IET Software

Research Article

IET Journals
The Institution of Engineering and Technology

EUDroid: a formal language specifying the behaviour of IoT devices

ISSN 1751-8806
Received on 5th December 2017
Revised 3rd May 2018
Accepted on 30th May 2018
doi: 10.1049/iet-sen.2017.0347
www.ietdl.org

Paolo Buono[1], Fabio Cassano[1], Alessandra Legretto[1], Antonio Piccinno[1] ✉

[1]*Dipartimento di Informatica, Universitá di Bari Aldo Moro, Bari, Italy*

✉ *E-mail: antonio.piccinno@uniba.it*

Abstract: Recent technologies are offering today many possibilities to end users, which ask for continuous support in a variety of situations. Internet of things (IoTs) and the proliferation of smart devices are offering many opportunities that raise the need to standardise protocols for their interoperability and interaction languages for their management. This study proposes EUDroid, a system composed of a mobile application and an IoT device used as a pill reminder to allow the patients to correctly take their prescribed drugs. A web server stores and manages the therapies that can be defined by the end users. The web server also manages the communication between the app and the device. In order to validate the management of the therapies, a formal language has been proposed. It describes the behaviour of different components of the IoT device, such as LEDs or buzzers, and defines when, with which delay, and for how long time a given event will last, to manage technical concepts related to smart devices for supporting them in following therapies more accurately.

1 Introduction

Computer users have rapidly increased in both number and diversity, including managers, accountants, engineers, teachers, scientists and health care workers [1, 2]. Many of these people work on tasks that rapidly evolve on a yearly, monthly, or even daily basis. Consequently, their software needs are different, complex, and frequently changing [3, 4]. Professional software developers hardly cope with such needs because of slow development processes. End-user development (EUD) helps to solve this problem. EUD is defined as 'a set of methods, techniques and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact' [5]. EUD, a discipline that encompasses techniques, methodologies, as well as socio-technical environments that allow end users, as older people and their caregiver, to act as professionals in domains such as Internet of things (IoTs). End users are the real owner of their own context and needs. Moreover, they often have real-time awareness of changes in their respective domains. Through EUD, end users can adapt the software they are using to fit their requirements. In a more recent view of EUD, it not only involves the context of software systems but also encompasses techniques, methodologies, situations, and socio-technical environments that allow end users to act as professionals in domains in which they are not professionals, including IoT [6].

In this article, we address the management of pills for a therapy. The choice of the domain is led by the need of many people to take, on a daily basis, one or more pills, for a given period (or forever). We designed and developed an IoT device that allows the management and taking of pills for the end users of this domain, who are mainly older people, or caregivers, that very often have no or low IT expertise. EUDdroid is composed of three main components: a smartphone app, the IoT device pill reminder, and a web server that manages the user's requests and the therapy scheduling. IoT devices are objects that can be sensed or controlled remotely across an existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit, in addition to reduced human intervention [7]. The main purpose of the project is to create a system that allows the composition and customisation of elementary events, in order to manage a pill-based therapy. The idea to use electronic devices to help older people to take pills is well known in the literature. Many of the existing works, however, do not allow end users to customise the behaviour of the device. Commercial solutions allow users only to define what pill should be dropped according to the calendar, but with limited personalisation possibilities. The final user is not allowed to deeply edit the parameters of the pills reminder and, to the best of our knowledge, no formal language has ever been defined to allow users to customise the device [8]. Moreover, it is not always possible for the end user, to control and edit the pill reminder through the smartphone. The pill reminder is not a new technology. Many researchers have proposed different solutions like proposed in 8–10]. We have been inspired to continue our previous work presented in [11]. This paper is structured as follows: Section 2 illustrates the different approaches provided by task automation tools. The proposed formal language is presented in Section 3. The EUDroid system is illustrated in Section 4. Finally, Section 5 concludes the paper.

