	PersonAAL	

Deliverable 1.1.b

Architecture Specification (Revised)

Responsible Unit: CNR

Contributors:USI

Document Technical Details:

Document Number	D.1.1.b
Document Title	Architecture Specification (Revised)
Version	1.0
Status	Final
Work Package	WP1
Deliverable Type	Report
Contractual Date of delivery	30-09-2017
Actual Date of Delivery	29-09-2017
Responsible Unit	CNR
Contributors	USI
Keywords List	Personalization, Web Applications, Architecture, Context-dependent systems
Dissemination Level	Public

Document Change Log:

Version	Date	Status	Author	Description
0.1	21/09/2017	Draft	Marco Manca, Fabio Paternò, Carmen Santoro, Luca Corcella	Initial version
0.2	25/09/2017	Draft	Marco Manca, Fabio Paternò, Carmen Santoro, Luca Corcella	Version for Review
0.3	27/09/2017	Update	Ivan Elhart (USI)	Update on Security
0.4	28/09/2017	Review	Ivan Elhart (USI)	Review
1.0	29/09/2017	Final	Marco Manca, Fabio Paternò, Carmen Santoro, Luca Corcella	Final

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Contents

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

1 INTRODUCTION

One of the goals of the PersonAAL project is to extend the time that elderly people can live independently in their preferred environments by using ICT technologies for personal healthcare.

In order to achieve this goal there is a need for a platform that is able to support personalization of the existing web applications used by formal and informal caregivers or even by the elderly users themselves. The platform can provide enhanced safety through continuous and personalised monitoring and personalised warning messages issued in risky situations; persuasive messages to stimulate the elderly in healthier habits (e.g. do more physical activity); and personalised applications taking into account specific elderly characteristics.
We focus on personalising Web application because they are widely used, can be accessed through different types of devices, and easily modified.

In this deliverable, we present how the platform for the personalisation of context-dependent applications by non-technical users has evolved during the project. While the overall architecture has not changed the various component modules have been enriched with further functionalities. In this deliverable we first describe the overall architecture, then the main platform components and how their functionalities have been updated during the second year of the project, and finally the information flow across the platform components.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

2 THE OVERALL ARCHITECTURE

The platform aims to provide support for elderly users who live alone or distant from relatives with increased risk of unrecognised decline onset. They may suffer from physical issues such as cardiovascular problems, reduced sight, irregular eating habits, and/or cognitive issues such as tendency to forget tasks and events with increased risk of social isolation, and depression.

Such target users are very diverse in terms of actual behaviour and needs. Thus, the platform allows formal and informal caregivers to modify the behaviour of the existing applications by specifying personalization rules. Such rules are structured in terms of triggers and actions that allow users to easily associate dynamic events with the activation of desired effects.

The main components of the platform (Figure 1) are:

- The Personalization Rule Editor, which allows caregivers to specify the personalization rules specific for the target users and contexts of use;
- The Adaptation Engine, which informs the applications when any change is necessary;
- The Context manager, which is a middleware able to detect information from sensors, objects, and applications and communicate it in a more logical structure to the other platform components;
- Secure Authentication Server, which comprises various techniques to keep the platform from undesired accesses;
- Behaviour analysis, which detects when anomalous behaviour occurs (it is described in detail in D1.2);
- Persuasion, which aims to generate actions expressed according to persuasive techniques (it is described more in detail in another deliverable).

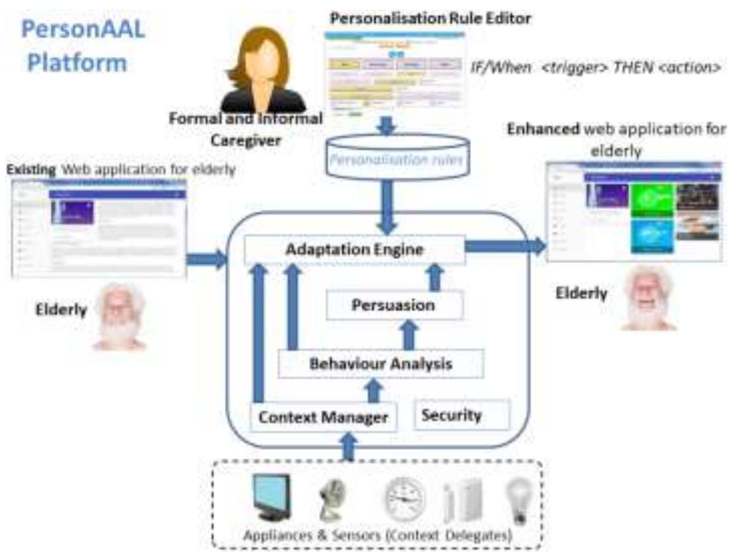

PersonAAL Platform
Personalisation Rule Editor
IF/When <trigger> THEN <action>
Formal and Informal Caregiver
Personalisation rules
Enhanced web application for elderly
Existing Web application for elderly
Adaptation Engine
Persuasion
Behaviour Analysis
Context Manager
Security
Elderly
Elderly
Appliances & Sensors (Context Delegates)

Figure 1: The Overall Architecture

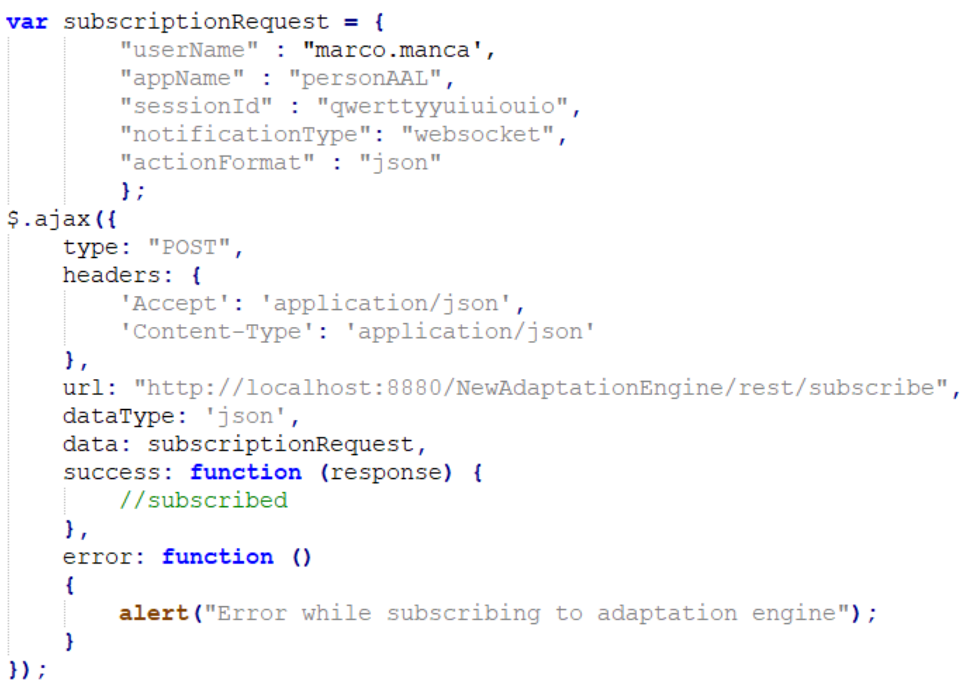
The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3 PLATFORM COMPONENTS AND UPDATES

3.1 Adaptation Engine

The Adaptation Engine is the module in charge of applying adaptation by sending to applications and objects the changes to apply. At the configuration time, it receives the set of active rules from the Authoring Tool along with the address of the relevant Context Server. The rules are associated with specific *application name and user name.* Thus, each user of the application can have different personalization rules. At run time, when an application subscribes to the adaptation engine, the adaptation engine loads the associated rules and subscribes to the Context Manager for the specified events and conditions. When a rule is triggered (i.e., when an event occurs or when a condition is verified) the context manager notifies the Adaptation Engine, which extracts the list of actions from the triggered rule and sends them to the concerned application for being properly applied. Sometimes multiple rules with conflicting effects can be triggered at the same time. In such cases, the adaptation engine is in charge of selecting the rule that should be applied using a priority approach.

