Journal of Visual Languages and Computing (xxxx) xxxx–xxxx

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

A visual language and interactive system for end-user development of internet of things ecosystems

Barbara Rita Barricelli*, Stefano Valtolina

Department of Computer Science, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milan, Italy

ARTICLE INFO

Keywords:
Internet of things
Event detection
Unwitting developers
Sociotechnical design
End-user development
eWellness
Visual language
Visual interactive system
Rule Editor

ABSTRACT

This paper presents the definition of a visual language and its implementation with the design of a visual interactive system for the collaborative management of Internet of Things (IoT) sensors (e.g., wearable fitness trackers, ambient sensors, fitness apps, nutrition apps, sleep trackers) for improving people's quality of life and promoting wellness awareness. The system, called SmartFit Rule Editor, is designed to be used by coaches and trainers of non-professional teams of athletes for monitoring and analyze fitness and wellness data streams and to support them in detecting relevant events and specifying rules for actions taking. Our research is framed under the scope of computer semiotics and semiotic engineering theories. This allows us to study how to support coaches and trainers as a community of domain experts – but not IT and IoT experts – to use elements of a visual language to indirectly manage physical devices and their data streams without the need to know technical specification of the devices, the apps, and the data. We apply a socio-technical approach to design being able to study the social and the technological aspects of the use of the Internet of Things ecosystem, considering them as closely interconnected and dependent. Such an approach underpins user-centered design and development methodologies in order to design the most suitable User eXperience according to users' culture, needs, context of use, and activity.

1. Introduction

Community of domain experts but not IT and Internet of Things (IoT) experts need an effective, easy-to-use and easy-to-learn strategy for managing physical devices and their data streams, without the need to know technical specification of the devices, the apps, and the data. The relationship between domain experts and IT experts has often been fraught with misunderstandings and even frustration. Part of this disconnection between the Communities of Practice [1] is a natural disparity between their work cycles. For a domain expert, the IT system development cycle appears to be a long, drawn-out process. Moreover, once the requirements are explained, it is not easy for domain experts to follow them through the implementation process once they are translated into technical forms. In the developers' defense, what they receive as requirements is often not sufficiently detailed and precise and many requirements are “refined” in the development process. More importantly, once the translation is accomplished, the process for changing a requirement requires going through the same complex consultation and translation process for the change. In several cases, the owners of the requirements (i.e. the domain experts) lose control over their expression and evolution when it is handed off to be embedded in a software solution, and never regain control. To this aim, business rule management systems (BRMSs) [2] enable a great leap forward in bridging the gap between domain and IT experts. Using BRMSs, the domain experts define their policies and rules and provide the clear communication between them and the developers implementing the application system solution. Thus, domain experts have control on exactly how their rules are being executed, and, perhaps more importantly, how the system is designed to facilitate change when business rules change. In traditional information systems, the rules and policies get hard-coded into the application. When changes need to be made, the entire software development life cycle must restart and all its phases need to be repeated: requirements analysis, system design, making sure it does not adversely affect anything else, implementing the change, testing it, and deploying it. With a BRMS, the business logic (again, the part most likely to require change) is separated out and can be changed without impacting the remainder of the application. An assessment of the impact of the change on these rules must still be done, but this is much simpler than a full system impact assessment and it can be done in a more understandable way by domain experts themselves. For this reason, rules should first be specified at the conceptual level, using concepts and languages easily understood by

* Corresponding author.
E-mail address: barricelli@di.unimi.it (B.R. Barricelli).

http://dx.doi.org/10.1016/j.jvlc.2017.01.004
Received 11 February 2016; Received in revised form 26 July 2016; Accepted 8 January 2017

the domain experts who are best qualified to validate the rules. This is particularly true for the quantified-self movement, the result of technology advances in the field of lifelogging, where thanks to the sensors used as small and wearable devices that are affordable and easily interconnectible, it has become a wide spreading phenomenon that sees people to keep track, by using rules, of their habits, health conditions, physiological data, and behavior, and to monitor conditions and quality of the environments in which they work and live. Typical BRMSs such as Drools,[1] OpenRules[2] or IBM Operational Decision Manager[3] offer different solutions for editing, managing and executing rules and in some cases they also provide functions for modeling, in a graphical way, the execution data-flows by applying a set of rules. Moreover, as described in Section 2, the literature reports several solutions for modeling business rules by using UML-Based Rule Modeling Language [3] or standard ontology modeling notation such as RDF or OWL, or other strategies based on Extended Tabular Trees or Decision Tables approach. Anyway, all these approaches require significant knowledge of standard modeling notation, and the programming is still the central metaphor. However, what is necessary to support nontechnical people in keeping track of their daily activities is a new manner for expressing rules that can meet their expectations and decision making attitude. This requirement pushes towards the design of new manners for visually specifying rules at conceptual level by adopting a graphical interaction strategy for expressing rules and related constraints and actions in an easier and more natural way. The emphasis of the interaction needs to be the communication process that takes place among users and the system in order to meet the users' expectations as thus defined in their mental model and tacit knowledge. Our approach, specifically implemented in the context of eWellness domain, aims at helping domain experts, in our case coaches and trainers, in expressing rules based on their working settings by using a set of conditions that if met, have to trigger suitable actions, therefore allowing the establishment of a correct interpretation and semiosis process among users and the system. The IoT panorama that characterizes current eWellness domain faces the problem to monitor a huge quantity of data collected by sensors and services that need to be exchanged together with their users' needs and/or preferences, in order to keep track and influence behaviors and critical situations. In this context one of the main problems is the need to express conditions and spatial-temporal and thematic relations that typically affect the sensors' data-stream management. In general, sensors besides spatial and temporal information provide thematic information in order to discover and analyze data. Thematic domain contains metadata that describe a real-world state from sensor observations, such as a sensor that is used for gather data about calories burned, heartrate, or duration of a physical activity.

Section 2 describes current solutions based on the use of decision tables, decision trees or mashup techniques that are still highly complex and negatively affect the effectiveness of the communication process between users and the system. On the other hand, a visual approach to temporal conditions definition could help the user in successfully expressing them without having to learn specific languages and their syntax. Defining suitable visual representations for temporal structures for rules needs the integration of theoretical and methodological work both from traditional areas devoted to temporal representation (logic, reasoning, and databases) and from the information visualization research field.

Stemming from these considerations, this paper presents the Rule Language we defined and its implementation in a visual interactive system that allows domain experts to unwittingly develop an IoT ecosystem [4–7]. We use the term unwitting developers to identify those people who are motivated in using specific software environments to reach their objectives and thanks to their motivation they are more keen to overcome any eventual difficulty. As reported in [4,5], programming is not these people's goal, instead constructing and deconstructing software objects are. The architecture of the SmartFit Framework is designed on Software Shaping Workshop (SSW) methodology [8], one of the most established methodologies for End-User Development (EUD) [9].

Specifically, this work results in the design of a Rule Editor as part of the SmartFit Framework whose aim is to gather, compute, and diffuse data originated and streamed by physical and social IoT devices, sensors, and applications. The graphical Rule Editor uses a notation able to facilitate team sport members in taking under control their athletes' physical activities and their nutrition and sleep behaviors.

The essential idea of using a semiotics-driven design is to define a new perspective in designing and developing interactive systems to support collaboration in multidisciplinary projects, especially adopting participatory design techniques. The key concept is to involve domain experts in activities for mapping and translating their professional knowledge into proper vocabularies, notations, and suitable visual structures of navigation among interactive systems interface elements. Such an approach to design is aimed at supporting communication among different communities of experts (e.g., IT, domain, HCI, and interaction design) and enabling their collaboration in designing together effective interactive systems for specific application domains.

The paper is organized as follows. Section 2 describes how EUD methods and techniques can be used to empower the end user in the IoT domain and illustrates the state-of-the-art solutions provided by Workflow Management Systems (WfMSs) and BRMSs in supporting nontechnical people in keeping track of their activities and behaviors. In Section 3, the methodology at the base of the design of the SmartFit Framework is illustrated and discussed in detail. Section 4 presents the architecture of the Framework and its systems. The Rule Language is presented in Section 5, while its implementation in the SmartFit Rule Editor is described in Section 6. The evaluation process and its results are reported in Section 7, while Section 8 concludes the paper offering an overview of future works and developments.

2. Related work

One of the key objectives in EUD is to provide people with the capability to create and modify software. In particular, the goal of EUD is to extend that capability to a wider range of people, beyond professional programmers, in a way that will help these people in achieving successful results in their daily activities. EUD represents the ideal approach for empowering end users and make them becoming unwitting developers in their own IoT environment [4–7]. As widely reported in the literature, EUD can be enabled by applying methods and techniques and by offering specific tools that support end users in the development of solutions with limited programming skills and knowledge about programming languages. Specifically, the solutions offered by EUD include tools for the customization of applications by parameters setting, control of a complex device (like a home-based heating system), and even scripting of interactive Web sites [10]. In the case of EUD, we are particularly interested in those who are not professional programmers, but might still engage in some kinds of programming activity. Significant researches have highlighted the differences between end-user programming from professional programming; with an intent-based differentiation: End-user programming produces program for personal use or for their Community of Interest [11]; Professional programming produces program to be used by larger and more generic groups of users [12]. In the IoT context, end-user programming allows users to configure, adapt, and evolve their software by themselves [13] and such tailoring activities, together with personalization, extension, and customization, aim at programming not (only) the user interface and the behavior of an interactive

[1] http://www.drools.org/
[2] http://openrules.com/
[3] http://www-03.ibm.com/software/products/it/odm

system, but embraces the whole IoT scenario of sensors and devices. Therefore, in such context, end-user programming activities need to distinguish according to three different levels on which they can operate: hardware, software, and data. The activities on hardware are those made on the devices via their bundled applications. They typically are configuration, personalization, and customization by setting parameters and choosing among existing behaviors. The activities on software target concern the applications used for controlling more than one sensor/device (even of different brands) and include tailoring by integration of existing and/or new functionalities, macros, visual programming, and programming by examples. The activities that can be made on data can be resumed in aggregation, filtering, and porting. What we are interested in this paper is those last type of activities addressed to supporting non-technical people in managing the workflow of data coming from the set of sensors and services that populated their working environment with special regard to eWellness context.

2.1. Systems for designing workflows

Current WfMSs provide users with various approaches for simplifying the acquisition of programming skills in the field of the workflow design. Commercial systems such as Talend Studio,[4] StreamBase Studio,[5] and Waylay.io[6] are designed for offering programming assistance by using metaphors in graphical interfaces, or by using abstract representations to hide complex programming concepts, or by providing helper codes to avoid syntactic errors. While Talend works on static data coming from fixed datasources, StreamBase and WayLay can receive and analyze continuous data streams and are specifically designed for IoT. Despite their different attitude to analyze dynamic flow of data, these systems adopt typical control strategies that aim at helping users to overcome their cognitive difficulties in programming by offering a graphical interface for designing workflows and data-flows as graphs of connected nodes representing tasks and data sources. These environments provide rich user interface support for the full application lifecycle, spanning feed integration, application modeling, development, data streams recording/playback, testing, and debugging. These environments offer the possibility to define UML-based modeling of data, states, and processes or to use standard ontology modeling notation such as RDF or OWL for representing the business data and relationships against which rules will execute, simplifying solution development. However, they are not just graphical modeling tools but they are a complete, unified graphical programming environment. They offer graphical transformation and mapping tools that allow developers to drag and drop a rich set of out-of-the box catalog functions or user-defined functions into the rules editor. Even if they are much simplified programming environments than today's Integrated Development Environments (such as Eclipse,[7] Visual Studio,[8] and Netbeans[9]), programming is still their central metaphor. For example, the screenshot in Fig. 1, depicting the StreamBase studio interface, presents a graphical representation that exposes technological constructs such as icons for carrying out classic Database Management Systems operations, function-based tasks or queries that can be performed by using a specific programming languages paradigm, i.e. StreamSQL.

For avoiding the need to learn a SQL-like language or other programming-oriented constructions, in some case unknown by non-technical people, interesting solutions can be found in the field of the BRMSs able to offer easier way for editing rules. In [14] the authors propose the integration of Drools and the XTT2 rules representation and the HQEd visual Rule Editor. The results of the modeling are translated in Drools Language (DRL) files, which can be executed by the Drools engine. However, the weakness of these approaches is that the XTT2 language has not been standardized and has several limitations as reported in [15] despite the high expressiveness of DRL. Another paper presents a solution for graphical modeling of rules [16] that are then automatically translated in the programming language supported by the adopted rule engine. In this case, the graphical editor is integrated in Drools Guvnor.[10] Drools Guvnor provides a guided text editor for writing rules that are then translated into the Drools rule engine compliant language. However, the expressivity of the visual language described in [16] is reduced with respect to drools textual language especially for what concerns the specification and processing of complex events. This language allows to specify a business rule can be expressed in form: ***When*** *something is true*, **Then** *do this*. Instead, the Guvnor text editor allows to specify more complex conditions on events. Apart from supporting logical combinations of conditions, the editor also includes rich timing comparison operators that allow to specify different timing relations between events, such as the event A happens before or after event B.

