Teaching End-User Development in the Time of IoT & AI

Fabio Paternò[0000-0001-8355-6909]

CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 56124 Pisa, Italy
fabio.paterno@isti.cnr.it

Abstract. The combination of the Internet of Things (IoT) and Artificial Intelligence (AI) has made it possible to introduce numerous automations in our daily environments. Many new interesting possibilities and opportunities have been enabled, but there are also risks and problems. Often these problems originated from approaches that have not been able to consider the users' viewpoint sufficiently. We need to empower people in order to actually understand the automations in their surroundings environments, modify them, and create new ones, even if they have no programming knowledge. It is thus important that the curricula of programs in several disciplines (artificial intelligence, computer science, human-computer interaction, psychology, design, …) discuss these problems and some possible solutions able to provide people with the possibility to control and create their daily automations. In this paper I propose a possible way to organise and structure teaching of the concepts, methods and tools for this purpose, and which can be adopted in the relevant curricula.

Keywords: Internet of Things, End-User Development, Tailoring Environments, Trigger-action programming.

1 Introduction

The main technological trends of recent years have been the Internet of Things and Artificial Intelligence. Their combination has made it possible to introduce numerous automations that can manifest themselves in different ways in our daily environments. Many new possibilities and opportunities have been created, but also risks and problems. Often these problems originated from approaches that have not been able to consider the human point of view sufficiently. In particular, the user has often been considered as a passive element with respect to the new possibilities instead of being the central subject. People in their lives often have dynamic needs, which sometimes stem from episodes, even unpredictable ones. The most effective automations are often the ones that can be dynamically customized and created to meet these changing and different needs that only the users know completely. Thus, we need to empower people in order to actually understand the automations active in their surrounding environments, modify them, and create new ones, even if they have no particular programming knowledge [13].

For such reasons it is of paramount importance that designers and developers be aware of such issues and of some possible solutions to provide people with the ability to control and create their daily automations. Thus, it is important that they are trained in courses aiming to allow attendees to gain knowledge and skills in addressing problems and solutions involved in end-user creation, control, monitoring, debugging automations that can be deployed in daily environments (such as home, office, shops, industries, …). Such courses should provide a discussion of the possible solutions in terms of concepts, techniques, and tools, with particular attention to those supporting the trigger-action paradigm. The courses should discuss how to enable people who are not professional developers to indicate the various dynamic events and conditions that can occur in their contexts of use (considering aspects related to user, technology, environment), and the possible associated actions.

These types of topics are interesting for courses in several areas: computer science, engineering, digital humanities, human-computer interaction. In general, such topics are to be aimed at those who want to understand the issues involved in introducing automations in daily environments, and the corresponding possible solutions that can empower end users in controlling, modifying and creating new ones. In case of research students, they allow participants to understand the relevant state of art in order to think about novel solutions in this area. Thus, there should not be any particular prerequisites for attending such courses. Some basic knowledge of Internet of Things technologies would make them easier to follow, but all the relevant concepts should be introduced in such a way to be understandable also to those who are not familiar with them.

The next sections present a description of the recommended content for such courses that is based on my personal experience in teaching them at the Digital Humanities degree of University of Pisa, and in tutorials at mayor HCI international conferences, as well as in the research work carried out for several years in relevant areas in national and international projects.

2 Content

The content for this type of course can be structured into four main parts: introduction, trigger-action programming, tailoring tools, intelligent services. The initial one should provide some background information and explain the motivations for addressing such topics. Thus, it should introduce the main current technological trends (internet of things and artificial intelligence) with their potentialities and risks, and end-user development, and explain why it can be useful to mitigate such risks. The next part should be dedicated to trigger-action programming, which is the programming paradigm that seems most relevant when considering contexts of use characterised by the presence of numerous connected objects and devices, with the need of tailoring the automations involving them according to the dynamic events that can occur. The various possible compositional styles to create the trigger-action rules need to be introduced with example of tools for each of them. Thus, the various compositional styles are described: visual data flow, wizard-like, block-based, and conversational. The forth part can be

dedicated to introducing additional support that can increase the possibilities of the tailoring environments. Thus, in this part it is possible to introduce the role of intelligent recommendations in this context, for example while people are creating trigger-action rules. Another important relevant topic is how to provide explanations that allow end users to understand why or why not a certain rule can be executed in a given context of use. One further aspect that is important to consider is also issues and possibilities that are available when deploying in real world platforms for executing such rules, and how to monitor their behaviour. Table 1 provides a summary of the course content.