Since the early version of the architecture, the communication between the application that should be personalized and the Adaptation Engine takes place through a web socket communication channel.



```
var subscriptionRequest = {
        "userName" : "marco.manca',
        "appName" : "personAAL",
        "sessionId" : "qwerttyyuiuiouio",
        "notificationType": "websocket",
        "actionFormat" : "json"
        };
$.ajax({
    type: "POST",
    headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    },
    url: "http://localhost:8880/NewAdaptationEngine/rest/subscribe",
    dataType: 'json',
    data: subscriptionRequest,
    success: function (response) {
        //subscribed
    },
    error: function ()
    {
        alert("Error while subscribing to adaptation engine");
    }
});
```

Figure 2: Subscription to the Adaptation Engine

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

In order to be personalized a web application has first of all to subscribe to the Adaptation engine, and when a rule is triggered the application has to apply the received actions. An example subscription to the Adaptation Engine is shown in Figure 2.

The basic steps that an application must perform in order to customize its behaviour by interacting with the Adaptation Engine are:

1. Open a web socket communication and get the ID of the opened session (session id);
2. Subscribe to the Adaptation Engine sending the following information (see Figure 2): user name, application name, session id, notification type and action format. The username and application name are used by the adaptation engine in order to retrieve the adaptation rules defined through the authoring tool; the session id is used by the adaptation engine to identify the user and to send the actions to the right client when a rule is triggered. The supported action format can be JSON or XML depending on the application preferences. The notification type specifies which channel the adaptation engine should use to send the actions.
   The adaptation engine returns: i) the connection status information about the web socket connection; ii) subscription status information about the state of the subscription to the adaptation engine (subscribed/not subscribed); III) names of the subscribed rule(s); iv) not subscription rules represents the name of the rule(s) that thrown an error on the context manager during the subscription phase; v) the description in natural language of the subscribed rules (it can be useful if the application want to show to the user the current personalization rules);
3. Wait for receiving the list of actions to apply from the Adaptation Engine (through the web socket opened before).
4. When the application receives the actions list from the Adaptation Engine it must invoke a function that for each action, depending on the action type (e.g. update, create, delete, etc.), calls the corresponding function that will apply the customization.

3.1.1 REST communication

The communication between adaptation engine and applications through web socket connection is useful because the actions are received directly by the client side part of the web application, thus it is very easy to apply the actions that change the appearance of the user interface. However, this type of communication can introduce some problems when the received actions perform changes in the appliance state, indeed it can happen that such changes involve the server side part of the application because the client side is not allowed to manage remote appliances. Moreover, it is possible to apply the actions only when the application is opened on the client browser; because when the application is closed the web socket connection is terminated and it is no longer possible send the actions.
In order to apply the adaptation rules also when the client is not connected, we decided to allow users to subscribe to the adaption engine specifying a different communication channel (see Figure 3).

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

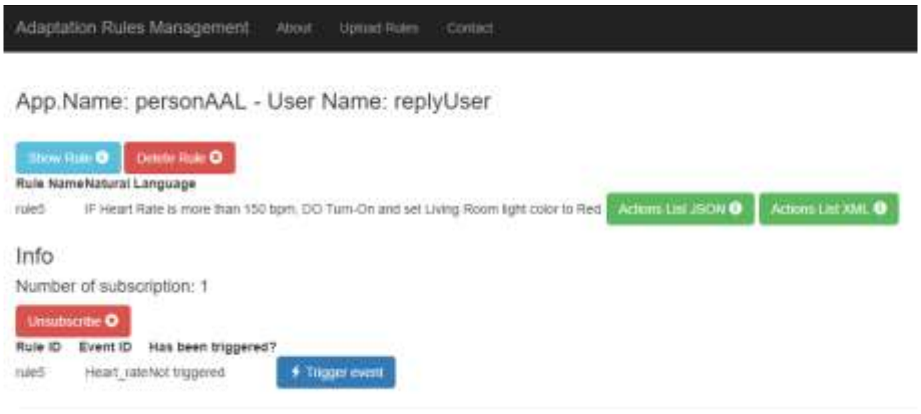
```
var subscriptionRequest = {
        userName :  userName,
        appName : appName ,
        actionFormat : "json",
        notificationType : "rest",
        notificationEndPoint : "http://taurus.isti.cnr.it:3000/api/receiveActions"
};
```

Figure 3: Subscription to the Adaptation Engine (REST notification)

In the new version of the Adaptation Engine the application can specify the notification type REST and indicates the application endpoint to which the adaptation engine must send the actions. If the action is "turn on a light" it can be applied even if the application is not loaded on the user browser.

3.1.2 Administration Panel

In order to manage the rule associated to an application and to a user we developed a private administration page for the adaptation engine (Figure 4).
In this page, there is the list of the rules stored in the Adaptation Engine, for each rule it is possible to analyse the correspondent code in the Event Condition Action (ECA) format, delete the rule, moreover it is also shown the natural language which described the rule. There are two additional buttons to show the list of action in XML and JSON format.
In the lower part, there are some information about the subscription: the number of subscription for the considered application and username (it worth noting that a user can subscribe to the adaptation engine for the same application through multiple devices, thus there can be multiple subscriptions and he has to receive the personalization actions in each device); there is a button to unsubscribe all the subscriptions; a field called "Has been triggered" which indicates the last time the rule has been triggered (for debugging purposes). Finally, there is the "trigger event/trigger condition" button which simulate the rule execution forcing the adaptation engine to send the actions to the subscribed devices (for debugging purposes).



App.Name: personAAL - User Name: replyUser

Rule Name	Natural Language
rule5	IF Heart Rate is more than 150 bpm, DO Turn-On and set Living Room light color to Red

Info

Number of subscription: 1

Rule ID	Event ID	Has been triggered?
rule5	Heart_rate	Not triggered

Figure 4: Adaptation Rules Administration Panel

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.1.3 Dynamic reference to context values in reminder and alarm messages

As a result of various discussions with partners, the need to be able to refer to context values within the text of a reminder or an alarm emerged. A partner has to define a rule to persuade the elder to have more physical activity. The context is able to store the number of steps currently performed by the user, thus it is possible to define a rule such as "IF the daily steps performed are less than 5000 DO send this message "You did a few steps today, go outside to have a walk!".

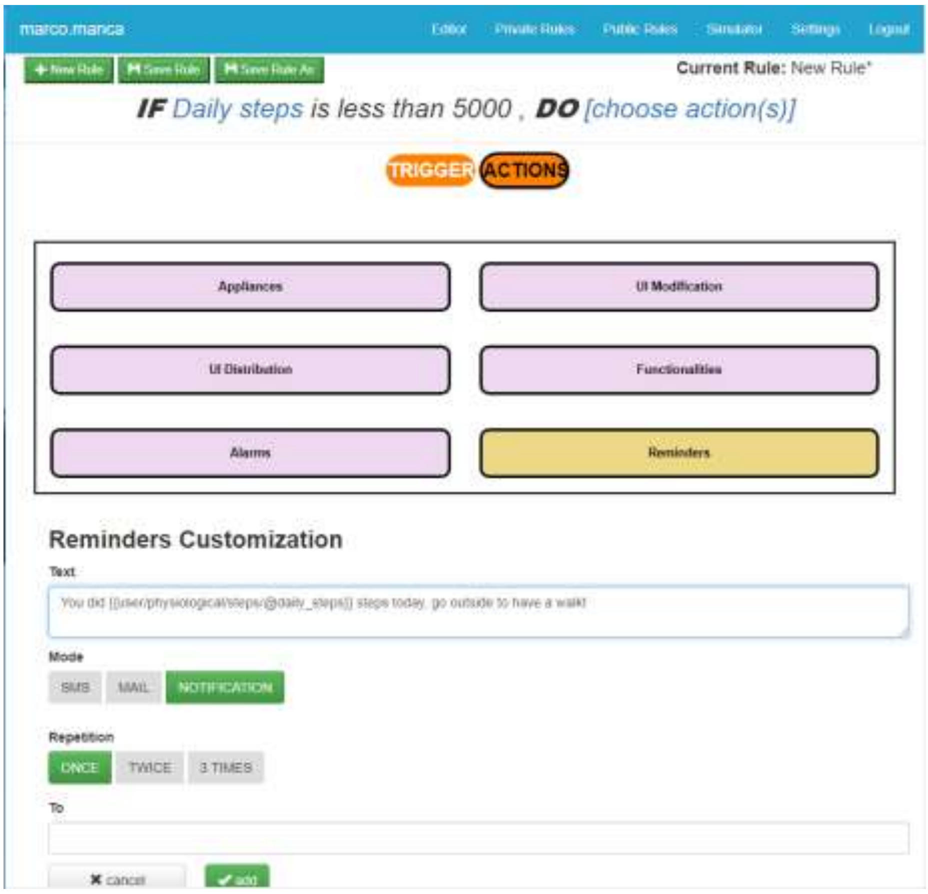
This message is too generic and the user may ignore it because he does not feel engaged enough. A way to increase the participation is to insert in the persuasive messages references to the current user context such as: "You did only 134 steps today, go outside to have a walk".