Other visual strategies typically used for modeling workflows are based on the use of mashup techniques for combining existing Web-based content and services for creating new applications. These strategies stem from the idea to adopt a control-metaphor based on the use of “spreadsheets” or “pipes”. Decision tables are one of the most popular way to present sets of related business rules inside spreadsheet tables. They are generally used to describe and analyze decision situations, where the state of a number of conditions determines the execution of a set of actions. BRMSs such as OpenRules or Drools allow users to configure different types of conditional logic to generate rules in the base of data entered into a spreadsheet. A disadvantage of the technique is that a decision table is not equivalent to complete test cases containing step-by-step instructions of what to do in what order. When this level of detail is required, the decision table has to be further detailed into test cases making it difficult to read and understand. Other solutions rely on the use of metaphor provided by the most famous system that apply them, it is Yahoo's Pipes.[11] it is based on the idea of providing a visual pipeline generator for supporting end users in creating aggregation, filtering, and porting of data originated by different sources. Systems like Bipio[12] or DERI pipes[13] typically use formula languages and/or visual programming for data transformation and mash-up. An advanced use of such visual paradigm is offered by WebHooks[14] that allows the end users to even write their personal API for enabling connections with new sources of data. In general, the visual strategies adopted by such Yahoo's Pipes-compliant solutions are promising techniques but, in our opinion, they present some lacks. Some studies [17] also found that, although these strategies tried to simplify mashup development, they are still difficult to use by non-technical users, who encounter difficulties with the adopted composition languages [18]. The preparation steps to setup the pipes or customizing the operators of these pipes require low-level technical skills such as data processing or programming. As a result, this mashup approach hurdles users in their everyday situations. Moreover, events in each stream of a IoT scenario are time and space dependent and so the related rules need to take into account these types of conditions. Nevertheless, in the described systems, time and space dimensions are almost neglected.

[4] http://www.talend.com
[5] http://www.streambase.com
[6] http://www.waylay.io
[7] https://eclipse.org
[8] https://www.visualstudio.com
[9] https://netbeans.org
[10] http://guvnor.jboss.org/
[11] https://pipes.yahoo.com/pipes/
[12] https://bip.io/
[13] http://pipes.deri.org
[14] https://developer.github.com/webhooks

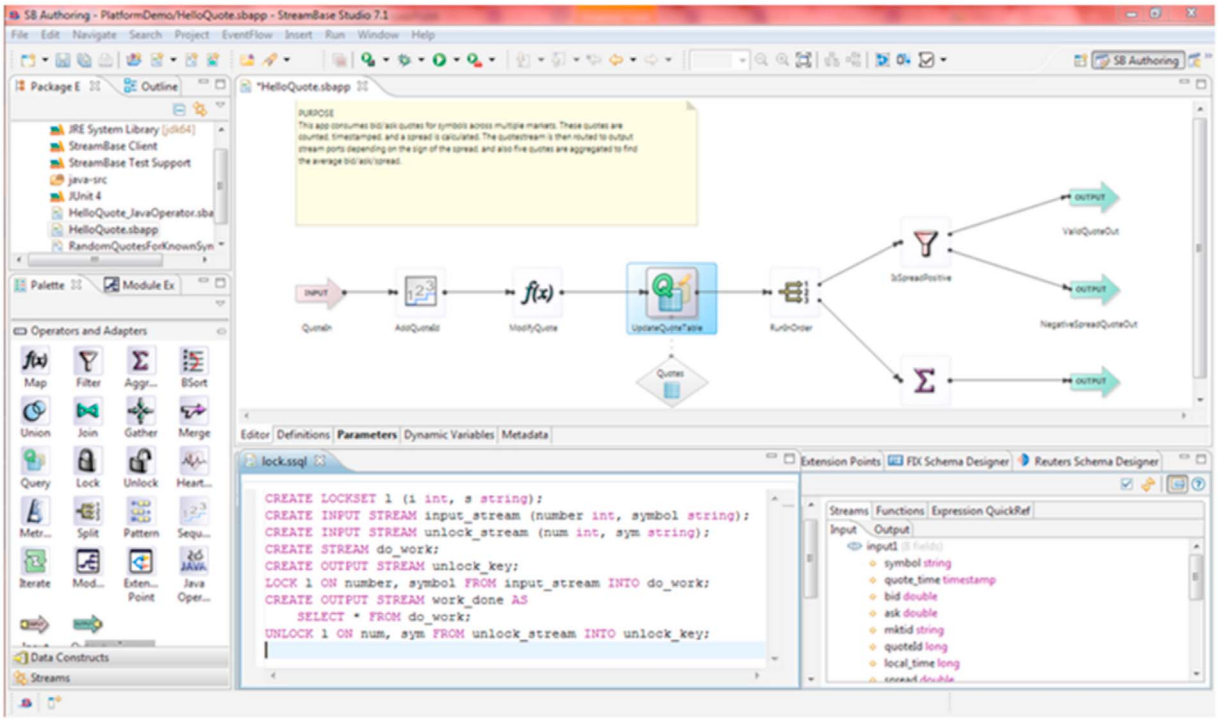

Fig. 1. StreamBase Studio screenshot. An example of a visual workflow design strategy.

2.2. From BRMs graphical interfaces to event-based visual notations

Solutions described in the previous section and currently adopted in several WfM and BRM systems or Yahoo's Pipes-like applications, despite being an easy strategy for editing rules, suffer of the problem to compel end users in acquiring technical knowledge to manage the workflow that characterize their working context. Moreover, there are many end users' daily settings where a programming environment is plainly inappropriate and inaccessible for the situation. For example, in an eWellness context, where a trainer has to monitor the activities of her/his athletes, she/he has a requirement for a lightweight, minimalist approach without the constraint to manage the whole workflow and data-flow that underlies their knowledge domain. For example, they do not wish to handle the amount of involved sensors and services on which data they need to execute integration, filter, join, transform operations in order to detect relevant events. What they need is a simplified environment to monitor events and to perform a set of rules to take into account for supporting their decision making. Recent applications in IoT field have focused their attention on the modeling tasks on the base of event-condition-action (ECA) [19] rules able to monitor regular day-to-day activities for detecting anomaly or specific situations. Famous examples are IFTTT[15] and Atooma.[16] These applications allow users to define sets of desired behaviors in response to specific events. This is made mainly through rules definition-wizards that rely on the sensors/devices states. Rules can be typically chosen among existing ones or can be tweaked through customization. These activities put in place a task automation layer across all sensors/devices in the IoT environment. The visual strategy aims at creating automated rules by using graphical notation for programming statements such as: “IF this DO That” or “WHEN trigger THEN action”. These solutions offer a very easy to learn solution based on the definition of ad hoc rules that can notify the end users when something happens, e.g., when their favorite sites are updated, when they check-in in some places or their friends do, or warn them when specific weather conditions are going to take place. However, the language is not expressive enough for the specification of more sophisticated rules based on time and space conditions. For example, in a scenario where we want to monitor data related to health conditions and behavior of a group of athletes, we need to keep track of the collected data through all the everyday or occasional activities that may influence on the people's quality of life, e.g., in wellness domain: weight, sport/fitness, food, and sleep quality. Apart from supporting logical combinations of conditions, we need to specify timing comparison between different events. For example that the athlete has taken less calories of those burned in a physical activity that has been taken place after lunch.

3. Research approach

Our research's approach is socio-technical, i.e. it focuses on both the social and the technological aspects without considering them as distinct but seeing them as tightly interconnected and related. Such an approach is based on user-centric design and development methodologies (User-Centered Design, but in particular EUD) in which the user is seen in her/his activity but also directly involved in the creation and design processes. In EUD literature, several methods and tools are available to support the participation of end users in design and development [20–23]. In our research, we adopt the SSW methodology [8]. SSW allows a multidisciplinary design team to design and develop interactive environments that support the activities of users acting a specific role in their community and having a specific application domain; are tailorable, customizable and adaptive to the working context; support the exchange of information among users belonging to different communities. The SSW methodology is evolutionary and participatory: the final user can customize and evolve her/his own virtual environment and she/he is involved in each step of the system development. A representation of the implementation of the SSW methodology for our research is depicted in Fig. 2.

At Metadesign level one or more IoT Engineers configure the IoT elements in the ecosystem and design the data flow that will gather and organize data to be served at the Design level interactive system. At Design level, Coach and Trainer (but eventually more than one for each

[15] https://ifttt.com
[16] http://www.atooma.com

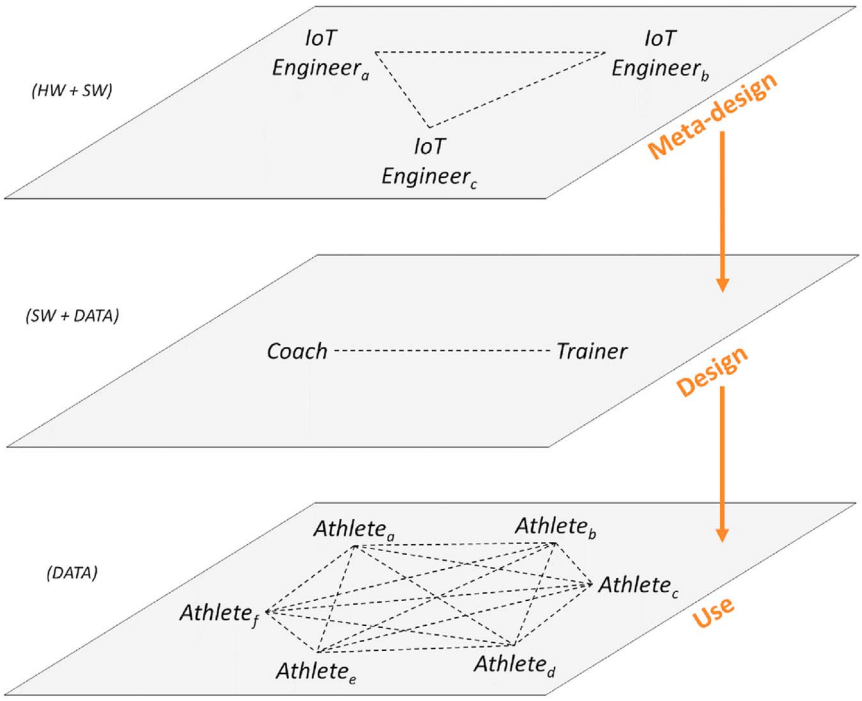

Fig. 2. The SSW implementation for SmartFit Framework.

category) will use their system to design the rules to be used at Use level. At Use level the Athletes will be provided with an interactive system for monitoring their own behavior and lifestyle and to share their data with the other athletes in their team. The vertical arrows in Fig. 2 show the flow that the meta-design, design, and use process follows, while the dashed lines indicate that the users at the same level are able to share among themselves the outcome of their metadesign, design and use activities. The SmartFit Framework architecture and its three interactive systems are described in detail in the next section.

Our research is framed in a computer-semiotics context. According to Andersen's definition [24], computer semiotics is “a discipline that analyzes computer systems and their context of use under a specific perspective, namely as signs that users interpret to mean something”. Computer systems are studied in computer semiotics from two different, though related, perspectives. First, they are utterances in which designers and developers formulate an understanding of the problem domain and of the application domain of the system [25–28]. Semiotic engineering [9] views interactive software systems as artifacts through which the communication between users and designers takes place. The designer sends users “a one-shot message” which unfolds into further two-way message exchanges, according to de Souza [27] explaining how and why they should communicate with the software application in order to achieve a specific goal. To decode and interpret the designer's message, the users have to communicate with the very message itself employed in the software. The message gradually unfolds all the meanings the designer encoded in it before the users. In this view, the message (i.e. the software application) serves as the designer's deputy and represents the mediator through which end users can interact with the system. From a semiotic engineering perspective, Human–Computer Interaction is seen as a process of “communication about communication” i.e. “meta-communication” [27]. Indeed, system designers tell users how and why to interact with the system they have created, and this explanation is embedded in the system's interface, which serves as a means for transferring the designers' messages to the users. Second, in so transferring messages, computer systems provide a formal symbol-manipulation mechanism that serves as a structured channel for communication through the system. The symbols are computed by the machine and the results of the processing appear as messages on the computer monitor. At that point, the human perceives the messages and transforms them back into signs. These two ways in which computer systems are part of sign-based communication are related: Skilled use of a software application depends on the user's understanding of how the formal symbol-manipulation relates to the problem and application domains. In other words, competent interaction with and communication through the system depends on the user understanding the utterance of the designers and developers [25,26]. Conversely, if the software is designed to structure navigation and content in ways unfamiliar to the user,the user will not be able to use the software for the task at hand or to communicate with other users in a collaboration context.

