Smart Home Control through Unwitting Trigger-Action Programming

Daniela Fogli, Matteo Peroni, Claudia Stefini
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia
Via Branze 38, Brescia, Italy
daniela.fogli@unibs.it, matteo.peroni89@gmail.com, claudiastefini@hotmail.com

***Abstract*—This paper describes ImAtHome, an iOS application for smart home configuration and management. This application has been built over the framework HomeKit, made available in iOS, for communicating with and controlling home automation accessories. Attention has been put on the design of the interaction with such an application, in order to make the interaction style as much coherent as possible with iOS apps and supporting users without programming skills to unwittingly create event-condition-action rules that, in other similar systems, are usually defined through "if-then" constructs. The results of a user test demonstrate that ImAtHome is easy to use and well accepted by end users of different age and background.**

Smart home; end-user develoment; rule-based programming

I. INTRODUCTION

The attention of research scholars and ICT companies is more and more attracted by Internet of Things (IoT) [1] and Ambient Intelligence (AmI) [2], as witnessed by recent conferences, journal special issues and commercial advertisements. These areas involve experts in several disciplines – electronics, artificial intelligence, cloud computing, network infrastructures, and software architectures, just to name a few – who are called on to create and set up a new generation of distributed multimedia systems sometimes referred as "sentient multimedia systems" [3]. However, research on new Human-Computer Interaction (HCI) paradigms is fundamental as well, especially for making AmI environments easy to use and possibly allow their inhabitants, without computer programming knowledge, to install, configure and modify them over time. Therefore, with a particular attention to the smart home, End-User Development (EUD) [4] approaches are being proposed, which are aimed at transforming end users (household members) from passive consumers of sensors, robots and smart devices scattered in the house to active producers of new and possibly coordinated behaviors of such hardware/software components [5].

The idea is indeed to provide the house inhabitants with methods and tools to modify and adapt home behaviors to their needs, in order to cope with the continuous request of user-system co-evolution [3]. This could be achieved by providing users with EUD tools that support them in creating simple commands to be activated manually (e.g. "I am at home, please switch the radio on") or automatically (e.g. "At 7 a.m. rise shutters and activate the coffee machine").

DOI reference number: 10.18293/DMS2016-017

From the analysis of the literature and commercial products [6][7] it emerges the event-condition-action (ECA) rule-based paradigm is the most used in user interfaces devoted to the configuration and adaptation of smart homes by non-expert developers and thus could be considered as a promising solution to create EUD tools in this field. Such tools allow the user to carry out a form of *trigger-action* ("if, then") *programming* [8]. The user is thus guided in setting up the "if" and "then" parts of a rule, by choosing them among lists (filtered-list metaphor), virtual puzzle pieces (jigsaw composition) or components to be put in a network (wired composition) [9].

This paper proposes a new interaction metaphor for rule creation aimed at supporting users to perform trigger-action programming in an "unwitting" manner, that is at helping them create antecedent and consequent parts of the rules, without requiring them to think in terms of "if-then" constructs like computer scientists naturally do. To achieve this goal, the proposed metaphor splits rule creation in two steps, namely the *definition of scenes* followed by the *definition of rules* starting from available scenes. Scenes are sets of device actions that can be also manually activated by the user, thus becoming high-level commands for the house. Rules are defined to make the house able of activating itself some given scenes on the basis of the occurrence of an event, possibly combined with one or more conditions. The metaphor also encompasses a more guided but easier way of defining events and conditions for triggering rules. These ideas have been implemented in an iOS mobile application, called ImAtHome. Terms such as "if", "then" or "do" never appear in ImAtHome, as well as it has been removed the constraint of defining the consequent only after the definition of the antecedent, often present in other similar applications (e.g. IFTTT, Atooma, Tasker, and others).

Another important aspect, often neglected in scientific literature, is the cost of transforming a traditional house into a smart home. Current solutions usually require the interaction with companies that provide global services for smart home installation and maintenance or, alternatively, the acquisition of home automation boxes (e.g., Zipabox, Zibase, Vera, and eeDomus); the latter, in turn, require some "guru" in the family (i.e. a software expert or someone interested in software programming) capable of taking care of system installation and personalization [10].

To overcome this problem, also in this case a smoother approach to home automation is advocated: the idea is to allow

end users to add gradually smart devices to their house, according to emerging needs and economic possibilities. To this aim, we have decided to develop our mobile application over the framework for home automation made available in iOS, namely HomeKit[1]. HomeKit is a framework for communicating with and controlling the smart devices available in a house. It provides the user with a way to automatically discover such devices and configure them. It also makes available functionalities for executing sets of actions to control groups of devices, by possibly triggering them using Siri – the voice-controlled virtual assistant available in iOS.

ImAtHome is thus built on top of HomeKit and proposes itself as a hub application, able to manage all the devices currently available in a smart home, as well as those that will be acquired and included in the future. This paper presents the design and implementation of ImAtHome, as well as a usability study carried out with a group of 14 users of different age and background.