For this reason, we added the possibility to refer to context values inside a reminder or an alarm message.

Figure 5 shows an example of a reminder definition. In the text field, there is an instruction defined in the following format {{xPath}}. This type of instruction specifies the path of a context entity inside the context manager. When the adaptation engine analyses the actions before sending them to the application, if it finds a message with this format it queries, the context manager asking the current value of the context entity described by the specified xPath. The context manager returns the current value to the adaptation engine which can send the message replacing the xPath with the actual value. In that case, the message will be: "You did only 134 steps today, go outside to have a walk".

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Ministero dell'Istruzione, dell'Università e della Ricerca
FCT Fundação para a Ciência e a Tecnologia
Forskningsrådet
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra



Current Rule: New Rule*

IF Daily steps is less than 5000 , *DO* [choose action(s)]

Reminders Customization

Text: You did [[user/physiological/steps/@daily_steps]] steps today, go outside to have a walk!

Mode: SMS MAIL [✓ NOTIFICATION]

Repetition: [✓ ONCE] TWICE 3 TIMES

To:

Figure 5: Dynamic reference to context values in a reminder message

3.2 Context Manager

The Context Manager provides all the functionalities to collect data from external contextual sources, store and analyse them. It is composed of a Context Server and a set of Context Delegates. At run time, contextual information is organized in a set of instances of interconnected classes, where each class can have one or more attributes.
Data arrive from various sources, such as sensors (temperature, noise, light, doors/windows closure, power absorption, etc.) or external services (e.g. weather forecast). Connection to sensors and data gathering is made by context delegates specifically developed for the type of sensor to interface with and for the type of platform where the software is running.
Such software is potentially distributed on various devices; they are considered to be part of the overall context management functionalities, and implicitly manage the appearance/disappearance of resources/sensors in the environment. This is possible because the Context Delegates are expected to update sensors values at specific time intervals, and the

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

entities stored in the Context Server are characterized by an update timeout. Thus, if a sensor fails, a timeout exception is raised on the entity referred by the sensor. For instance, if the temperature sensor of the living room is turned off or is uninstalled, then the Context Delegate responsible for that sensor stops updating the temperature entity for the living room on the Context Server. The subsequent update timeout will imply that the entity is set to disabled state on the Context Server.
The context manager is described through a context model which defines all the context entities, the relationships between them, and their attributes.
A specific, domain-oriented version of the context model is created to better suit the needs of a certain application domain. The domain-oriented context model, rather than replacing the generic one is a refinement of it, and it acts as an intermediary between the end-user Authoring Tool and the Context Manager. The domain-oriented context model is used as input to structure the domain-dependent Authoring Tool, in such a way to parameterize the user interface for rule triggers selection accordingly. The creation of a domain-oriented context model from the generic context model is performed with the support of domain experts. There is no limit to the definition of new classes in the domain-dependent context model, neither in terms of number nor in terms of names or connections among them.
This distinction between generic and domain-specific context model, facilitates the reusability of the context-dependent platform. For example, the User class can be modelled differently in different applications, each one with an associated domain-specific context model, which refers to the same generic context model.
The context model is stored as a XML data schema which is automatically transformed in classes and instance variables and it provides a general reference vocabulary.

3.2.1 Context Model and RESTful services Updates

During the face to face meetings and the various calls with the partners of the project, we decided to update and expand the context model in order to store the data coming from the BITalino chest band developed by Plux, to model the medication needs managed by the application developed by IMB Norway and to describe the user motivation detected through the Reply application.
The data sensed by the BITalino chest band are: respiration rate, heart rate, body temperature, walking steps and body position. Thus, we extended the context model adding those attributes under this hierarchy: User -> Physical and Mental -> Physical.

The data are sent from BITalino every 5 minutes and include the respiration and heart rate, the body temperature in degrees, the number of steps performed from the previous transmission and the list of the body position detected in the time interval.
The type of body position detected by the device are: standing, supine, prone, right and left.
We created a RESTful service which allow to update all the physical attributes through a single communication, below there are the specifications.
We also developed a couple of services (one to update and the other to get the value) for each single physiological attribute:

SET physiological values:	
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/physiological"

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Supported method	POST
Data Type:	JSON

Data Format:	var physiologicalObj = {
	"heartRate":101,
	"numberSteps":12,
	"positionArray":[
	{"position":supine,"timeStamp":1493387947576},
	{"position":prone,"timeStamp":1493387950587},
	{"position":stainding,"timeStamp":1493387951587},
	],
	"respirationRate":20,
	"temperature":38
	}

GET physiological values:	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/physiological
Returned Data Type:	JSON

SET heart rate	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/heartRate/{heartRateValue}

GET heart rate	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/heartRate
Returned Data Type	int

SET body position	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/body_position/{body_positionValue}

GET heart rate	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/body_position
Returned Data Type	String

SET respiratio	

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission | Ministero dell'Istruzione, dell'Università e della Ricerca | FCT Fundação para a Ciência e a Tecnologia | Forskningsrådet | Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra

n rate	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/respirationRate/{respirationRateValue}

GET respiration rate	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/respirationRate
Returned Data Type	int

SET steps	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/steps/{stepsValue}

GET steps	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/ steps
Returned Data Type	int

SET body temperature	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/ bodyTemperature/{bodyTemperatureValue}

GET body temperature	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/ bodyTemperature

We also added to the context model the following attributes in order to model the medication data coming from the IBM application.
The purpose of the IBM application is to improve medication adherence, and thus to prevent secondary complications and medication-related safety risks. The application will assist formal and informal caregivers in monitoring medication intake, assessing adherence and persistence to treatment as well as offer elderly citizens personalized guidance in self-management of medication intake in the form of reminders, education and motivation.
The main functionalities are to allow elderly patients and caregivers to manage multiple medications, and to use the technology that is developed in this project to create a personalized experience for each of the stakeholders.
The first group of attributes model the medication planned event and they are available following this path in the authoring tool: user -> activity - >behaviour - >medication - > medication planned. The names of the attributes are: Notification time, medication name and dosage.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

The second group of attributes model the medication occurred event and they are available following this path in the authoring tool: user -> activity - >behaviour - >medication - > medication occurred. The names of the attributes are: Registration time, medication name and dosage.

These two groups of attributes model when a medication should be taken (medication planned) and if and when a medication has been taken (medication occurred).

We developed the following REST services to update the medication planned/occurred attributes:

SET medication planned	
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/medication_planned"
Supported method	POST
Data Type:	JSON

Data Format:	var medicationObj = {
	"notification_timestamp": 1506073998,
	"notification_time": 2017-09-22T09:53:18,
	"medication": "aspirine",
	"dosage": "100mg"
	};

GET medication planned:	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/medication_planned
Returned Data Type:	JSON

SET medication occurred:	
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/medication_occurred"
Supported method	POST
Data Type:	JSON

Data Format:	var medicationObj = {
	" registration_timestamp": 1506073998,
	" registration_time": 2017-09-22T09:53:18,
	"medication": "aspirine",
	"dosage": "100mg"
	};

GET medication	

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission | Ministero dell'Istruzione dell'Università e della Ricerca | FCT Fundação para a Ciência e a Tecnologia | Forskningsrådet | Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra

occurred:	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/medication_occurred
Returned Data Type:	JSON

The Remote Assistant application developed by Reply addresses mainly elderly people with a good degree of independence with the goal of preventing decline rather than safety purposes, and therefore includes functionalities related to fitness and well-being as well as support for monitoring vital parameters such as heart rate and body position.
The benefits of a physically active lifestyle are well known and demonstrated, however older adults often conduct sedentary behaviour. Motivation is a critical factor in supporting sustained exercise, which in turn is associated with important health outcomes.
Knowledge about personal factors influencing motivation and exercise involvement can be used to tailor interventions to the individual and to improve long-term adherence to regular physical activity.
The application combines the main individual motivation factors, assessed using the EMI-2 questionnaire [1], with context information detected through Plux BITalino biosignal sensors. The goal is to dynamically adapt the application accordingly in order to reinforce the motivation of the specific users and to encourage them in the most effective way to exercise.
In order to achieve this goal, we extended the context model adding the user motivation entity; the path inside the Authoring environment is the following: User -> Activity – Motivation. The possible values of the motivation attribute are: wellness, health, social and fitness.

