpose of this channel is to provide to secondary users a full set of social interaction services and at the same time to keep him well informed about the overall primary user social interaction (through the DVPIS@Office and DVPIS@Home). The set of services here deployed is the following:

- **Instant Message:** the secondary user can send a free text message that will be instantly sent through the middleware infrastructure and delivered to the Primary User by the DVPIS@Home. A short history of instant messages will be always visible on this channel.
- **Reminder:** the secondary user can setup a reminder with free text, which will be delivered at customized time. The delivery will be again enabled by the DVPIS@Home.
- **Questions:** the secondary user can setup a question that will be sent in the actual time or at a customized time. The user can choose between two kinds of questions:
    - **Yes/No:** the question will automatically contain two possible answers that will be *Yes* or *No*.
    - **Free answers:** the user can create his own set of possible answers to attach to the question. The question will be displayed by the DVPIS@Home with a sort of multi-choice dialog.

In addition, the secondary user will always be kept informed about the social message flow directed towards the Primary User. This will be done by populating a timeline with all necessary info. Further details are provided in the User Manual.

Channel 2: Communication amongst Secondary Users. This channel is dedicated to the secondary users who are following the same primary user. It is meant to provide a separate space to secondary users where they can freely speak about topics and problems related to the primary users. This area will also be the first way to directly open a communication space between formal caregivers and informal caregivers within the GiraffPlus project. Possible realistic use cases could be:

- *A relative of the Primary User (informal secondary user) is worried about some physiological measurement readable from the DVPIS@Office. Through this environment the relative could then express this concerns through this space where also formal caregivers could give a professional interpretation of the registered data. Moreover, other secondary users could join the conversation adding useful details or information to enhance the evaluation of the problem.*
- *People could start discussions about some system configuration changes that could be needed (for example sensor replacement or movement from a room to another, or request for a new activity to be monitored and so on).*
- *Organize and schedule routine calls with the Giraff Robot amongst secondary users.*
- *Since also the GiraffPlus-Engineer might be allowed to have access to this channel, some secondary user could here report any software or hardware defect of the system.*

A sort of asynchronous message exchanging service implements this communication. The secondary user can create these messages, so called **post**, anytime and they will be instantly published in a common area and visible to all users. Moreover the GiraffPlus-Engineer can also publish on this area and can also create a sort of Announces (**System Post**) that will have a different layout and anonymous sender.

The list of deployed services on this channel that any secondary user can use is:

- **Post:** the secondary user can send a free text message that will be instantly published on the post container area. Posts are always visible to all secondary users and a short history published post is always loaded when the people panel is loaded.
- **System Post:** the GiraffPlus-Engineer can send a post with a different layout. Useful to perform official communication to all secondary users through the DVPIS@Office.
- **Reminder:** user can setup a reminder with free text, which will be delivered with a customized time. The secondary user can create reminder directed to him or any other secondary user.
- **Private chat:** user can select another secondary user and take a private conversation with him/her. No one else will be able to read this conversation.
- **User online status:** user can monitor which other secondary user is currently online and in addition, can even see if such user is monitoring the same house he/she is monitoring.

Also on this channel the secondary users have a timeline with some information placed in time. The information displayed is all reminders that he created and all reminders set to him by other secondary users. The use case diagram of the current People panel is shown in Figure 18.

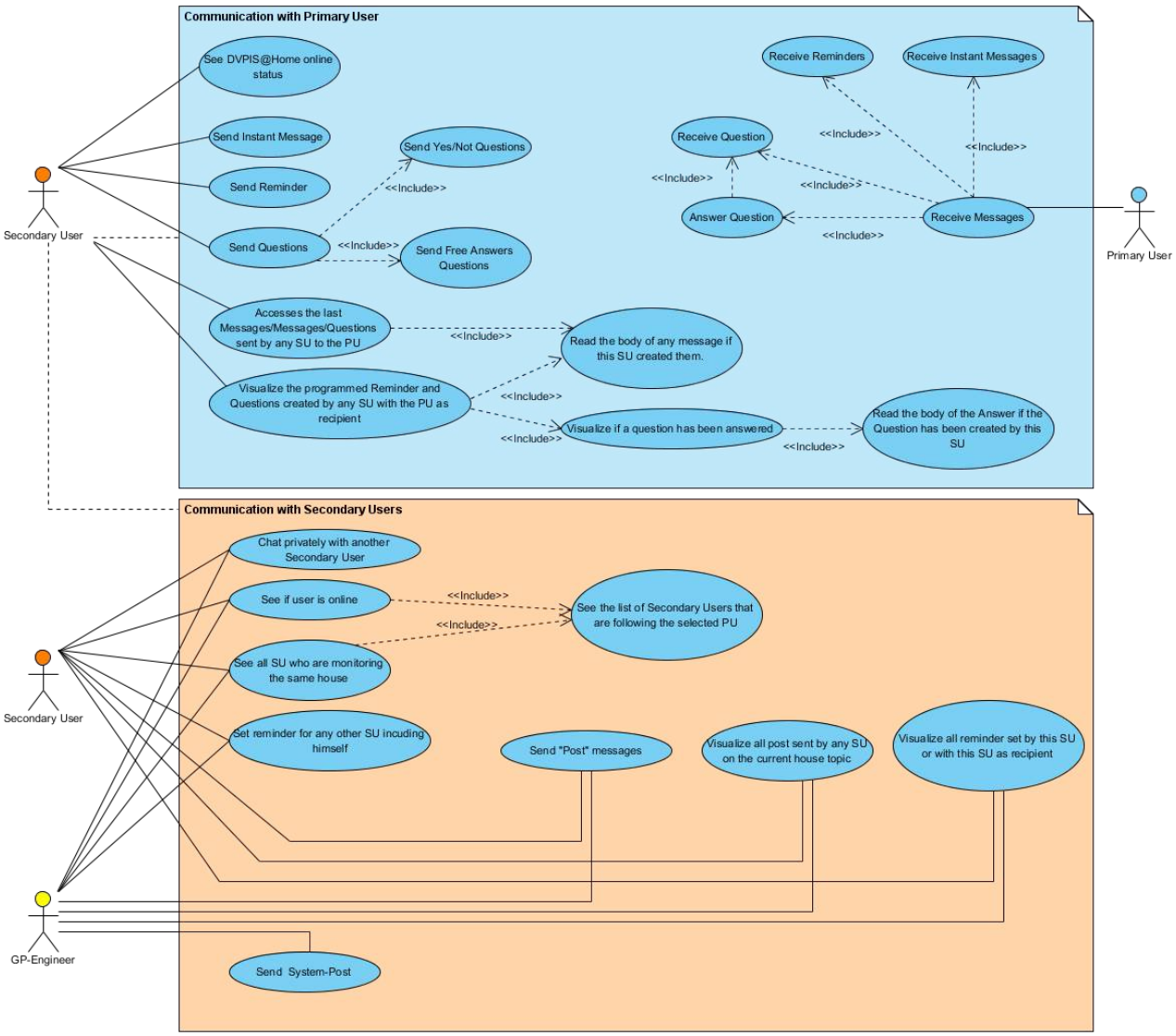


Figure 18 - Use case diagram of the People environment

Software Implementation and other modules interaction. The main software components implementing this new environment have been developed in a separate OSGi Bundle: **people.gui**. The module contains an internal message dispatcher with the purpose to spread the incoming messages to all linked internal subcomponents who are listinening for real time messaging. The message objects are wrapped into an internal hierarchy in order to properly display in specific graphical components the different kind of messages allowed by the model. Some singletone manager class take care of consinstency between different components and in some case act like an internal cache to prevent reconstrunction of entire panels. Finally the module has been integrated into the DVPIS@Office and linked to real message dispatchers offered as OSGi Services

by the PerS module. Historical data are instead retrived directly from the Caching API which is responsible to also fetch data from Database when necessary.

Mostly we have two main sources of information during the execution of the DVPIS@Office: real time data and historical data and this distinction is also heavily reflected into the People panel. The historical data are mostly used to initialize all the graphical components and the real time data are meant to keep all the panels up to date with respect to what is happening during the execution time. However on the people panel context, we need a slightly addition of information about the events. Specifically it is importat not only to know when a scheduled message (e.g. reminder) has been delivered, but also, when a scheduled message is supposed to be delivered and this information we need to retrieve as soon as this kind of message is being created. Therefore the timeline representation graphical component has been enhached with the possibility to have the view of a portion of the future and the PerS API have been enriched with methods that provide to the DVPIS@Office the information about future scheduled events. In this way the user can have both in Channel 1 and in Channel 2 the complete view about the quantity of the message. A different set of icons, custom tooltip and additional graphical elements (e.g. arrows) will provide info about the type of the messages (e.g. message type and possible connection between messages) . Also, a vertical line will mark the time-now highliting the future events from the past events. We also assume that all delivered events are the past events.

Figure 19 represents the People panel module's architecture and its interaction with the other modules that provides all the information needed.

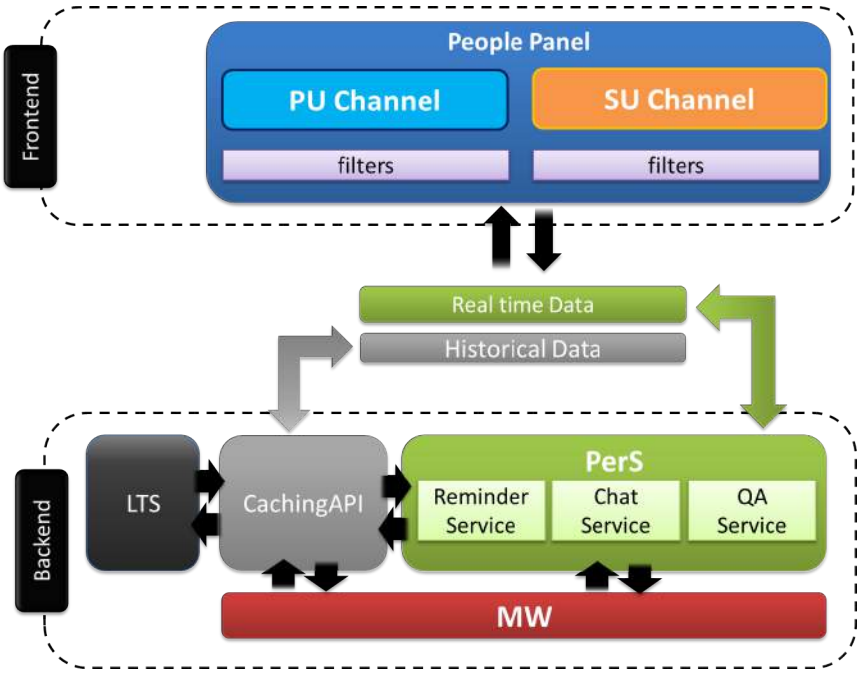


Figure 19 - People Panel modules interaction

All the real time information, marked with green colors is fetched from the PerS module, which is internally connected with the middleware for publish and retrieve real time data. The PerS module also exposes three different services; each one of them is responsible for an aspect of the People panel functionality set, in details:

- **Reminder Service:** provides functionalities to setup reminders. Notify when a new reminder is being delivered or created by any secondary user within the range of potential accessibility.
- **Chat Service:** provides functionalities to send instant messaging, both private chat messages and post messages.
- **QA Service:** provides functionalities to create and deliver questions to primary user. It also notify when the primary user has taken an answer of some question.

Regarding the historical data, as said above, the People Panel directly fetches required data from Caching API.

In the frontend, all retrieved data are filtered and then displayed.

