TAREME Platform

Website

Co-funded by the AAL program

Criteria for open platform in AHA and AAL domains which TAREME complies with:

Open Source

Open Standards Based

Shared Common Information Model

Federatable

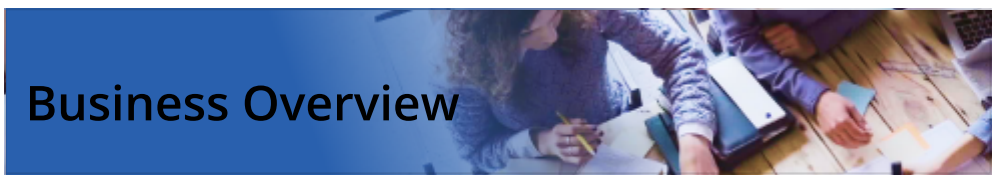
Vendor Technology Neutral

Supports Open Data

Open APIs

Open Usage

AHA Experts' Voices: Dr. Fabio Paternò on the impl...



Business Overview

TAREME (Trigger-Action Rule Editing, Monitoring, Executing) platform provides support for executing and analysing personalized automations in Internet of Things scenarios. The platform allows the creation and execution of trigger-action personalization rules that can change the state of connected smart objects and devices, send alarms or reminders, and modify applications' state depending on contextual events. The execution of rules is supported by a middleware infrastructure able to manage and detect when contextual triggers generated by available sensors and devices are verified. Analytics is supported through a tool providing information about the personalization rules executed in users' contexts, which can help in better understanding their actual use and personalization preferences.

The personalization platform is not domain-dependent and can be applied to different types of scenarios. This platform also includes also a Tailoring Environment that enables end-users to specify expressive trigger-action rules with various possible compositions of triggers and actions and with a clear distinction between events and conditions defining triggers, in a way that is understandable by end-users.

Target Scenario: Ambient Assisted Living

Remote monitoring: Remote monitoring services and health related interventions are strongly personalized to specific individuals' requirements, preferences, abilities and motivations, which can radically differ among the older persons, and even dynamically evolve over time for the same person depending on changing user needs and context-dependent conditions.

Establishment of rule: A caregiver (or even an older adult) can set up a rule that better monitors what the older adults do during the night, or whether their expected physical and/or cognitive exercises are performed during the day.

A rule can switch on a light for a specific interval of time so that the bedroom-bathroom path can be better illuminated when motion is detected during the night. Another example is a rule that changes the light's colour according to the current user's emotional status since lights can play an important role in the older adult's mood.

Remote analysis: In addition, in such contexts, caregivers can benefit from a rule monitoring tool that allows them to understand which kind of rules have been triggered in a specific period, to better understand the automations that have more frequently executed during the older adults' daily life. For instance, high use of reminders can be interpreted as a sign that the older adult is increasingly relying on external support to remind the various activities to carry out, which might need further investigation in some cases.

Target Scenario: Smart Home

Customized automation: Great variety of activities occur in the domestic environment and people's interactions with the devices and objects available in the household. This emphasizes on the need to support the easy customization and personalization by end users. An example of customized automation is creating a rule that notifies parents when children are back home from school and activates the external surveillance system in the house.

Another rule could check gas leaks in the kitchen, and in such case, an alarm can be directed to alert inhabitants currently at home (as well as other people that can provide help), and at the same time activates the automatic opening of the kitchen's windows. In this case, a rule monitoring tool can be useful to better understand the kinds of automations that the family members are most interested in, i.e. whether they regard controlling the equipment at home, or whether they involve triggers more related to inhabitants' safety (e.g. presence of smoke or gas) or health (heart rate, time spent in the bathroom).