SET motivation	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/motivation/{motivationValue}

GET motivation	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/motivation
Returned Data Type	String

In order to improve the Persuasion Module there is the need of store the history of the context entities along the time in order to provide more persuasive messages based on the values the attributes took in a period of time.
Every time the context manager receives an update it stores the new value and update the context entities history. We developed a set of RESTful services to access the history values:

GET N last Values	

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission | Ministero dell'Istruzione, dell'Università e della Ricerca | FCT Fundação para a Ciência e a Tecnologia | Forskningsrådet | Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra

Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/contextEntityName/{entityName}/history/getNlastValues/{getNlastValues}
Returned Data Type	List of last n values

GET get Values On Date	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/contextEntityName/{entityName}/history/getValuesOnDate/{date}
Returned Data Type	List of values

GET get Values between Dates	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/contextEntityName/{entityName}/history/getValuesBetweenDates/date1/{date1}/date2/{date2}
Returned Data Type	List of values

GET get Values from Date to now	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/contextEntityName/{entityName}/history/getValuesFromDateToNow/{date}
Returned Data Type	List of values

GET get Values before Date	
Supported method	GET
REST endpoint	https://giove.isti.cnr.it:8880/cm/rest/user/{user_id}/contextEntityName/{entityName}/history/getValuesBeforeDate/{date}
Returned Data Type	List of values

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3 Authentication Server

In the initial architecture specification document, D1.1.a, we described the main security objectives and identified appropriate security mechanisms to protect both data in motion (when information is exchanged between the platform components) and data at rest (when information is stored within the platform). Based on the analysis, we also identified the main high-level security requirements for the PersonAAL platform that focus on supporting secure (encrypted) communication, full access mediation, strong authentication, usable authentication, and peer-reviewed implementations of security protocols and mechanisms. Since the initial architecture specification, we have been concentrating on providing strong and usable authentication mechanisms relaying on peer-reviewed and trusted implementation of security protocols.

Collecting, processing, storing, and sharing personal information within the platform must be safeguarded by strong and secure authentication mechanisms due to the sensitivity of information and according to the security regulations and standards (described in D1.1.a). One example of strong authentication is a universal 2-factor authentication that uses a standard login credential (i.e., user name and password) with additional pin number, verification code, or security token generated by an external hardware device. However, a high level of security and the usage of additional external hardware devices usually influences and reduces the usability of software, in particular for elderly people.

To support strong but usable authentication within the platform, we implemented a server-based security component (i.e., authentication server) with a set of authentication modules and API calls that ensure data confidentiality and data integrity and that allow only authenticated users, web applications, and platform components to access and process information within the platform. The authentication server is based on the implementation of the state-of-the-art "OAuth 2.0" authentication protocol and hosted on professional and secure hosting platform that provides uninterrupted and continuous operation. The server provides a single sign-on mechanism that allows the use of a single users account across all platform applications and components. To counterbalance a high level of security with usability requirements for elderly users and caregivers, we provide a set of authentication software components and modules that feature a unique and secure log-on interface. The authentication modules are available for all development technologies and development platforms used by the project partners, i.e., Java, JavaScript, PHP, NodeJS, and Android. In addition, the server provides a set of secure API calls available to the platform components. The set of API calls that platform components can used is listed and described in Integrated Platform document (D1.5.a).

3.4 Personalization Rule Editor

The editor to personalize the adaptation rules follows the End-User Development (EUD) paradigm that it is based on the idea that end-users can adapt the software to their own goals.
The Authoring Tool is Web-based and enables the creation of (multiple) trigger-action rules in an intuitive manner. It is flexible in the order in which rules can be created: users can start either from triggers or from actions, and it provides the possibility to reuse previously specified rules as a starting point in order to create new ones.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

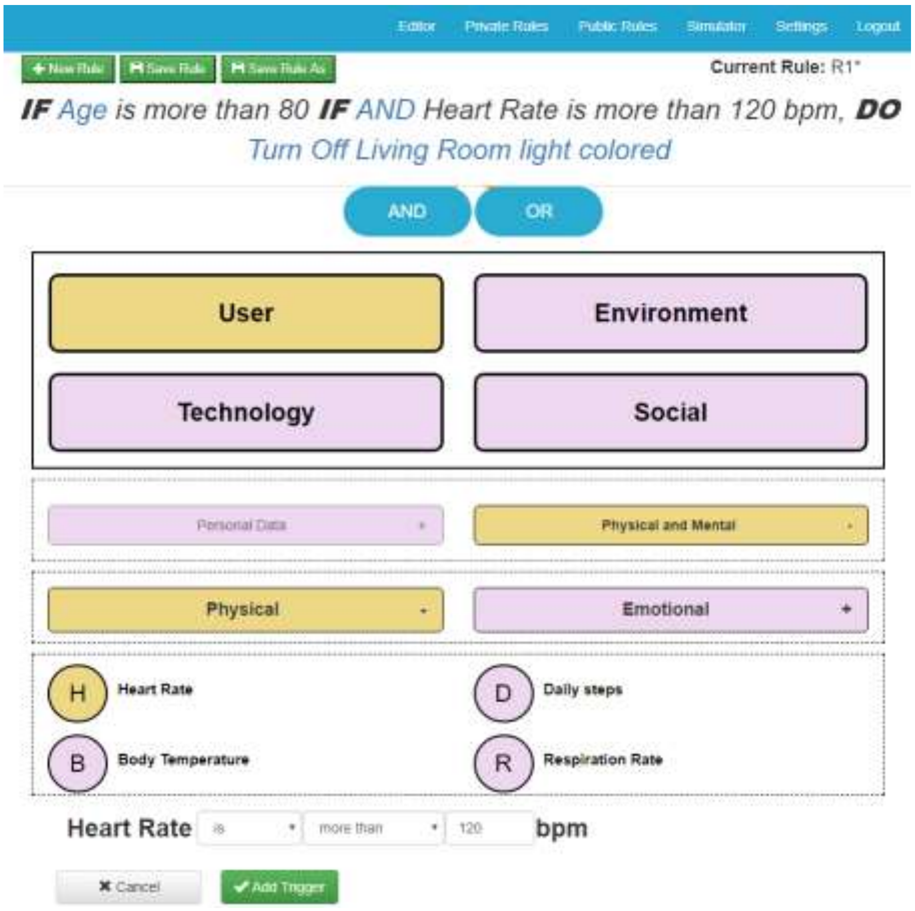
Each rule is composed of two parts: a trigger part, and an action part. Thus, its basic structure is the following one:

IF/WHEN <trigger_expression> DO <action_expression>.

The trigger_expression defines the event(s) and/or the condition(s) that activate the rule application. The action_expression defines the action(s) that should be carried out when the rule is triggered.
Trigger_expression and action_expression can be either an elementary or a composite expression. We also provide the possibility of combining multiple triggers and/or multiple actions. As both the evaluation of events and the evaluation of conditions result in Boolean values, multiple triggers can be combined by using basic Boolean operators, namely AND and OR. It is also possible to use the NOT operator in the definition of conditions. The Actions are sequentially performed.

In the editor, the selection of the relevant concepts is performed by navigating in the hierarchy of concepts associated with each contextual dimension (user, environment, technology, social). For this purpose, each contextual dimension is refined by means of a number of conceptual levels until basic elements are reached. In order to show only the relevant elements to the user, the refinements are presented in an interactive manner (i.e. only the decomposition of the element currently selected by the user is expanded), and highlighting the element(s) selected by the user through a different colour. The selected concepts are highlighted in the tool by the yellow colour.
Figure 6 shows an example in which the yellow rectangles represent the path from the dimension to the event/condition aspect: User -> Physical and Mental -> Physical -> Heart Rate. This was a design choice aimed at reducing the cognitive load for the user (limit the number of elements to be visualised at a certain point, so that users can focus only on the actually relevant ones).
Basic elements are represented differently from the elements that are in the higher levels of the decomposition: indeed, the latter ones are presented by using a rectangular icon with the name of the concerned category inside, while the first ones are shown by using circled shapes with 1-2 for letters identifying them.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.