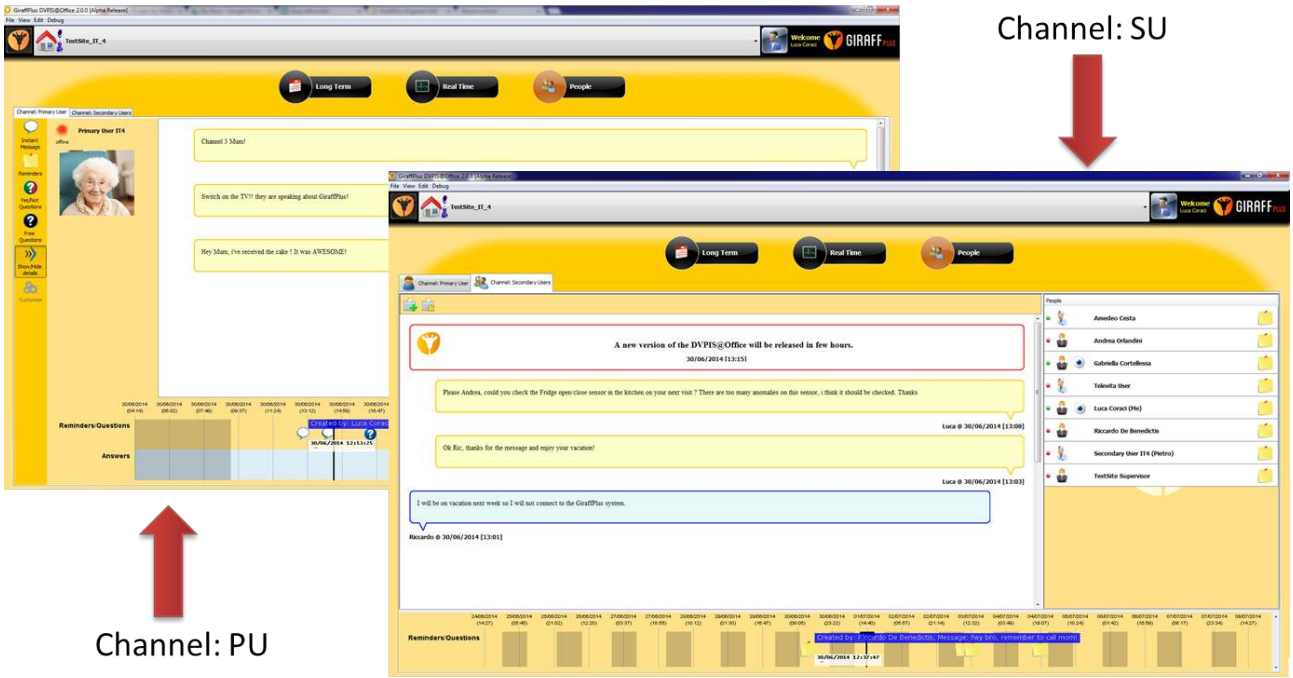


Figure 20 - Implementation of the People Panel in the current version of DVPIS@Office

The final result of the people panel currently deployed is shown in Figure 20.

3.4.3 Auto Update of the DVPIS@Office software

An additional improvement contained in this version of the software is the possibility to automatically update the DVPIS@Office once the tool is launched and new updates are available.

From this version onwards, the DVPIS@Office is equipped with an updater tool, existing as an OSGi bundle, called **giraffplus-updater**. This module will allow the software to self-update without any manual intervention. The updating system is designed to be robust and potentially extendible enabling the developers to be able to have complete control about deployed software instances. From the user perspective the updating system will be totally transparent and automatic. When the module detects an available update, the application alerts the user about it and the user can decide whether to proceed with the update or not. If the user decides to update the software, the update will be downloaded and the DVPIS@Office restarted with the new update active.

The update system functioning is based on the structure of the software release that basically relies on the folder where the modules executables reside (*./bundle* folder into the distribution pack) and a cache folder (*./felix/cache/runner* folder into the distribution pack) used by the software while is running. This separation is fundamental to allow the giraffplus-updater to fully operate when the DVPIS@Office is running. The strategy is rather simple: if a new update is

detected, the module will checkout the manifest file of that specific update. On this manifest file the list of modules to be updated is described that will replace the corresponding executable file in the bundle folder with the new ones downloaded from the DVPIS@Office update server (http://pst.istc.cnr.it/giraffplus). After the completion of this task the cache needs to be deleted in order to be refreshed with the new files at the next run of the DVPIS@Office. This operation is done via script (batch / bash) by the launcher of the DVPIS@Office. The script will also completely restart the DVPIS@Office that will run with the new update.

Version Management. On the server side, for each version after the release of the **giraffplus-updater** module, a correspondent folder exists named with the version number. On this folder two files are needed to make the updating system to work:

- ***nextversion* file:** this version contains info about which version is the next one. If this file exists and the field that declare the next version contains not null or wrong values the updating system will start the procedure to update the DVPIS@Office.
- ***manifest* file:** contains all info required to update the DVPIS@Office from the previous version to the current version where the manifest file lies.

It's important to note that the updating system will continue to update the system by iterating the check on *nextversion* file until a version with no next version declared is found. Therefore, if for instance the current version is the 2.1 and versions 2.2, 2.3 and 2.4 have been deployed, by updating the 2.1 the DVPIS@Office will be restarted with the 2.4 release.

The detailed workflow of this module is shown in the following activity-diagram:

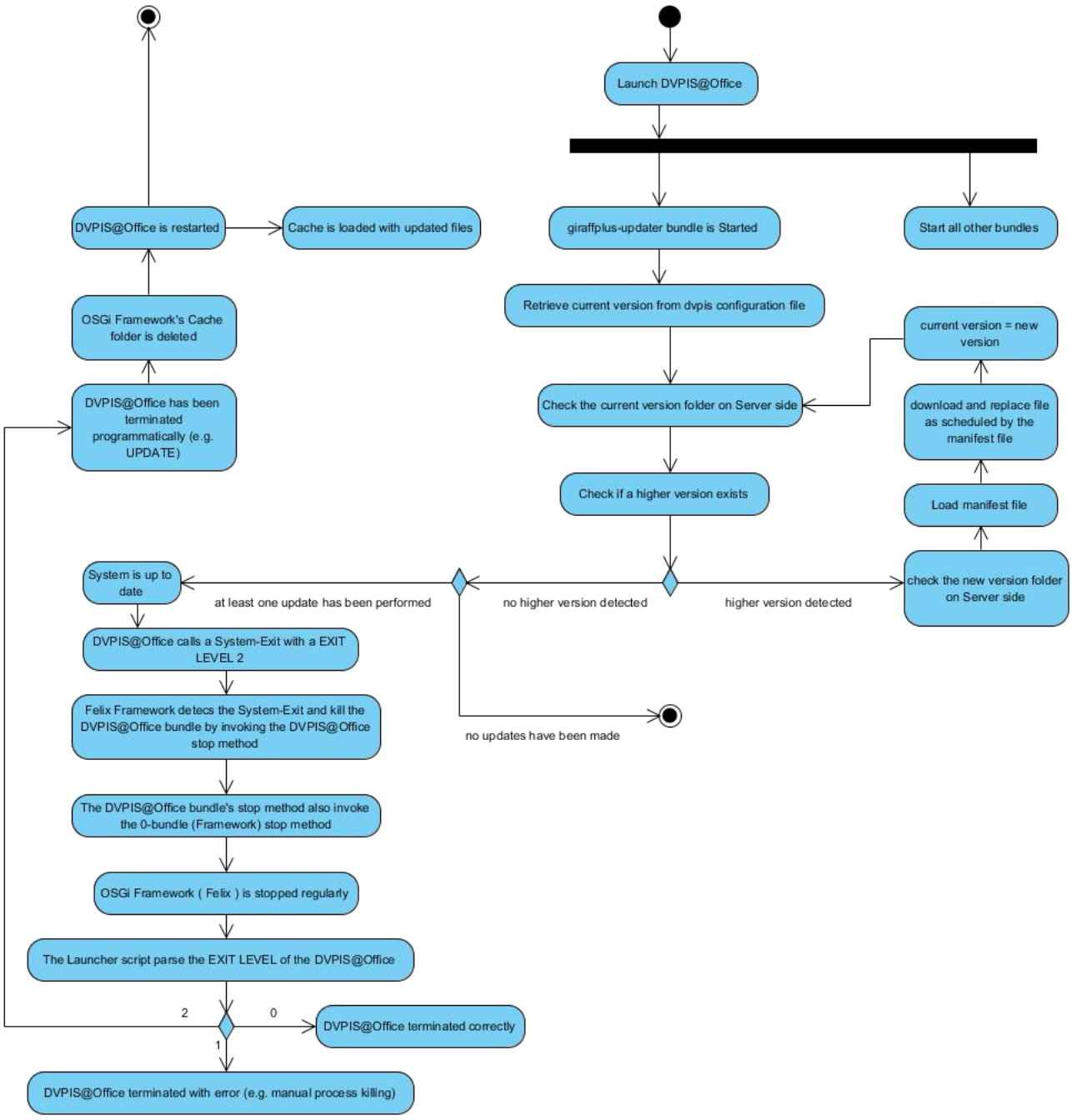


Figure 21 - Activity diagram of the auto-update feature

4 The DVPIS@Home: New Features

As already introduced in deliverable D4.1 we have designed an environment, called DVPIS@Home (also shortly referred to as @Home in the following), that runs on the telepresence robot and contributes an additional information channel between outside the home to inside and vice versa. In the Alpha version, we designed a few initial functionalities for pure demonstration purposes. We have now created a stable version of @Home that represents a consolidation of the initial ideas and is an additional GiraffPlus functionality that can be installed and used in the test sites.

The Beta version of @Home has the structure depicted in Figure 22 where we have identified an entry-screen and the possibility to access three different functionalities during its use:

- **Avatar**: this functionality preserves the "telepresence" service that the Giraff robot provides. The Giraff application has been indeed embedded within the @Home so as to maintain the possibility for secondary users to visit the older user's apartment through the telepresence robot.
- **Messages**: an environment has been designed to allow the primary user to receive messages from secondary users or reminders and suggestions (hance this functionality is directly connected with the capabilities introduced with the Primary User Channel previously described). Messages and reminders are provided in both textual and spoken form. Specifically we have developed a message listener that collects messages coming from the middleware and gathered them with a specific panel that "mimic" the functionality of a mail client or a messenger on a smartphone. In addition to adapting the font size to the user we have decided to integrate an off-the shelf text-to-speech translator to give the user the possibility of "reading aloud" the messages and listening them again and again (the use of the text-to-speech is intended as a facilitator given that sight problems are frequent in older adults).
- **Personal Data**: this panel allows to show personal data to the primary user (e.g., physiological measures), and to endow the system with a shared space between the primary user and the secondary users that could foster a discussion on the health status and habits of the old person. This implements a requirement emerged from several of the tests and allow to have the telepresence and some information content visualized.

Figure 22 - The functionalities of the DVPIS@Home

It is worth emphasizing that the current module has been produced according to the elicited user requirements (e.g., *1.c.2 Shared view of data between PU and SU*) and we plan now to start an evaluation based on use in some of the test sites and also in ad hoc evaluation sessions in the lab to elicit further users' suggestions.

4.1 @Home Software architecture

The @Home aims at enriching the features provided by the standard Giraff robot exposing some additional services to the primary user. To achieve this goal, different pieces of software had to be integrated in a unique working system.

On one hand there is the existing software from Giraff Technologies that runs on the robot, which is responsible for the standard robot behaviour and user interface. An "obvious" key requirement is that all these features had to be preserved, and @Home should not interfere with the existing behaviour.