Learn more about TAREME Platform

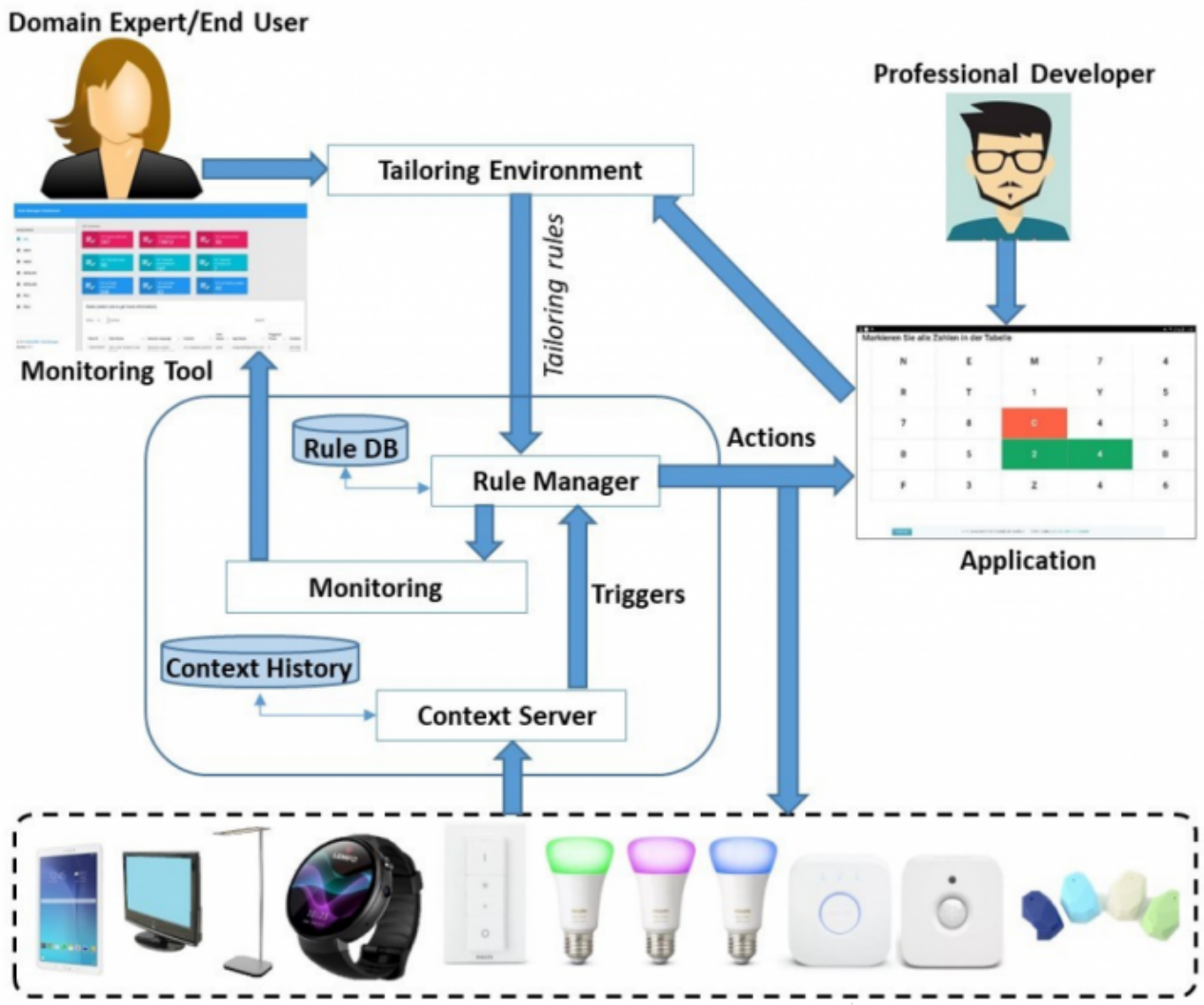
Technical Overview

Platform's architecture

The architecture of the considered software platform includes a Tailoring Environment through which even people without programming background (e.g. domain experts, end users) can specify the desired personalization rules. The Tailoring Environment sends such rules to a Rule Manager, which receives information from the Context Manager when the triggers involved in the rules are verified: when this happens, the Rule Manager sends the actions that should be executed by target applications and appliances.

The Context Manager is a software composed of one server and various delegates. The purpose of the Context Delegates is to communicate directly with the various sensors or other entities able to generate events and be informed when their associated variables change. When this happens, the Context Delegates communicate such changes to the Context Server, according to a pre-defined vocabulary used to specify the triggers.

There is also a Monitoring module which aims to provide support for analysing the use of the personalization platform, by showing relevant information to users interested in it (e.g. in Ambient Assisted Living scenarios they can be caregivers or platform managers). For gathering the relevant information, the Monitoring module receives data from the Rule Manager concerning the rules, their state and when they have been executed. This module is a key added value because it allows relevant stakeholders to focus on the personalization that has actually been put in place by users, thereby of actual interest for them. Applications can be integrated with the platform in order to: receive actions (which were included in relevant rules) indicating requests of modifications to make, and/or send events generated by the application itself (in this case the application acts as a Context Delegate, by sending the information associated with the occurred event(s) to the Context Server).