2 Task automation tools

EUD methodologies allow users to customise their systems to support personal, situational needs [12, 13]. Non-technical users need to orchestrate ecologies of smart objects to satisfy specific, and sometimes unexpected, needs [14]. Many of the current solutions consist of pre-packaged-specific software for remotely controlling single smart objects that cannot be easily adapted to the requirements deriving from specific domains and contexts of use. The so-called task automation (TA) tools, which combine social services, data sources, and sensors, are also gaining importance. They have become popular as they offer very easy paradigms to synchronise the behaviour of objects and applications [15, 16]. Through Web editors, users can synchronise the behaviour of smart objects (devices generally equipped with sensors capable to detect different types of events occurring in an observed environment) and/or actuators (actions determining a state change in the environment or in the monitored system) by graphically sketching the interaction among the objects [17]. For example, this can be done either by using graphs to represent how events and data parameters propagate among the different objects to achieve their synchronisation, or by defining event condition action (ECA) rules, a paradigm largely used for the specification of active systems [18–20], which in the IoT domain can be fruitfully exploited to express how and when some object behaviours have to be activated in reaction to detected events. Recently, different Web tools have

revisited the ECA paradigm to address the problem of TA [16]. In particular, they support the definition of ECA rules to synchronise the behaviour of smart objects and services. Many of these tools are designed for non-technical people and offer wizard procedures that guide users during the composition process.

Desolda *et al.* [21] presented a detailed review of TA for both web and mobile. The most popular tools for the Web, based on wizards are: (i) if this then that (IFTTT) (https://ifttt.com/, last accessed 3 May 2018), (ii) *elastic.io* (https://www.elastic.io/, last accessed 3 May 2018), (iii) *Zapier* (https://zapier.com/, last accessed 3 May 2018); (iv) *itDuzzit* (http://cloud.itduzzit.com/, last accessed 3 May 2018), (v) *WigWag* (https://www.wigwag.com/, last accessed 3 May 2018). A graph-based representation is also adopted in *Node-RED*, which is meant for professional users since it supports advanced rule customisation by means of nodes, representing control statements, functions (written in JavaScript), and debug procedures. We focus on TA services that also run on mobile devices, as in the case of Resonance-Ai (formerly Atooma) (http://resonance-ai.com/, last accessed 3 May 2018), an app that allows for composing of device functions, Web services, and smart objects connected to a mobile device. The rule composition follows an 'If-Do' paradigm. Multiple triggers, actions and filters can be included in each rule. *AutomateIt* (https://automateitapp.com/, last accessed 3 May 2018) and *Tasker* (https://tasker.joaoapps.com/, last accessed 3 May 2018) support the creation of rules limited to the composition of app and functions available on mobile devices. ECCE toolkit [22] aims at supporting the definition of ecologies of smart objects. This focuses on the way smart objects can connect to a server, so that an ecology of such objects can be set-up and managed. An XML-based language is used for describing the properties of smart objects, then, the corresponding code to handle the smart objects behaviour is generated on the platform server. Connected devices are synchronised by means of the Web server component. The synchronisation consists in coupling some events generated by one object to the operations exposed by other objects.

Manipulation of physical objects was also investigated as a programming paradigm to define smart object behaviours. For example, AutoHAN (http://www.autohan.nl/, last access 3 May 2018) implements a new paradigm for interaction with abstract functions of home appliances through special cubes that act as one-button remote controls. Each cube is devoted to a task or function, and the users can associate such functions by holding one face of a cube against an appliance. The expressive power of the language is achieved by placing two or more cubes next to each other, and instructing AutoHAN to store the configuration of Cubes. Such configuration can then be used to schedule home-automation processes. SiteView exploits the manipulation of physical objects for creating ECA rules [23]. Actions are programmed by placing physical objects, representing, for example appliances, in a world-in-miniature area which is a small scale picture of the active environment. The users can see a rule representation in a rule display and can simulate the rule results in an environment display.

A diverse use of user-defined event-action rules is provided in [24]. An authoring tool supports the development of context-aware cross-device user interfaces (UIs) through the creation of rules in which different types of events can activate actions, indicating that how the UI should adapt to the detected context. The tools described above cover several relevant aspects of TA. However, in relation to the composition paradigm, which represents the main aspect addressed by this work, their potential benefits are still limited [25]. On one hand, tools such as IFTTT, elastic.io, Zapier, and itDuzzit address non-technical users only assisting users in the creation of 'basic' rules, i.e. rules that synchronise one event with an action and do not include any additional conditions for rule activation. This is also true for some research works illustrated in the literature. For example, the approaches reported in [26, 27] allow the reuse of pre-defined 'recipes' or 'schemas of digital experiences'. New rules can be defined by only modifying and adapting pre-defined rules. Thus, end users are not allowed to define their own rules, which can be needed if the pre-defined ones do not accommodate their situational or unexpected needs. On the other hand, tools like Node-RED allow one to create more complex rules, but they also require advanced skills. In order to simplify the user interface and make the rule composition consistent, we developed a formal language that aims to address the pill-tacking therapy.