The paper is organized as follows: Section II discusses related works in the IoT and AmI fields, with particular reference to the smart home and user interfaces for their configuration and management. Section III describes the HomeKit framework. Section IV illustrates the development and the operation of the app ImAtHome. Section V provides some implementation details, while Section VI discusses the results of a usability evaluation with 14 users. Finally, Section VII draws some conclusions and proposes hints for future work.

II. Related Works

The idea to enable users to program the behavior of their smart home has been discussed in literature by several research scholars (e.g., [11][12]). For example, in [11], the "Media Cubes" programming language is proposed: it is based on the physical arrangement of infrared remote cubes; they represent abstract functions whose combination leads to the creation of complex behaviors. In the e-Gadgets project [12], instead, a visual editor is proposed, where end users can define "synaptic associations" (cause-effect relationships) between smart appliances available in a home.

More recently, the ECA rule-based paradigm has been proposed in several approaches to EUD applied in IoT or AmI. One of the most complete frameworks for AmI based on a rule-based approach is that described in [13]; a subsequent work of the same authors present three different graphical user interfaces for rule creation [14], even though no usability study is reported. In many cases, the proposed toolkits and languages require users to have some expertise in computer programming and hardware/software technologies. Barricelli and Valtolina have delineated an extension of the ECA paradigm pairing it with the use of formula languages [15]. Coutaz and colleagues [16] have presented a programming environment, called SPOK, which combines rule-based and imperative programming. Demeure et al. [10] described a field study involving 10 households using different home automation systems for a long period of time. In all households, there was always only one member of the family in charge of installing, configuring and managing home behavior modification. This family member was always a male adult, knowledgeable in hardware/software technologies. Therefore, the pure end user, neither expert in software programming nor interested in it, seems practically excluded from the use of such kinds of tools.

Dahl and Svendsen carried out a preliminary comparison among three composition paradigms (filtered lists, wiring composition and jigsaw puzzle composition) for rule creation [9]. From it, filtered lists, where condition-action compositions are obtained by selecting conditions and actions from respective lists, resulted to be the most intuitive for readability; whilst, jigsaw puzzle composition was considered by participants the most playful and engaging type of interaction. The filtered lists metaphor is recently adopted by several commercial and research applications, such as IFTTT, Atooma, Tasker, Locale, and others. Ur and colleagues [8] have proved that IFTTT has a pretty usable interface to create rules, even though these can contain only one event or condition, and only one action. Lucci and Paternò [17] have compared Tasker, Locale and Atooma, all allowing the user to create rules with complex antecedent and consequent parts. In this study, Tasker resulted the best tool in terms of expressiveness and Atooma resulted the easiest to use by end users. The user study reported in [6] compared Atooma and IFTTT in terms of usability and user preferences, by considering both users with a background in computer science and users without this background. The System Usability Scale (SUS) [18] composite score indicated that Atooma has a higher usability; moreover, users appreciated the user interface of Atooma much more than that of IFTTT. A systematic literature review in the IoT and AmI areas is presented in [7], focused on the research works that present tools supporting EUD for smart home configuration and management. From the papers selected through the review, eleven tools have been identified and examined. All tools are based on a rule-based paradigm: end users are supported by visual interfaces to compose events and/or conditions with actions, using structures like 'if-condition(s)-then-action(s)' or 'when-event(s)-then-action(s)'. A qualitative comparison of a subset of these eleven tools is then presented in [7], by considering the design principles for smart home control discussed in [19]. From this analysis, Tasker resulted to be the only tool able to satisfy most of the design principles (six out of seven). However, as also underlined in [17], Tasker appeared as more suitable to users with some knowledge in computer programming than to actual end users. In this paper, we propose a different approach to the creation of ECA rules, which, on the one hand, is aimed to be powerful enough for modeling a huge variety of home behaviors, and on the other hand, would like to support users performing such programming activity in an unwitting manner, as advocated in [20].

Finally, IT companies such as Google, Apple, Samsung and so on, are currently proposing their solutions in this field. However, to allow controlling a variety of devices they may require buying some specific hardware, as in the case of Google Smart Home Media Center, or proprietary accessories as in the case of Samsung. This has consequences on scalability and the possibility for users to create their own rules. On the other hand, Apple proposes HomeKit as a framework for communication with accessories and provides some indications to build apps on it. We have thus chosen to study HomeKit and develop an

[1] https://developer.apple.com/homekit/

application able to exploit it, but open to the interaction with any kind of device compatible with this framework.

III. THE HOMEKIT FRAMEWORK

HomeKit is a framework made available in iOS for communicating with and controlling connected home automation accessories that support Apple's HomeKit Accessory Protocol. Using HomeKit a developer can create complex applications that allow managing the interaction with accessories at a high level, without worrying about low-level technical details. HomeKit is mainly a communication protocol that supports the integration and interoperability of different kinds of accessories.