Furthermore there are the services provided by the DVPIS@Office which is composed by several separated modules that are built upon the OSGI framework. One of these modules is the middleware, which among other things enables the communication between the different instances of the DVPIS@Office and the sensors.

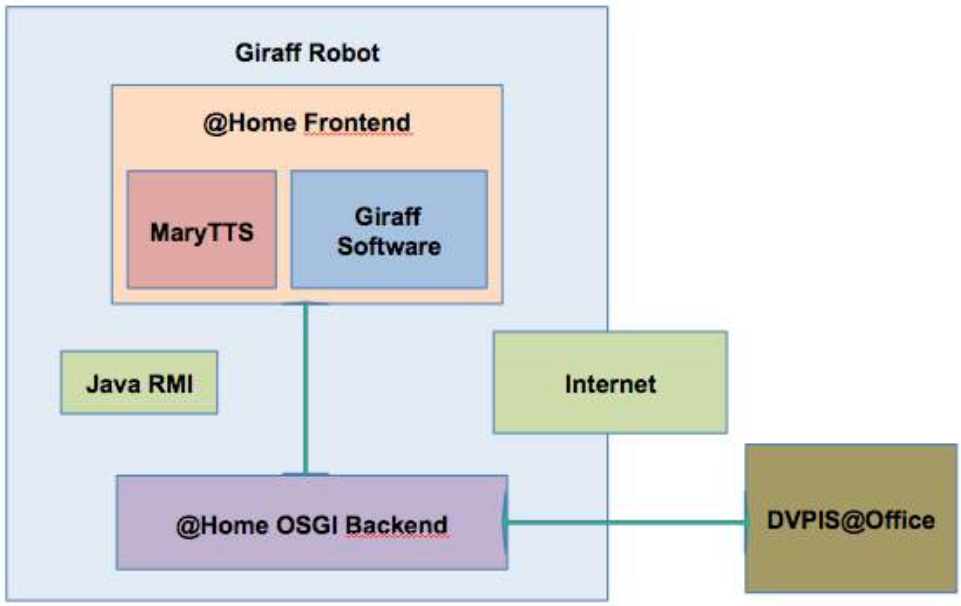


Figure 23 - The @Home software architecture

To make use of these services, the DVPIS@Home had to rely on the same framework. Unfortunately, the Giraff software is meant to be extended by third parties in a way that does not play well with complex frameworks like OSGi. Technically, the Giraff software can load additional libraries (jar files in Java parlance) on startup, run some code from these libraries and let them initialize the rest of the system. A software that runs in the OSGI framework requires a somewhat more complex startup: making the Giraff software initialize such a framework would be cumbersome and error-prone.

An alternative that has been investigated was packing the Giraff software in an appropriate module (a bundle in OSGI parlance), to let it run inside OSGI as whatever other module. The Giraff software, however, relies on some native libraries, spawns external processes, and in general expects to be run from the directory in which it was installed, and not from a module inside OSGI. Moreover, this approach would also have broken the auto update mechanism of the Giraff Software.

To overcome this limitation, the fundamental architecture of the @Home system has been subdivided into two components (see Figure 23):

- The *frontend* implemented as a regular java .jar file, that handles the user interface and extends the Giraff software;
- The *backend*, which is an instance of the OSGi framework that exports its services to the fronted.

The two components communicate through Java RMI (Remote Method Invocation): they live in two different instances of the Java Virtual Machine, and as such all the technical differences related to the different underlying frameworks are hidden each other.

4.2 @Home implementation

This subsection adds technical details describing the implementation choices done to obtain the current version of the @Home software Beta release.

4.2.1 The *frontend*

The frontend is responsible of handling the user interaction: it displays the user interface on the robot screen, takes care of speech synthesis, and displays user-relevant data; in simple words, it does everything the application is supposed to do except the low-level communication with the DVPIS@Office system, which is handled by the backend.

Figure 24 shows the different components of the @Home frontend.

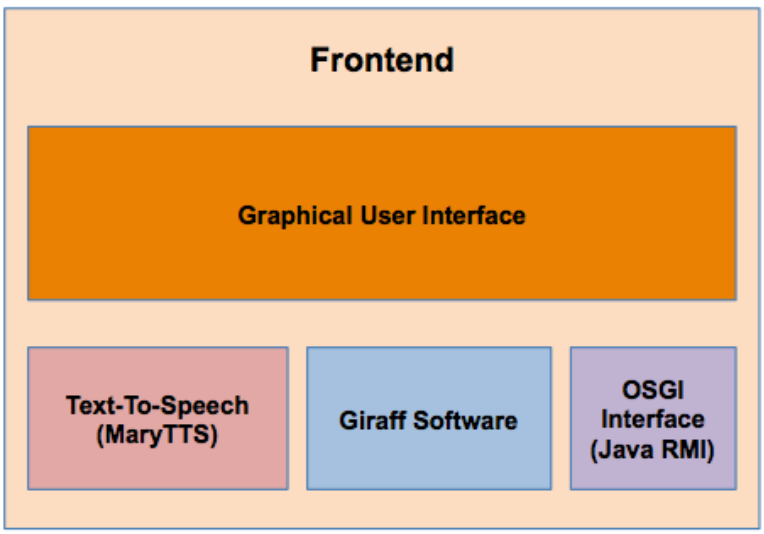


Figure 24 - The subdivision of functionalities inside the @Home frontend

4.2.1.1 The @Home Graphical User Interface

As sketched in Figure 22 the user interface is organized according to a set of views, each pertaining to a specific aspect:

- The main view, showing the normal Giraff application (the avatar)
- The messages view, deputed to the handling of text messages received from the People functionality of the DVPIS@Office

- The Data view, which displays (at the moment) the user data gathered by the sensors during the last week (shared space between primary and secondary users).

The views can be switched by touching the corresponding icon on the "belt" on the lower part of the screen. Obviously, switching to a different view does not interrupt the activity that is ongoing on that view: that is, switching to the messages view does not interrupt an in-progress call, or switching to the main view does not stop messages from being received.

The existing robot features are preserved: when in the main view, the robot behaves exactly as before, the only difference being the two belts on the top and on the bottom of the screen. Also, if an incoming call is detected the application immediately switches to the main view to let the user answer or refuse the call.

4.2.1.2 The Text-To-Speech services

The DVPIS@Home is endowed by TTS (Text-To-Speech) features, so that messages that are sent from the DVPIS@Office can be read out loud by the robot, to help elderly users that might have difficulties in reading too much text from the screen.

To achieve this, MaryTTS (http://mary.dfki.de/), developed by DFKI (Deutsche Forschungszentrum für Künstliche Intelligenz), has been employed as the engine of choice.

The pros of using this system are

- It is written in Java, so it plays well with DVPIS
- It is reasonably self-contained
- it is free (licensed under the LGPL license)
- It supports multiple languages besides English

In detail, MaryTTS has been embedded in the @Home by means of the *LocalMaryInterface* API, although we developed an interface to connect to remote MaryTTS servers too, in case this need will arise in the future. In both cases, the final output from MaryTTS is a raw audio stream which the standard Java Audio System can be fed from, which is quite convenient.

It is worth underscoring how the DVPIS@home is by no means tied to MaryTTS. Its Text-To-Speech interface is quite general and can be easily adapted to different TTS systems.

4.2.1.3 The embedding of the Giraff application

The original Giraff application runs "inside" the @Home, as shown in Figure 24. This is possible by leveraging the API provided by Giraff. In detail, the DVPIS@home is loaded by the Giraff startup scripts instead of by the regular application. The DVPIS, in turn, loads an instance of the Giraff application inside its virtual machine. This is achieved by using the *GiraffIntegration* class provided by Giraff, which enables to initialize the application and obtain a reference to its "screen", which can then be embedded inside the DVPIS@home. The DVPIS is aware of the state transitions inside the Giraff application (e.g., call in progress, user hang up, etc.) by means of the *GiraffApplicationListener* interface, again provided by Giraff.

4.2.2 The interface to the *backend*

As previously stated, the Frontend and the Backend communicate through Java RMI (Remote Method Invocation). At the low level, the two subsystems are each represented by a facade object (*ServerInterface* and *GiraffApplicationInterface* respectively) that is exported through RMI for the other part to use it. If necessary, the frontend takes care of executing the external processes that are needed for the overall system to work (the rmiregistry tool and the OSGI backend). In this way, the existing Giraff startup sequence does not need to be modified: the DVPIS installation program writes a couple of configuration files that instruct the Giraff startup scripts to load the DVPIS Frontend instead of the Giraff application, and the Frontend then takes care of the initialization of the subsystems it needs.

4.2.3 Internationalization

The internationalization is handled with the basic tools that the Java platform provides (resource bundles based on property files). However, it is not possible to detect the user language at runtime, since the language of the operating system upon which the software runs is usually English: the actual language that is displayed in the Giraff application can be configured at runtime by using the sentry web interface, but currently there are no APIs available to obtain this information. For this reason, currently the language selection in the @Home software must be performed ad-hoc (usually just after the installation) by right-clicking on the GiraffPlus logo in the top part of the application.

5 The Install and Maintenance Service: New Features

In year 1 of the GiraffPlus project partners expressed a clear need for a simple graphical user interface that would enable them to easily enter and edit the configuration description for each of the test sites. For this purpose we created the GiraffPlus EngineerUI web application described in

deliverable D4.1. Since then the GiraffPlus EngineerUI has has been completely rewritten in order to keep up with the changes in the project.

5.1 Implementation details

This section addresses one by one the single aspects that have deserved attention.

5.1.1 RESTful web service

The RESTful web service, which decouples the GiraffPlus EngineerUI from the GiraffPlus Long Term Storage web service, has seen many changes in the last year. In a first phase the development of the open-source Play Framework (http://www.playframework.com), has been followed and has been used to implement the RESTful web service. As a consequence the version of the Play Framework has been updated from 2.0 to 2.2, which called for a rewrite of the web service due to changes APIs in the new version of Play Framework. In the second place to change of the authentication mechanism in the GiraffPlus software ecosystem has been decided and a transition from a password-based to a certificate-based approach realized. In the scope of the GiraffPlus LTS, a Certificate Agency web service has been developed which securely issues X.509 certificates for GiraffPlus users and software components, which are used to secure communication between software components and to secure users' access to these components. To adapt to this approach a needed rewriting of parts of the Play Framework has been needed to deal with certificate-based authentication against the GiraffPlus LTS. In addition, changes in the GiraffPlus LTS schema required a rewrite to reflect changes in the data entities.

5.1.2 The GiraffPlus EngineerUI Web Application

The GiraffPlus EngineerUI web application saw a complete rewrite in the past year. From the initial version of the EngineerUI, which was quite unintuitive and required a deep understanding of GiraffPlus LTS data entities, we created a more user-friendly web application, which now guides GiraffPlus Engineers in a wizard-like fashion through various steps needed to completely configure or edit configuration of any test sites. Figure 25 shows the welcome screen of the GiraffPlus EngineerUI web application. As can be seen in the figure the welcome screen enables the GiraffPlus Engineer to create a new configuration entity (home, user, organization, sensor type or room type configuration entity) or to edit an existing configuration.