Table 1. Possible course structure

Subject
Introduction Course
The technological trends (IoT + AI)
The dark side of intelligent automations
Trigger-action programming
Automation specification exercise
Environments for end user creation of automations
Real world deployment, execution, monitoring
Exercise with tool for end user automation creation
Intelligent automation recommendations
Explainable automations
Augmented reality support for serendipitous creation
Final Discussion

2.1 Introduction

This type of course should start with an introduction to the main current technological trends (internet of things and artificial intelligence). For this purpose, it can be useful to report current forecasts[1] that indicate how in the next years we expect that the number of connected objects populating our homes, cars, working environments will be continuously increasing while the number of devices (smartphones, laptops, PCs) will only increase slightly. Artificial intelligence will be exploited more and more in this context in order to create automations based on the data that can be collected. However, the following will include a discussion of the problems end users encounter when their viewpoint is not considered (see for example the study reported in [18]), such as the learning system fails to understand user intent or the system's behaviour is hard to understand. Thus, one of the main challenges in the coming years is how to obtain tools that allow users to control and configure smart environments consisting of hundreds of

[1] https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

interconnected devices, objects, and appliances, tools that allow people to obtain "*humanations*": automations that users can understand, monitor, and modify.

At this point there should be a brief introduction to end-user development, its importance to empower people to customize their applications, the main approaches in terms of metaphors and programming styles that have been considered, and the aspects that have to be addressed in order to provide effective solutions. Various generations of contributions based on main technological trends (graphical desktop, Web, Mobile, …) have been put forward in this area over time. The last generation aims to empower users to exploit ecosystems of smart things characterised by the presence of various types of sensors and actuators.

2.2 Trigger-Action Programming

The trigger-action programming paradigm [1, 3, 7, 8] is suitable for end-user development because of its compact and intuitive structure, and it is relevant for the Internet of Things characterised by the presences of a multitude of sensors and actuators, since it connects the dynamic events and/or conditions with the expected reactions. It is relevant also because it does not require the use of complex programming structures or particular algorithmic abilities. This approach has been used in several domains, such as home [17], ambient assisted living [102], robots [9], finance [5]. In particular, it should be explained that triggers represent situations / events that are relevant to the users and may relate to their emotional or physical state, or the surrounding environment or available devices or applications. Trigger information is derived from various sensors (for example, movement, proximity, light, noise, breathing, heartbeat, ...) and applications and services. Actions represent what the objects, device, applications available are able to do. For example, they can control objects (such as turning on / off lights, opening / closing doors, activating TV / radio), or activate reminders, alarms, user interface changes. The triggers can be composed of events and / or conditions. In this discussion it should also be introduced the aspects that sometimes are unclear for end users [1, 8] when they have to approach automations in internet of things scenarios. For example, sometimes it is not immediate to distinguish between events and conditions. Failure to understand the distinction between events and conditions can cause unwanted behaviours (e.g. opening a door at the wrong time or turning on the heater when it is not needed). For example, these two rules have different effects (and in one case it can be rather annoying):

- When I get home the bell rings
- If I'm at home the bell rings

Thus, it is useful to explain that an event happens in a point in time: when user enters a room, when it starts to rain, when kitchen temperature exceeds 30 degrees, at 8 o'clock. While a condition is a state that lasts for a longer period of time: while user is inside a room, as long as it's raining, if kitchen temperature is over 30 degrees, between 8:00 and 11:30. When writing the rules, the use of different keywords helps to differentiate them. Thus, often "WHEN" is used for events, and "IF" or "WHILE" are used for indicating conditions. Other useful trigger operators can be introduced for specific

cases. There are rules that are triggered if an event does not occur in a specific interval of time (e.g. when medicine has not been taken between 10 a.m. and 11 a.m.). In other cases, there are rules that are triggered when a specific ordered sequence of events occurs (e.g. “When the user enters a room and then he exits”; or “When the temperature becomes more than 20 degrees and then the humidity level becomes more than 50%”) or when an event occurs a specified number of times (event iteration), e.g. “When the user goes to the bathroom 5 times during the night”.

It is possible to compose events and conditions through logical or temporal operators (although it may not be immediate to understand the result) [8]. In particular, the composition of an event and a condition is an event that should occur when the condition is verified. The composition of two events is still an event but is an event with a very low probability, since it is unlikely that two events can happen at the same time. The composition of two conditions is something more frequent since it indicates the intersection of the times when the two conditions are verified.