A. User Interface Guidelines

The user interface of ImAtHome has been developed by following the iOS guidelines for user interfaces and the following more specific HomeKit User Interface Guidelines[2]:

- **Setting up homes** by defining three types of locations: homes, rooms and zones (groups of rooms, such as "upstairs"). Rooms, such as "kitchen" or "bedroom" are the basic organizational concept and will contain the accessories. At least one home must be specified; it will include rooms, and will optionally contain zones. Users must be supported in the creation, modification and deletion of homes, rooms, and zones.
- **Managing users**, who, according to their privileges (Admin or iCloud account holder) may carry out different activities: setting up homes, adding accessories, creating scenes or just adjusting the characteristics of accessories.
- **Adding and removing an accessory** in an easy way, also by means of automatic discovering of accessories. Users must be supported in the configuration of the accessory by assigning it a name, home, room and zone (optional). Users must be able to easily identify the accessory they are configuring.
- **Facilitating the creation of scenes** to adjust the characteristics of multiple accessories simultaneously. Each scene is therefore a set of actions on any number of accessories.
- **Siri integration** to activate scenes with voice commands. HomeKit allows Siri to recognize home, room, and zone names; therefore, Siri can support statements like "Siri, turn off the living room lights".
- **Using a friendly and conversational language**, in order not to intimidate the user with acronyms or technical terms.

Furthermore, HomeKit supports the execution of rules (called "triggers" in HomeKit), which are ways to activate a scene based on conditional relationships concerned with time, location, and the behavior of other accessories.

Therefore, an additional guideline in the HomeKit Developer Guide suggests to **help users set up triggers**, by facilitating as much as possible the creation of the conditional relationships. In the design and development of ImAtHome we have carefully considered this issue.

B. HomeKit Accessory Simulator

HomeKit Accessory Simulator is a tool that allows one to simulate the presence of some accessories in the smart home. Such accessories correspond to those ones that an app would automatically discover in a home. HomeKit Accessory Simulator builds a simulated wireless network to which all accessories added by the developer are connected. Each accessory may have a variety of characteristics to be controlled; through the simulator, the developer can add a characteristic to a class of accessories or a personalized characteristic to a single accessory. Optional characteristics can also be removed. This simulator has been very useful for setting up the experimentation of ImAtHome.

C. Communication in ImAtHome through HomeKit

A variety of companies are developing accessories compatible with HomeKit, such as conditioners, thermostats, light bulbs, cameras, secure locks, carbon monoxide sensors, and so on. As a consequence, several dedicated apps are being developed to control such different devices. Actually, the Apple Store contains at least one app for each accessory class mentioned above and it may also happen that more than one app for the same accessory class are available, usually developed by different producers. The advantages given by the compatibility with HomeKit are that accessories can be controlled through Siri and may be included in the creation of a scene. However, the main drawback is that each accessory keeps on being controlled only by its corresponding app, according to the architecture schematized in Figure 1. Therefore, scenes or rules that involve different types of accessories cannot be created.

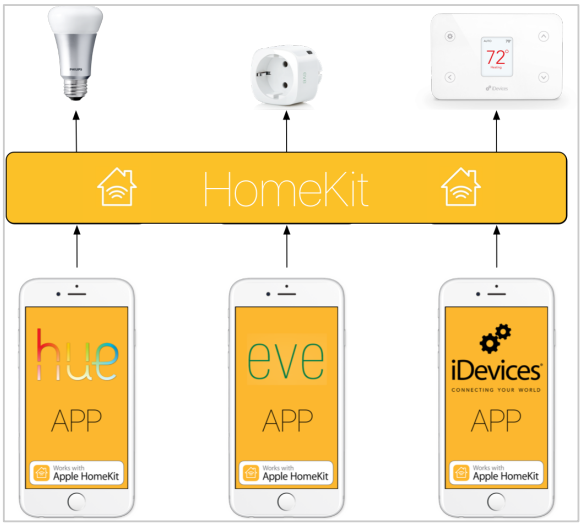


Figure 1. Communication with accessories through HomeKit.

[2]https://developer.apple.com/homekit/ui-guidelines/

The idea underlying ImAtHome is to exploit the common communication protocol for interacting with many kinds of accessories or combinations of them (see Figure 2). In other words, ImAtHome proposes itself as a hub for controlling one's own smart home: all compatible accessories are made available on the smartphone in a unique app, with the same interaction style and the possibility to work in combination one another.

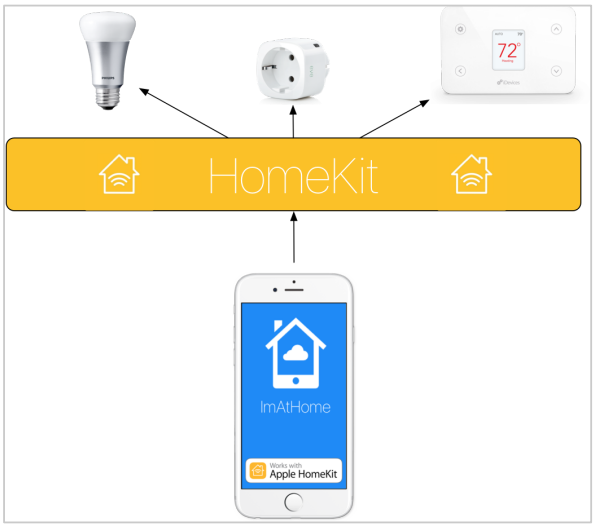


Figure 2. Controlling the smart home through HomeKit with ImAtHome.