Specifically, to our work, we apply the semiotic model of digital communication among IT experts, domain experts, and end users that we published in [29].

To describe this semiotic approach, the communication process that takes place among all the stakeholders in a participatory design process, exploiting on the participation of domain experts in the knowledge management design, is enabled by the means of the design of usable and effective interfaces, that according to semiotic engineering theory can be seen as composed of messages (embodying the implicit information).

From a preliminary comparison with coaches and trainers we involved in the design process and test of our prototype, we understood that such users need to have control over the conditions and the action sequences of their task and automate the triggering of its execution once the specified conditions are met. They do not want to be distracted by the complexity of the composition paradigm of a workflow but they want to focus on their daily tasks that have to be defined in an easy way also by nontechnical users without the IT assistance.

As part of the participatory design process, we organized a focus group involving coaches and trainers of a teenager soccer team of Milan affiliated to the Centro Sportivo Italiano (Italian Sport Centre), highlighted as ECA rules could well represent the message that a coach or a trainer wants to communicate. These communication strategies were devised by experts who collaborated in three participatory design sessions. During these sessions four domain experts were involved: Three coaches and one trainer. A first session, lasting approximately

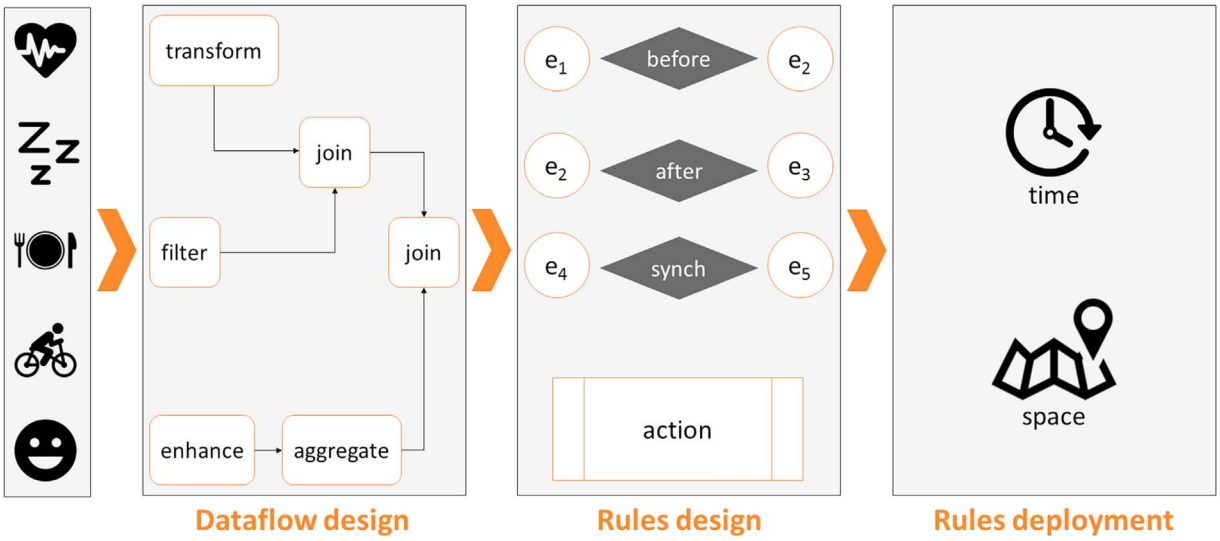

Fig. 3. The SmartFit Framework.

three hours, was set up for defining a set of tasks to be carried out in order to monitor specific situations and conditions of members of a non-professional soccer team. During the session, coaches have highlighted the need to monitor specific situations and trigger related corrective actions such as "If yesterday my athlete overtook his caloric intake of 3000, today I suggest him to run more than 1 hour" instead of having to design the whole workflow for monitoring data coming from her/his athletes' sensors and services. The results of this session have highlighted how by keeping track of events related to athletes' habits, in terms of physical activity, nutrition, sleep and so on, can help the coaches in understanding the variety of the team members and finding successful schemes of training. A second session was set up for discussing about possible solutions for helping coaches and trainers in creating rules in a digital environment by means of a visual language. Also this session requested a debate of around three hours. According to the results of this session and as outlined in [30], a summary of common characteristics that point to the need for defining ECA rules is:

- Users need to take the control of rule conditions, including (i) the choice of resources and (ii) the specification of spatio-temporal condition combinations. For example, the user may to be interested in selecting specific sensors and services for monitoring the calories intake before a heavy physical activity.
- Users need to take the control of actions triggered by rules, including (i) the choice of resources and (ii) the specification of action sequence for execution. For example,since the user want to monitor the calories intake and the minutes of heavy physical activities, she/he may to be warned in case of critical situations such as that the caloric intake is too low with respect to the time spent on heavy physical activities.
- Users want to delegate the checking of condition fulfillment and the subsequent initiation of rule execution. For example, the user may enable automatic triggering, by email or sms, of suggestions or warning in case some critical conditions are met.

From literature analysis [31,32], it emerges that the ECA rule-based paradigm turns out to be one of the most promising ones for supporting non-expert users in configuring smart devices in intelligent environments. Other studies such as [33,34] claimed as the use of ECA rules is well suited for monitoring wellness events. Regardless of the application domains, the literature explains how the use of visual tools based on IFTTT-like approaches is a suitable solution for supporting EUD strategies based on ECA rule [35,31]. These studies showed that the perceived usability of these visual tools is good and that the average user can successfully carry out trigger-action programming, also when it is extended with the possibility of creating multiple triggers and actions.

For these reasoning, starting from the IFTTT-like approaches, our idea aims at extending the ECA language for providing coaches and trainers with a familiar graphical notation for editing rules, that can be stored and shared among all member of the community to which they belong. An initial proposal of our Rule Language is published in [36] and is refined and extended in this paper (see Section 5).

The third and last session was devoted to the design of the SmartFit Rule Editor interface, as is described later in Section 6.1.

The final SmartFit Framework is based on a cloud-based architecture able to host the ECA rules that coaches and trainers can personalize and use in their context of use without the need to reedit them or to define the whole workflow. In contrast to traditional systems that host networked applications on dedicated server hardware, a cloud computing model allows applications to be provided over the network "as a service" supplied by an infrastructure provider. As argued in [30], our cloud-based architecture aims at offering a new layer of service, named a "rule as a service" for allowing coaches and trainers to share their ECA rule definitions and in this way improving collaboration, integration, and community-based cooperation within their organizations. Share rules allow users to exploit common experiences in order to discover existing solutions that can fit their needs. This increase the sense of belonging to a common community that help to find fruitful solutions based on shared experiences.

4. The SmartFit Framework

All the interactive systems used by IoT Engineers, Coaches and Trainers, and by the Athletes, together with the IoT devices, constitute the SmartFit Framework. SmartFit aims at offering a set of graphical visual environment for exploiting the potentials of an IoT environment in the domain of non-professional athletes training. In Fig. 3 the different environments are illustrated showing the flow of the process, starting with the connection of the IoT devices, passing through the Data flow design system, called StreamLoader (for the Engineers), the Rules design system, called SmartFit Rule Editor (for Coaches and Trainers), and ending with the deployment of the rules for their actual use in everyday life (for the Athletes). Up to now, this one is the only system that still has to be completed.

The architecture at the base of our SmartFit Framework has been designed to help members of non-professional sport teams, in configuring and managing a network of sensors and services which the aim of

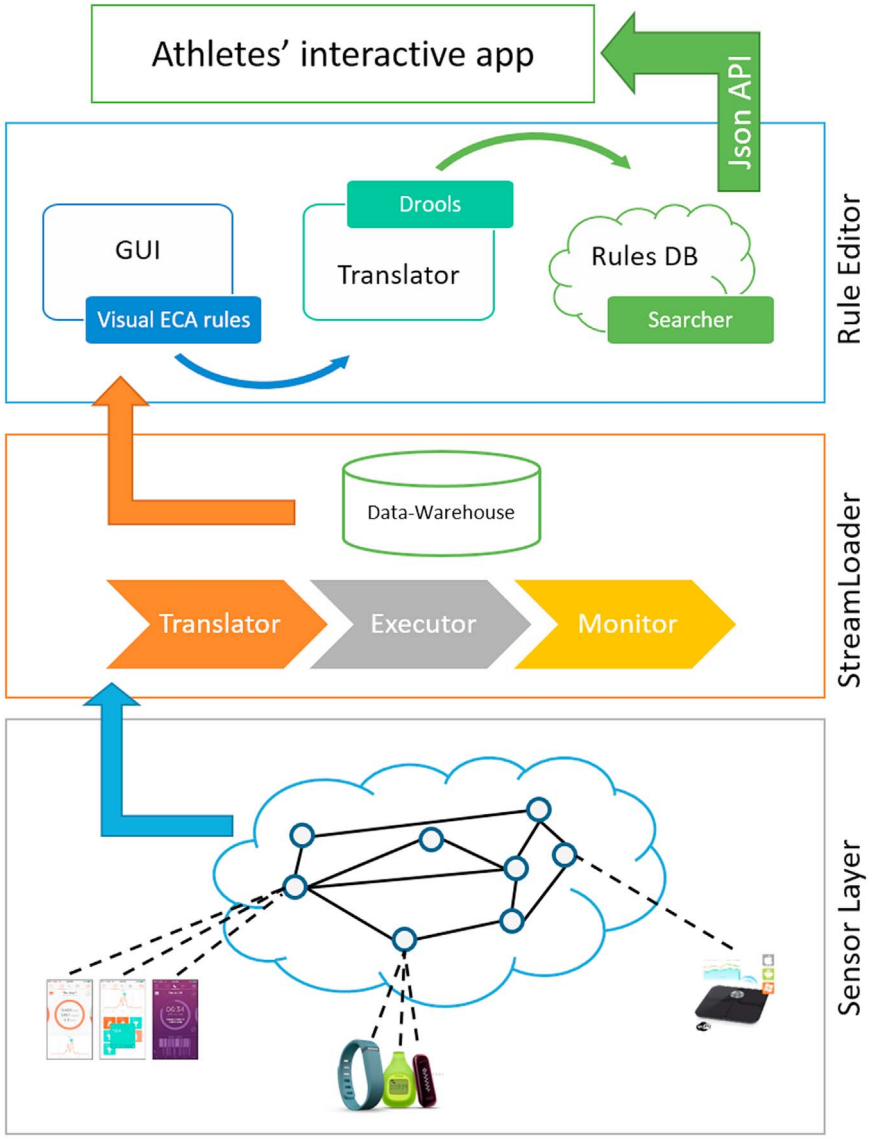

Fig. 4. SmartFit cloud architecture.

providing coaches and trainers with strategies for monitoring their athletes' activities. As depicted in Fig. 4 the architecture of the SmartFit Framework is composed of four layers: Sensor Layer, StreamLoader, Rule Editor and Interactive App. The first one is the physical network of sensors and services while the other three are tailor-made environments designed to address specific needs of three different Communities of Practice that co-exist in a eWellness scenario: IoT engineers, Coaches and Trainers, and Athletes.

Sensor layer: At sensor layer, each node of the network is in charge of managing a bunch of sensors and can execute a set of Extract, Transform, Load stream processing operations. Sensors are handled through a distributed publish–subscribe system [19]. Each time a sensor is published, its type, schema, and frequency of data generation are made available to subscribers. When a conceptual dataflow is realized into the StreamLoader, a translator module is in charge to execute it at network level. This module described in [37,38] aims at providing a network control protocol stack able to interpret the logical description of the data flow and dynamically to coordinate the network configurations, such as dataflow execution, segmentations, and QoS parameters. When the dataflow is graphically described by the IoT sensors/devices engineers in the StreamLoader it is translated and executed in the network and monitored. The executor module coordinates their execution.