The actions too have a temporal dimension [8], which can be classified in instantaneous (e.g. send email), extended (e.g. brew coffee), and sustained (e.g. turn the light on). In some cases, the temporal aspects of both triggers and actions can generate some ambiguity. For example, if we consider the rule

IF I am at home DO send email to John

We have a condition as a trigger, which means something that is verified for some time, and an instantaneous action. This can be interpreted in two ways: one is that the action should be performed multiple times as long as the condition is true, the other one, more realistic in this case, is that the action should be performed once. One further potential ambiguity can occur when combining a condition and a sustained action. For example:

IF time is between 8 and 11:30 pm, DO turn on the living room light

What should happen after 11.30 ? If users express interest for a similar rule, probably they expect that the light should be on in the indicated period of time but then it should be switched off.

When writing automations in trigger-action format, which can be considered personalization rules, there can be different styles [4]: device-centric, a personalization whose subject is the physical medium with which it is executed (e.g “when the motion sensor in the kitchen becomes active”); information-centric, a personalization whose subject is the underlying information, regardless of the physical medium with which it is manipulated; people-centric, a personalization where users, their actions, and/or feelings are at the center of the interaction, independently of any physical and virtual medium (e.g. “when I enter the bedroom”).

At this point, the participants should be asked to actually write some examples of automations with different complexity in natural language in pairs (one trigger and one action, two triggers and one actions, two triggers and two actions …). During this exercise it should be discussed together the rules created, and how well they specify the desired behaviour in order to allow participants to better understand possible ambiguities and problems in their execution.

2.3 Environments for End-User Creation of Automations

Once the basic concepts have been introduced, it is possible to present the possible approaches for supporting the composition of automation rules, and then examples of specific tools. Tailoring environments can guide the rule development process through different approaches:

- Data flow (focused on how information goes through the various parts)
- Wizards (aiming to drive users by limiting their possible selections)
- Visual blocks (Using visual cues to suggest possible compositions)
- Conversational (exploiting natural language and AI support)

Node RED[2] is an example of the data flow approach; it supports the creation of nodes and flows for connecting together objects and online services. It includes many predefined nodes. The user will have to define some behaviours, which requires writing some JavaScript code.

Probably, the best-known example of the wizard approach is IFTTT[3]. It is a Web and mobile environment that allows users to create and share rules, called «applets», in the form if trigger then action. Triggers and actions can be chosen from existing services (for example, Facebook, Evernote, Weather, Dropbox, etc.). In 2017 it was estimated that 320,000 applets involving 400 service providers were installed more than 20 million times. Services are peripherals, hubs, wearable devices, devices in the car, online services, social networks, messaging, ... IFTTT developers have recently changed their business model. Now, there is a limited version for free that can be used to create at most three applets, and then there is an IFTTT Pro version, which can be used by paying a subscription fee, which supports multiple features (multi-step, conditional logic, queries, filter code, multi-action). In general, IFTTT suffers from some limitations. It does not distinguish between events and conditions, it is not easy to extend the list of the connected applications, and it shows long lists of potential channels to compose where it is easy to get lost.

Another example of wizard-like tool is TARE (the Trigger-Action Rule Editor)[4], which is part of a more general platform for supporting creation, execution, and monitoring of personalization rules (TAREME [11]). In order to help users to find the relevant triggers and actions, they are organised in a hierarchical structure with the main categories at the top level, and then more specific elements are detailed. The triggers are classified according to three main contextual aspects (see Figure 1): users (for example associated with physiological or emotional aspects or their activities), environments (for aspects such as noise, light, temperature), and technology (associated with the available devices and appliances). The actions can be reminders, alarms, or effects on appliances or devices and their functionalities. On the left side there is also an interactive sidebar that indicates what has been done and what should be done to complete the creation of the rule, while on the top part there is a natural language description of

[2] https://nodered.org/
[3] https://ifttt.com/create
[4] https://tare.isti.cnr.it/RuleEditor/login

the rule being edited. When selecting the triggers, users are explicitly asked to indicate whether it should be an event or a condition in order to make them more aware of the difference.