3 EUDroid formal language

In order to validate the management of the behaviours of the different components of an IoT device (such as LEDs or buzzers, and defines when, with which delay, and for how long time a given event will last), a formal language has been adopted. A formal specification can describe unambiguously the behaviour of a user interface and may help the designer to easily extend the functionalities of the app, making sure that such functionalities will not be in contrast with the previous behaviour. Once the behaviour of the system is defined (in terms of syntax and semantics), new widgets or features can be added by adding their semantics. In the following, we formally describe the basic components and the logic of the system. We have developed the EUDroid language, getting inspired by a previous work [28].

The EUDroid formal language has two parts:

- The first part involves the electronic parts. As the language converts rules into commands, the devices need to be working and reachable by the controller part of the system.
- The second part involves the ECA rules description which associate the logic/EUD part to the electronic and the actuators.

An 'elementary component' is an electronic device that receives input or provides output. The elementary components are characterised either as sensors or as actuators. Sensors mostly collect and analyse information while actuators perform operations or 'carry out a behaviour'. In our case study, the elementary components are associated to further information:

- Delay (DY), which is the time (expressed in seconds) that the component has to wait before activate itself.
- Duration (DU), which is the time (expressed in seconds) the component keeps active before it turns off.

Actuators are electronic parts that are able to interact with the world (e.g. stepper motors, fans, buzzers). The case study presented in this article includes five actuators:

- red LED (LR)
- blue LED (LB)
- green LED (LG)
- yellow LED (LY)
- buzzer (B)

The LED colour might have a different meaning for the user, thus each of them is going to be considered as a separate entity.

Due to the electronic logic, each of the described actuators (but the buzzer in the proposed case study) has two different states:

- High level means that the actuator is active and
- Low level means that the actuator is its inactive state.

The buzzer is in the active state (it emits sound) when the level is low, while it is in the inactive state (mute) when the level is high.

Another elementary output component is the message (M). It is considered as an output component (because it comes out from the system) and it appears as a text string on some output devices (such as the mobile app). The text to be included in the message M is set by the user in the app settings.

Definition 1: AT defines the whole actuator set. In the proposed case study $\mathrm{AT} = \{\mathrm{LR}, \mathrm{LB}, \mathrm{LG}, \mathrm{LY}, B, M\}$.

This set can be extended according to the available devices in the user home environment.

The main purpose of our formal language is to create a general tool that allows all the devices to be controlled by the app. In order to define the formal language, we need to introduce some concepts

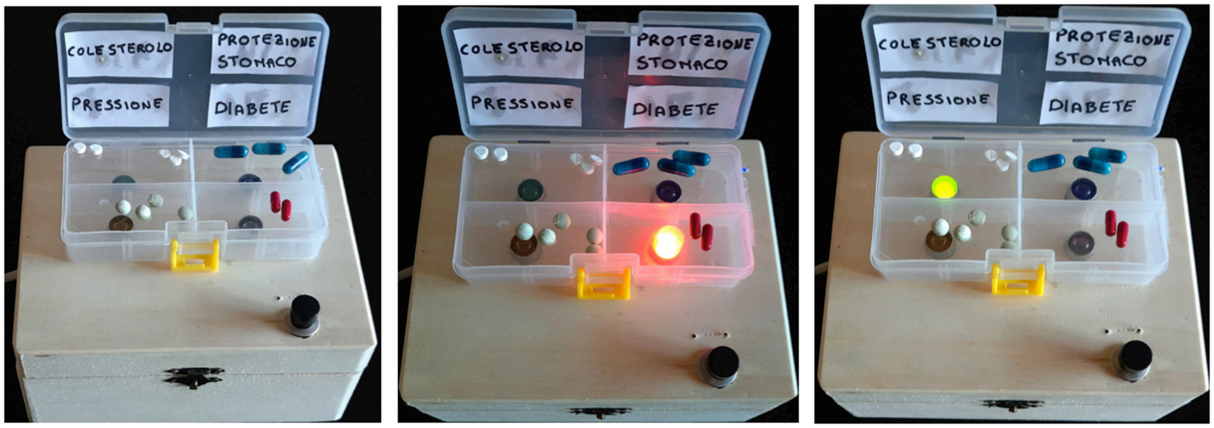

Fig. 1 *Pill reminder prototype in the following status: off, diabetes pills, and pressure pills*

about the 'Event, Trigger, Action', a paradigm that is going to complete the language used by the device:

Definition 2: An elementary event (e) is formed by an action and a trigger.

Definition 3: An action (A) consists in the execution of operations (i.e. turning on/off elementary components).

Definition 4: A trigger is the logical condition to execute one or more actions.

The user can define a custom event based on some formal characteristics, then he/she can associate one or more actions to be complete if a trigger is launched by the system. More elementary events create a so-called composite event (CE):

Definition 5: A CE is a set elementary events, composed of more trigger-actions relations.

In the reported case study, all the events are related to the calendar and both a 'start' and a 'duration' of the action. The trigger event can be launched by the calendar item. This item is composed of the start date of the trigger (SD) and the execution time (T). So, an elementary component c can be defined as

$$c = \langle x, \mathrm{DY}, \mathrm{DU} \rangle, \quad \text{where} \quad x \in \mathrm{AT}, \mathrm{DY} \in N, \mathrm{DU} \in N \tag{1}$$

For example, $c = \langle \mathrm{LR}, 120, 60 \rangle$ means that the red LED will be turned on after 2 min (120 s) from the activation of a given trigger, and will last for 1 min (60 s).

A sequence of any number of elementary components c, as expressed in (1), can be defined as

$$A = \langle c_1, c_2, \ldots, c_n \rangle \quad \text{where} \quad c_i \in C, 1 \leq i \leq n \tag{2}$$

considering C as the set of elementary components.

As the calendar is at the centre of the logic of the proposed scenario, an elementary event is composed as a pair of trigger action. The formal description of an elementary event is.

Definition 6: The elementary event is described as

$$e = \langle \mathrm{CL}, A \rangle, \quad \text{where} \quad \mathrm{CL} = \langle \mathrm{SD}, T \rangle \tag{3}$$

An evolution of Definition 6 is the definition of a CE:

Definition 7: A CE is defined as

$$\mathrm{CE} = \langle e_1, e_2, \ldots, e_n \rangle \quad \text{with} \quad e_i \in E, 1 \leq i \leq n \tag{4}$$

where E is the set of elementary events.

The Arduino controller is able to read and interpret the commands from the web server. Each command cm has a fixed grammar.

Definition 8: The Arduino controller reads and interprets the commands sent in the form:

$$\mathrm{cm} = \langle p, s, \mathrm{DY}, \mathrm{DU} \rangle \tag{5}$$

where p defines the pin position on the Arduino board and s the state (high or low) of the elementary component.

4 Proposed solution

In this section, we are going to show some case studies with which the EUD formal language can be used and applied. The proposed formal language can be easily extended according to the different actuators available and connected to the home environment. EUDroid can be used in different case studies. For example, by changing the actuator's behaviour, it allows the control of one or more window blinds according to the user preferences. In this kind of case study, at a given time (e.g. when the users have to wake up), the blinds are risen in the same moment the user alarm clock rings.

To prove how the formal language works, we have developed a pill reminder to improve people's pill therapy. The pill reminder prototype is composed of:

- a pill reminder box with the actuators (LEDs, buzzer, button) which allow the interaction with the user;
- a web server that stores the information and checks if a specific event has occurred; and
- an app to specify the therapy and the pillbox behaviour.

4.1 Pill reminder IoT device

The web server hosts a database with the ID of the pill reminder and the available commands for the device. In this case study, the device, based on Arduino, has three tasks: (i) checks the Web Server for commands associated to its code; (ii) if there are some it reads the first event and run it according to the delay and duration, then it reads the others, in turn; (iii) if the user presses the button, the device resets the status of the pill reminder. It stores the information sent through the app and listen for occurrences of specific events, to send a trigger to the pill reminder, and receives (and process) the button push event, to notify when a pill has been taken. In case of more than one pill to be taken together the user first takes all pills then presses the button. Different behaviours can be set by the user in the app. As shown in Fig. 1, the physical prototype has currently four compartments for pills. At the bottom right, there is the button to reset the device status when the pill (or the pills) is taken. It is the only user input on this device and, when it is pressed, its status changes to 'LOW', deactivating all the actuators. At the top right there is a small hole for the buzzer (Fig. 1).