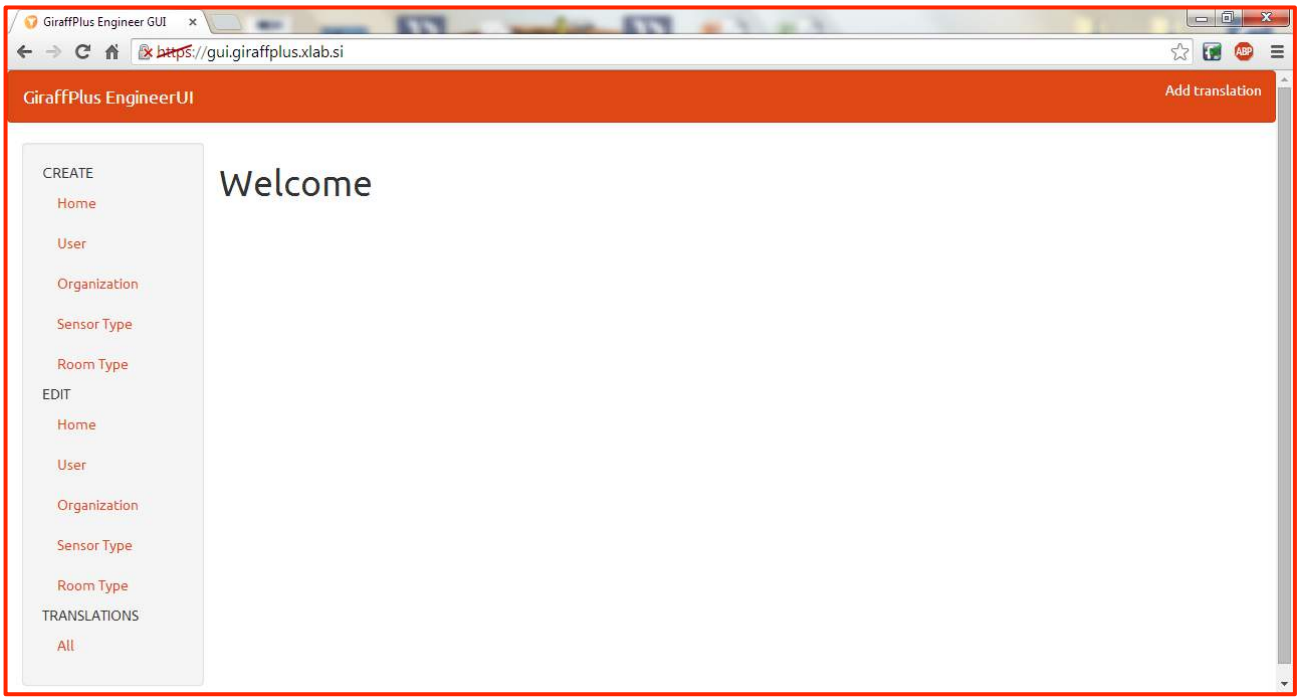
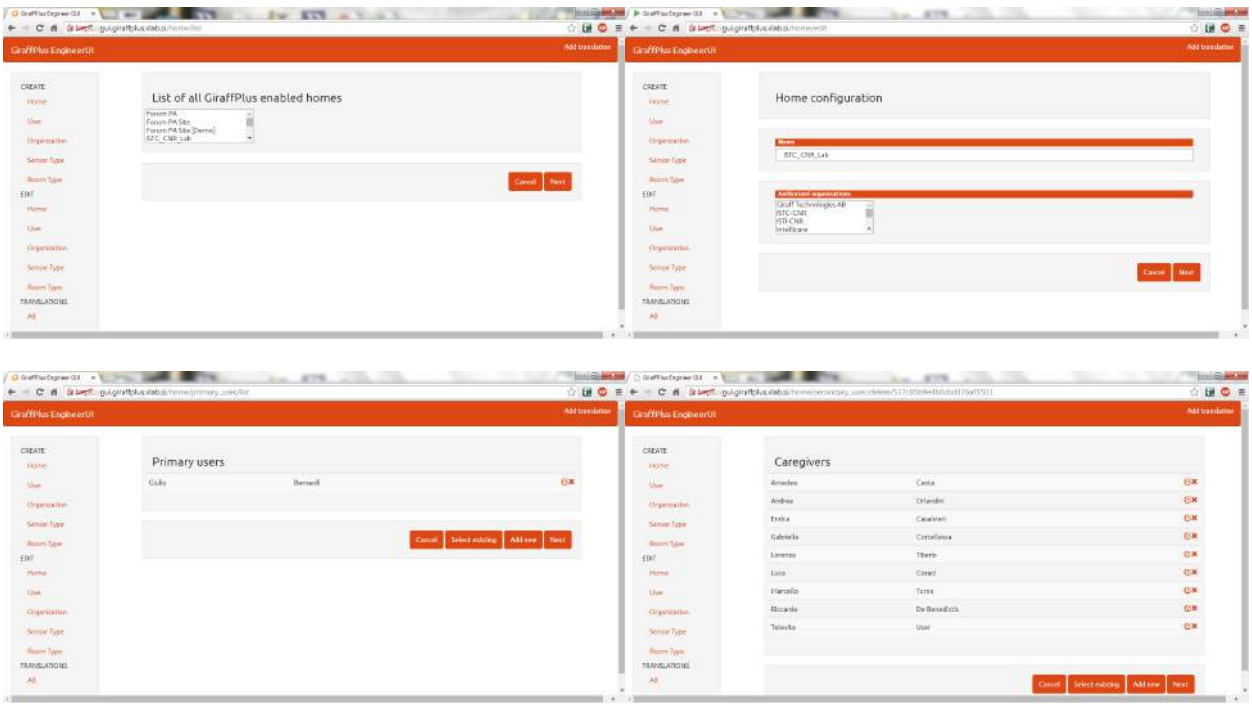


Figure 25 - GiraffPlus EngineerUI welcome screen



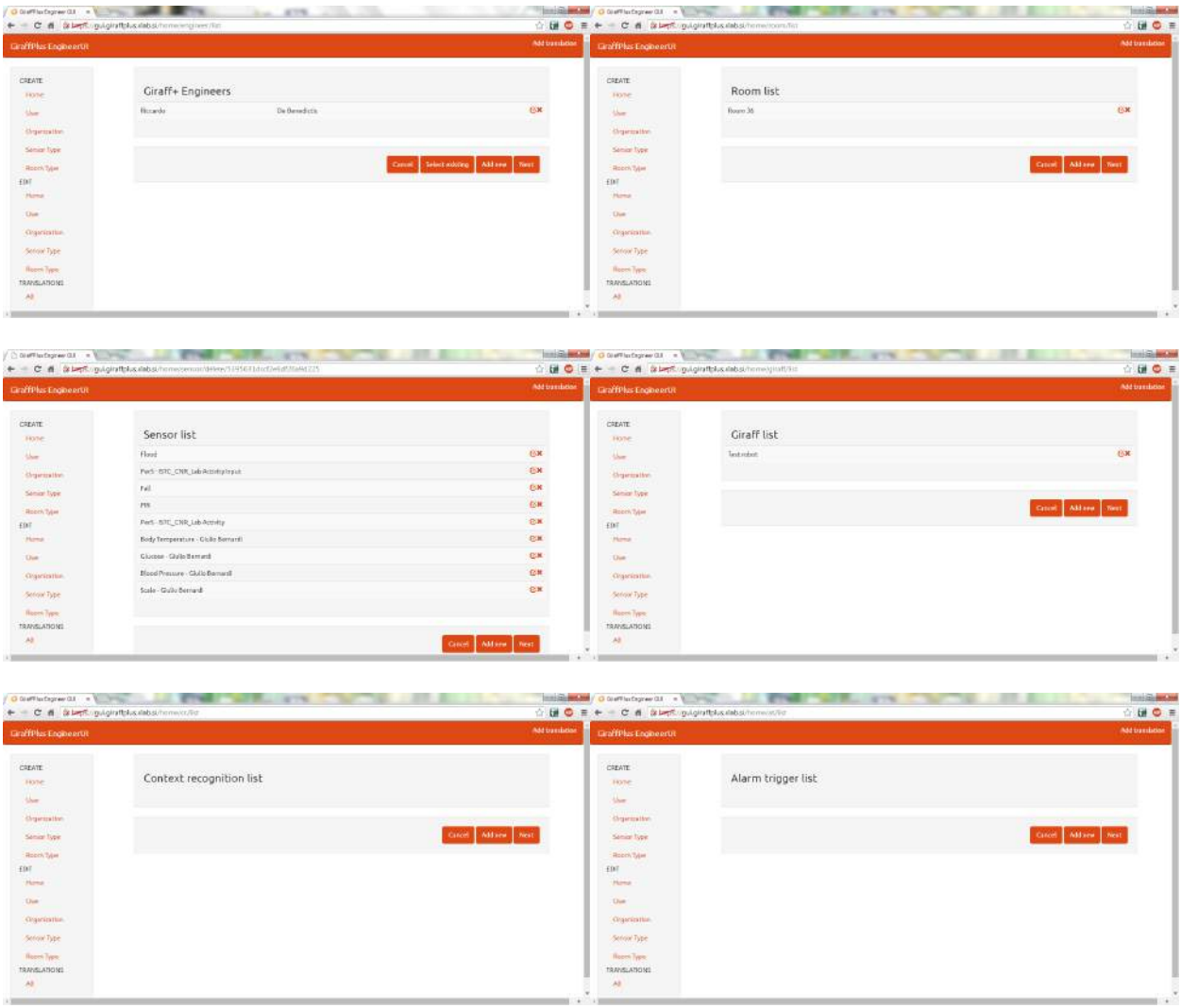


Figure 26 - Configuration of a home

An example of the use can be seen in Figure 26 which show the process of editing a home configuration in series of simple steps (from left to right, top to bottom). First engineer selects a home to edit (top left), then the engineer is asked to edit the basic information about the home – name and the organizations responsible for that home (top right). In the second step the engineer is required to enter the information on primary users of the home (second row, left) – either select users already in the system or enter information about the new user (see Figure 27). Having entered the information on primary users the engineer is required to enter information on caregivers (second row, right), followed by the information on engineers responsible for this home (third row, left).

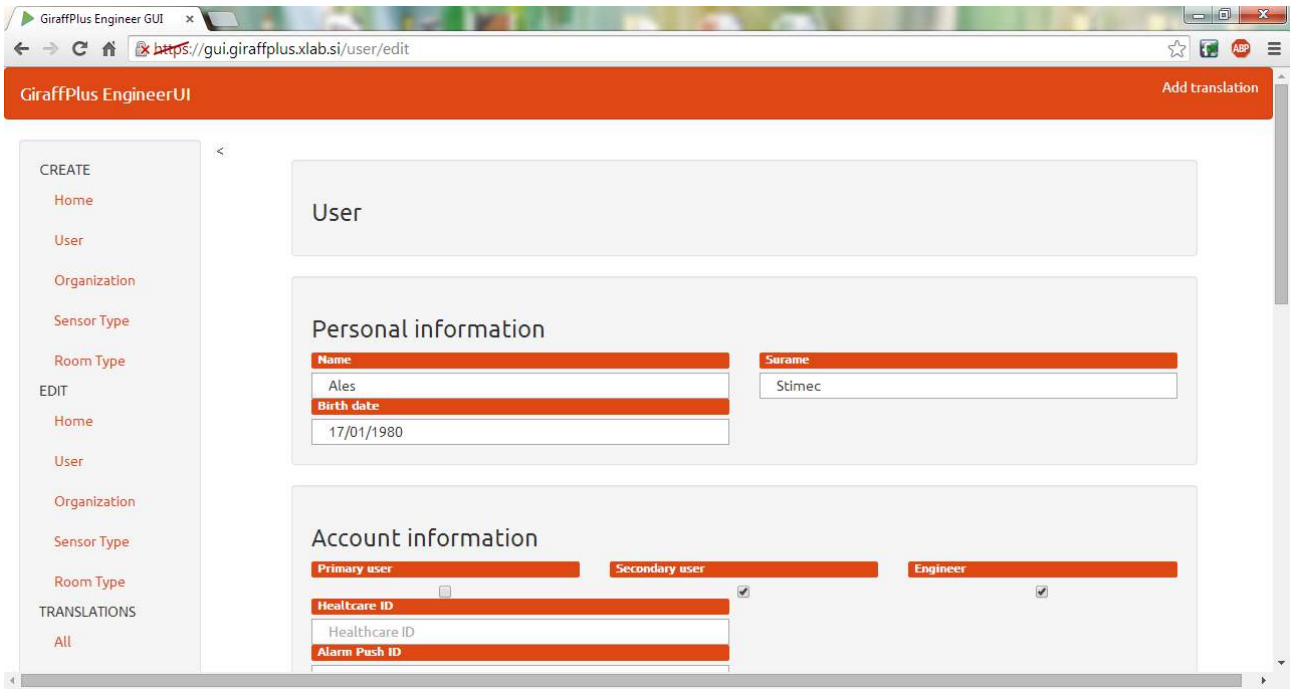


Figure 27 - Editing user information

In the following steps the engineer is required to enter information on rooms in the home (third row, right), sensors deployed in the home (fourth row, left), Giraff robots present in the home (fourth row, right), context recognition rules active in the home (fifth row, left) and alarm trigger rules (fifth row, right). Finally the engineer is presented with a request for certificate password as seen in Figure 28, which is used to generate a home certificate. This certificate can be downloaded either in BKS or JKS format and is used to authenticate the middleware running in the home (JKS) or on an android device (BKS) against other software components in the GiraffPlus ecosystem.