StreamLoader: By means of the StreamLoader system, IoT sensors/device engineers are in charge of connecting, maintaining, and setting up the devices and sensors to be used by the Smartfit and its inhabitants (coaches, trainers, and athletes). StreamLoader allows the design of data flows by aggregating data sources and applying operators to them for filtering, transforming, and composing the gathered data. More details about this environment are referred in [38,39]. The StreamLoader output is a flow of events that can be used by coaches and trainers, for defining rules to monitor particular situations and to adopt suitable actions according to the occurrence of given conditions. As reported in Fig. 4, the final flow of events is stored in a data-warehouse that is then forwarded to the Rule Editor to be analyzed by coaches and trainers. Exploiting the Rule Language explained in the next section, the Rule Editor enables coaches and trainers to act as end-user developers by designing the ECA statements to be used to supervise athletes' performances and lifestyle and they also analyze the gathered data in their interactive system. StreamLoader produces streams of events according to the multigranular space, time and thematic data model.

Data produced by sensors or services are heterogeneous in spatial and/or temporal granularities (e.g., temperature in a room versus

temperatures in a geo-graphical area or data acquired one time for day or every hour), in thematics (data about calories burned or data about heartrate). Relying on the concepts of temporal and spatial granularities, an event is stored as a tuple of data coming from a sensor which generated values having a spatial reference about its location and that are captured at a given time period according to given thematic. Therefore, an event is a value represented at a given spatio-temporal granularity for which thematic information is added. Granularities are used for identifying correlations among data produced by different sensors and for imposing consistency constraints in the composition of sensor data produced by heterogeneous devices. We remark that whenever a sensor is not able to produce the spatio-temporal information of the produced data, this information is added by the publish–subscribe system that we adopt in our architecture.

Rule Editor: The Rule Editor has been designed to offer the possibility to exploit the expressivity of our ECA language specification through a visual interaction. By means of graphical user interface (GUI), coaches and trainers can express their intent through a familiar and easy to manipulate interaction strategy. At the same time, the GUI forwards this visual specification to a translator module to translate it into an “executable” language to provide to a rule engine. Specifically, the translator module translates the ECA rules expressed by the visual specification in DRL files, which can be executed by the Drools engine. Coaches and trainers act as non-technical users and for this reason the Rule Editor needs to provide them with interactive strategies able to hide the complexity of the underlying transport module. Moreover, in a typical non-professional sport team, coaches and trainers are involved as volunteers or semi-pros that need to re-use existing ECA rules for taking advanced by training and monitoring strategies of others colleagues or more expert professional. To make this efficient, the Rule Editor stores generated rules into a repository that manages them for one or more teams. From a technical point of view, designing storage architectures for these rule definitions presents several challenges and opportunities. Tackling these problems requires a combination of architectural optimizations to the storage devices and layers of the memory/storage hierarchy as well as techniques to manage the flow of data between the software layers and storage. Our cloud computing data center is modeled upon a simple design-for-failure infrastructure. It uses low-cost, purpose-built, scalable solutions while still utilizing a standard delivery model. The peculiarity of our architecture is to provide a service specially designed for exposing rules through a JSON API. The rules are stored without specifying their validity, that is, without indicating the time period in which the rules are enabled. In this way, the rules can be reused and customized by other users according to their needs. Finally, to effectively manage policies implemented as large numbers of rules, the Rule Editor provides the possibility to search the repository for specific rules, based upon criteria related to the spatio-temporal-thematic granularity used for storing the flow of events in input to the Rule Editor. Moreover, the Rule Editor enables the assignment of additional metadata properties (built-in or customized to your own needs) such as who authored the rule, when the rule was last modified or the effective date range for the validation of the rule. These properties are in addition to the ability to search based on the vocabulary terms used in rule definitions and the actions taken as results.

Interactive App: Finally, at the last layer of our architecture, an interactive app acts for supporting athletes that can be seen as the real end users, to passively use the data gathered by the IoT sensors and devices to improve their lifestyle and sport performances. This tailored interactive system can be used to have a view on their behavior and performances at any time during the day.

5. Rule Language

A business rule is a statement that describes a policy or procedure [40]. A business policy defines the scope or spheres within decisions can be taken. In this context, a policy needs to be translated into a more specific statement that specifies the details of how the policy is enforced. These statements are the business rules and are used for explaining the conditions and actions that unambiguously enforce the policy. For example, if a coach wants to monitor her/his athletes' behaviors about their diet and quality of sleep she/he could define rules for checking the athletes' calories intake and the hours of sleep. In creating these rules, the coach has to specify what happen when a particular set of conditions occur, by defining a list of actions to trigger. For example, the coach can specify that if the conditions are met, a warning is sent by sms. In a rule, the condition evaluation is not tied to a specific evaluation sequence or carried out at a predefined time, but it happens continually, at any time during the life time of the sensors or services; whenever the condition is met, the actions are executed. According to this rule definition, SmartFit aims at providing non-professional sport team members with an environment for rules that are used for monitoring events related to athletes' habits, in terms of physical activity, nutrition, sleep and psychological conditions. This editing system is designed for providing a means of expressing rules in a format and language that is familiar to the policy managers, the sport team members in charge to define rules. The main aim of SmartFit is to offer a ECA language specification through a visual notation that can unambiguously express the coaches and trainers' intents through a familiar and easy to manipulate interaction strategy, and at the same time a language that can be translated into something “executable” by the software application. A peculiar characteristic of this language is the need to monitor events that are time dependent. In IoT contexts of use, rules are designed around the requirement to manage the processing of real time or near real time events, and for this reason, time becomes an important variable of the reasoning process. In what follows the formal description of the language used for expressing rules is presented.

5.1. SmartFit Rule Editor Language

The Rule Editor Language we defined allows the domain experts (coaches and trainers) to specify composite rules on different kinds of events that are detected by IoT devices, sensors and applications used by the athletes. In the notation used for the language definition, we used square brackets to denote optional components. A CompositeRule, as it can be seen by its definition, is composed of one or more Rule(s) and is associated to an Action. An Action in the current implementation of the Language in SmartFit Rule Editor is message that is sent to the athletes via their SmartFit environment. Moreover, a CompositeRule might have the indication of a ValidityInterval, that is the period of time considered for the data stream filtering; all events taking place outside the specified ValidityInterval would be ignored. A Rule is defined by the Event that has to take place and a specific condition on that Event. For example, an Event in the eWellness case could be “Calories Intake” and as a condition “greater than 1500”. In our work, we define the Event as the stream of Data that is gathered by a specific device. It is important to underline the fact that Data might not be only composed of numbers or strings, but could be defined by a complex structure that has one or more parameters. As an example, physical activity detected by a fitness monitoring device could be described by duration, calories burned, heartrate, distance, location etc. Therefore, a parameter is defined as the couple ParName (name of the parameter) and ParValue (the value recorded by the fitness monitoring device). Condition can be of two types: on value or on time. ConditionOnValue allows to express a filter on the value of a specific parameter using common mathematical and logical operators. ConditionOnTime, on the other hand, allows to express conditions on temporal frame in which the events specified by the rules take place. An important aspect of this language is that is simplified in comparison to Drools one, as discussed later in this section.

```
CompositeRule ::= Rule [AND | OR] Action
[ValidityInterval]
Rule ::= ON Event WHENEVER Condition
Event ::= Data GATHERED BY DEVICE (DId)
Data ::= Parameter
Parameter ::= ParName ParValue
ParValue ::= Integer | Real | String | Date | Set
Action ::= SEND MESSAGE TO USER_APP (AId)
ValidityInterval ::= TODAY | YESTERDAY | 1 DAY BEFORE | 2
DAYS
BEFORE | 3 DAYS BEFORE | 1 WEEK BEFORE | 1 MONTH BEFORE |
6 MONTHS BEFORE | SINCE THE BEGINNING
Condition ::= ConOnValue | CondOnTime
CondOnValue ::= ParName Operator ParValue
Operator ::= = || < || > || IN | NOT IN | BETWEEN | NOT
BETWEEN | IS NULL | IS NOT NULL
CondOnTime ::= Rule starts BEFORE | AFTER | WHEN Rule
starts
[Range] [and ends BEFORE | AFTER | WHEN Rule ends
[Range]]
Range ::= +/− Integer
```

5.2. Overcoming drools complexity

To implement the Drools rules editing in an interface would force the user to select among a set of different types of presets that are not easily recognizable without effort. Drools has a set of 13 temporal operators: After, Before, Coincides, During, Finishes, Finished by, Includes, Meets, Met by, Overlaps, Overlapped by, Starts, and Started by. All 13 operators have different meaning but some of them are not very distinguishable from one another. The different combinations of rules temporal conditions are depicted in Fig. 5. We described the rules temporal conditions with an approach based on Allen's work on interval-based temporal logic [41]. All these combinations can be codified using the Rules Language we defined. This simplification, in respect to Drools syntax, allows to create easy-to-use interfaces and a fast process of composite rules creation. All the combinations can be codified using the Rules Language we defined. This simplification, in respect to Drools syntax, allows to create easy-to-use interfaces and a fast process of composite rules creation. Let us propose an example for proving the reduced complexity in the SmartFit Rule Language is the creation of a complex temporal condition:

Rule ***R1*** *starts* ***before*** *Rule* ***R2*** *starts* (*Range*: ±**15** ***min***) *and ends* ***after*** *R2 ends.*

The creation of such condition takes place in SmartFit by asking the user to select only the parameters written in bold in the condition text. The interaction with the system is very easy and fast and results in the creation of a natural language sentence that is very easy for the user to understand, even if she/he reads them a long time after it has been created (i.e. recall is not needed to understand the interaction with the system; instead, recognitions work effectively).

The same temporal condition, if expressed in Drools, would force the user to understand which one of the available operators to use. Actually, one operator is not enough to represent entirely the condition that the user is asked to edit. In fact, to express that two rules are overlapping there is need to choose the Overlaps operator but it is not possible to express the fact that the first Rule (R1) starts before and ends after the second Rule (R2). Therefore, a combination of more than one operator would be necessary, increasing in this way the complexity of the user interaction and augmenting user's cognitive effort.

6. SmartFit graphical Rule Editor

By exploiting the Rule Language described in the previous section, the rules can be specified by means of the graphical Rule Editor. Through a visual notation, coaches and trainers can specify conditions to monitor of athlete well-being on the base of the flow of events provided in output by IoT engineers at meta-design level with the SmartLoader environment. A screenshot of the initial screen of Rule Editor is shown in Fig. 6.

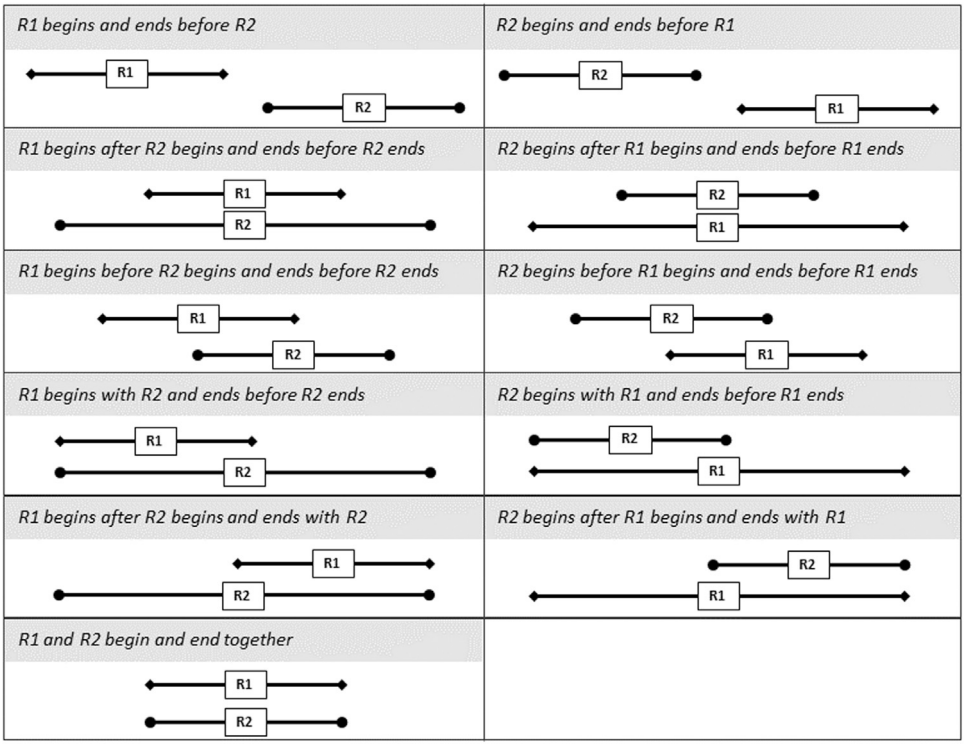

Fig. 5. All 11 possible combinations that can be codified using Rule Language.