The communication between the device and the Web Server is performed through the local area network of the user's home.

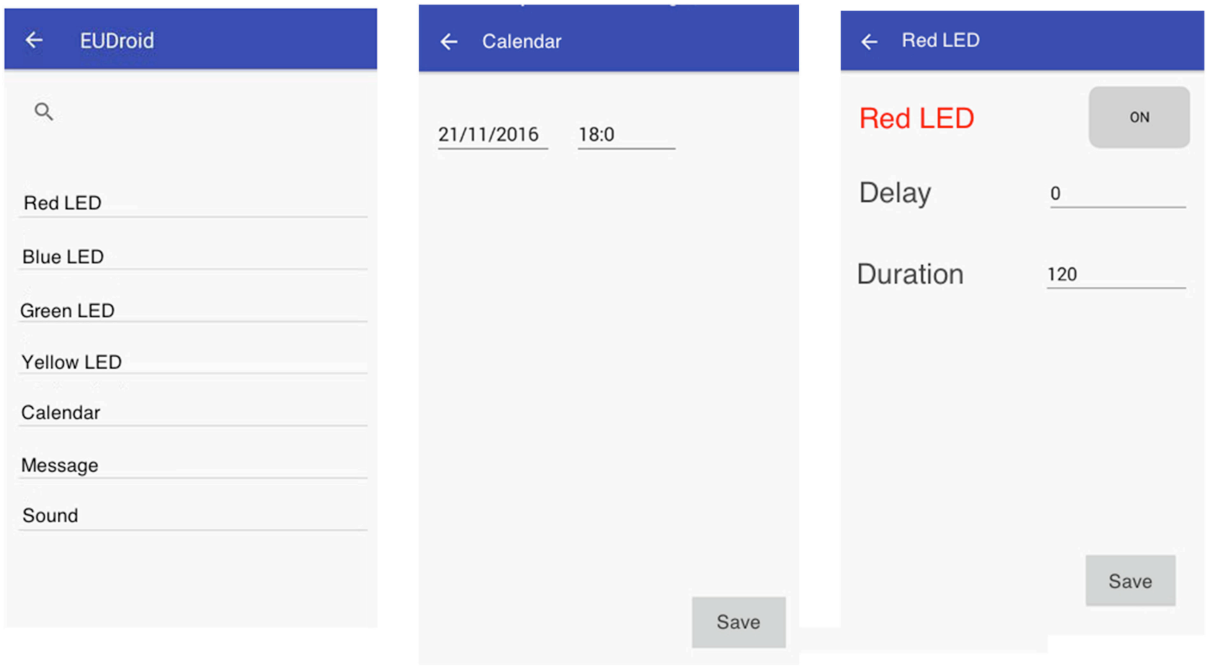

Fig. 2 *EUDroid user interface*
(a) EUDroid app main window, ***(b)*** Date and time specification, ***(c)*** Red LED settings