IV. The Application ImAtHome

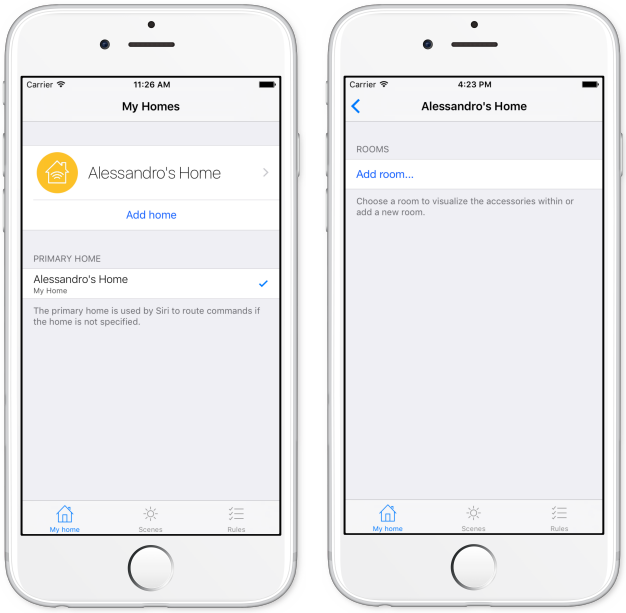
The design and development of ImAtHome has been performed according to a user-centered approach, by involving users in the discussion and evaluation of paper mock-ups and interactive prototypes throughout the software development lifecycle. The following subsections illustrate the interaction with ImAtHome to create homes, scenes and rules.

A. ImAtHome Interface Structure

The features made available by HomeKit have guided the ideation of the app structure. First, the interface includes a section, entitled "My home", where the user can access or define a new home (Fig. 3(a)) and its rooms (Fig. 3(b)). By selecting the item "Add room..." in Fig. 3(b), the user can create a room and associate it with a name and an icon among those available (Fig. 4(a)). The new room, "Living room" in the example, is thus added to the list of rooms previously defined by the user (Fig. 4(b)). Accessories are similarly shown as a list associated with a room. The user may bind accessories to rooms inserting their configuration code by means of OCR technology.

Furthermore, the HomeKit database distinguishes between action sets (scenes) and triggers (rules). Action sets may be related to triggers through conditional relationships. Therefore we have decided to add two different sections in the app, one where the user can find or define her/his scenes ("Scenes") and the other devoted to the creation of rules ("Rules"). Scenes are actually sequences of actions that the user may manually activate; whilst rules represent automatic activation of one or more scenes, under some specific conditions (that trigger the rules).

The three sections are accessible through the bottom tab bar.

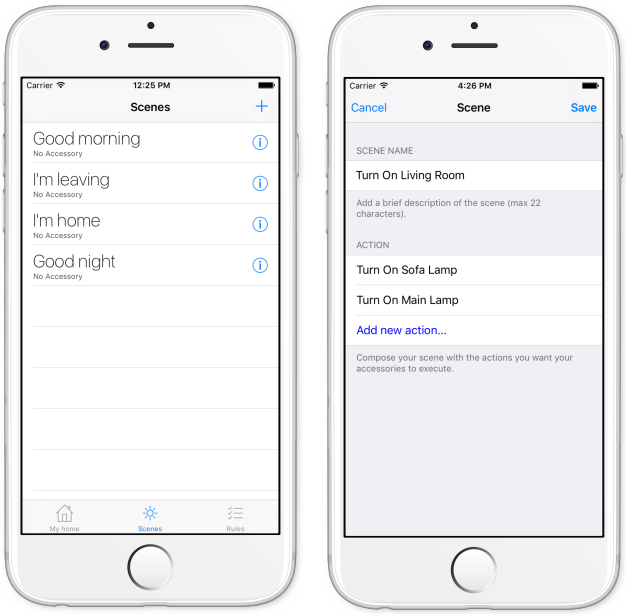


(a) (b)

Figure 3. Section "My home" on the left (a) and screen for adding a room to the home on the right (b).

(a) (b)

Figure 4. Hierarchical navigation in section "My home": screen for room creation on the left (a) and list of rooms on the right (b).



(a) (b)

Figure 5. Section "Scenes" with the list of pre-defined scenes on the left (a) and screen for scene creation on the right (b).

B. *Scene definition and activation*

The section "Scenes" presents a default list of void scenes available in the HomeKit database, that is "Good morning", "I'm leaving", "I'm home" and "Good night" (see Fig. 5(a)). The user may complete them or create a new one by clicking the '+' button in the right upper corner (as it is usually requested in iOS apps to add a new object). In the new screen - Fig. 5(b), the user can give a name to the scene (e.g., "Turn On Living Room" to indicate a scene that turns on the lights in the living room) and define a set of actions (in the example: "Turn On Sofa Lamp" and "Turn On Main Lamp") by setting the characteristics of the accessories involved in each action. When the new scene is saved, the list shown in Fig. 5(a) is updated. Here the user can manually activate the created scene with a tap or by pronouncing its name, thus giving control to Siri for scene execution.