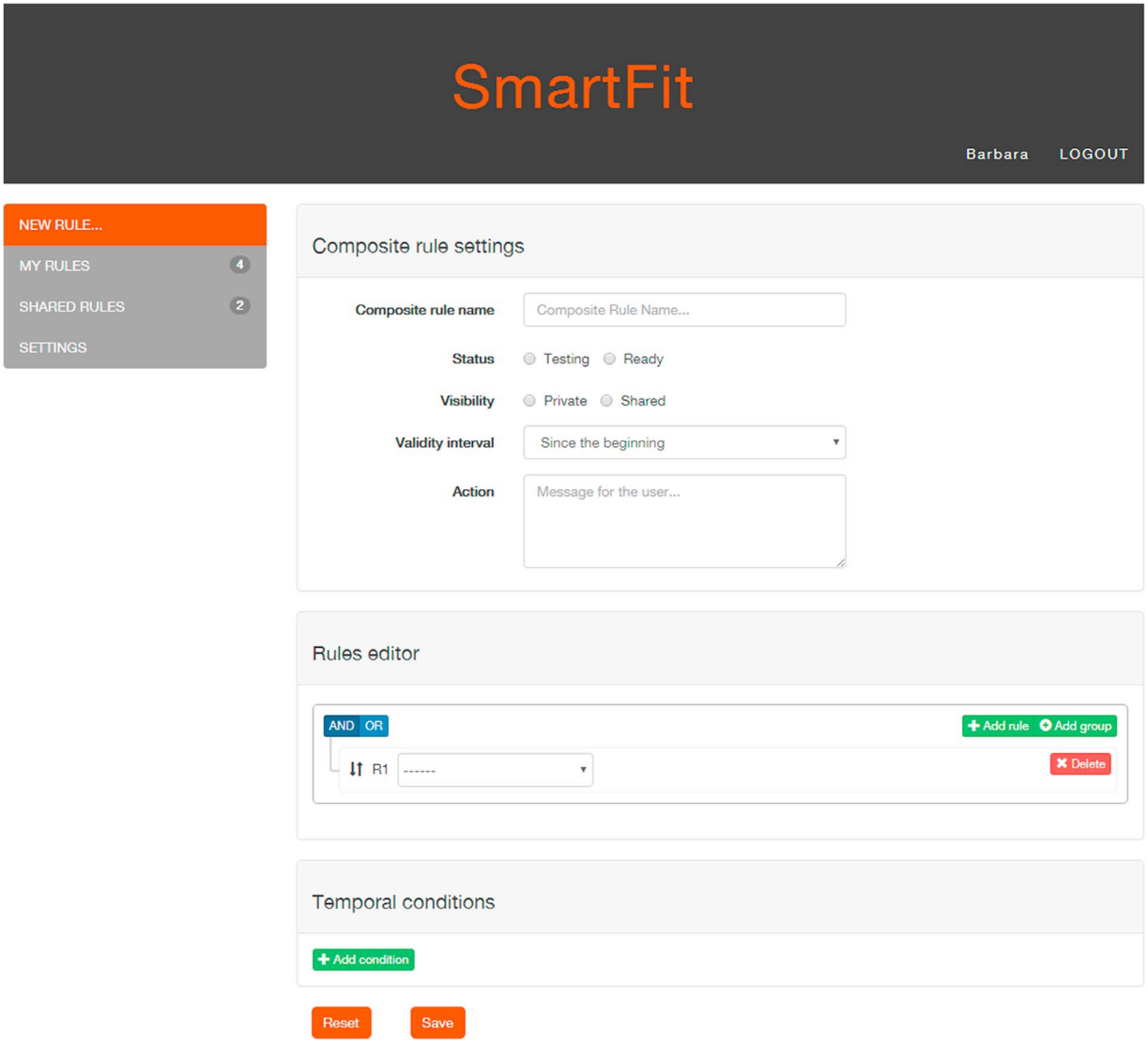

Fig. 6. The Rule Editor system.

6.1. Design rationale

The design of the SmartFit Rule Editor interface is an outcome of the third session of the focus group we organized with the domain experts. We adopted the PICTIVE method (Plastic Interface for Collaborative Technology Initiatives through Video Exploration) [42], based on collaborative prototyping and visual communication paradigms, and used a combination of deliberately low-tech design tools (e.g., paper and pencil, colored labels, and sticky notes) with high-tech video recording facilities. A mock-up prototype of SmartFit Rule Editor was created using low-tech design tools. Recording facilities have been used to capture the process of prototyping, in order to use the recordings as a guide for the implementation of the system or to explain the logic that exists behind the design decisions.

The prototype showed us what features the domain experts expect to find in the system and how they see the composite rule creation process organized.

As to the features, the requests were to have four sections in the system:

1. *New rule*: This section is the one that is further described in this paper, that is the composite Rule Editor.
2. *My rules*: In this section the users find all the rules they created with the possibility of manage them (modify, delete). Among the other details that the users can change for the rules, it is also possible to set the sharing status of the rule, i.e. to be shared or not with others.
3. *Shared rules*: This section is dedicated to all the rules that have been shared with others and to those that were created by other users and that were adopted by the Editor user. Shared rules can be modified by the user to adapt them to their needs.
4. *Settings*: This final section allows to set some properties of the system (e.g., colors) that can be customized to enhance the user experience quality.

In designing the composite rule creation process, the domain experts pointed out the need to have it split into three main steps:

1. *Composite rule settings*: This first step allows the user to assign a name to the composite rule, to specify if the rule is ready to be used or in a testing phase, to share the rule or maintain it private, to set a validity interval of application of the rule and finally to define the action that has to be taken when the rule is eventually verified.
2. *Rules editor*: This represents the core of the SmartFit Rule Editor, in that allows the user to create the composite rule, using some logical operators (and, or) and grouping and reordering the rules if needed.
3. *Temporal conditions editor*: This last step is the one that allows the user to define specific temporal conditions that need to be satisfied to relate the rule with the time dimension.

```
{
  "day": "30-01-2016",
  "data_flow": {
    "step_counter": {
      "type": "sensor",
      "total_steps_day": "9000",
      "num_steps": [
        {
          "start": "11:00",
          "stop": "11:30",
          "steps": "7000"
        },
        {
          "start": "13:00",
          "stop": "14:30",
          "steps": "2000"
        }
      ]
    },

 "hours_sleep_daybefore": "7"
  },

 "calories_intake": {
     "type": "mobile_app",
     "total_cal_day": "4500",
     "calories": [
       {
         "time_start": "08:00",
         "time_end":   "08:20",
         "num": "1000"
       },
       {
         "time_start": "12:00",
         "time_end":   "12:45",

         "num": "2000"
       },
       {
         "time_start": "20:00",
         "time_end":   "20:40",

         "num": "1500"
       }
     ]
   }
}
```


Fig. 7. JSON flow of events.

6.2. The graphical editor

As depicted in Fig. 8 the graphical interface is based on drop-down menus that are populated by using the attributes that characterize the JSON of the flow of events produced by the IoT Engineers with the SmartLoader environment. At this development stage of the Rule Editor, we are able to manage events sampled on a daily basis. For each day, the dataflow reports a set of events that can be collected at different time.

For example, Fig. 7 presents a flow of three events caught at day "30-01-2016" and that concern the number of steps, the hours of sleep of the day before, and the calories intake at breakfast, lunch and dinner. An example of CompositeRule creation is given in Fig. 8, where the user has defined four Rule conditions:

- R1: Hours of sleep less than 7.
- R2: Calories intake at dinner greater than 1500.
- R3: Number of steps less than 8000.
- R4: Activity duration less than 45 min.

The rules are built in this way: R1 AND R2 AND (R3 OR R4). The meaning of the CompositeRule created in this example is: "if the hours of sleep of the day before are less than 7 AND if the calories intake at dinner (before the sleep) is greater than 1500 AND (if the number of steps is less than 8000 at day OR the duration of physical activity is not less of 45 min at day THEN send the athlete a message that warns about the behavior and performances."

The Rule Editor aims at allowing non-technical people to specify rules by using simple drop-down menus. The conditions can be composed of combining groups of statements connected by using the operator AND or OR. The order of the conditions that can be changed by the user just by dragging and dropping the statements into the right

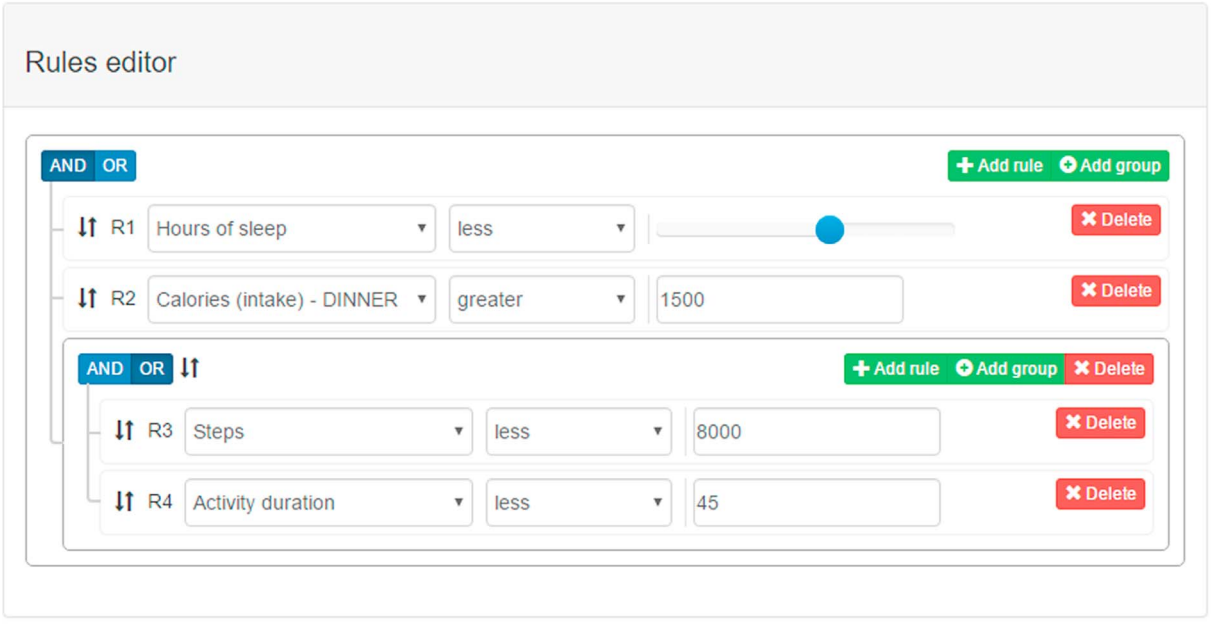

Fig. 8. The composition of the rules of a CompositeRule.