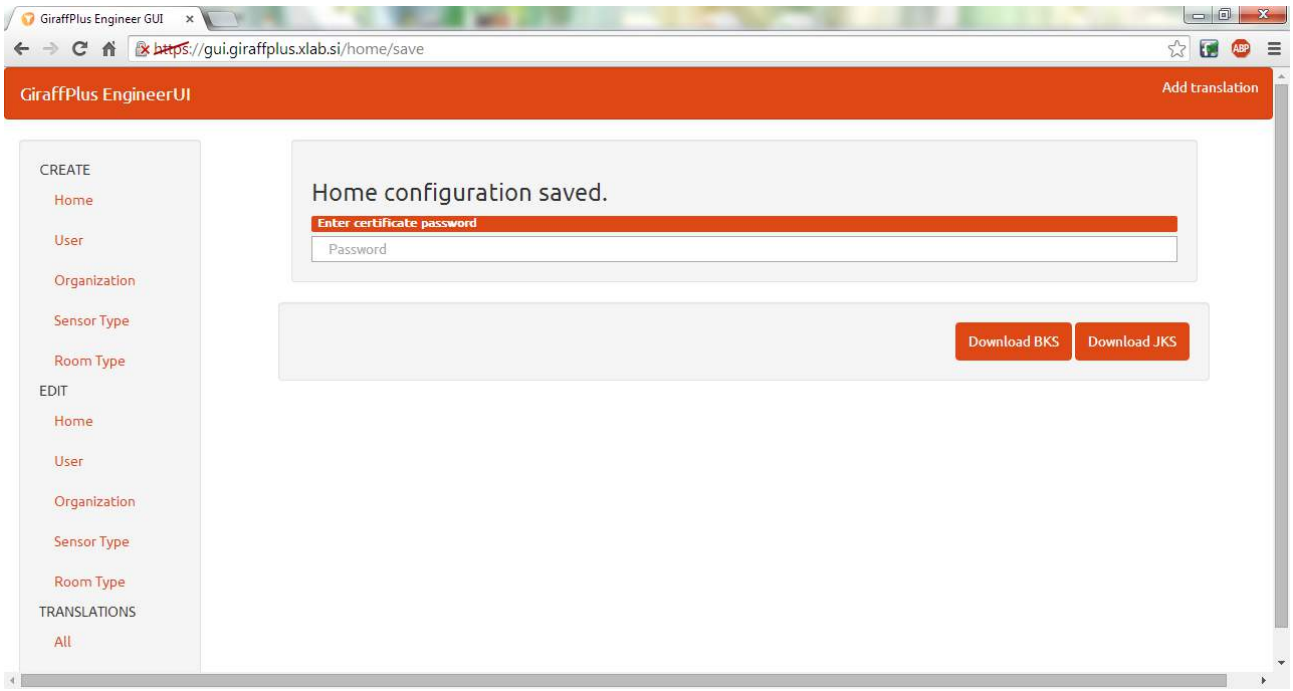


Figure 28 - Final step of the home configuration where the engineer obtains the home certificate

One of the latest additions to the EngineerUI was the part where GiraffPlus engineers can add translations for the internationalization feature of the DVPIS. As seen in Figure 29 engineers are able to list existing translations (left), which also warns them of missing translations, or add a new translation (right).

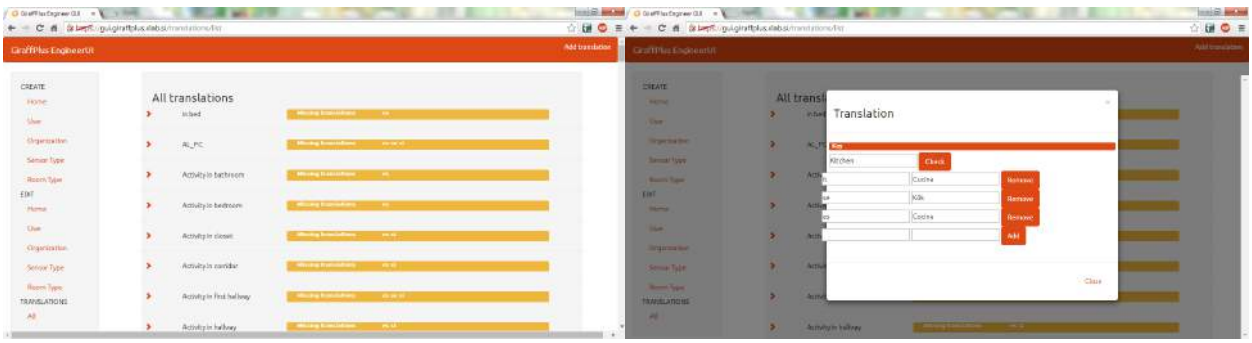


Figure 29 - Listing and adding translations

6 Personalization in GiraffPlus

In addition to the data visualization services described above, another aspect of the GiraffPlus system is the provision of personalized and proactive interaction services. The DVPIS back end is

indeed responsible for preparing the "personalized information to the users" and for offering support for different types of services like *reminders*, *reports*, *proactive suggestions*, *warnings* and *alarms*.

6.1 Definition of Personalization in GiraffPlus

The general idea behind the personalization services in GiraffPlus is shown in Figure 30 (but see (De Benedictis, Cesta, Coraci, Cortellessa, & Orlandini, 2014)). A user model is created that enables the personalization of services. In the current version of the software, two different type of information are used to create a user model, that is a combination of: (a) *static user* parameters, considered immutable during the care process, describing both general data (e.g., his/her age), medical condition (a chronic disease), and also his/her attitude toward interaction and technology (e.g., the degree of extrovertion, to be used for example, to avoid sending a person an excess of undesired information or recommendations in the case he/she is not willing to accept them); (b) *dynamic user* parameters, whose values are extracted from the physiological monitoring of the person and from observing variability of other continuous data flows like, for example, the monitored interactions that the user has with the system (both through DVPIS and through the Giraff robot) as well as the state of specific sensors dedicated to th2e alarm detection.

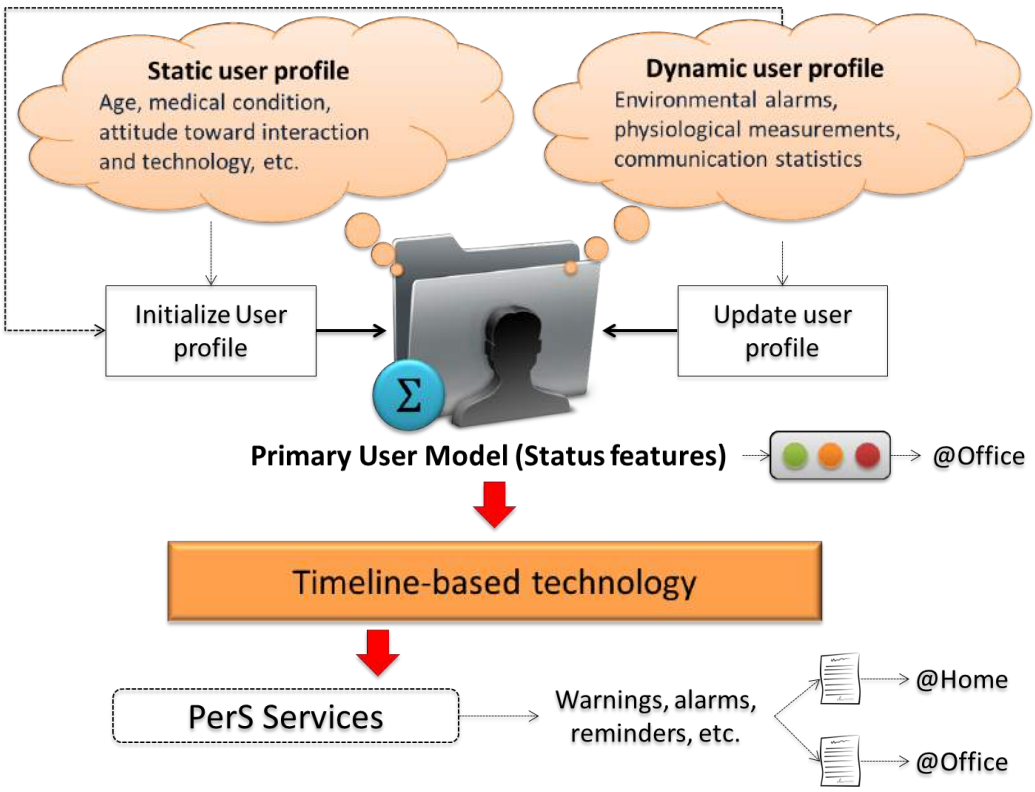


Figure 30 - The personalization service idea in GiraffPlus

These features are then passed to a *classification* process the result of which is a simplified user model indicating different aspects of the socio-health status of the person (*status features*).

Specifically the current implemented indicators (status features) are **Alarms status**, **Physiological status** and **Social Activities status**. For each of such dimension, a level is computed that can range among three values: *green = good*; *yellow = warning*; *red = risk*. In order to achieve this elicitation, the current implementation makes use of *naive Bayes classifiers* for extracting different features of the user profile. These classifiers can handle numeric parameters thus allowing us to include temporal aspects inside the classification (e.g., the social status could make a transition from "good" to "warning" in case the social interaction with the primary user does not occur for a long time). Although the collection of this data is still an on-going process, we have been able to gather different training sets for different classifications.

This classification is used to identify the set of the "colour based" indicators shown in the semaphores in Figure 30 that, in turn, correspond to those shown to the secondary user in the home page of the DVPIS@Office in Figure 5. Additionally, these indicators are passed to a timeline-based technology module that plans and triggers personalized stimuli toward the involved users. The planned stimuli are generated toward both the primary and secondary users. More specifically, by reasoning on dynamic parameters of the primary user, we pursue a "proactive service" in the form of *warnings*, *alarms* and *reminders*, modulated by the user model information. In broad terms the pursued idea is to create a proactive service at home by reasoning on user personal data and modulating system messages directed to primary users. It is worth observing that the possibility to deliver "system generated" messages to the primary user through the DVPIS @Home interface is a basic enabler for such a functionality. Additionally the user model can also be used to decide the type of messages, services and reports to send to secondary users through the DVPIS@Office.