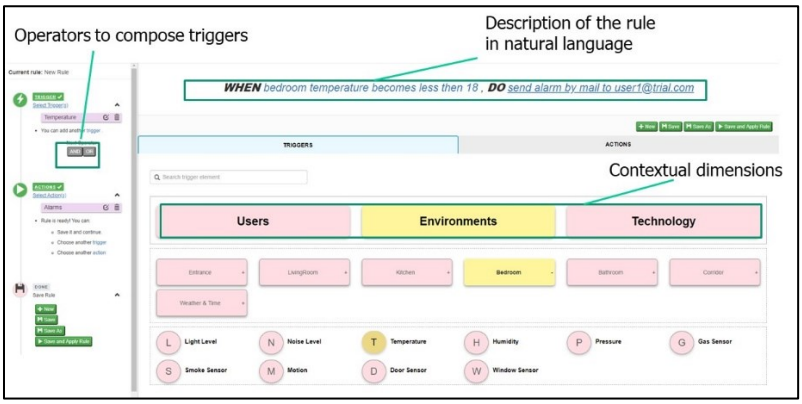

Figure 1. The TARE User Interface.

The block-based approach aims to apply the puzzle metaphor in the composition process of the automations. This type of metaphor has been long considered in the end-user development area starting from the Scratch[5] project first, and then the App Inventor[6] environment for developing mobile apps. Potentially, it can stimulate more creative, easy, satisfying, engaging experiences. It is used in several domains, such as education, IoT, industrial robotics. There are libraries, such as Google Blockly, that facilitate its implementation in specific tools. In this area, the puzzle elements should be designed in such a way that their shapes drive the process of creating rules structured in terms of triggers and actions. An example contribution in this area is BlockComposer[7] [14]. The user interface of this tool has three main parts (see Figure 2).

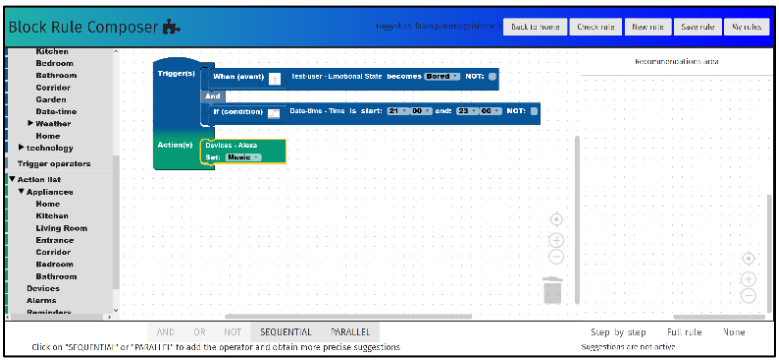

Figure 2: The BlockComposer User Interface

[5] http://scratch.mit.edu
[6] http://appinventor.mit.edu/explore/
[7] https://giove.isti.cnr.it/demo/pat/

On the left side there is the list of elements available for creating the rules, in the central part there is the main working area where the puzzle associated to the rule should be composed, and in the right side there is an area where recommendations of possible useful and relevant elements are presented. Thus, the rule block has two separate sections, one for inserting trigger blocks, one for action blocks. The “trigger” blocks are blue, and the “action” ones are green. The same colour is used for the part of the “rule” block indicating where triggers and actions can be inserted. Blocks that define the composition and the behaviours of triggers and actions (such as and/or/not operators, group and parallel) use the same colour with a different saturation level. To make clear the distinction between events and conditions, a modal window is shown every time a user selects a trigger type block. In both cases, an appropriate icon, a description of the temporality it models, and two possible cases are shown. The different representations used for events and conditions in the modal window are then maintained on the trigger blocks in the workspace.

The conversational approach seems relevant and interesting to support the specification of flexible automations in trigger-action format since it allows people to indicate the desired automation through natural language. However, the conversation should be able to remove possible ambiguities and allow the users to indicate precisely the desired effects. RuleBot[8] [6] is an example of conversational agent dedicated to supporting the end user in creating automations in environments populated by sensors and smart objects, with the ability to manage multiple triggers and actions, and clearly distinguish between events and conditions, which was not supported by previous solutions. It has been implemented with DialogFlow, and is able to perform a conversation with the user thanks to the use of machine learning and natural language processing techniques, and can manage the creation of personalization rules even starting from complex inputs. Figure 3 shows an example of conversation in Italian, where the chatbot asks the user for input, the answer is that if at 9 he has not taken the medicine then a message should be sent him to remember it. At this point the chatbot asks the user for the text that should be used in the remainder.