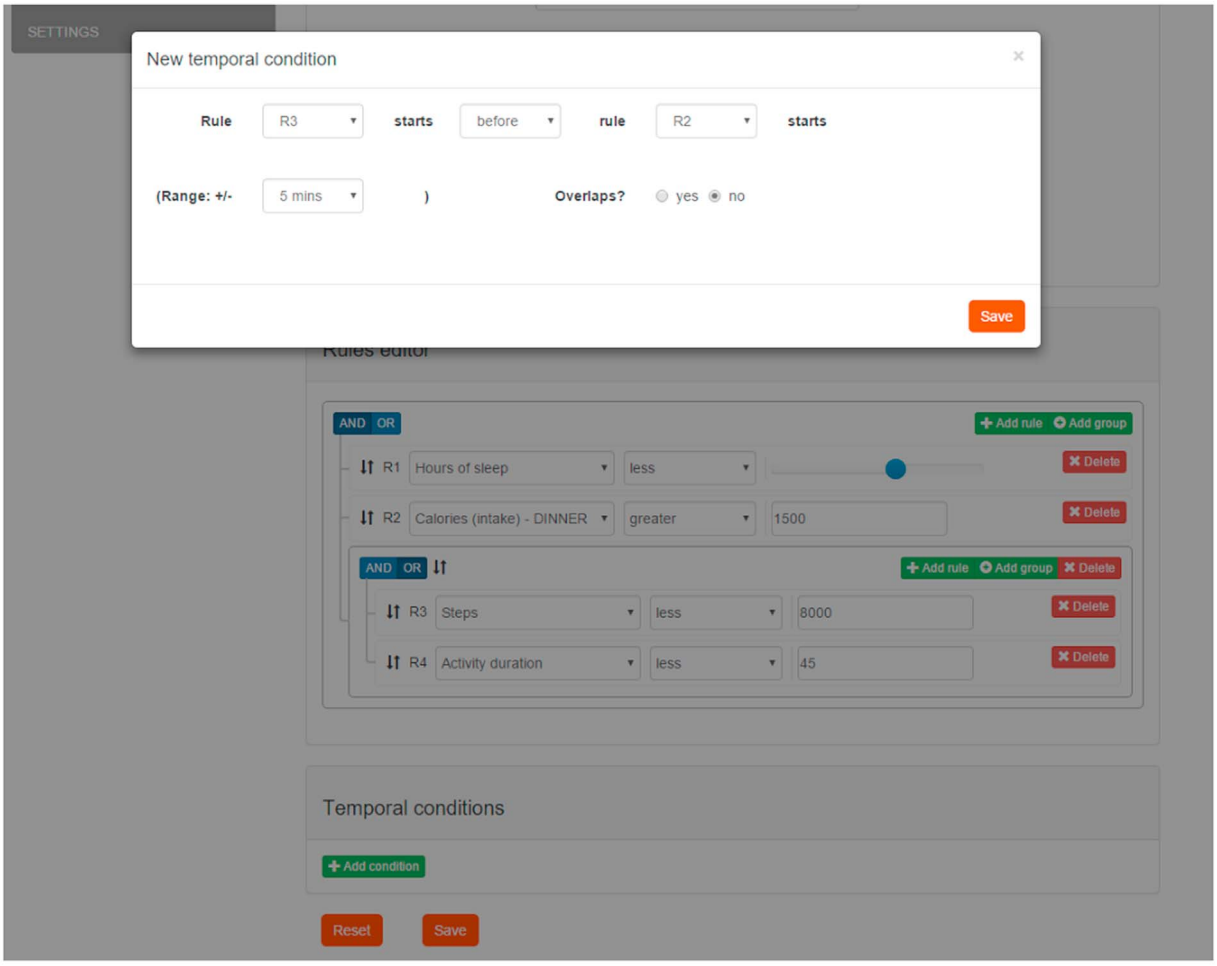

Fig. 9. The creation of a temporal condition in the Rule Editor system.

position.

The user can collect data for a given period of time set by using the "validity interval" parameter (see Fig. 6). In our example the data are collected since three days before. Temporal conditions are defined as depicted in Fig. 9 and use the automatically assigned names of the Rules as elements to be composed (R1, R2, R3, ...). An example of temporal condition is that R3 (number of steps less than 8000) has to take place before R2 (calories intake at dinner greater than 1500) with a range of ± 5 min. In other words, the dinner must begin within 5 min after the user has finished to walk (for less of 8000 steps). Another complex temporal condition is depicted in Fig. 10. In this case, R4 (activity duration less than 45 min) begins before R2 (Calories intake at dinner greater than 1500) and ends after ± 10 min. In other words, the trainer wants to check if her/his athletes eat too much and how, if they eat not seated at a table but when they are on the move and quickly (in less than 45 min). For creating a such temporal condition, the user needs to select the overlap flag. After having indicated an overlap between two rules, the user can also specify if the event of the first rule has to end after or before the second one and by when.

After the composition of a rule or a set of rules, SmartFit can translate them in DRL files, which can be then executed by the Drools engine.

In Fig. 11 is reported the final translation in DRL matching the rules generated by the user that can be executed by the Drools engine. The window:time are used for collecting events of interest by defining a time-window (three days in the examples that is 72 h). The engine will automatically disregard any sensor reading older than specified time-windows and keep the collected values consistent. The command: this before [0m,5m] $calories (in bold in the code) is used for translating conditions that need to be performed on events in the stream that are ordered by a timestamp. The parameter [0m,5m] means that the previous pattern will match if and only if the temporal distance between the time when $steps finished and the time when $calories started is maximum 5 min. In our case, the fact that the number of steps is taken into account only before the timestamp associated to the attribute $calories.

The outcome of the use of the system is a translation into DRL that can be executed by the Drools engine. Once a rule is created, it is stored in a repository for further re-use or for sharing it among members of a community of trainers and coaches. In this way, it is easy for other users to customize existing rules according to their needs.

7. Evaluation

To evaluate the SmartFit Rule Editor, a user test has been carried out. It is important to underline that this experiment was not intended to evaluate the tool performances in converting the rules into DRL, but to evaluate and discuss the approach for visual and interactive creation of rules in a Web-based system. The user test consisted asking 10 participants (HCI experts) to create a CompositeRule following a set of tasks specifications. Since the goal was not to measure performance in terms of time of execution, we did not collect times and duration of the single tasks but we focused on the interaction with the element of the interface. For better understanding the opinion of the participants, we collected information about their profile through an initial and a final questionnaire. The initial questionnaire was devoted at collecting demographic information about the participants, while the final one was composed of three different parts:

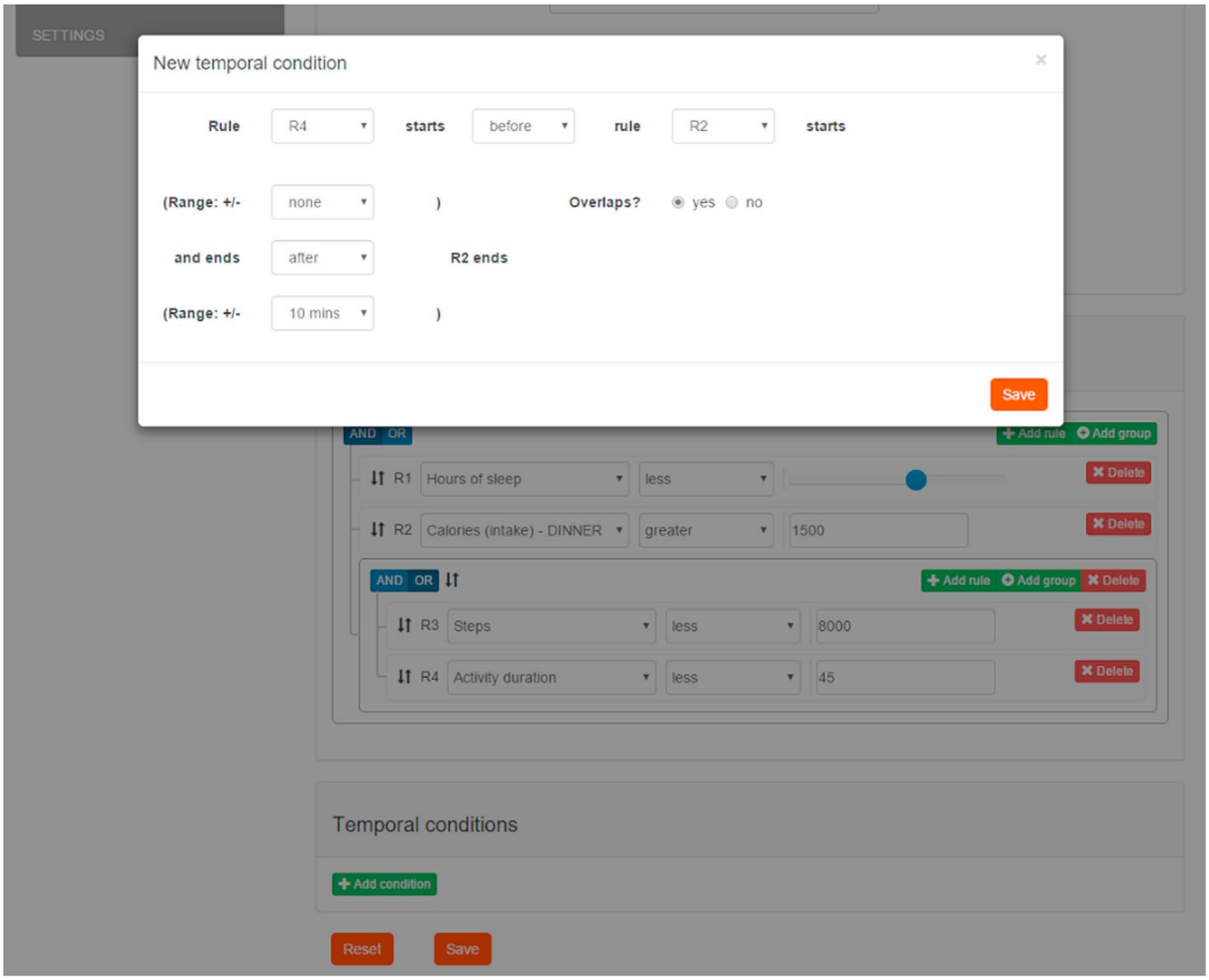

Fig. 10. Another example of temporal condition created in the Rule Editor system.

1. *SUS* (*System Usability Scale*) [43]: A 10-item attitude Likert scale (see Fig. 12).
2. *CUSQ* (*Computer Usability Satisfaction Questionnaire*) [44]: A 19-item attitude Likert scale (see Fig. 13).
3. *UEQ* (*User Experience Questionnaire*) [45]: A 26-item attitude Likert scale that allows to measure both usability (efficiency, perspicuity, dependability) and user experience (originality, stimulation) aspects (see Fig. 14).

A further unstructured interview with each participant concluded the user test sessions and allowed us to gather further information and suggestions on how to improve the SmartFit Rule Editor environment.

7.1. Participants

All the participants had normal color vision and normal or lens-corrected visual acuity. The participants were then welcomed individually and asked to read the Participant Information Sheet and invited to pose any questions about the test objectives and modalities of conduction. Once the participants were satisfied with the answers, they

```
rule "Hours of sleep/Calories intake/physical activities check"
when
       $hours : Number( IntValue < 7 ) from entry-point "hours_sleep_daybefore" over win-
                dow:time( 72h)
       $calories_dinner: calories_intake.calories.num (calories.time_start == "20:00")
       $calories : Number( IntValue > 1500 ) from entry-point $calories_dinner
                over window:time( 72h)
       $steps : Number( IntValue < 8000 ) from accumulate (step_counter(($n_step:
                num_steps.steps) over window:time(24h), sum ($n_step), this before
                [0m,5m] $calories)) over window:time(72h)

then
        // warning
end
```

Fig. 11. The translation of the CompositeRule in DRL.

	Strongly disagree 1	2	3	4	Strongly agree 5
1. I think that I would like to use this system frequently					
2. I found the system unnecessarily complex					
3. I thought the system was easy to use					
4. I think that I would need the support of a technical person to be able to use this system					
5. I found the various functions in this system were well integrated					
6. I thought there was too much inconsistency in this system					
7. I would imagine that most people would learn to use this system very quickly					
8. I found the system very cumbersome to use					
9. I felt very confident using the system					
10. I needed to learn a lot of things before I could get going with this system					

Fig. 12. The System Usability Scale (SUS) questionnaire.

were asked to read and sign the Informed Consent Form. The participants were then invited to sit on a chair positioned in front of a monitor and to fill out the pre-test demographic questionnaire. The characteristics of the participants are summarized in Figs. 15 and 16.

7.2. Protocol

The participants have been asked to perform the following tasks using data that we provided:

1. Set up the general settings of a CompositeRule.
2. Creation of three rules with the editor.
3. Creation of two temporal conditions to put in relation two of the three rules created before.

The user test has been performed with a think-aloud protocol and with the presence of an observer (direct observation). We decided not to measure time of task executions but to focus on the task performance and the eventual problems encountered.

7.3. Results

All participants completed successfully the user test in a reasonable time, i.e. less than 5 min (even if we did not use time as metric), creating correct rules and temporal conditions. Some problems have been highlighted during the test and have been confirmed by the unstructured interviews and the final questionnaires.

7.3.1. SUS

The result of SUS questionnaires is 72.5, which is above 68, the result considered by SUS method as the average. Therefore, the results of the SUS questionnaire are above average, even if the results highlight that the usability of the system still needs improvements. The average of the positive questions answers is 3.72, which gives indication of the positive attitude of the users in respect with the system and its features. The average of the negative questions is 1.92, which is a confirmation that work can be done on the system and its interface to improve the overall usability. Among the positive questions, the two that scored the higher results are “I think that I would like to use this system frequently” (average result: 4) and “I thought the system was easy to

	Strongly disagree 1	2	3	4	Strongly agree 5
1. Overall, I am satisfied with how easy it is to use this system					
2. It was simple to use this system					
3. I can effectively complete my work using this system					
4. I am able to complete my work quickly using this system					
5. I am able to efficiently complete my work using this system					
6. I feel comfortable using this system					
7. It was easy to learn to use this system					
8. I believe I became productive quickly using this system					
9. The system gives error messages that clearly tell me how to fix problems					
10. Whenever I make a mistake using the system, I recover easily and quickly					
11. The information (such as online help, on-screen messages, and other documentation) provided with this system is clear					
12. It is easy to find the information I needed					
13. The information provided for the system is easy to understand					
14. The information is effective in helping me complete the tasks and scenarios					
15. The organization of information on the system screens is clear					
16. The interface of this system is pleasant					
17. I like using the interface of this system					
18. This system has all the functions and capabilities I expect it to have					
19. Overall, I am satisfied with this system					

Fig. 13. The Computer Usability Satisfaction Questionnaire (CUSQ).