C. *Rule creation*

In ImAtHome, the user may decide to use the pre-defined or user-defined scenes to create rules. Let us suppose that the user would like to create a rule that switches on the lights of the living room (the scene illustrated above) when she arrives at home, but only if it is after 5 P.M. To this aim, she must access the third section of the application and click on the '+' button in the right upper corner. As a consequence, the screenshot in Figure 6(a) is shown. Here the message at the top tells the user to choose among three options to trigger her scenes by "Time", "Position" or "Another accessory". These ones correspond to the three conditional relationships for triggering scenes, which are supported in HomeKit. Selecting one of them allows defining the "event" part of an ECA rule. However, differently from the interaction with other tools, here the user does not need to know that the "if" part of an "if-then" construct must be created. In the example, the user selects the "Position" option. As a consequence, a screen appears where the user

1) defines the details of an event related to her position; in the example in Fig. 6(b) she selects "When I arrive". Then she taps on "Choose a position" and, as a consequence, a map appears centered in the current position of the user; if she is at home, she can simply save that position, otherwise she may look for a specific address through the search bar;
2) defines an additional condition ("after 5:00 P.M." in Fig. 6(c));
3) selects one or more scenes that must be activated. In the example in Fig. 6(d), the user checks "Turn On Living Room".

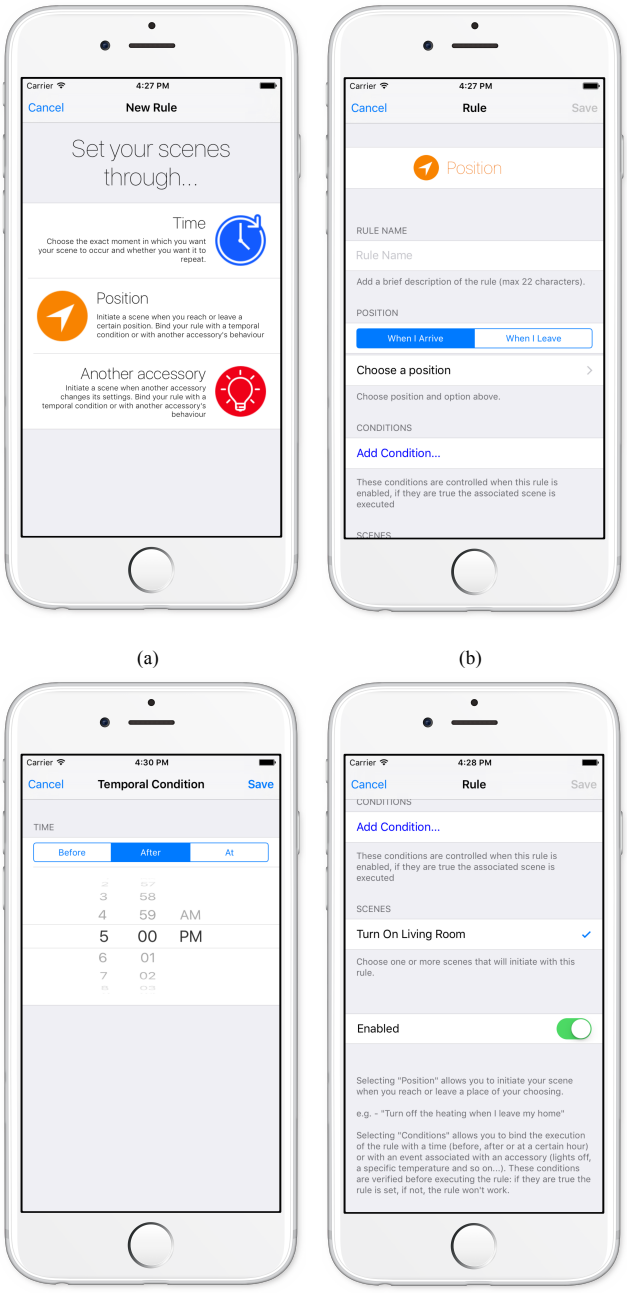
Note that with steps (2) and (3) the user actually creates the condition part and action part of an ECA rule respectively, without being aware of it. Moreover, differently from other user interfaces for ECA rule definition, ImAtHome requires to define action sets (scenes) first, and then relate them to events and conditions. This allows users to activate scenes manually if needed and use them in several different rules.

V. IMPLEMENTATION

The implementation of ImAtHome has been carried out in Swift, the programming language for iOS presented by Apple in 2014. As already mentioned, it has been built over the HomeKit Application Programming Interface (API), which provides a variety of classes for low-level interaction with home accessories. On activation, ImAtHome creates an object of the class HMHomeManager, made available by HomeKit to add or remove a home to/from its database. For each home, HomeKit creates a database on iCloud, which contains all its objects (rooms and accessories in our case). This database is always synchronized with the user's iOS device; therefore, to show the user the most recent data, the app ImAtHome continuously monitors the database updates. In particular, ImAtHome exploits HomeKit API to 1) discover the accessories available in the environment compatible with the communication protocol, and add them to the database associated to the home; 2) access the properties of the accessories; 3) modify the current values of the accessory properties, thus executing actions (e.g. switch on a light in a given room with a certain lighting level and a specific color). At a higher level, the app is organized in four groups of classes: three main groups manage the behavior and appearance of the three app sections respectively (My Home, Scenes and Rules); whilst, the last group includes all other classes supporting the operation of the main classes.