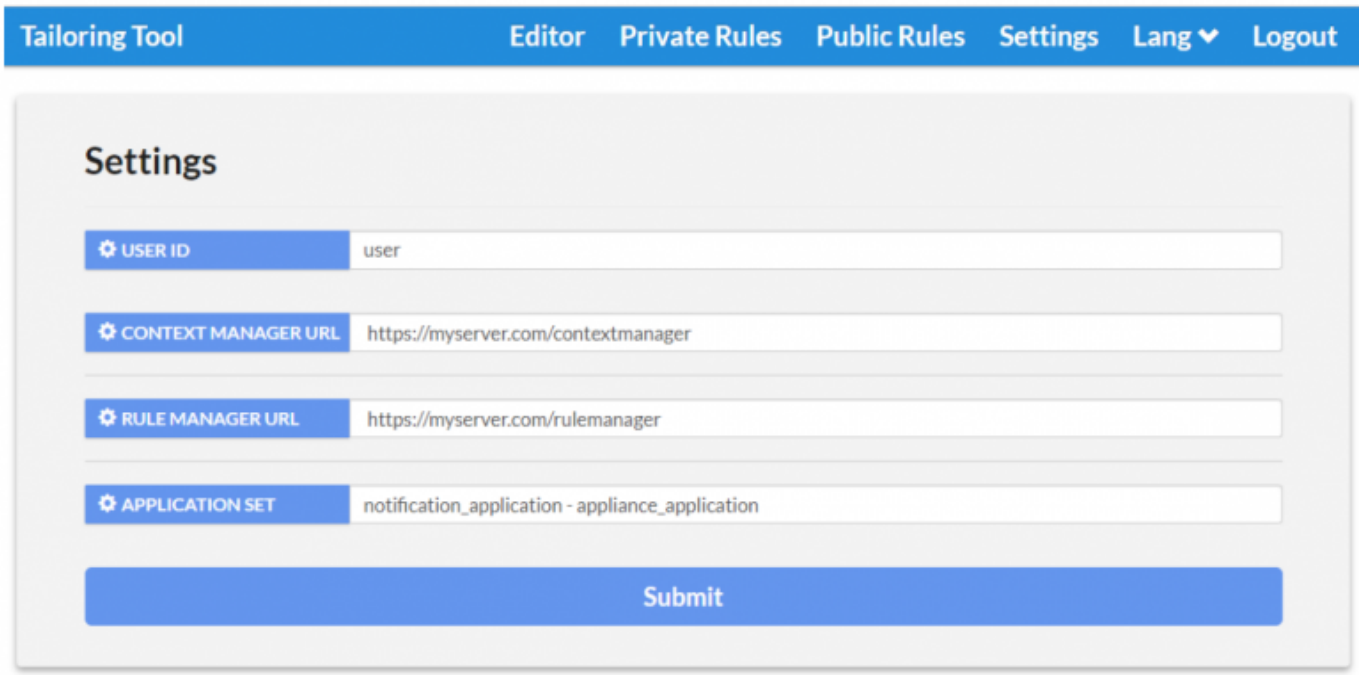


An Overview of the Main Components of the Platform and Their Communications

The Tailoring Environment

The Tailoring Environment is the EUD environment through which users specify personalization rules. On the one hand, it shows relevant triggers, which describe the main aspects that may change in the context, categorized under a hierarchical representation having three main dimensions (Users, Environments, Technology) at its highest level.

On the other hand, it also provides users with a structured representation of actions, mainly depending on the specific application(s) considered. The Tailoring Environment is designed to be a generic environment which can be easily configurable, making its customization to support specific applications, domains and contexts an easy task.