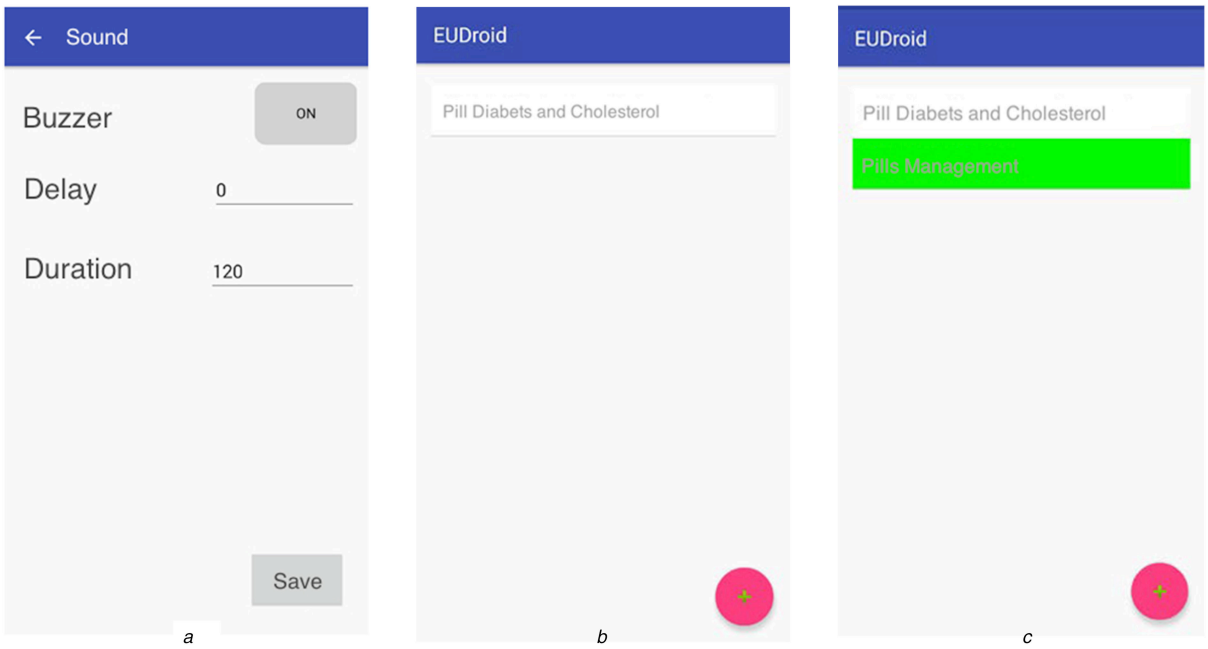

Fig. 3 *Screenshots of the app's UI*
(a) Buzzer setting with delay and duration, ***(b)*** 'Pill diabetes and cholesterol' are ready, ***(c)*** 'Pills management' therapy is activated

4.2 EUDriod android App

The main purpose of this app is to allow the user to create one or more elementary events as described in Section 3. By creating multiple elementary events, it is possible to create one or more composite events. For the sake of the simplicity, we have decided to report the working example of the app using the pill reminder box; however, the app behaviour is general and can be associated to other case studies. There are various elementary target events that can be chosen by the user: LED, calendar, message, sound (Fig. 2*a*). The user's approach to the application follows the ECA paradigm where, if a specific condition has occurred in an event, the system reacts, activating an action [29]. In Fig. 2*b*, the user selected a 'Calendar' trigger (which is the condition the system checks) and, then, the action to take (Fig. 2*c*). Thus, in the app the user can, step by step, create his/her own rules. Those are sent to the web server and converted according to the formal language described in Section 3 and (when needed) converted into command and sent to the target device. The communication between the app and the server is two way, thus the app first checks if any event for the pill therapy (or therapies) is already stored inside the web server. The user can associate, to a single specific event (e.g. two or more pills to be taken at the same time), multiple actions by the pill reminder. As shown in Figs. 3*b* and *c*, once the event has been created, a type of pill can be associated. The user has to choose which LED is going to be turned on for each pill. This piece of information is sent and coded into the command form shown in Definition 1 and its scheduling is managed in the same way by the web server.

The interaction between the box and the app is mediated by the web server as shown in Fig. 4. The web server acts like a middleware for the entire system. Each new rule created by the user is going to be stored inside the web server's database (Fig. 4,

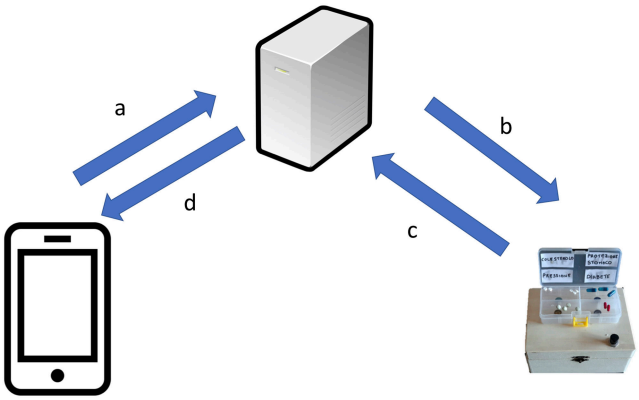

Fig. 4 *Communication between the pillbox and the Android app uses the web server as middleware*