VI. SYSTEM EVALUATION

A user test has been carried out to evaluate the usability of ImAtHome. The experiment has been conducted by using the iPhone simulator included in the development environment and the HomeKit Accessory Simulator. The Italian version of the app has been used to facilitate the interaction with Italian participants.



(a) (b)

(c) (d)

Figure 6. ECA rule creation in ImAtHome.

A. Methodology

The user test involved a total of 14 participants (8 males and 6 females). Their ages ranged as follows: 3 users in 20-24, 8 users in 25-35, and 3 users in 50-65 (average age equal to 33). They held different education degree and represented varied professional backgrounds. They included 8 students, 3 office workers, 1 housewife, 1 unemployed and 1 retired. Twelve participants held an Apple device. As to computer science knowledge, 8 participants declared to have low or medium knowledge, whilst 6 declared themselves as experts. Participants were asked to carry out five tasks of increasing complexity: the first two tasks devoted to the creation of a smart home, with rooms and accessories; the third one for creating a scene; and the last two tasks for defining a simple rule and a complex rule respectively. No previous training was provided to the users.

During task execution, we collected quantitative data related to execution time and percentage of task completion. Since a think aloud protocol was adopted, comments of participants were annotated as well. Then, after the test, we submitted to the participants a post-questionnaire to gather opinions about their experience of use and a SUS questionnaire [18] to obtain an estimation of ImAtHome usability.

B. Quantitative Results

Table I reports the execution times of participants (with average value and standard deviation) and the optimum execution times of one developer. Execution times of participants were always about 3 times the optimum time, except for Task 3 that was the first task devoted to the creation of a scene. The last task was the most complex in terms of actions to perform, but its average execution time was less than that of Task 4, thus demonstrating that ImAtHome is easy to learn. In general, also considering that participants did not receive any previous training on the application, execution times are highly satisfactory. The percentages of task completion reported in the last row of Table I confirm such positive results.

A further analysis on the time spent to carry out the tasks was performed dividing the users in two groups: the former including the 8 participants that declared to have low or medium knowledge in computer technologies, the latter with 6 participants declaring themselves as experts. A t-test was adopted to compare the execution times: no significant difference was found between the two groups, demonstrating once again that ImAtHome allows all kinds of users to easily become "unwitting trigger-action programmers".

C. Qualitative Results

Comments of the users gathered during test execution highlighted some cosmetic problems that were easily solved after the experiment. Most of the users made positive (and sometimes enthusiastic) observations on the interaction process adopted for creating homes and rooms and populating them with accessories. Some difficulties were encountered in the execution of Task 3, which required to create a scene for switching off all the lights: before choosing the right way, a participant tried at first to modify a pre-defined HomeKit scene; another participant observed that the task was not easy to understand, but he also admitted that he did not read the app instructions first, which instead would have provided him with useful examples; finally, one participant told us that it was not clear when and where to save the scene. Fewer difficulties were encountered in the creation of the first rule (Task 4): only one user had been not able to combine an event ("when I move from home") with a condition ("if it is after 9.00 P.M."). As already mentioned, Task 5 was very successful, despite its complexity: indeed, some participants commented that the interaction with the app became familiar after few interactions.

TABLE I. TASK EXECUTION TIME AND COMPLETION PERCENTAGE

User	Tasks				
	Task 1	*Task 2*	*Task 3*	*Task 4*	*Task 5*
U1	02:00	01:48	02:30	03:43	04:54
U2	01:46	02:00	01:30	04:30	02:57
U3	00:40	01:16	01:00	04:30	02:25
U4	01:23	01:30	01:38	03:03	03:35
U5	01:27	01:47	01:55	03:00	02:30
U6	01:20	02:03	02:30	02:15	02:55
U7	00:38	01:03	01:44	01:40	01:53
U8	00:39	01:35	02:00	01:30	02:40
U9	01:23	01:30	01:17	05:34	04:06
U10	00:41	01:30	00:35	02:20	02:10
U11	01:40	01:59	03:10	02:10	03:00
U12	01:27	01:19	01:08	01:45	01:55
U13	01:30	02:47	01:10	04:10	04:12
U14	00:50	01:34	01:29	04:41	02:40
Avg.	**01:15**	**01:42**	**01:41**	**03:12**	**02:59**
SD	00:27	00:25	00:40	01:16	00:52
Opt.	**00:30**	**00:50**	**00:18**	**01:00**	**1:00**
% Compl.	**100%**	**100%**	**93%**	**93%**	**100%**

D. *Findings from post-questionnaire*

The post-questionnaire included the following questions:

1. Did you find the interaction with the app pleasant and funny?
2. If compatible accessories would be present in your home, should you use ImAtHome?
3. Did you find the user interface coherent with the other iOS apps?
4. Do you think that with some limited training ImAtHome would be easier to use?
5. Did you find the language of the application easy to understand?