Current Rule: R1*

IF *Age is more than 80* ***IF*** *AND Heart Rate is more than 120 bpm,* ***DO*** *Turn Off Living Room light colored*

Heart Rate is more than 120 bpm

Figure 6: Editing a trigger expression in the tool

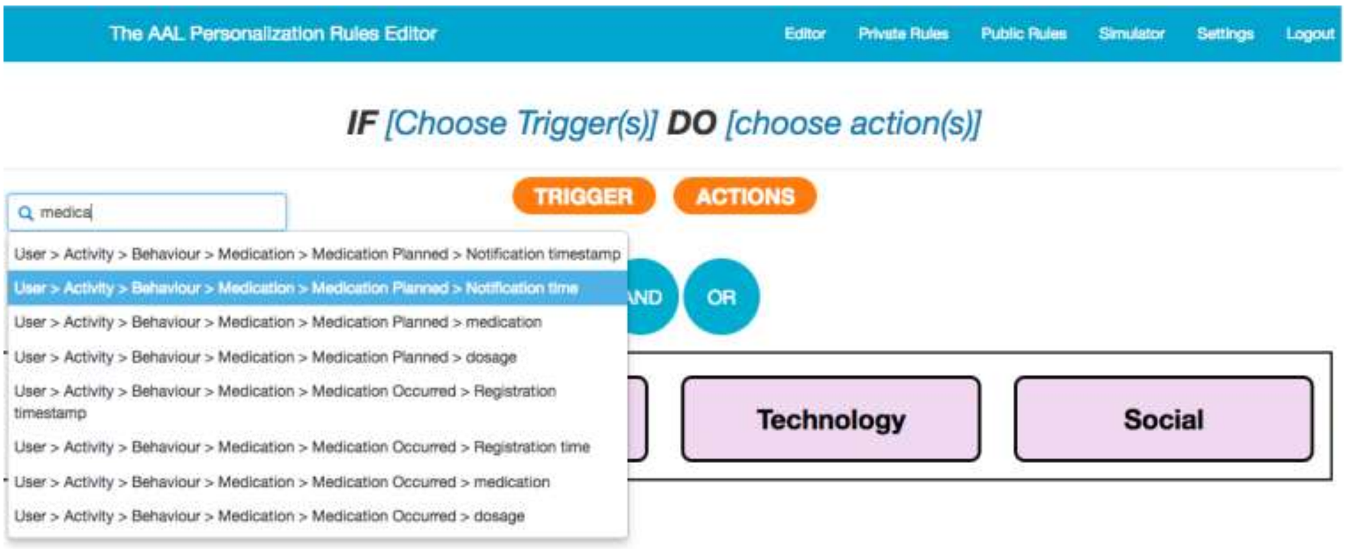
In a similar way users can specify the desired actions. The actions presented in the Authoring tool respect the following classification: Appliance (actions which modify the state of appliances, devices and physical objects), UI Modification (actions which modify the aspect of the user interface elements of the target application), UI Distribution (totally o partially distribute a user interface on to different devices), Functionalities (actions which activate functionalities on the target application), Alarm (actions which send messages that highlight potentially dangerous situations), Reminder (actions which send message to remind something to the user). . When one of the main action types is selected, the tool shows the supported corresponding application-dependent actions.

3.4.1 Search function

The usability test performed in March 2017 by Terzstiftung shows that the structure of the Rule Editor is clear and well-structured, the structure of triggers and actions resulted clear to the users, and it helps the user to get along with its various elements. However, once deep in the tree structure users might get lost. For instance, some of them reported to have difficulties to

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

remember where they can find the next steps, because the whole structure is not visible anymore.
To overcome this problem, we have developed a search function helping users to more easily find a specific hierarchy element without exploring the whole tree (of triggers and/or actions).



User > Activity > Behaviour > Medication > Medication Planned > Notification timestamp
User > Activity > Behaviour > Medication > Medication Planned > Notification time
User > Activity > Behaviour > Medication > Medication Planned > medication
User > Activity > Behaviour > Medication > Medication Planned > dosage
User > Activity > Behaviour > Medication > Medication Occurred > Registration timestamp
User > Activity > Behaviour > Medication > Medication Occurred > Registration time
User > Activity > Behaviour > Medication > Medication Occurred > medication
User > Activity > Behaviour > Medication > Medication Occurred > dosage

Figure 7: The search function to find triggers and actions

As a result of this study a new user interface for the rule editor has been designed and implemented. It supports users in finding the relevant aspects of the context to use in order to specify the triggers in the personalization rules (see Figure 7).
Thus, it is sufficient to enter the desired concept and the tool shows the paths of the elements where it is considered within the context model. Then, the user can easily select the most relevant one among those listed. When the user selects the desired concept, the user interface shows it in the logical structure of trigger classification, and provides users with the possibility to edit the corresponding attributes (see Figure 8).

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission
Ministero dell'Istruzione, dell'Università e della Ricerca
FCT Fundação para a Ciência e a Tecnologia
Forskningsrådet
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

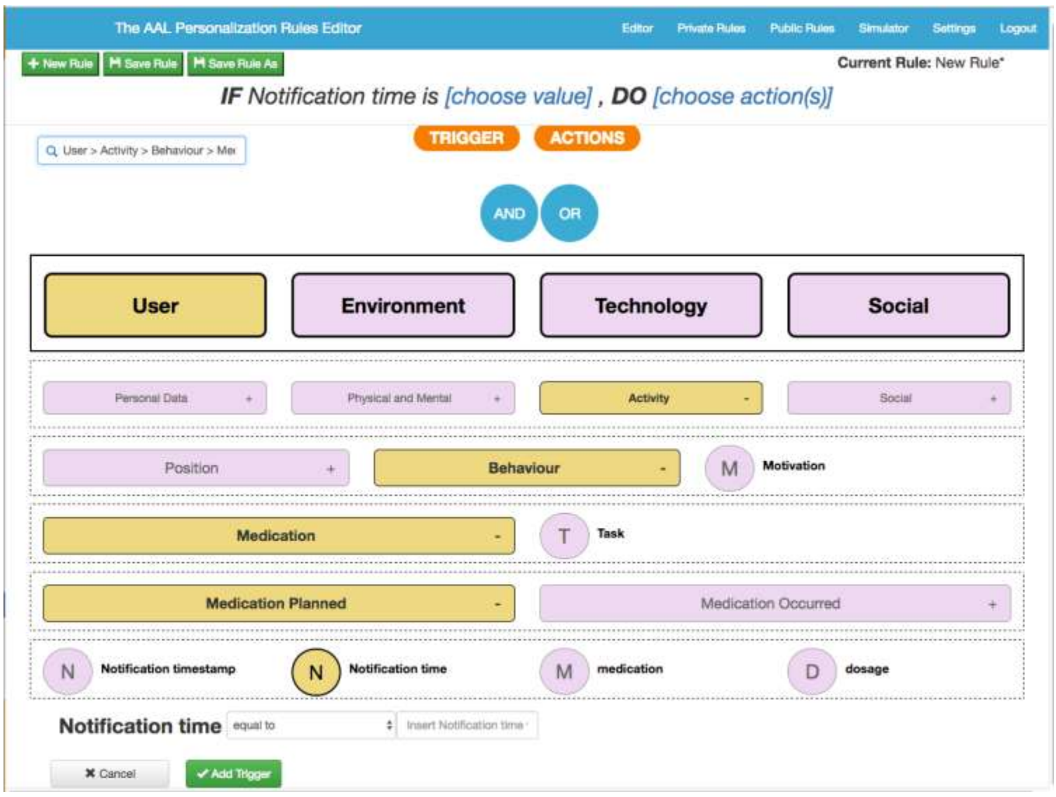


Figure 8: The selected element is presented in the context structure

3.4.2 Internationalization and multiple language support

In order to help out German, Norwegian and Italian partners in performing the user test using the Rule Editor we implemented the support for the internationalization and multiple language feature. The supported languages are German, Norwegian and Italian.
The Spring Framework used to implement the Rule Editor provides an object called Locale Resolver which resolves the language by getting the predefined attribute from user's session.
In order to implement the translation, we provide to the project partners the list of terms that should be translated and then we put them in different properties files (there is one properties file for each supported language):

Language English	messages_en.properties	editor.contextDimension.user=User
Language German	messages_de.properties	editor.contextDimension.user=Benutzer
Language Norwegian	messages_no.properties	editor.contextDimension.user=Bruker
Language Italian	messages_it.properties	editor.contextDimension.user=Utente

Normally, a Java properties file is used to store project configuration data or settings and it is composed of a key and a value (key = value).