	1	2	3	4	5	6	7	
annoying	O	O	O	O	O	O	O	enjoyable
not understandable	O	O	O	O	O	O	O	understandable
creative	O	O	O	O	O	O	O	dull
easy to learn	O	O	O	O	O	O	O	difficult to learn
valuable	O	O	O	O	O	O	O	inferior
boring	O	O	O	O	O	O	O	exciting
not interesting	O	O	O	O	O	O	O	interesting
unpredictable	O	O	O	O	O	O	O	predictable
fast	O	O	O	O	O	O	O	slow
inventive	O	O	O	O	O	O	O	conventional
obstructive	O	O	O	O	O	O	O	supportive
good	O	O	O	O	O	O	O	bad
complicated	O	O	O	O	O	O	O	easy
unlikable	O	O	O	O	O	O	O	pleasing
usual	O	O	O	O	O	O	O	leading edge
unpleasant	O	O	O	O	O	O	O	pleasant
secure	O	O	O	O	O	O	O	not secure
motivating	O	O	O	O	O	O	O	demotivating
meets expectations	O	O	O	O	O	O	O	does not meet expectations
inefficient	O	O	O	O	O	O	O	efficient
clear	O	O	O	O	O	O	O	confusing
impractical	O	O	O	O	O	O	O	practical
organized	O	O	O	O	O	O	O	cluttered
attractive	O	O	O	O	O	O	O	unattractive
friendly	O	O	O	O	O	O	O	unfriendly
conservative	O	O	O	O	O	O	O	innovative

Fig. 14. The User Experience Questionnaire (UEQ).

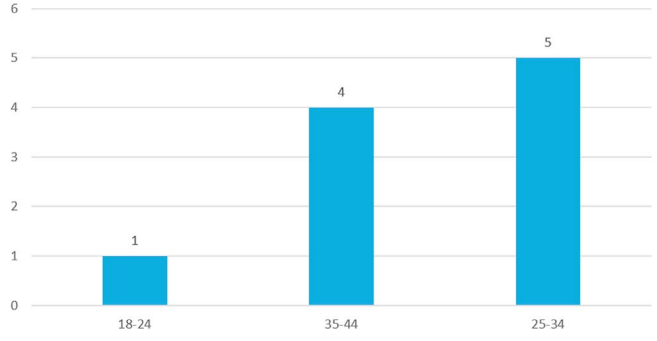

Fig. 15. The age groups of the test participants.

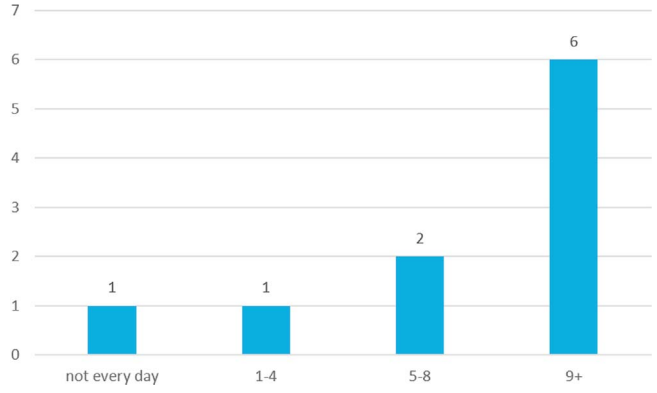

Fig. 16. The hours of use of computers by the test participants.

use" (average result: 3.9). This suggests that the participants recognized the utility of the system and were not afraid of using it. Among the negative questions, the one that resulted in the lowest average is "I needed to learn a lot of things before I could get going with this system" (average result: 1.4). This gives us hints about the easiness to use the system but teaches us that more can be done to improve the results.

7.3.2. CUSQ

The CUSQ questionnaires result is very positive, according to all metrics used by the CUSQ model: System Usefulness (SYSUSE), Information Quality (INFOQUAL), Interface Quality (INTERQUAL), and Overall Satisfaction (OVERALL). For all the factors, the results average is 4 (on a scale from 1 to 5). The questions with the higher results (both 4.2/5) are "It was easy to learn to use this system" and "The interface of this system is pleasant". The questions with the lowest results (both 3/5) are "The system gives error messages that clearly tell me how to fix problems" and "The information (such as online help, on-screen messages, and other documentation) provided with this system is clear". This reflects perfectly the comments that were collected during the final unstructured interview that are discussed later in this section. These two results reinforce the belief that one major improvement in the system design would be the creation of help tools, tooltips and an initial tutorial to help the users in speeding up their learning curve and to help them in fixing errors that may happen during the use of the system.

7.3.3. UEQ

In the UEQ questionnaire the items are on the Likert scale opposites are two concepts with opposite meaning. The items constitute six different scales that are aimed at offering results categorized into attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty. The six scales can be structured as illustrated in Fig. 17.

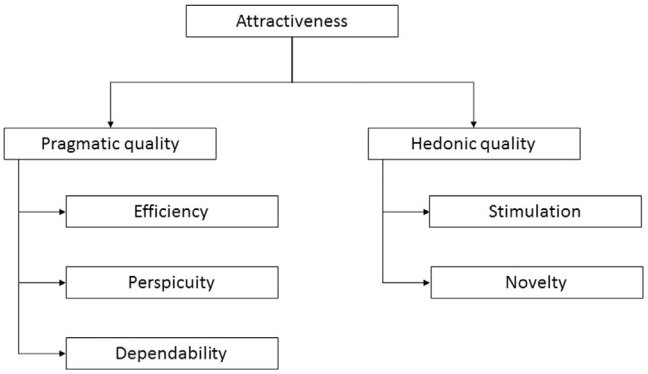

Fig. 17. The structure of the six scales of UEQ.

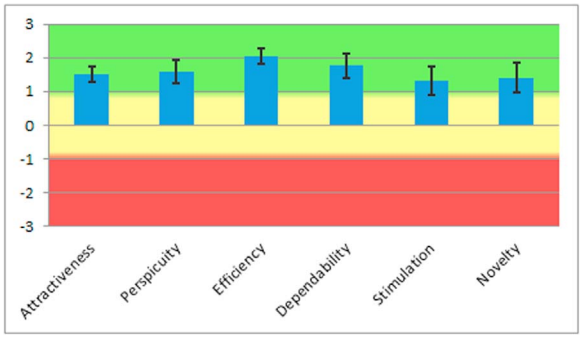

Fig. 18. The chart of the results of UEQ evaluation.

While attractiveness can be seen as a standalone value, perspicuity, efficiency, and dependability can be grouped in a Pragmatic quality dimension (that is focused on goal), and stimulation and novelty are grouped in Hedonic quality dimension (that is, not goal-focused).

We used UEQ to test if the Rule Editor system has sufficient user experience in its current implementation. As can be seen in Fig. 18, and according to the UEQ methodology, all scales show a positive evaluation, i.e. they are greater than 0.8. The values for the UEQ scales are reported in Fig. 20 (Column *Mean*).

UEQ methodology suggests to focus on a restricted range of results, because of the extreme unlikeliness of values above +2 and below −2. The chart with the restricted scale is shown in Fig. 19.

Particularly, the items in the questionnaire that scored the highest results were part of the Efficiency scale: “Fast” and “Efficient”. The confidence intervals for the scales are reported in Fig. 20.

7.3.4. Unstructured interviews

During the unstructured individual interviews that we took after each user test, some important suggestions for improving the system emerged:

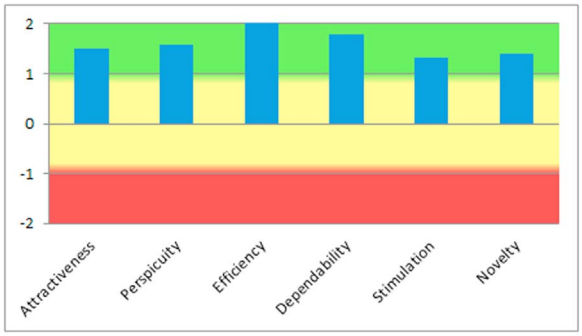

Fig. 19. The chart of the results of UEQ evaluation with the scale restricted to +2/−2.

- The organization of the page suggests the users to divide the creation of the CompositeRule into 3 distinct steps: general settings, rules editing, temporal conditions editing. Specifically, a split navigation interaction design pattern[17] is suggested. As described by one of the participants:

  “*Splitting the rules and temporal conditions creation in more steps would help the user in navigating the system. I would suggest to divide the process into three steps.*”
- It would be better to introduce some tooltips, help icons, and eventually an initial tutorial to better explain to the user all the features that the system offers. Specifically about the tutorial, one of the participants said:

  “*An initial tutorial could be useful to teach the user how to use the system. I learned it fast but it took me some time to get the entire flow of the interaction.*”
- The rules that are created with the editor are automatically assigned with an ID with the format R < number>. This ID is used in the temporal condition editing for referring to specific rules created before. Two of the participants suggested to allow the user in creating natural language names to denote the created rules, to help the user in recognizing the rules to be used for the creation of the temporal conditions:

  “*It could be an idea to let the user assign a name to the rule to improve the selection of rules during the temporal conditions creation phase.*”

  “*You should consider to allow the user to chose short names for the rules to make it easier the creation of temporal conditions.*”
- One of the participants pointed out that the buttons that allow the use of logical operators on the rules (AND and OR) are not very visible so as the direct manipulation feature available for reordering the rules is not:

  “*I would put more in evidence the buttons of the logical operators and the feature of drag-and-drop of the rules.*”

8. Conclusion

8.1. Discussion

The design of the Rule Editor stems from the study of current solutions based on the use of visual strategies for designing workflows or business rules without requiring any knowledge of software engineering nor programming from the end users. Recent EUD researches in this problem space have highlighted several approaches ranging from mash-up techniques to visual programming strategies based on the use of “spreadsheets” or “pipes”. One thing that is consistent in all these solutions is that the programming is still the central metaphor. On the contrary, steps should be taken for helping users in expressing tasks based on their working settings by using a set of conditions that if met have to trigger suitable actions. This emphasizes the need to take care of communication process that takes place among all the stakeholders in a participatory design process, exploiting on the participation of domain experts in the knowledge management design, allowing successful communication processes design. Specifically, this process is enabled by the means of the design of usable and effective interfaces, that according to semiotic engineering theory [27,28] can be seen as composed of messages (embodying the implicit information). Systems such as IFTTT and Atooma are going in the right direction but we need a usable and effective interfaces, that according to the semiotic engineering theory will be able to fit the domain expert's mental model and tacit knowledge, therefore allowing the establishment of a correct interpretation and semiosis process by all the stakeholders. In other words, we need an interface that by using a visual notation can support coaches and

[17] http://www.welie.com/patterns/showPattern.php?patternID=split-navigation

	Mean	Std. Dev.	N	Confidence	Confidence interval	
Attractiveness	1.500	0.377	10	0.234	1.266	1.734
Perspicuity	1.575	0.553	10	0.343	1.232	1.918
Efficiency	2.050	0.350	10	0.217	1.833	2.267
Dependability	1.775	0.595	10	0.368	1.407	2.143
Stimulation	1.325	0.667	10	0.414	0.911	1.739
Novelty	1.400	0.709	10	0.439	0.961	1.839

Fig. 20. Confidence intervals (p=0.05) per scale.

trainers' decision making through the use of a formal language for creating complex rules in which, due the nature of the IoT data, the temporal conditions have to be taken in greater account.

8.2. Future developments

We are currently working on several activities and research issues related to the SmartFit.