Answers to the above questions were given on the qualitative scale {"definitely no", "no", "fairly", "yes", "definitely yes"}. We then translated the participants' assignments to the 0-4 numerical scale and computed the average values. The following results were obtained: Q1=**3.07**; Q2=**3.29**; Q3=**3.67**; Q4=**3.29**; Q5=**3.36**. As to Q3 (related to coherence), we gathered the answers of 12 out of 14 participants, since the remaining two participants did not hold any Apple device. Participants that encountered some difficulties in the execution of Tasks 3 and 4 asserted that, with a limited training, the application became very easy to use.

E. *SUS evalutation*

The overall usability of ImAtHome was finally evaluated through the SUS questionnaire, by providing an average cumulative score of the 14 participants equal to 86.6, pretty higher than the conventional threshold equal to 70, adopted for declaring that a system is easy to use [18]. More precisely, Figure 7 shows the SUS scores of all the 14 users; notice that they are all (except one) above or equal the threshold.

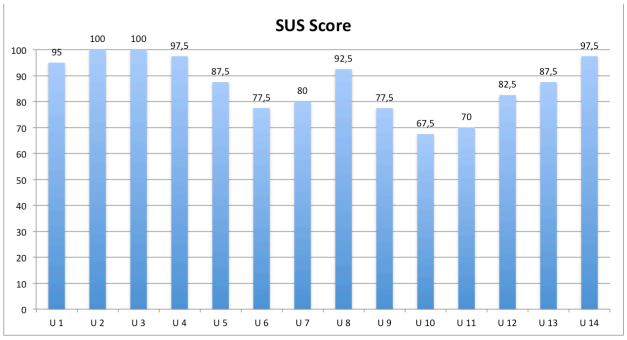


Figure 7. SUS cumulative scores of the 14 users.

VII. DISCUSSION AND CONCLUSION

This paper has presented ImAtHome, an iOS application that allows a home inhabitant, without any programming skills, to control home automation accessories and create scenes and rules for defining complex behaviors of a smart home. The application is scalable, because if a new accessory, compatible with HomeKit, is acquired, it will be automatically recognized by the application and its services will be presented to the users as it already happens for the other accessories. With respect to other approaches proposed in the literature and available commercial tools, ImAtHome allows the user both to manually activate some sets of actions (scenes) or to use them within rules to obtain automatic behaviors of the home. Furthermore, the same scene, once created, can be used several times in different rules; whilst, in other user interfaces based on the "if-then" paradigms, antecedents and consequents of rules must be always explicitly defined.

Integration with HomeKit allows ImAtHome to be used through vocal commands. Indeed, Siri is able to recognize the words associated with accessories and scenes. Therefore, vocal commands can be used to change the characteristics of an accessory and to activate pre-defined or user-defined scenes. An interesting extension could be the creation of a parser for scene and rule creation from the fragments of a vocal command.

As to future work, we are planning to extend the experimentation. Beyond involving a higher number of users, it would be interesting to compare "if-then" or "when-then" interfaces of existing tools (e.g. IFTTT, Atooma, and Tasker) with our "unwitting trigger-action programming" style. Not only measures of user performance, but also users' acceptance and appreciation for the interaction style would represent useful information for future development of these kinds of interfaces.

Another important issue is the extension of the application to the case of multiple user control of a smart home. Indeed, as underlined in [19], more than one person usually inhabits a

home and household activities may be collaborative or in competition (such as TV control or music choice). Therefore, we foresee a future where the collective and participatory evolution of a sentient multimedia system takes place through the simultaneous, but coordinated, intervention of all the interested actors [3]. However, to achieve this goal, some limitations of the current version of HomeKit must be overcome. Currently, it allows associating one's own apple account with a new home, and thus add or modify accessories, rooms, scenes and rules, but it does not allow other Apple accounts to do these activities on the same home. In other words, a home can be shared with other users, but a "guest" user can only control the accessories and activate existing scenes; whilst, she/he cannot actually modify the HomeKit database.

Under the hypothesis that future versions of HomeKit will be released to cope with this issue, we have started to think about the problems that would affect a multi-user approach to smart home control. First, user profiling and permission control should be supported; this would require a usable interface, possibly based on suitable visual languages, like those proposed in [21][22][23]. Second, a variety of social mechanisms, from collaboration to competition, from delegation to reciprocity, should be implemented to stimulate participation. To address this problem, we have proposed the idea of a collaborative application enriched with gamification techniques aimed at motivating all household members to participate in the shaping of their smart home [24][25]. Third, giving household members the possibility to intervene simultaneously in accessory or scene activation and in the creation of rules working on shared spaces may lead to incoherencies and conflicts among rules; suitable solutions must be carefully studied to address these problems.

REFERENCES

[1] L. Atzori, A. Iera, G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54(15), 2010, pp. 2787-2805.

[2] F. Sadri, "Ambient intelligence: A survey," ACM Computing Surveys vol. 43(4), 2011, pp. 1-66.

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

[4] H. Lieberman, F. Paternò, V. Wulf, V. (eds.), End User Development. Dordrecht, The Netherlands: Springer, 2006.

[5] F. Cabitza, D. Fogli, R. Lanzilotti, A. Piccinno, "End-User Development in Ambient Intelligence: a User Study," Proc. 11th Biannual Conference on Italian SIGCHI Chapter (CHItaly), ACM, New York, NY, USA, 2015, pp. 146-153.