In other words, the status of the indicators is used as a trigger for a strategy to decide which messages and appropriate frequency should be selected for the users. For example, observing value transitions (e.g., when the physiological status change from "good" to "warning") we can send simple questions like "have you taken your daily pill?" or issuing dietary suggestions like "avoid salt while cooking". These alerting messages can be sent, more or less often, both inside the house, to reassure or advise the primary user through the telepresence robot, and outside the house, to send an alerting message to a designated care person, etc.

6.2 Using Planning as an approach to enable personalization

This kind of behaviour is obtained by exploiting a particular temporal planning approach, known as timeline-based (we will adapt terminology from [2]), which allows us to easily represent information in time. We then enhanced this process with a classification system, which enables us to include complex domain knowledge inside the system, as shortly described in the following.

A *timeline* is, in generic terms, a function of time over a finite domain. Events on timelines are called tokens and represent information units over temporal intervals. For this purpose, tokens are represented through predicates extended with extra arguments belonging to the Time domain $\mathbb{T}$ (either real or discrete).

A *token* is an expression of the form:

$$n(x_0, \dots, x_k)@[s, e, \delta]$$

where n is a predicate name, $x_0, \dots, x_k$ are constants, numeric variables or object variables, s and e are temporal variables belonging to $\mathbb{T}$ such that $s \leq e$ and δ is a numeric variable such that $\delta = e - s$. A token $n(x_0, \dots, x_k)@[s, e, \delta]$ asserts that $\forall t$ such that $s \leq t \leq e$, the relation $n(x_0, \dots, x_k)$ holds at the time t.

Given the mutable nature of the user dynamic information and of their classifications in time, we have addressed the user modeling problem by making use of two timelines for each primary user that represent, respectively, their dynamic information and their current internal status classification. The use of different parameters allows to model different aspects of the dynamic information (such as current physiological parameters) and user profiles (such as current social profile, current physiological profile, etc.).

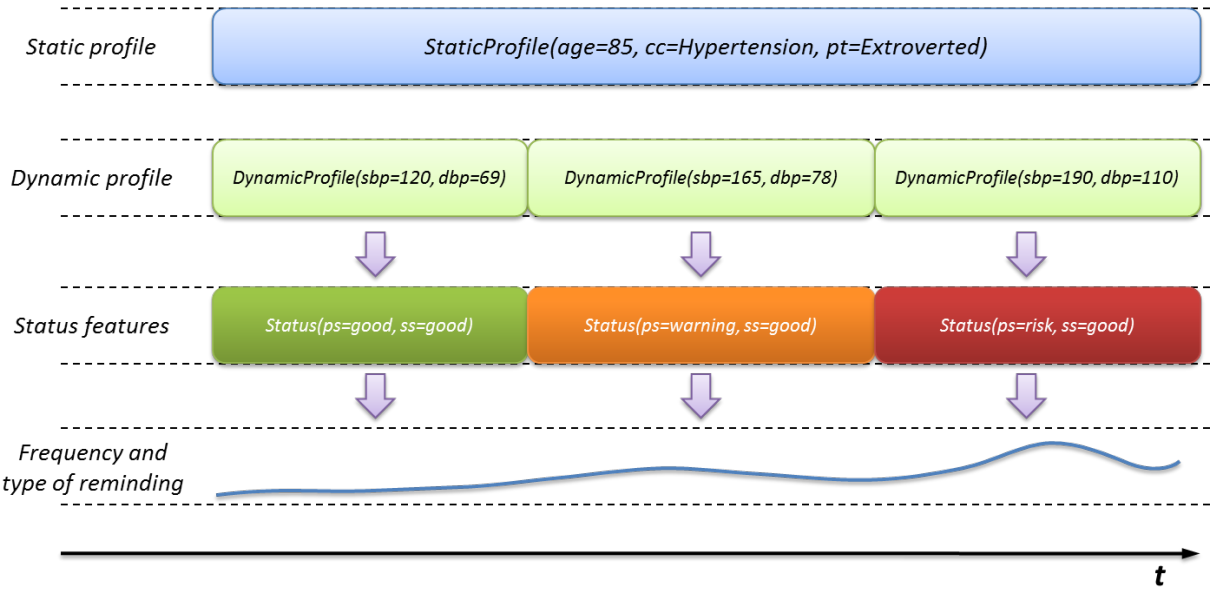


Figure 31 - User modelling through timelines

Figure 31 shows the personalization timelines of a sample primary user. In particular, in the sketched example we have used: (a) a predicate $StaticProfile(age, cc, pt)$[2] to represent the user static profile (characterized by age, chronic condition and personality trait), (b) a predicate $DynamicProfile(sbp, dbp)@[s, e, \delta]$ to represent dynamic features (characterized by systolic blood pressure and diastolic blood pressure from time s to time e) and (c) a predicate $InternalStatus(ps, ss)@[s, e, \delta]$ to represent the user classification (characterized by physiological status and social status from time s to time e).

Following the representation in Figure 31 it is possible to notice how the changing of the user's dynamic information, in particular his/her blood pressure, is associated to a change in his/her physiological status resulting in a transition from `good` to `warning`. This transition can automatically trigger a message to the primary user such as "did you take the blood pill today?" as we were mentioning before. In general state transitions can make the system recognizing dangerous situations and requiring a more significant intervention sending, for example, a reassurance message for the primary user together with an alert for the associated caregiver describing the current status of the old person. These kinds of "rules of behavior" are generalized in a concept usually called compatibility (the causal knowledge in temporal planning).

A *compatibility* is a tuple $c = \big(name(c), R(c)\big)$, where:

- $name(c)$ is the master (or reference) predicate and is an expression of the form $n(x_0, \dots, x_k)$, where n is a unique predicate symbol with respect to a timeline (i.e., no two compatibilities in a given timeline can have the same predicate symbol), and $x_0, \dots, x_k$ are its associated variable symbols.

- $R(c)$ is a requirement, i.e. either a slave (or target) predicate, a constraint among predicates or a conjunction of requirements.

Compatibilities define causal relations that tokens should comply with in order, for the plan, to be valid. Specifically, predicates can be used to represent specific events such as profile/status transitions, primary user's answers to specific questions, environmental configurations, etc. By adding slave predicates to compatibilities and by temporally constraining them it is possible to associate complex consequences to such events, up to enable the system to automatically follow the rehabilitation process associated with a therapy established by the caregiver. Once one of these events occurs, the associated compatibility is applied generating new stimuli for the old person such as new messages and/or new reminders. The timeline-based technology will be

[2] Since this feature is constant in time, its timeline never changes value hence the predicate has no temporal arguments.

responsible for placing these stimuli in time and for dispatching them when the moment comes. Compatibilities can be defined by secondary users by means of a simple editor provided by the @Office service allowing the GiraffPlus system to be tailored to specific needs of different caregivers.

As an example, in order to clarify this aspect, physiological classification follows a schema like $\langle age, cc, ms, bphour, sbp, dbp, hr, class \rangle$ associating the parameters age, chronic condition, taken medicines, blood pressure measurement hour, systolic blood pressure, diastolic blood pressure and heart rate to a class having values among "good", "warning" and "risk"[3].

It is worth noticing that personalized messages can include reports about the status of the home, statistics about sensors and habits about users thus we exploited this architecture to cover both requirements *1.c.3 Alarms in real time* and *1.d.1 Personalized reports* specified in Section 3.2. Finally, we have extended the static profile of secondary users with information about their preferences allowing us to cover requirement *1.d.3 Personalize the type of information sent to different secondary users*.

Table 4 shows the sample data currently used for the training of the Bayesian classifiers. Since the current training set is clearly insufficient as well as allegedly erroneous (at present the training set has been defined following common sense knowledge, it clearly needs a check with medical personnel in the near future), we are planning to allow some specific authorized users to "correct" the behavior of the system in time, generating a database of training data that can be possibly used for further purposes. Since the current state of the primary users is directly available to the system, a correction can be easily asserted by the secondary users by simply establishing a new classification, according to their common habits, for the associated primary users. In other words, the secondary user's correction task could be simply realized by allowing the them to assert the current classification (in terms of "good", "warning" and "risk") for each of wrong classifier (Alarms, Physiological and Social). The basic idea here is to extract knowledge from the secondary users avoiding to force them the learning of (more or less) complex mathematical functions and/or languages often not very intuitive for non-experts, for the definition of rules.

It is worth underscoring that the parameters that are currently used for establishing the static and the dynamic profile might be insufficient or inadequate for correctly classifying primary users. For this reason, a further discussion with final users involved into the GiraffPlus project might be appropriate in the last part of the project lifespan.

[3] An instance of training data, corresponding to the "risk" state of Figure 31, is $\langle 85, "Hypertension", "Diuretics", "15:30", 190, 110, 80, "risk" \rangle$. Since the user would be classified as "risk", a similar instance would produce an alarm for the caregiver.

Age	Last alarm past time	Last connection past time	Last blood pressure measurement hour	Last systolic blood pressure	Last diastolic blood pressure	Last heart rate	Is DVPIS connected	Is home connected	Is available	Alarm classification	Physiological classification	Social classification
			15	120	90	65					Green	
			12	125	95	60					Yellow	
			16	150	100	70					Red	
	0									Red		
	1									Red		
	5									Yellow		
	10									Yellow		
	20									Green		
	30									Green		
		0										Green
		1										Green
		5										Green
		10										Yellow
		20										Red
		30										Red
			13	190	130	75					Red	
			14	210	150	80					Red	
			10	220	140	71					Red	
			10	120	90	60					Green	
		50										Red
		100										Red
		1000										Red
			8	125	63	59					Green	

Table 4 - Initial training set for DVPIS home page semaphores

6.3 Generating personalized services

As mentioned earlier, in addition to being just a mere visual feedback for the secondary user, the classification process can be exploited by the timeline-based technology as a trigger for the generation of *proactive messages* toward both primary and secondary users. Depending on the compatibilities defined for the specific test case, the timeline-based technology reacts to the users' class changes generating a plan for the future interaction with the involved user. The execution of this plan will result in the generation of *new reminders*, *new warnings* and/or *alarms*, as well as *specific questions*, proposed to the primary users through the telepresence robot, whose answers are conceived to modify the behavior of the personalization services.

Messages can be sent to involved users exploiting the different technologies that might be available to them. Since we want to minimize the introduction of technology in the homes of the elderly, most of the messages directed to the primary users, are now sent through the telepresence robot, exploiting the DVPIS@Home service described in Section 4. Secondary users might have different needs. Most of the messages directed to the secondary users are sent to the DVPIS@Office service as already described in Sections 3.3.3 and 3.4.2. An exception that is worth mentioning is constituted by the **alarm messages** for which a greater reachability is demanded. At present time these messages are sent to secondary users by standard email (being confident in the use of smartphones by secondary users for providing a prompt reading). In this regard, we are currently evaluating the possibility of extending this service with the possibility to send SMS messages so as to reach faster even those secondary users who are not equipped with smartphones.