First we are going on with tests to evaluate the validity-in-practice of the system features in order to extend, amend or otherwise recommend how they can be improved or why they are inappropriate. At the moment, we are monitoring the use of our system in the context of activities of a teenage soccer team during its seasonal championship. We provided the soccer team with Fitbit trackers for collecting biological, activity, and sleep-related data and coaches and trainers with SmartFit Framework for allowing them to keep track of events related to athletes' habits, in order to understand the variety of the team members and find successful schemes of training. At the same time, we are working on research studies to investigate new rule storage strategies. In the current prototype, all rules are stored according to a streamed representation into the database. Therefore, fetching all rules associated with a request carried out by a user requires accessing a single, even though very large, tuple. In the current version all rules are fetched, even those that are not enabled. We thus plan to evaluate alternative strategies to improve the performance of the rule selection process.

A second research direction deals with developing an ontology-based representation of our Rule Language to enhance interoperability.

A third research direction is related to the development of a comprehensive authorization model for SmartFit and to investigate security and dependability of the proposed system.

We also will investigate the consequences that the sharing of composite rule may have on the meta-communication process among coaches, especially regarding the opportunity of assigning natural language descriptions of the composite rules to enable the coaches to better understand the goal of the rules use.

Finally, a last research direction is focusing on the extension of the scope of the conditions for embracing fuzzy aspects. This extension aims at using linguistic values of the fuzzy conditions for expanding their interpretation context. A linguistic variable is represented by a quintuple of $\langle v, T, X, g, m\rangle$ where v is name of the linguistic variable, T is the set of linguistic terms applicable to variable v, X is the universal set of values, g is the grammar for generating the linguistic term, m is the semantic rule that assigns to each term $t \in T$, a fuzzy set on X. To illustrate our approach, consider a sleep monitor. Let v represent a linguistic variable with a graphical distribution based on four parameters. Very_Low for hours of sleep is represented using trapezoidal function as Very_Low (2 h, 3 h, 4 h, and 6 h). Current IoT applications use simple statements such as “Hours of sleep $\leq$3 hours” to indicate when the value is very low. Using our extension of the Rule Language, users can specify the statement “Quantity = $Very_Low”: a set of values are related to “$Very_Low” in this comparison, rather than one single value. Fulfillment threshold is allowed to specify the condition with a degree value in the range of [0, 1]. For example, we can use 0.75 as the threshold to indicate that the value of hours of sleep is very low with the degree of 0.75. As a result, a value in the range [2.75, 4.5] indicates that “the number of hours of sleep is very high with a threshold of 0.75”. This strategy will further increase the efficiency of our approach in order to balance the human and the machine empowerment by adopting a EUD paradigm that will allow us the possibility to assist Communities of Interest in completing their daily tasks in an easier way.

8.3. Conclusive remarks

This paper describes a methodology to graphically model rules by using a visual notation able to facilitate team sport members in taking under control their athletes' physical activities and their nutrition and sleep behaviors. The Rule Editor is part of the SmartFit Framework that aims at gathering, computing, and diffusing data originated and streamed by physical and social IoT devices, sensors, and applications. The cloud architecture of the Framework has been described and it is composed of three environments: one for supporting IoT engineers in configuration the network of sensors and services from which the data are collected, a second environment is the Rule Editor and finally a third one that can be used by athletes for monitoring their own behavior and lifestyle and to share their data with the other athletes in their team. After presenting strategies of EUD in the IoT context, aimed at giving the end users more freedom and power to assemble different data sources/sensors in an ad hoc and personalized solution, we described the implementation of the SSW methodology for the EUD in the IoT and specifically for what concerns the design of the Rule Editor which aims to support coaches and trainers in creating rules in the eWellness domain. The paper also presents a Rule Language developed for the eWellness domain. Such language allows coaches and trainers to express complex rules and temporal constraints through a visual interface that the evaluation described in Section 7 reports to be easy to use, efficient and fitting the user' mental model.

Specifically, the evaluation highlights positive attitude of the users in respect with the system and its features and the fact that participants recognized the utility of SmartFit and were not afraid of using it. Other valuable features of SmartFit are the possibility to translate the rules in DRL files and to store them in a cloud computing data center for supporting a sharing policy among members of the team sport communities.

References

[1] E. Wenger, Communities of Practice. Learning, Meaning, and Identity, Cambridge University Press, Cambridge, UK, 1998.
[2] W.M. Van Der Aalst, A.H. Ter Hofstede, M. Weske, Business process management: a survey, in: Business Process Management, Springer, 2003, pp. 1–12.
[3] S. Lukichev, G. Wagner, Uml-based rule modeling with Fujaba, in: Proceedings of the 4th International Fujaba Days, 2006, pp. 31–35.
[4] M. Petre, A. Blackwell, Children as unwitting end-user programmers, in: Proceedings of VL/HCC 2007, 2007, pp. 239–242.
[5] M.F. Costabile, P. Mussio, L. Parasiliti Provenza, A. Piccinno, End users as unwitting software developers, in: Proceedings of the 4th International Workshop on End-User Software Engineering, ACM, New York, NY, USA, 2008, pp. 6–10.
[6] M.F. Costabile, P. Mussio, L.P. Provenza, A. Piccinno, Advanced visual systems supporting unwitting EUD, in: Proceedings of the Working Conference on Advanced Visual Interfaces, ACM, New York, NY, USA, 2008, pp. 313–316.
[7] B.R. Barricelli, A. Marcante, P. Mussio, L.P. Provenza, S. Valtolina, G. Fresta, Banco: a web architecture supporting unwitting end-user development, IxD & A 5 (2009) 23–30.
[8] M. Costabile, D. Fogli, P. Mussio, A. Piccinno, Visual interactive systems for end-user development: a model-based design methodology, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 37 (6) (2007) 1029–1046.
[9] H. Lieberman, F. Paterno, V. Wulf (Eds.), End User Development, Springer, 2006.
[10] A. Sutcliffe, End-user development, Commun. ACM 47 (9) (2004) 31–32.
[11] G. Fischer, Communities of interest: learning through the interaction of multiple knowledge systems, in: Proceedings of the 24th IRIS Conference, vol. 2001, Department of Information Science, Bergen, 2001.
[12] A.J. Ko, B.A. Myers, H.H. Aung, Six learning barriers in end-user programming systems, in: 2004 IEEE Symposium on Visual Languages and Human Centric Computing, IEEE, Washington, DC, USA, 2004, pp. 199–206.
[13] V. Pipek, M.-B. Rosson, V. Wulf, End-User Development: 2nd International Symposium, IS-EUD 2009, Proceedings, vol. 5435, Siegen, Germany, March 2–4, 2009, Springer, Berlin Heidelberg, Germany, 2009.
[14] K. Kaczor, G.J. Nalepa, Ł. Łysik, K. Kluza, Visual design of drools rule bases using the xtt2 method, in: Semantic Methods for Knowledge Management and Communication, Springer, Berlin Heidelberg, Germany, 2011, pp. 57–66.
[15] K. Kaczor, G.J. Nalepa, Critical evaluation of the xtt2 rule representation through comparison with clips, Knowl. Eng. Softw. Eng. (KESE8) (2012) 46.
[16] D.D. Bona, G.L. Re, G. Aiello, A. Tamburo, M. Alessi, A methodology for graphical modeling of business rules, in: 2011 Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS), IEEE, Washington, DC, USA, 2011, pp. 102–106.
[17] A. Namoun, T. Nestler, A. De Angeli, Conceptual and usability issues in the composable web of software services, in: International Conference on Web Engineering, Springer, Berlin Heidelberg, Germany, 2010, pp. 396–407.
[18] F. Casati, How end-user development will save composition technologies from their continuing failures, in: International Symposium on End User Development, Springer, Berlin Heidelberg, Germany, 2011, pp. 4–6.
[19] J. Widom, S. Ceri, Active Database Systems: Triggers and Rules for Advanced Database Processing, Morgan Kaufmann, San Francisco, CA, USA, 1996.
[20] C. Stary, Tadeus: seamless development of task-based and user-oriented interfaces, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 30 (5) (2000) 509–525.
[21] G. Fischer, J. Grudin, R. McCall, J. Ostwald, D. Redmiles, B. Reeves, F. Shipman, Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments, in: Coordination Theory and Collaboration Technology, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 2001, pp. 447–472.
[22] R.R. Penner, E.S. Steinmetz, Model-based automation of the design of user interfaces to digital control systems, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 32 (1) (2002) 41–49.
[23] C.B. Brodie, C.C. Hayes, Daisy: a decision support design methodology for complex, experience-centered domains, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 32 (1) (2002) 50–71.
[24] P.B. Andersen, A Theory of Computer Semiotics: Semiotic Approaches to Construction and Assessment of Computer Systems, Cambridge University Press, Cambridge, UK, 1990.
[25] Y. Dittrich, How to Make Sense of Software – Interpretability as an Issue in Design, Technical Report, University of Karlskrona Ronneby, 1998.
[26] Y. Dittrich, Computer Anwendungen un Sprachlicher Kontext. Zu den Wechselwirkungen Software normaler und formaler Sprache bei Einsatz und Entwicklung von Software, Peter Lang, 1997.
[27] C.S. de Souza, The Semiotic Engineering of Human–Computer Interaction (Acting with Technology), The MIT Press, Boston, MA, USA, 2005.
[28] C.F. Leito, C. de Souza, Semiotic Engineering Methods for Scientific Research in HCI, Morgan and Claypool Publishers, San Rafael, CA, USA, 2009.
[29] S. Valtolina, B.R. Barricelli, Y. Dittrich, Participatory knowledge-management design: a semiotic approach, J. Vis. Lang. Comput. 23 (2) (2012) 103–115.
[30] J.W. Ng, Task as a service: extending cloud from an application development platform to a tasking platform, in: 2015 IEEE World Congress on Services (SERVICES), IEEE, Washington, DC, USA, 2015, pp. 294–301.
[31] B. Ur, E. McManus, M. Pak Yong Ho, M.L. Littman, Practical trigger-action programming in the smart home, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 2014, pp. 803–812.
[32] F. Cabitza, I. Gesso, Reporting a user study on a visual editor to compose rules in active documents, in: Emerging Research and Trends in Interactivity and the Human–Computer Interface, 2014, pp. 182–203
[33] H. Boley, T.M. Osmun, B.L. Craig, WellnessRules: A Web 3.0 Case Study in RuleML-Based Prolog-N3 Profile Interoperation, Springer, Berlin, Heidelberg, 2009, pp. 43–52.
[34] F. Wang, K.J. Turner, An Ontology-Based Actuator Discovery and Invocation Framework in Home Care Systems, Springer, Berlin, Heidelberg, 2009, pp. 66–73.
[35] F. Cabitza, D. Fogli, R. Lanzilotti, A. Piccinno, Rule-based tools for the configuration of ambient intelligence systems: a comparative user study, Multimed. Tools Appl. (2016) 1–21.
[36] B.R. Barricelli, S. Valtolina, Designing for end-user development in the internet of things, in: End-User Development: 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26–29, 2015, Proceedings, Springer International Publishing, 2015, pp. 9–24.
[37] M. Mesiti, L. Ferrari, S. Valtolina, G. Licari, G.L. Galliani, M. Dao, K. Zettsu, StreamLoader: an event-driven ETL system for the on-line processing of heterogeneous sensor data, in: Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15–16, 2016, Bordeaux, France, 2016, pp. 628–631.
[38] M. Mesiti, S. Valtolina, L. Ferrari, M. Dao, K. Zettsu, An editable live ETL system for ambient intelligence environments, in: WF-IoT, 2015, pp. 393–394.
[39] S. Valtolina, B.R. Barricelli, M. Mesiti, End-user centered events detection and management in the internet of things, in: Current Trends in Web Engineering, Springer, Berlin Heidelberg, Germany, 2015, pp. 77–90.
[40] D.B. Rules, What are they really, The Business Rules Group, Formerly, known as the GUIDE Business Rules Project, Final Report, Revision (Online Guide) 1, 2000, pp. 1–77.
[41] J.F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (11) (1983) 832–843.
[42] M. Muller, Pictive: an exploration in participatory design., in: Proceedings of the ACM Conference on Human Factors in Computing Systems, ACM Press, New York, NY, USA, 1991, pp. 225–231.
[43] J. Brooke, SUS: a quick and dirty usability scale, in: P.W. Jordan, B. Weerdmeester, A. Thomas, I.L. Mclelland (Eds.), Usability Evaluation in Industry, Taylor and Francis, London, 1996.
[44] J.R. Lewis, Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use, Int. J. Hum.-Comput. Interact. 7 (1) (1995) 57–78.
[45] M.S. Bettina Laugwitz, Theo Held, Construction and evaluation of a user experience questionnaire, in: A. Holzinger (Ed.), USAB 2008, Lecture Notes in Computer Science, vol. 5298, Springer, 2008, pp. 63–76.