[6] F. Cabitza, D. Fogli, R. Lanzilotti, A. Piccinno, "Rule-based Tools for the Configuration of Ambient Intelligence Systems: a Comparative User Study," Multimedia Tools And Applications, DOI: 10.1007/s11042-016-3511-2.

[7] D. Fogli, R. Lanzilotti, A. Piccinno, "End-User Development Tools for the Smart Home: A Systematic Literature Review," In: N. Streitz and P. Markopoulos (Eds.): DAPI 2016, LNCS 9749, Springer International Publishing Switzerland, 2016, pp. 1–11.

[8] B. Ur, E. McManus, M. Pak Yong Ho, M.L. Littman, "Practical trigger-action programming in the smart home," In: SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 2014, pp. 803-812.

[9] Y. Dahl, R.-M. Svendsen, "End-User Composition Interfaces for Smart Environments: A Preliminary Study of Usability Factors," In A. Marcus (Ed.), Design, User Experience, and Usability. Theory, Methods, Tools and Practice, Vol. 6770, Berlin Heidelberg: Springer, 2011, pp. 118-127.

[10] A. Demeure, S. Caffiau, E. Elias, C. Roux, "Building and Using Home Automation Systems: A Field Study," In: Díaz, P., Pipek, V., Ardito, C., Jensen, C., Aedo, I., Boden, A. (eds.) End-User Development. LNCS, vol. 9083, Springer International Publishing, 2015, pp. 125-140.

[11] A. F. Blackwell, "End-user developers at home," Communications of the ACM vol.. 47(9), 2004, pp. 65-66.

[12] I. Mavrommati, A. Kameas, P. Markopoulos, "An editing tool that manages device associations in an in-home environment," Personal and Ubiquitous Computing, vol. 8(3-4), 2004, pp. 255-263.

[13] M. García-Herranz, P. A. Haya, A. Esquivel, G. Montoro, X Alamán, "Easing the Smart Home: Semi-automatic Adaptation in Perceptive Environments," Journal of Universal Computer Science, vol. 14(9), 2008, pp. 1529-1544.

[14] M. García-Herranz, P. A. Haya, X. Alamán, "Towards a Ubiquitous End-User Programming System for Smart Spaces," Journal of Universal Computer Science, vol. 16(12), 2010, pp. 1633—1649.

[15] B. R. Barricelli, S. Valtolina, S., "Designing for End-User Development in the Internet of Things," In: Díaz, P., Pipek, V., Ardito, C., Jensen, C., Aedo, I., Boden, A. (eds.) End-User Development, vol. 9083, Springer International Publishing, 2015, pp. 9-24.

[16] J. Coutaz, A. Demeure, S. Caffiau, J. L. Crowley, "Early lessons from the development of SPOK, an end-user development environment for smart homes," Proceedings of 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp), Seattle, Washington, pp. 895-902, 2014.

[17] G. Lucci, F. Paternò, "Understanding End-User Development of Context-Dependent Applications in Smartphones," In S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt & M. Winckler (Eds.), Human-Centered Software Engineering, vol. 8742, Springer Berlin Heidelberg, 2014, pp. 182-198.

[18] S. Borsci, S. Federici, M. Lauriola, M., "On the dimensionality of the System Usability Scale: a test of alternative measurement models," Cognitive Processing, vol. 10(3), 2009, pp. 193-197.

[19] S. Davidoff, M. K. Lee, C. Yiu, J. Zimmerman, A. K. Dey, "Principles of Smart Home Control," In: Dourish, P., Friday, A. (eds.) UbiComp 2006: Ubiquitous Computing. LNCS, vol. 4206, Springer, Berlin Heidelberg 2006, pp. 19-34.

[20] 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 (WEUSE '08). ACM, New York, NY, USA, 2008, pp. 6-10.

[21] M. Giordano, V. Loia, G. Polese, G. Tortora, "A system for user friendly pervasive computing management," In Proceedings of the 3rd International Conference on Intelligent Environments (IE'07), Ulm, Germany, September 2007, pp. 282-287.

[22] M. Giordano, G. Polese, G. Scanniello, G. Tortora, "A system for visual role-based policy modelling," Journal of Visual Languages & Computing, vo. 21(1), 2010, pp. 41-64.

[23] L.Caruccio, V. Deufemia, C. D'Souza, A. Ginige, G. Polese, "A Tool Supporting End-User Development of Access Control in Web Applications," International Journal of Software Engineering and Knowledge Engineering, vol. 25(2), 2015, pp. 307-331.

[24] F. Benzi, F. Cabitza, D. Fogli, R. Lanzilotti, A. Piccinno, "Gamification Techniques for Rule Management in Ambient Intelligence," In: B. De Ruyter, A. Kameas, P. Chatzimisios and I. Mavrommati (Eds.), Ambient Intelligence, Springer International Publishing, 2015, pp. 353-356.

[25] D. Fogli, R. Lanzilotti, A. Piccinno, P. Tosi, "AmI@Home: a Game-Based Collaborative System for Smart Home Configuration," In: Proceedings of International Working Conference on Advanced Visual Interfaces (AVI '16), ACM, New York, NY, USA, 2016, pp. 308-309.