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

The previous table shows an example of the different files where the key is always the same for each language while the value represents the translation of the considered term.

In the user interface, we put the reference to one of the key listed in the properties file and when the page is requested by the user, the Locale Resolver load the right value from the properties file of the selected language.

Exploiting the internationalization support provided by the spring framework it is very easy to introduce the translated version on a new language; other partners should only translate the term listed in the English properties files and we just have to compile the Personalization Editor with the new language file.

3.4.3 Simulator Conflict detection

The End User Development (EUD) tools such as the Personalization Rule Editor allows users to control and personalize their applications. In this approach one interesting aspect is how people can test and possibly assess whether the modified/created behaviour of the application actually results in the expected outcome. This need is especially relevant in AAL domains, where incorrect behaviour of applications or actuators can eventually even have safety-critical consequences.

Another key point is how people can test the correctness of the rules and possibly identify errors in it, e.g. triggers/actions that they might have forgotten (or inappropriately added) in the current rule specification. A way to reduce the likelihood of errors in the specification of rules is to allow users to simulate the conditions and events that can trigger a rule and the effects that they will bring about. Debugging is the process of finding the cause of the identified errors in the behaviour and fixing/removing them. A promising approach for the debugging part is represented by the Interrogative Debugging paradigm [2], in which the system directly answers “why” and “why not” questions. In this regard, we have also considered previous studies on the use of “why” and “why not” explanations to improve the intelligibility of context-aware intelligent systems [3]

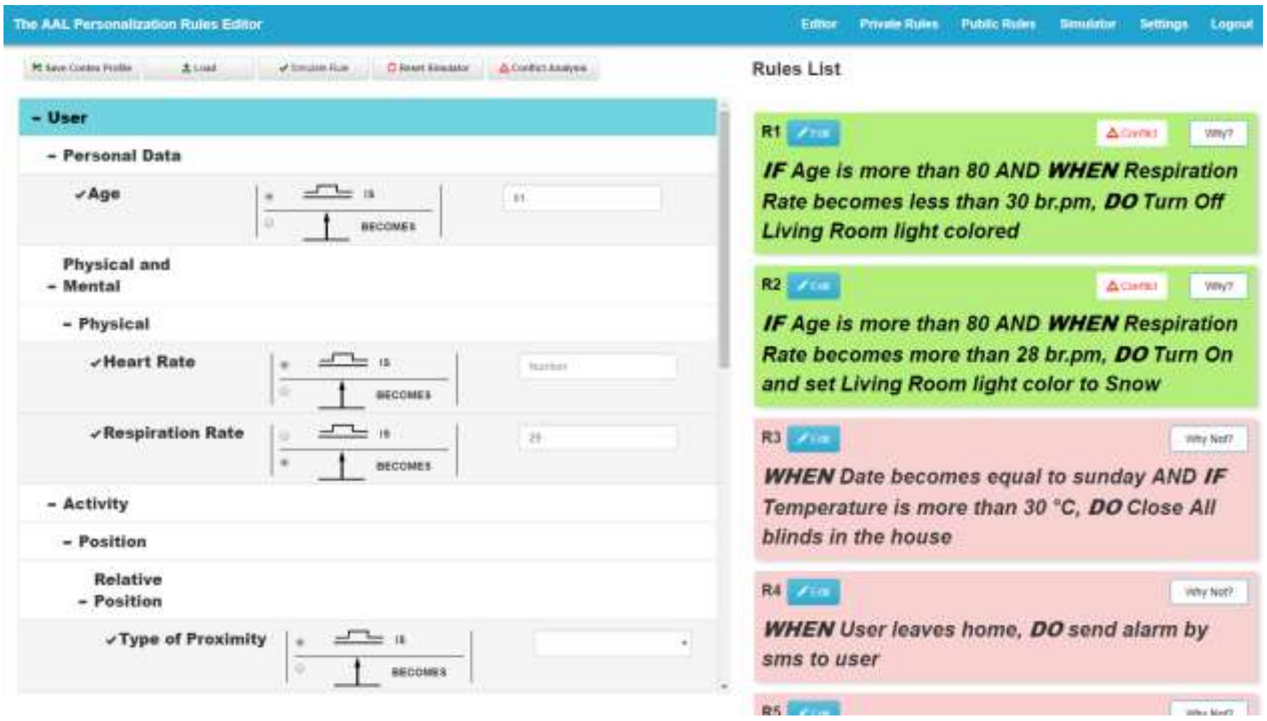
By observing users in various trials specifying trigger-action rules, and considering previous work in the area, we identified some features that an environment to enable users to define context-dependent personalization rules and manage their execution should satisfy:

- It should be able to statically identify conflicting rules. The environment should analyse the defined rules and identify which of them are in conflict (e.g. all the windows in the house should be open in the morning [from 9 a.m.], all the windows will be safely closed in the evening [after 9 p.m];
- it should be able to simulate the rule execution through an interactive user interface;
- it should be able to identify rules which may conflict depending on the values that their triggers can assume at run-time. Rules such as “IF it is raining, the windows should be closed” and “IF time is after 9 am the windows should be open” are in conflict only when both the conditions are verified (it is raining and it is after 9 am).

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

The goal is to allow end users without programming experience to be able to debug the execution of the personalisation trigger-action rules. In such rules one particularly important issue is that often the triggers depend on events and conditions that are not immediately verified (for example, lights should be off at night), thus it is problematic to test whether the specified rules behave as desired. For this purpose, we found important to introduce the possibility of simulating the state of the context of use by allowing users to assign values to the various aspects (e.g. time, location, environmental conditions) and then automatically checking whether the indicated rules are triggered for those specific values of the context of use entities.

The simulator presents the hierarchical structure of the context model in a tree view (see Figure 9). At the beginning the context dimensions are collapsed and only the dimensions and entities involved in the rule triggers are expanded, thus users do not have to search in the context model for the elements involved in the defined triggers.



- User
  - Personal Data
    - ✓Age — IS / BECOMES — 81
  - Physical and Mental
    - Physical
      - ✓Heart Rate — IS / BECOMES — Number
      - ✓Respiration Rate — IS / BECOMES — 29
  - Activity
    - Position
      - Relative Position
        - ✓Type of Proximity — IS / BECOMES

R1: ***IF*** *Age is more than 80 AND* ***WHEN*** *Respiration Rate becomes less than 30 br.pm,* ***DO*** *Turn Off Living Room light colored*

R2: ***IF*** *Age is more than 80 AND* ***WHEN*** *Respiration Rate becomes more than 28 br.pm,* ***DO*** *Turn On and set Living Room light color to Snow*

R3: ***WHEN*** *Date becomes equal to sunday AND* ***IF*** *Temperature is more than 30 °C,* ***DO*** *Close All blinds in the house*

R4: ***WHEN*** *User leaves home,* ***DO*** *send alarm by sms to user*

R5

Figure 9: The environment for executing rules in a simulated context of use.