The “Settings” Panel of the Tailoring Environment

“User id” specifies the name of the user who is currently using the Tailoring Environment. This information is important to correctly associate the rules created through the Tailoring Environment with the correct user;

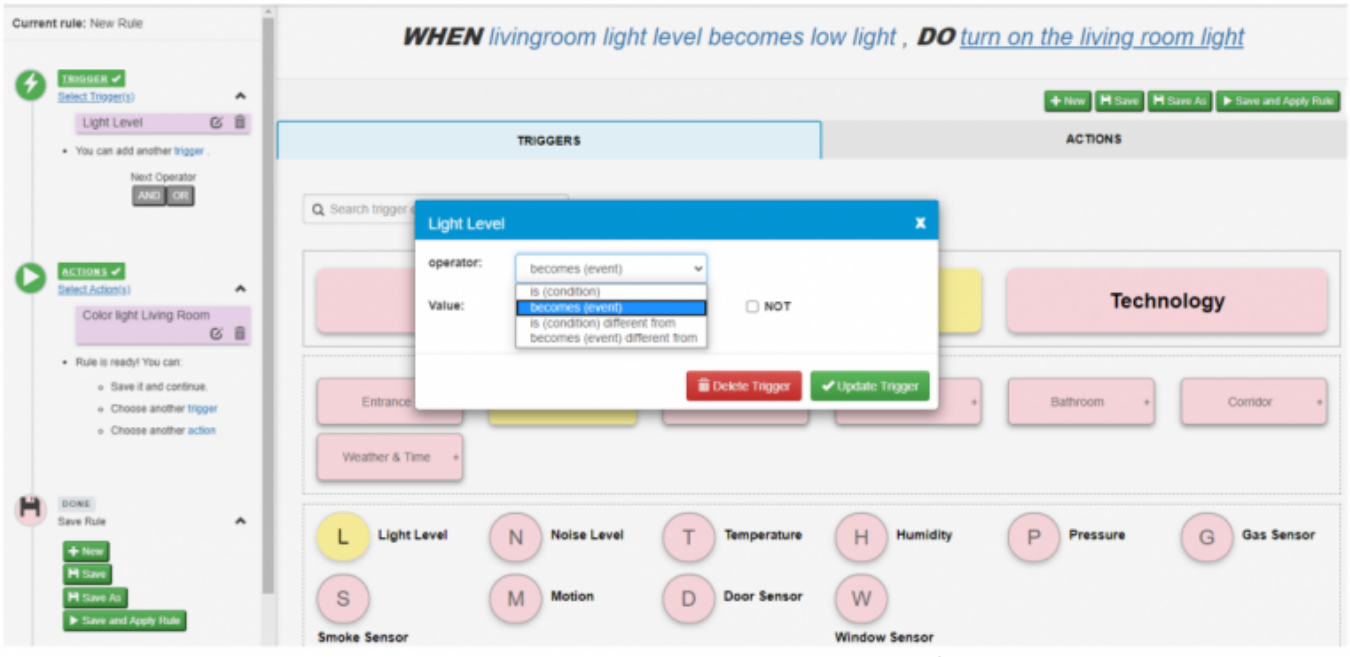
“Context Manager URL” specifies the URL of the specific instance of the Context Server used. This information is essential to identify the triggers that the Tailoring Environment has to show, which depend on the considered context;

“Rule Manager URL” specifies the URL of the Rule Manager. This information is key to understand the URL to which to send the verified rules for their execution.

“Application Set” specifies the name of the application(s) considered. This information is important to identify the actions that can be executed.

The Tailoring Environment also supports the explicit distinction between events and conditions when specifying the trigger. This distinction was introduced considering previous user studies, which have shown that EUD environments sometimes confuse end users concerning the difference between these two concepts. Thus, by introducing the explicit distinction, one can better stimulate users in thinking about their difference. In addition, the top part of the central panel of the Tailoring Environment provides feedback in natural language of the created rule, to express the specified behavior in a more immediately understandable manner.

Moreover, it is possible to define rules that are triggered if an event does not occur in a specific interval of time (e.g. When the medicine has not been taken between 10 a.m. and 11 a.m.), which were found useful in some scenarios. This has been supported thanks to the introduction of the NOT operator associated with an event in the proposed tool, differently from other solutions that do not support it (e.g. IFTTT). Another useful feature of the rule language used is the possibility to define rules that are triggered when a specific ordered sequence of events occurs (e.g. “If the user enters inside a room and then he exits”; or “When the temperature becomes more than 20 degrees and then the humidity level becomes more than 50%”), or when an event occurs a specified number of times (event iteration), e.g. “If the user goes to the bathroom 5 times during the night”.