Although the system is open to extensions, the personalization service currently handles the following kind of messages (most of which have already been described in Sections 3.3.3 and 3.4.2):

- **Chat messages**, mostly sent by secondary users for secondary users, are free text messages that are instantly delivered to selected recipients. Although the graphical user interfaces distinguishes about chat messages and post messages (see Sections 3.3.3 and 3.4.2), the personalization service uses a simple flag to distinguish them.
- **Reminding messages**, created by secondary users or automatically generated by the system, can be delivered both to primary users, through the @Home service, and to the secondary users, through the @Office service. These messages are typically used to remind medications to primary users even if they can be fully customizable. Reminders can have a *time*, in which the reminder is delivered, a *delay*, after which the reminder is delivered, and a *period* representing the interval of time between periodic reminders. It is possible to associate delivery conditions to reminders so as to slightly adapt messages deliveries to the actual state of the user. Finally, reminders can be exploited by the personalization service to deliver proactive messages to primary users such as "avoid salt while cooking".

- **Questions**, similar to reminders, can be created by secondary users or can be automatically generated by the system but, unlike reminders, can be delivered only to primary users. The main difference between reminders and questions is the possibility to associate answers to questions and, in turns, to associate consequences to answers in terms of compatibilities. This means that a question like “have you taken your daily pill?” can be associated to the answers “yes” and “no”. In case the answer from the elder is negative, the compatibility associated to the “no” answer could generate a new message like “take it soon!”, the state of the user could be changed considering his/her easiness in forgetting things and a new daily reminder could be activated hopefully reducing future oversights.
- **Mails** are automatically generated by the system when alarms are detected and are sent to all the secondary users associated to the home in which the alarms has occurred.
- **Therapies**, initially associated to the homes, are simple collections of “reminders” and “questions” activated at first test site initialization.
- **Reports** can be generated by the personalization system as a summary of the events happened inside a home together with statistics on some important events and on activities detected by the context recognition.

As can be derived from the implementation details described in the Section 6.4, all these messages share a common structure called “Event” and thus they all can be combined at will as consequences of other events, as well as being potentially constrained by means of times of happenings, delays and periods.

6.4 Implementation details

The Personalization Services (PerS) exploits the GiraffPlus middleware (see D2.3) as a communication infrastructure. The aim of the middleware is to provide a publish/subscribe mechanism for accessing the context information about the physical environment and physiological data. In particular, the Personalization Services relies on two specific “virtual” sensors each home is supposed to have. The two sensors are intended to provide support for communication toward the home, basically, for creating new content, and from the home to the outside, for delivering personalized content both to the primary and to the secondary users.

Figure 32 describes the component diagram representing the personalization services high-level architecture. Exploiting an underlying reasoning environment based on timeline-based technology, a personalization server runs on each of the homes of the GiraffPlus system waiting for new content to be personalized coming from satellite services. When some special event occurs (temporal triggers, alarms, etc.) the personalization server emits new content for the different clients, addressed to both primary and secondary users, which are connected to the system. On the other side, both primary users and secondary users generate new content for the personalization services (e.g., create new reminders, answer to questions, request personalized

reports, etc.) and receive, at proper time, personalized content to be shown to the users through the @Home and/or the @Office services.

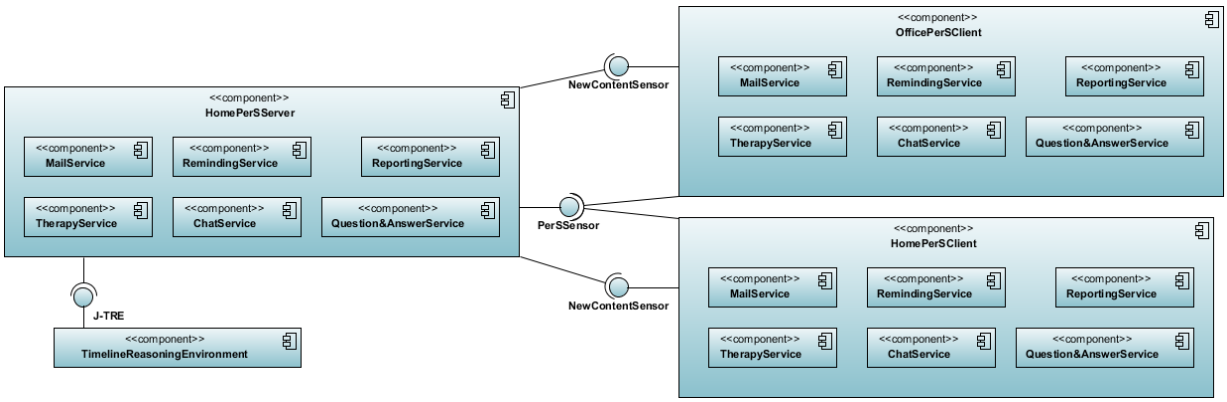


Figure 32 - PerS Architecture

Finally, Figure 33 represents all the communications messages that can be sent among PerS server and different PerS clients connected to the GiraffPlus system. Since most of these messages have a common structure, the Event class is taken as the root of the PerS hierarchy. Without going into details, we can see that an event has a *sender* parameter, representing the sender of the event, a *recipients* parameter (i.e., the users which will receive the personalized content), a *delay* parameter, that temporally constraints the event after some specific time, a *time* parameter, temporally constraining the event at a specific date (in case the specified date has already passed then the reminder is displayed immediately) and a *period* parameter, representing the interval of time between two instances of the same event when the given event should occur at a periodic rate. The presence of a set of *effects* associated to an event is used to represent target predicates for the compatibility associated to the event (see previous section for theoretical background about compatibilities). Finally, the *conditions* associated to the events describe execution conditions that must be verified in order for the event to be executed. By describing a condition as holding we specify that the event will remain pending until the condition is verified (e.g., a secondary user sends a message to a primary user who is not currently available because, for example, the system has detected the primary user is currently sleeping; the message will be delivered to the primary user as soon as the system recognizes that the user is available).

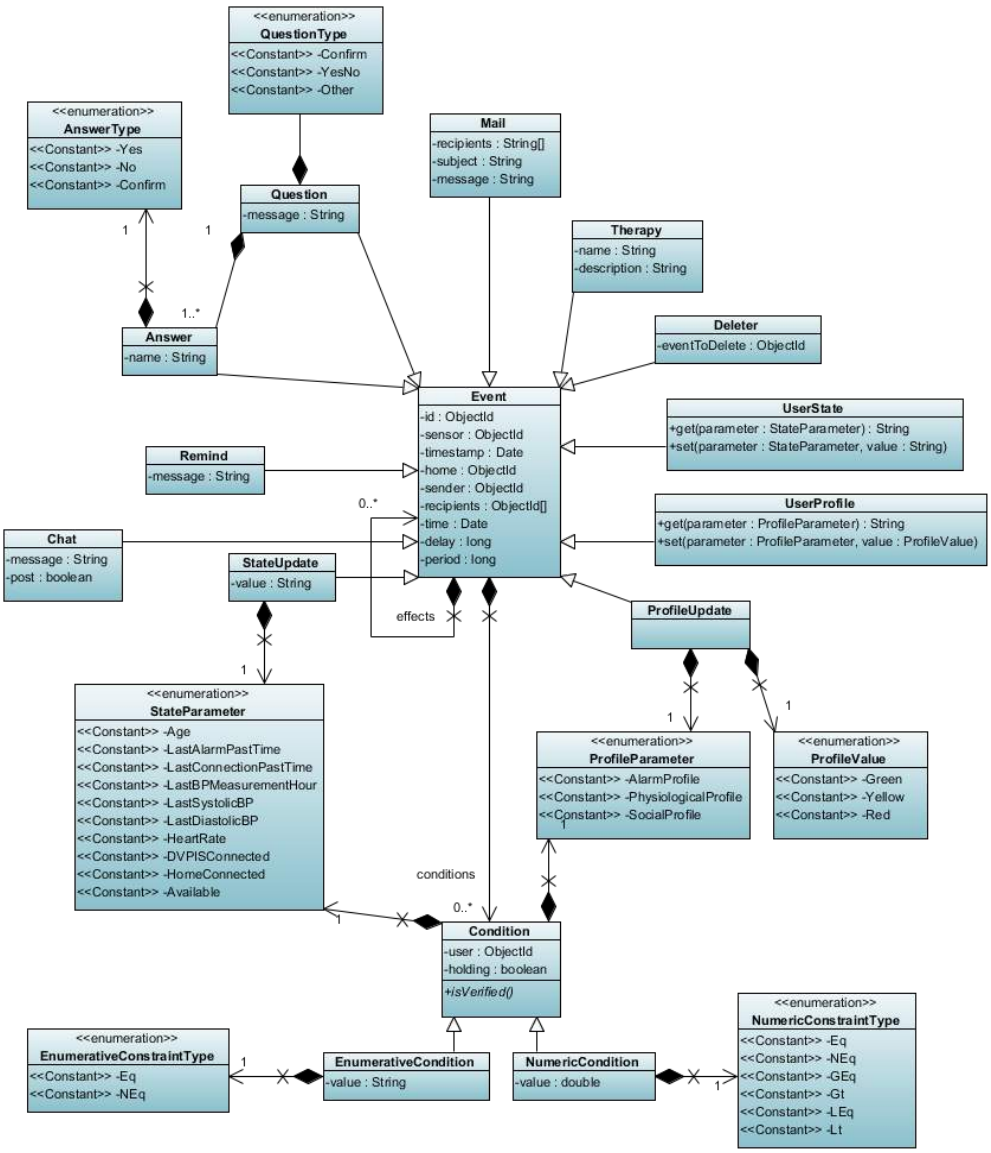


Figure 33 - Communication events in the Personalization Services

7 References

[1] R. De Benedictis, A. Cesta, L. Coraci, G. Cortellessa and A. Orlandini, "A User-adaptive Reminding Service," *Intelligent Environments (Workshops),* pp. 16-27, 2014.

[2] Cortellessa, G.; D'Amico, R.; Pagani, M.; Tiberio, L.; De Benedictis, R.; Bernardi, G.; Cesta, A., "Modeling Users of Crisis Training Environments by Integrating Psychological and Physiological Data," in *Proceedings of IEA/AIE-11, Siracuse, NY*, 2011.

A Appendix

A.1 Instructions and system requirements for the DVPIS@Office

A demo version of the software can be downloaded from the following links:

- WINDOWS users:
  http://pst.istc.cnr.it/giraffplus/demo-releases/dvpis_office_beta_setup.exe

- MAC OS users:
  http://pst.istc.cnr.it/giraffplus/demo-releases/dvpis_office_beta.zip

This appendix contains the system requirements and the instructions for installation of the DVPIS Beta Release including both the @Office and the @Home component.

A.1.1 System requirements

For Windows Users there are no particular system requirements. For non-Windows users the following requirement is mandatory:

Mandatory Requirement	Downloadable at
Java Development Kit, Version 7+	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Table 5 - System requirements

In fact the distribution for Windows user includes Java7.

Note: It is worth noticing that the embedded Giraff Pilot (integrated within the DVPIS) is available only for Windows users. Non-window users should use the pilot of the telepresence robot as separate software due to the fact that the Giraff Pilot software only runs on Windows machines (for example under MacOS it is possible to run the Pilot inside the Parallel environment).

A.1.2 Instructions for Installation

From this version on, an installer has been developed for WINDOWS users while MAC OS users can use the previous methodology to run the software: unpack & start. Below a detailed description of required step to correctly initialize and start the DVPIS@Office.