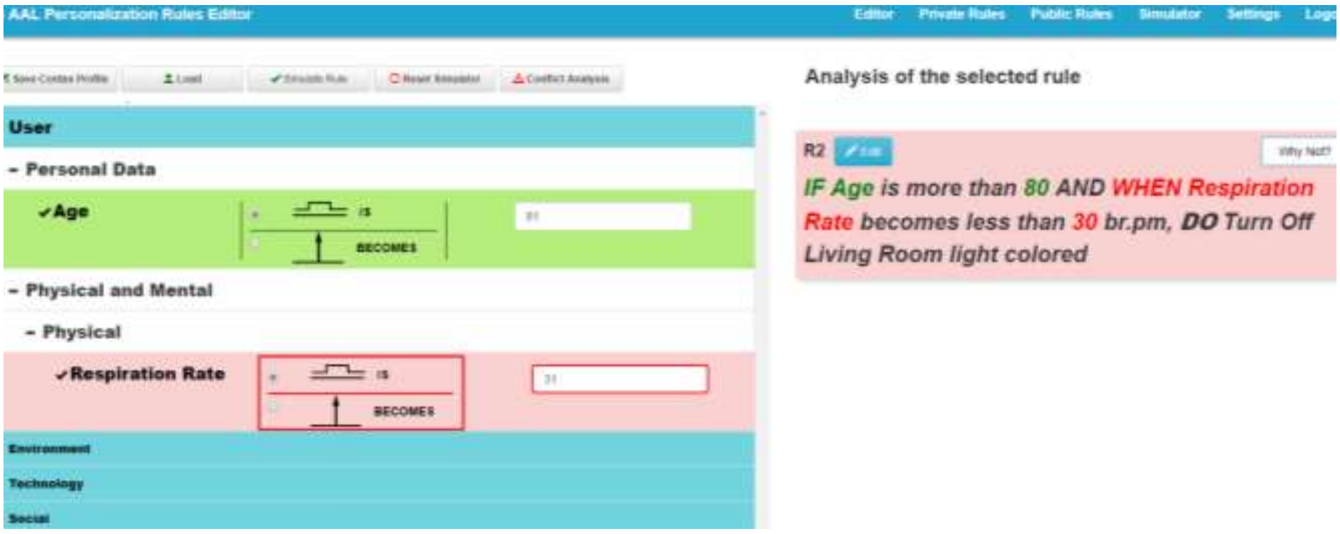
In addition to the trigger values, the simulator also allows users to specify whether the values inserted for each contextual entity are considered as events or conditions. Thus, for each of them there is a radio button with the two possible choices accompanied by a representative icon and the keywords IS (condition) and BECOMES (Event). The simulator helps users to understand the difference between events and conditions; in fact, the rule will be indicated as verified only if the user specifies the right option between event and condition in addition to providing the correct values.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission
Ministero dell'Istruzione, dell'Università e della Ricerca
FCT Fundação para a Ciência e a Tecnologia
Forskningsrådet
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Figure 9 shows the simulator user interface: on the left part of the interface there is the editable simulated context model, while on the right part there are the rules indicated by the user. The simulated context model has the following values: "Age is 81" (condition) and "Respiration Rate becomes 29" (event). When the user selects the Simulate Rule button, the tool checks whether the defined rules will be triggered when the state of the context is that indicated on the left side.

The results are shown by changing the background colour of the verified rules to green, while the colour of the rules that are not verified becomes pink. Moreover, as a result of the simulation some buttons are displayed in each rule container: the first one allows users to edit a rule directly from the simulator, another one is the button "Conflict" which is displayed only if the rule is verified and it conflicts with other rule(s). The last button is the "why/why not" button: the why button activates the highlighting of the contextual elements that are verified, the other one points out why the rule is not verified.



IF Age is more than 80 AND WHEN Respiration Rate becomes less than 30 br.pm, DO Turn Off Living Room light colored

Figure 10: Simulator. Why Not Example

Figure 10 shows how the interface changes after clicking on "why not" button: only the considered rule is displayed and only the context elements related to the rule are shown and expanded in the tree representation of the context model on the left side. The verified triggers are highlighted in green (both on the context tree and in the natural language on the right part) while the not verified ones in pink. For example, if we consider the Respiration Rate element inFigure 10, the trigger which includes this element is verified WHEN the heart rate becomes less than 30 bpm, while in the context description for respiration rate there is the IS operator along with the value 31 indicating a stable condition: the simulator highlights these wrong values through a red border in order to indicate the reasons why the current rule fails.

In rule-based environments it is possible to have potential conflicts in the effects that the rules can have. In this case, conflict resolution strategies should be put in place able to identify potential conflicts and help find possible solutions, even by involving users in this process.

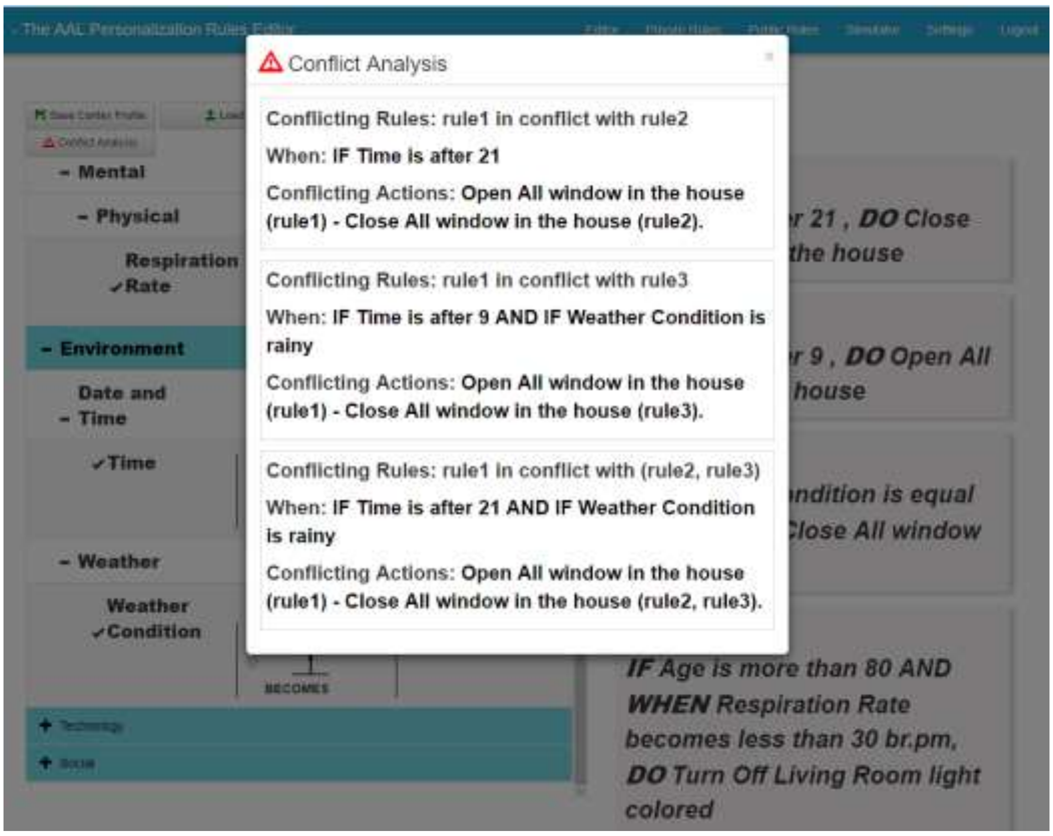
The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

European Commission
Ministero dell'Istruzione, dell'Università e della Ricerca
FCT Fundação para a Ciência e a Tecnologia
Forskningsrådet
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

If we consider the following rules

- rule1: IF Time is after 9, DO Open All window in the house
- rule2: IF Time is after 21, DO Close All window in the house
- rule3: IF Weather Condition is rainy, DO Close All window in the house

we can observer that these rules are in conflict because rule 1 opens all the windows while rule 2 and rule 3 close them all.



Conflicting Rules: rule1 in conflict with rule2

When: IF Time is after 21

Conflicting Actions: Open All window in the house (rule1) - Close All window in the house (rule2).

Conflicting Rules: rule1 in conflict with rule3

When: IF Time is after 9 AND IF Weather Condition is rainy

Conflicting Actions: Open All window in the house (rule1) - Close All window in the house (rule3).

Conflicting Rules: rule1 in conflict with (rule2, rule3)

When: IF Time is after 21 AND IF Weather Condition is rainy

Conflicting Actions: Open All window in the house (rule1) - Close All window in the house (rule2, rule3).

Figure 11: Conflict Detection and Analysis Functionality

We developed a new tool in order to support users in detecting the conflicts between the rules and Figure 11 shows an example of the output produced by the conflict detection functionality. As we can see three conflicts are detected in the example. The results of the conflict detection execution show three important pieces of information: the name of the conflicting rules, when they conflict, and the actions which cause the conflict.
In the example above, the first and the second rules conflict when time is after 21. The first and the third rules presented before conflict when time is after 9 and it is rainy. The first rule is in conflict with the second and the third ones when time is after 9 and it is rainy.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

4 COMMUNICATION ACROSS PLATFORM COMPONENTS

4.1 Personalization Data flow diagram

Below is the data flow diagram of the communications between the modules that compose the personalization platform.