Distinguishing between Events and Conditions in the Tailoring Environment

[Learn more about the Panel of the Tailoring Environment]

The Context Manager

The main task of the Context Manager is to maintain an up-to-date picture of the current situation of the considered context of use (e.g. the house of a specific user), and to inform the other modules of the platform -the Rule Manager in the first place- when relevant updates to such a context occur, i.e. when an event happens or a condition is fulfilled. Therefore, the Context Server communicates with the Rule Manager only when the triggers specified in a rule are verified. The Context Manager is a distributed module, composed of a Context Server and some Context Delegates.

The Context Delegates are software components that communicate with the sensors and the appliances available in the considered context: they get raw information and send it, in a suitable format, to the Context Server, by using a RESTful service that this module exposes to this aim. In turn, the Context Server, according to the data received from the Context Delegates, updates a database storing the current and past values (historical data) of the attributes that the various elements considered in the concerned context dynamically assume over time, and represented using a common format.

More specifically, the description of the types that such contextual entities can assume, and the hierarchical structure in which they are organised are represented in a context meta-model specified in an XSD format. When the Context Server is compiled, this XSD file is automatically translated into a set of Java classes, and various instances of Java objects are created to define the state of the elements composing the current context (e.g. the various instances of users, environment and technologies). In order to update such Java objects, the Context Server provides just one RESTful service to receive the data from the various Context Delegates which, using this REST service will be able to update the various attributes of the context. The input

parameters of this service aimed to update the information received from the Context Delegates are: i) the id of the corresponding context dimension; ii) the xPath value indicating where the context attribute to update is located within the context model structure; iii) the new value gathered from the sensor.

For example, the trigger representing the event “user has fallen” is defined in the context model as an attribute of the “User” dimension called “layingDown”, and it assumes a Boolean value: this trigger is defined under the following hierarchy: “user -> physical -> laying down”. To update the layingDown attribute, the context delegate sends three parameters to the REST service: the user id, the xPath of the attribute to update (in this case “user / physical / @ layingDown”) and the current value of the attribute (in this case “true”). The advantage of this single-service solution is in terms of platform evolvability. Indeed, when there is the need of updating the context meta-model (because of a new context element type, or an existing context element type has been changed, or removed), it is sufficient to modify the XSD definition of the context metamodel, since the REST service above mentioned can still be used without requiring any further changes.

This is because: i) the JAVA classes that are expected to manage the values of the attributes of the updated context are automatically generated when compiling the Context Server; ii) the above-mentioned RESTful service will still be able to support changing the values received for the newly generated context element(s). For instance, suppose that there is the need to introduce a new attribute providing information about user movements (i.e. “isMoving”), and this attribute needs to be added under the “user/physical” xPath hierarchy (namely: “user / physical / @isMoving”). In this case, while the XSD describing the structure of the context needs to be modified, there is no need to create a new, ad-hoc service to receive and update values of this newly introduced attribute, because the RESTful service described before can still be used to change the value of the concerned “isMoving” attribute.

This is realised through the JAVA reflection mechanism (namely: the capability of an executing Java program to examine upon itself, i.e. obtain the names of all the fields or methods of a JAVA class), which makes it possible to automatically derive, within a Java program, the name of the method to call (by using the information included in the parameters mentioned above), and consequently update it. This type of solution is modular and evolvable because it can easily manage the flexible introduction of various sets of sensors and applications, according to the specific needs to address, in a dynamic manner.

The Monitoring Tool

The main goal of this module is to provide its users with information about what happens in the end-user context (e.g. seniors' homes), both in terms of activities done by the users (as detected by sensors), and in terms of personalizations that have been put in place through the specified rules. To this aim, this module receives input from the Rule Manager, specifically the personalization rules that have been sent for execution, and the time when they have been actually triggered.

The information about triggered rules can represent valuable data to understand what is currently and actually going on in one or more end-user sites, to identify the personalization aspects which users are focusing on most, the types of routines they have put in place and the frequency with which such automations occur.

Indeed, users of the Tailoring Environment could create/add several rules in their repositories over time. Thus, some rules may appear in the repository, but they are not currently exploited because the users did not want to have them active at the time. In general, the information about the triggered or active rules provides data that have been of actual interest for their users. The monitoring tool can also provide more detailed information, such as how many times a specific rule has been triggered.