A.1.2.1 Installation of DVPIS@Office for WIN users

Once the download is complete, just double click on the .exe file and accept the permission to run the installer. An installer wizard will be presented. Please follow the guided procedure to correctly install the software as shown in the following list of pictures:

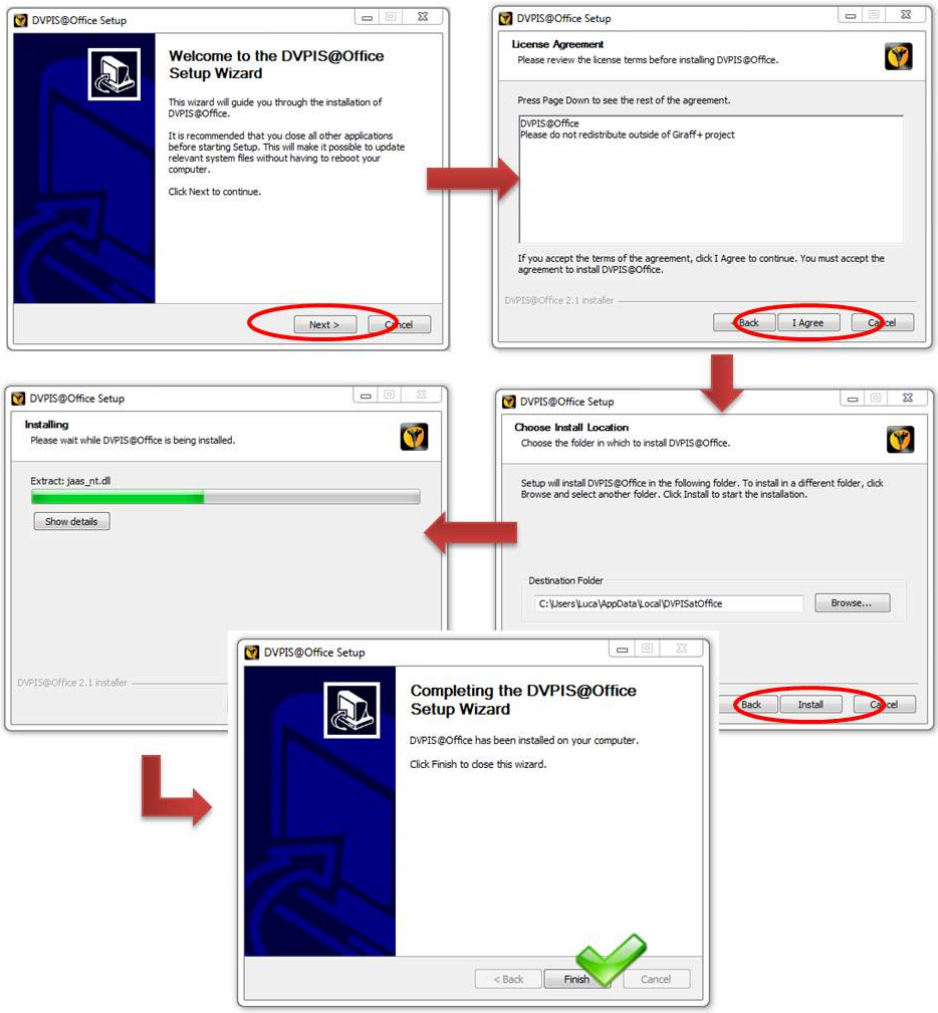


Figure 34 - Guided installing procedure of DVPIS@Office

When the installing procedure has terminated, two executable items will be added to your windows application start menu. Both of them will start the DVPIS@Office. The one of those labelled with "(debug mode)" will enable a DOS-like console that is used for debug purposes.

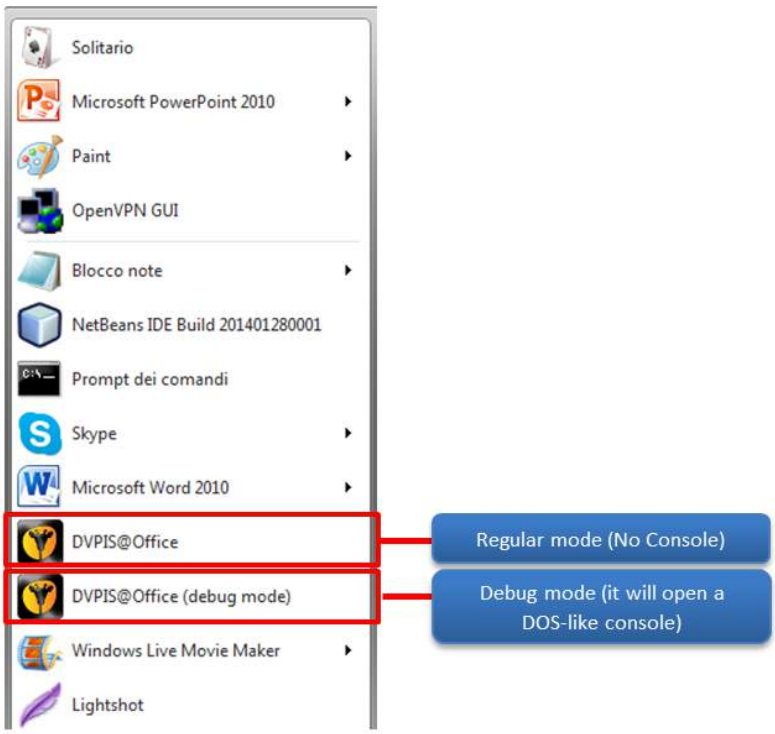


Figure 35 - Two DVPIS@Office application starting mode

A.1.2.2 Uninstallation of the DVPIS@Office for WIN Users

To uninstall the DVPIS@Office it is sufficient to use the provided uninstaller reachable by browsing "All Programs" in the WINDOWS start menu as shown in the next picture:

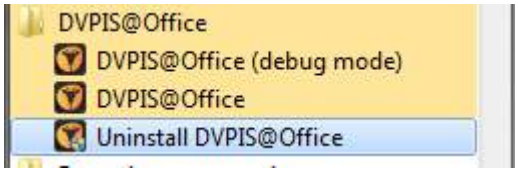


Figure 36 - uninstalling the DVPIS@Office

A.1.2.3 Installation of the DVPIS@Office for MAC users

The distribution dedicated to MAC OS users is presented as a zipped package that contains all software components. To start the DVPIS@Office please download the package at the proper link shown above, unzip the package and click on the application icon as shown in the Figure 37.

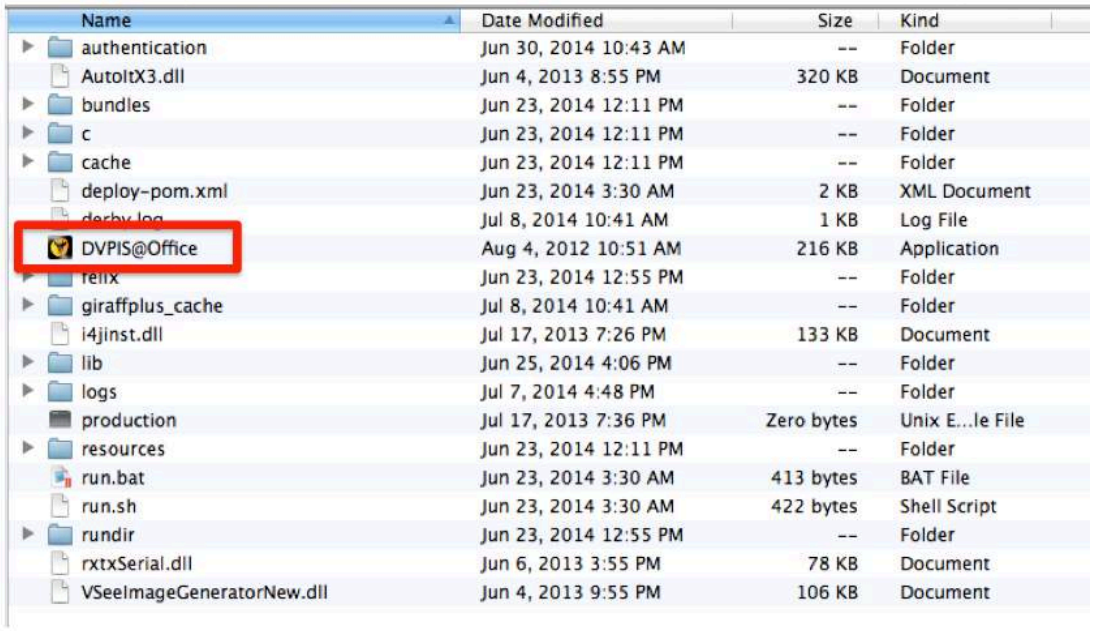


Figure 37 – Main application folder of DVPIS@Office on mac and application icon to start the software

To enable the console please open the MAC shell and browse to the DVPIS@Office main folder directory and just execute the run.sh by typing:

```
./run.sh
```

A.1.2.4 Uninstallation of the DVPIS@Office for Mac Users

Since this distribution is completely self-contained, to uninstall the product it's enough to completely delete from the system the main DVPIS@Office folder.

A.1.3 Login

Figure 38 - Login Dialog

The screenshot above shows the login prompt.

The DVPIS@Office does support internationalization; hence, before performing the login it is possible to select a language by clicking the combobox shown in the picture. 

In order to access the application you must own your authentication file, which is a Java Key Store File (.jks)[4]. Each authentication file is unique and personal; GiraffPlus-Engineers and only them, have the responsibility to create and maintain those file. Once you get your authentication file, you have to import it into the application by using the "browse" button and selecting the .jks file in your local system. **This step is required by only the very first access or occasionally if you need to access on the same machine with another user account.**

When the certificate is set, it's enough to insert in the password field your own password decided in the Engineering GUI with the GP Engineer and to enter the DVPIS@Office.

[4] http://en.wikipedia.org/wiki/Keystore

A.2 Instructions and system requirements for the DVPIS@Home

A.2.1 System Requirements

The following java should be installed on the robot or computer where the DVPIS@Home is installed.

Mandatory Requirement	Downloadable at
Java Development Kit, Version 7+	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

A.2.2 DVPIS@Home installation

The software is meant to be installed on a Giraff Robot with a touch screen. If needed it can also be installed on a Windows PC for testing purposes, but then no integration with the Giraff software is obviously possible.

The latest version of DVPIS@Home can be downloaded from

http://pst.istc.cnr.it/giraffplus/athome/athome_setup.exe

The software can be installed by simply running the downloaded file. If you are not installing the software on a real robot you will also have the option of choosing the installation directory.

At the end of the installation, the program will remind you of installing a certificate into the installation directory: the exact path and file name is shown on the screen, but usually the certificate will be “certificate.jks” and should be put inside the program installation directory.

To run the software, just restart the Giraff application (or restart the robot).

If you installed the software on a standard PC, double click on the “DVPIS@Home” shortcut on your desktop or under Start Menu.

A.2.3 Language selection

The first time you run DVPIS@Home you might want to select the language of your robot (English by default). To do this, right click on the top bar (the one with the GiraffPlus logo) and choose your language: the software will automatically restart. If you are not running the software on a real robot, you have to start the application again manually.

A.2.4 Uninstallation

Close DVPIS@Home if running, then click on “Uninstall DVPIS@Home” under Start Menu.

Warning: the installed certificate will be deleted too.

A.2.5 Advanced options

Warning: regular users do not need to change advanced options, they are provided as a facility to developers only.

Configuration data is stored in a file called athome_startup.properties, located in the “dvpis” directory inside the “Application Data” user directory: this means that the exact path will be something like C:\Users\your user name\AppData\Roaming. The file is a java properties file, which means that every property is encoded as a “key = value” pair, one per line.

The following properties are recognized:

user.language	A two-letter language identifier, like “en” for English or “it” for Italian
home_dvpis	The path to the home_dvpis folder (the backend), for developers only
fullscreen	True or false. Defaults to true on real robots, false otherwise.
dvpis.verbose	True or false. If true it shows debugging information for developers.