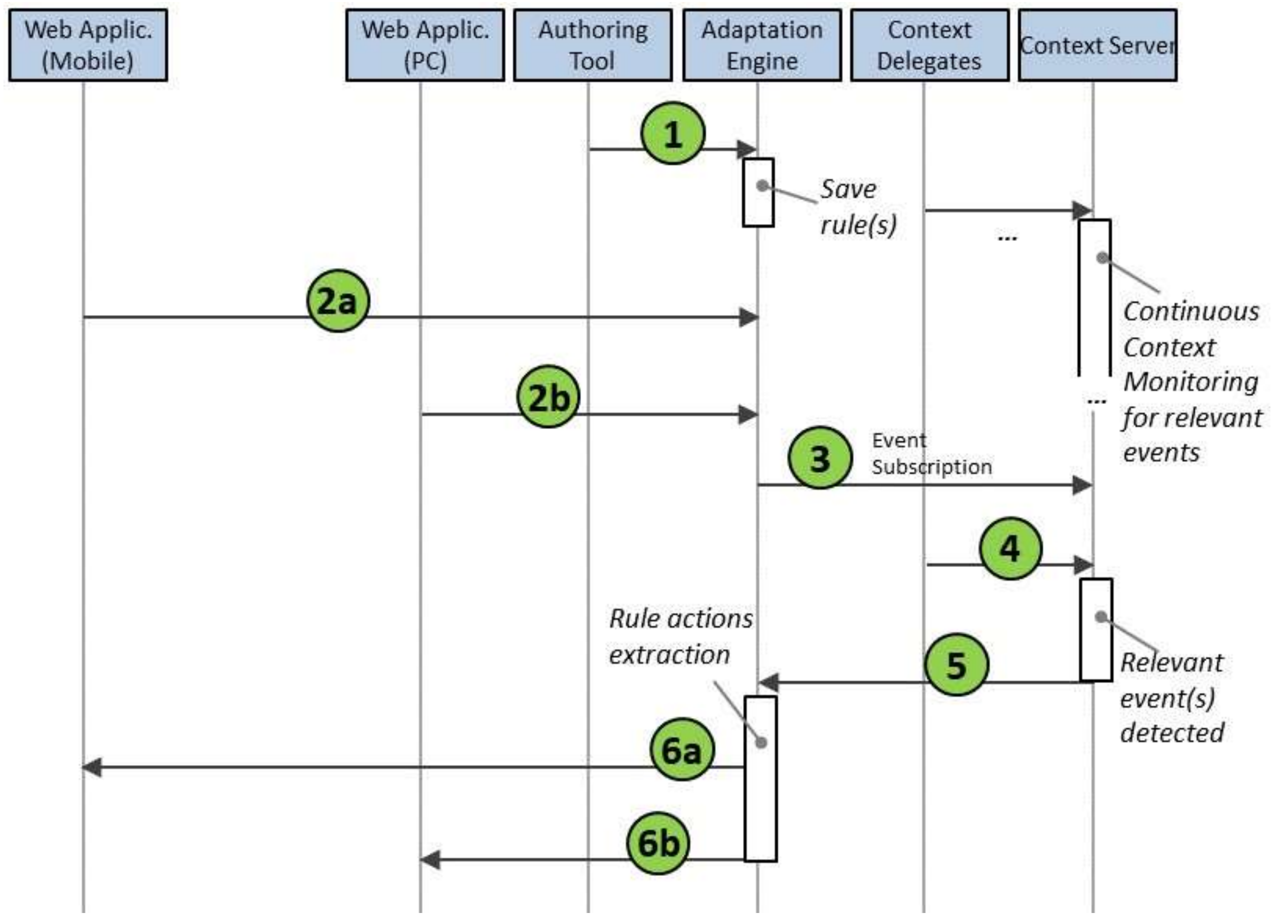


Figure 12: Personalization Data flow diagram

The rules created through the Authoring Tool are saved on the editor and then are sent to the Adaptation Engine module (1). Afterwards, when the application is deployed it subscribes to the Adaptation Engine (2a and 2b), which is in charge of subscribing to the Context Manager in order to be informed of the occurrence of the events or condition defined in the rules (3). Sensors or external services continuously update the Context Server by providing actual data

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Ministero dell'Istruzione, dell'Università e della Ricerca
FCT Fundação para a Ciência e a Tecnologia
Forskningsrådet
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

(4). For each previously subscribed event and condition that are satisfied in the current context, the Context Manager notifies the Adaptation Engine (5), which extracts the list of actions from the verified rules and pushes them to the application (6a and 6b) which interprets and applies the received actions.

4.2 Authoring Tool data flow diagram

The authoring tool carries out some specific interactions with the context manager. When the authoring tool is used by a user, the description of the domain specific context model is loaded from the context manager. Then the context model hierarchical structure is used to display the possible triggers, and users can interact with the editor. The context model defines special elements (e.g. devices and physical object) called "dynamic elements" by specifying only their structure and the attributes that characterize them (e.g. state or location) and by indicating that their instance (i.e. the current devices that are currentely present in the context) can be retrieved by querying the context server. When a user clicks on such dynamic elements the authoring tool queries the context manager in order to receive the list of dynamic elements that are registered on the context manager at run-time.

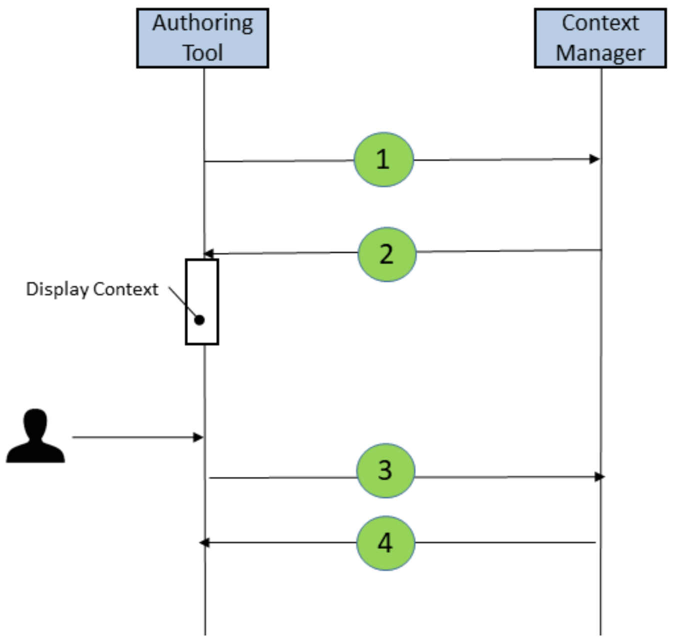


Figure 13: Authoring Tool Data flow diagram

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

5 CONCLUSIONS

The platform has evolved during the project, mainly in terms of refinements of its functionalities to better support the applications developed. The authoring environment for specifying the personalization rules can be accessed in four languages (English, Italian, German, and Norwegian) to facilitate its use in the user tests and trials, and it has some debugging functionalities that help users in understanding whether the specified rules indicate the expected behaviour. The adaptation engine can send actions to REST endpoints and not only use Web sockets. The adaptation rules can be associated to different users for the same application. In addition, there is a page to access the adaptation engine that shows whether and when the rules have been triggered, and provides the possibility to generate manually the trigger. In the rules it is possible to manage actions that refer to dynamic values of the context (e.g. the number of steps performed in the day). The context manager has been extended to support gathering of various physiological, medication and motivation attributes.

Thus, the platform seems sufficiently solid for supporting the trials planned for the last year of the project.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

6 REFERENCES

1. David Markland, and Lew Hardy. 1993. The Exercise Motivations Inventory: Preliminary Development and Validity of a Measure of Individuals' Reasons for Participation in Regular Physical Exercise: Personality & Individual Differences, 15, 289-296
2. Andrew J. Ko, and Brad A. Myers (2005). A framework and methodology for studying the causes of software errors in programming systems. Journal of Visual Languages and Computing, 16(1), 41–84.
3. Brian Y. Lim, Anind K. Dey, and Daniel Avrahami, Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems, Proceedings CHI 2009, pp. 2119-2128, ACM Press, 2009.

The project PersonAAL is cofounded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.