arrow ‘a’). The check that the rule is correctly created according the formal language is made directly in the app. The web server then constantly controls and schedules all the rules. When an event occurs it sends a command to the pill reminder box, which activates all the needed actuators (Fig. 4, arrow ‘b’). In the presented case study, the only ‘feedback’ coming from the device is the button pressed by the user. The activation signal of the button stops the server's rules used to check that the user have correctly taken the pill (Fig. 4, arrow ‘c’). This action avoids any further reminder of the current event and the conditions for the next event are then considered. After a given amount of time, if no interaction has occurred between the user and the pill box reminder, the server will start the ‘reminder procedure’ which involves the activation of a buzzer and (if nothing happens) the sending of a message to the user's smartphone (Fig. 4, arrow ‘d’). For example, the buzzer can be activated (typically as consequence of the missing pill assumption) at the same time of the red LED.

5 Conclusions

In this paper, we have proposed EUDroid, a formal language that allows users to manage the devices behaviour in the home environment. We have developed a pill reminder system as an example of the formal language potentiality and the ECA rules. The system is composed of three parts: a web server that stores the information of the rules and converts them into commands for the pill reminder device, an app that support the user in the creation of the rules which the pill reminder must obey and, finally, the pill reminder device (with all the actuators for each pill therapy).

As future work, a formal evaluation with users must be performed. The pill reminder has only four fixed boxes for the pill therapy; this means that for those people having the need to take more than four pills, two (or more) pillboxes are needed. To overcome this problem, a modular pill reminder box is going to be implemented. There is no control that the user has effectively taken the pill. Once the button on the pill reminder is pressed, it only goes back to the ‘waiting’ state, sending a message to the web server that the pill has correctly been taken.

The formal language can be extended to address further needs. Currently, the device setup is made manually. An automatic recognition of the device and an automatic connection to the app are under development.

6 References

[1] Burnett, M.M., Scaffidi, C.: ‘End-user development’, in Zahirovic, A., Lowgren, J., Carroll, J. (Eds.): ‘*Encyclopedia of human-computer interaction*’ (2011)

[2] Scaffidi, C., Shaw, M., Myers, B.: ‘Estimating the numbers of end users and end user programmers’. 2005 IEEE Symp. Visual Languages and Human-Centric Computing, Dallas, TX, USA, 2005, pp. 207–214

[3] Cabitza, F., Fogli, D., Piccinno, A.: ‘Fostering participation and co-evolution in sentient multimedia systems’, *Journal of Visual Languages and Computing*, 2014, **25**, (6), pp. 684–694

[4] Fogli, D., Piccinno, A.: *Co-evolution of End-User Developers and Systems in Multi-tiered Proxy Design Problems, Lecture Notes in Computer Science*, (Springer, Berlin Heidelberg, 2013), **7897**, pp. 153–168

[5] Lieberman, H., Paternò, F., Klann, M., *et al.*: ‘End-user development: an emerging paradigm’, in Lieberman, H., Paternò, F., Wulf, V. (Eds.): ‘*End user development*’ (Springer, Dordrecht, Netherlands, 2006), pp. 1–8

[6] Fischer, G., Fogli, D., Piccinno, A.: ‘Revisiting and broadening the meta-design framework for end-user development’, in Paternò, F., Wulf, V. (Eds.): ‘*New perspectives in enduser development*’ (Springer, Cham, Germany, 2017), pp. 61–97

[7] Bartolomeo, M.: ‘Internet of things: science fiction or business fact’. A Harvard Business Review Analytic Services Report, Technical Report, 2014

[8] Crema, C., Depari, A., Flammini, A., *et al.*: ‘A smartphone-enhanced pill-dispenser providing patient identification and in-take recognition’. 2015 IEEE Int. Symp. Medical Measurements and Applications (MeMeA), Torino, Italy, 2015, pp. 484–489

[9] Dong, H., Vanns, N.: ‘Designing an innovative pill dispenser: an undergraduate level case study of inclusive design’, *Design J.*, 2009, **12**, (1), pp. 95–115