This value is highly dependent on the purpose of each particular rule. In the following, we will better detail the information provided by the Monitoring Tool, which displays various types of structured information. Therefore, to facilitate its description, the project team schematized its layout as structured into four main parts. The content of such four parts are further detailed in the following four sections.

Learn more about the layout of the Monitoring Tool and the platform configuration

Contextual Overview

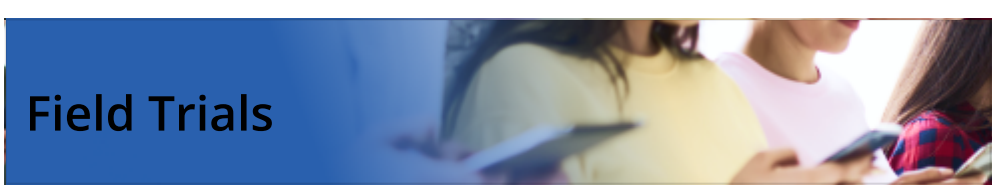
This section provides background information to better introduce the context of the research which was carried to support the development of the platform. The design of the platform goes back to the concept of an automatic environment for specifying personalization activities in IoT scenarios through trigger-action rules. That environment just allowed end users to create, modify, save, delete and reuse trigger-action rules. Such initial architecture, over the

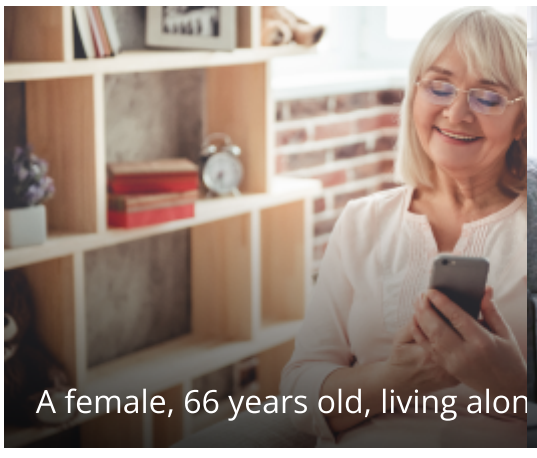
years, significantly evolved from a first ideation to a concretely working solution supporting the actual execution of the actions contained in rules, and even providing ‘smarter’ support in some specific cases. According to the evolution of the platform, also the included environment for managing rules significantly improved.

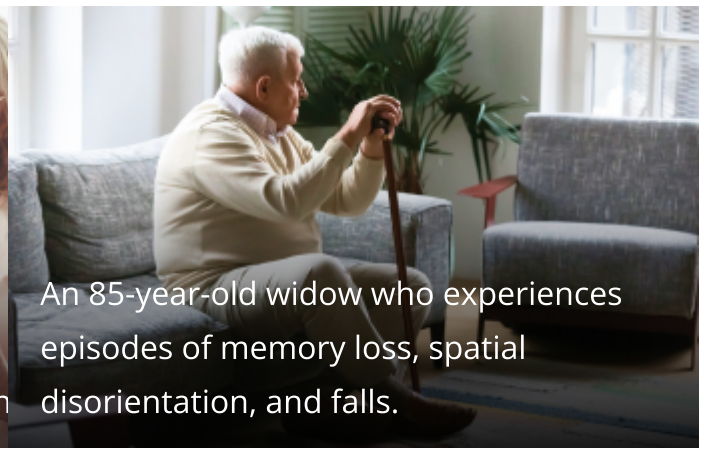
Despite some promising results derived from usability studies involving EUD tools, it became soon clear that even providing users with usable EUD tools, without proper support, they may easily define rules triggering actions that are in conflict, or not resulting in the intended behaviour. Therefore, the developed solution supports end-user debugging the simulation of the context in which trigger-action rules are expected to be executed. In particular, the platform supports functionalities not only for simulating rules but also for ‘debugging’ them by providing conflict resolution support as well as interactive explanations.

Considering the increasing availability of humanoid robots in various domains of everyday life the platform enables those who are not programming experts to personalize robot behaviour according to contextual information detected by both the robot and the available IoT objects and sensors available in a specific context, and in this way to link the robot behaviour to what happens around it.

Learn more about the TAREME platform







A male, aged 81, who was quite autonomous in the management

An 83-year-old widower who had diabetes, walked with difficulty and took several medicines

An 78-year-old widow, living alon

An 80-year-old widow, who suffered from heart rate alterations.