[10] Yang, G., Xie, L., Mäntysalo, M., *et al.*: ‘A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box’, *IEEE Trans. Ind. Inf.*, 2014, **10**, (4), pp. 2180–2191

[11] Buono, P., Cassano, F., Legretto, A., *et al.*: ‘A homemade pill dispenser prototype supporting elderly’. Int. Conf. Web Engineering, Rome, Italy, 2017, pp. 120–124

[12] Ardito, C., Buono, P., Costabile, M.F., *et al.*: ‘End users as co-designers of their own tools and products’, *J. Vis. Lang. Comput.*, 2012, **23**, (2), pp. 78–90

[13] Costabile, M.F., Fogli, D., Mussio, P., *et al.*: ‘Visual interactive systems for end-user development: a model-based design methodology’, *IEEE Trans. Syst. Man Cybern. A, Syst. Humans*, 2007, **37**, (6), pp. 1029–1046

[14] Daniel, F., Matera, M., Weiss, M.: ‘Next in mashup development: user-created apps on the web’, *IT Prof.*, 2011, **13**, (5), pp. 22–29

[15] Lucci, G., Paternò, F.: ‘Analysing how users prefer to model contextual event-action behaviours in their smartphones’. Int. Symp. End User Development, Madrid, Spain, 2015, pp. 186–191

[16] Piccinno, A., Fogli, D., Lanzilotti, R., *et al.*: ‘Supporting end users to control their smart home: design implications from a literature review and an empirical investigation’, *J. Syst. Softw.*, in print

[17] Atzori, L., Iera, A., Morabito, G.: ‘The internet of things: A survey’, *Comput. Netw.*, 2010, **54**, (15), pp. 2787–2805

[18] Ceri, S., Daniel, F., Matera, M., *et al.*: ‘Modeldriven development of context-aware web applications’, *ACM Trans. Internet Technol. (TOIT)*, 2007, **7**, (1), p. 2

[19] Daniel, F., Matera, M., Pozzi, G.: ‘Managing runtime adaptivity through active rules: the Bellerofonte framework’, *J. Web Eng.*, 2008, **7**, (3), p. 179

[20] Pane, J.F., Ratanamahatana, C., Myers, B.A.: ‘Studying the language and structure in non-programmers’ solutions to programming problems’, *Int. J. Hum.-Comput. Stud.*, 2001, **54**, (2), pp. 237–264

[21] Desolda, G., Ardito, C., Matera, M.: ‘Empowering end users to customize their smart environments: model, composition paradigms, and domain-specific tools’, *ACM Trans. Comput.-Human Interact. (TOCHI)*, 2017, **24**, (2), p. 12

[22] Bellucci, A., Díaz, P., Aedo, I., *et al.*: ‘Prototyping device ecologies: physical to digital and viceversa’. Proc. 8th Int. Conf. Tangible, Embedded and Embodied Interaction, Munich, Germany, 2014, pp. 373–376

[23] Beckmann, C., Dey, A.: ‘Siteview: tangibly programming active environments with predictive visualization’. Adjunct Proc. UbiComp, Seattle, WA, USA, 2003, pp. 167–168

[24] Ghiani, G., Manca, M., Paternò, F.: ‘Authoring context-dependent cross-device user interfaces based on trigger/action rules’. Proc. 14th Int. Conf. Mobile and Ubiquitous Multimedia, Linz, Austria, 2015, pp. 313–322

[25] Barricelli, B.R., Valtolina, S.: ‘Designing for end-user development in the internet of things’. Int. Symp. End User Development, Madrid, Spain,, 2015, pp. 9–24

[26] Kubitza, T., Schmidt, A.: ‘Towards a toolkit for the rapid creation of smart environments’. Int. Symp. End User Development, Madrid, Spain, 2015, pp. 230–235

[27] Zancanaro, M., Not, E., Petrelli, D., *et al.*: ‘Recipes for tangible and embodied visit experiences’, 2015

[28] Buono, P., Costabile, M.F., Covino, E., *et al.*: ‘A visual tool for multidimensional data analysis’, 2005, pp. 333–338

[29] Almeida, E.E., Luntz, J.E., Tilbury, D.M.: ‘Event-condition-action systems for reconfigurable logic control’, *IEEE Trans. Autom. Sci. Eng.*, 2007, **4**, (2), pp. 167–181