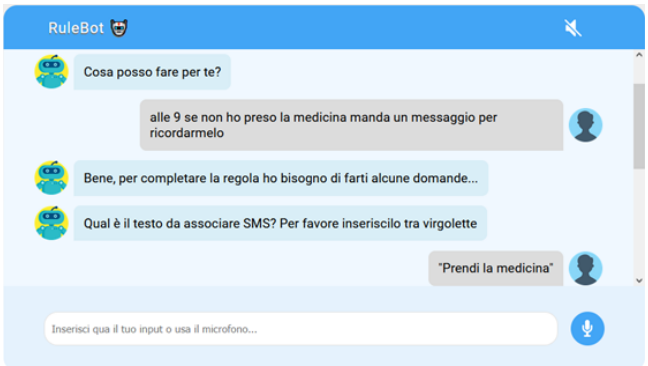

Figure 3: The RuleBot User Interface

[8] https://africa.isti.cnr.it/

Some initial studies [6] aiming at comparing the usability of the different compositional styles found that with the wizard style, where the user is guided in performing the relevant interactions, it is easy and fast to create simple rules, but creating more complex and elaborate rules may become long and tedious. Moreover, the chatbot requires first sometime before users get used to how the tool expects them to indicate the automations, in particular the types of natural language expressions that it is able to elaborate immediately, and then it becomes direct and efficient in understanding and activating the requested rules. In general, the performance of a chatbot depends on how the associated intents have been modelled, and the level of training received.

Real world deployment. In order to make the discussion more concrete and interesting, examples of deployments of these types of tools in real world should be provided. In this way it is possible to better understand the possible issues and advantages in deploying this approach in real cases studies. For example, some real experiences have been carried out in the PETAL European project, which aims to support remote older adults assistance through personalisation rules specified by caregivers or the older adults themselves. A number of trials have been carried out for some months with a personalization platform deployed in the homes of older adults with mild cognitive impairments, and some tools have been proposed to monitor when the rules have been created, activated and executed [11]. Another experience was carried out with a students' home, which was equipped with a number of sensors and appliances, and it showed some possible conflicts that can occur amongst rules created by different inhabitants of the same apartment [2]. Further experiences with small families have been recently reported [15]. Results and issues in such experiences can provide useful insights to understand how to deploy and use tools for personalising automations in general. In this context, it can also be useful to review the various social needs that such automations can help users to address (such as security, safety, well-being, health, energy saving, sociality) by providing specific examples.

Then, course participants should be asked to specify some automations in the form event / condition / action, with at least two rules including at least two triggers, and each rule should refer to different contextual aspects, by using two different visual tools publicly available on the Web (e.g. IFTTT and TAREME). In this way, they can take direct experience on these types of tools, and discuss possible advantages and disadvantages.

2.4 Intelligent Support

Intelligent support can be introduced in several ways in the process of creating automations. One is the use of recommendations. Rule editing is a process comprised of multiple steps, where appropriate suggestions should consider what has already been inserted. Beginners can benefit from being guided to discover the tailoring environment being used by seeing the structure of the rules and the possible next steps to take in order to complete the editing process. Advanced users can discover new possibilities

on their own. There are various possible types of recommendations relevant in trigger-action programming. In collaborative filtering the basic idea is that if user A has specified a rule, which is also indicated by user B, then there is a good probability that other rules specified by user B can still be relevant for user A. Recommendations obtained through generalization of the content of some part of the existing rules can be possible. For example: if a rule says that when the user enters the bedroom, then the lights should be turned on, one possible suggestion can be that when the user enters any room, then the lights should be turned on. Another type of recommendation can be obtained by trigger refinement, which means to narrow when the trigger should be fired by adding conditions that make the rules more suitable to meet needs more precisely specified. Other recommendations can be based on the actual user behaviour to detect their preferences. For example, they can be device-oriented: if the user prefers to use some specific device, then it can be meaningful to suggest rules that exploit it, for example for sending alarms. One further aspect is to have location-dependent recommendations, which aim to provide suggestions when the users are in a specific area that they seem to prefer. There are also time-oriented recommendations, which are rules with triggers associated with specific periods of the day, when the user is more inclined to receive information (e.g. reminders).

BlockComposer supports two policies in recommendations while users create rules: step-by-step in which the tool provides suggestions for the next element to include in the rule under editing; or full rule, in which complete rules are suggested. It exploits a data set of rules publicly available[9], and the generation of the recommendations is based on the use of compact prediction trees. RuleSelector [16] follows an approach where a set of rules are proposed based on the user behaviour detected through the smartphone sensors, from which the user can select the most relevant. The rules are proposed based on some metrics, for example, by applying confidence measures of the likelihood of the user performing action a when the context descriptors in the pre-condition frequent context itemset are true. Trace2TAP [19] is a different solution aiming to combine trigger-action programming and automated learning. For this purpose, it takes as input traces of user behaviour, and automatically generates rules that could automate a good portion of the observed instances of human actions. It clusters rules based on the similarity of their actuations, and ranks both clusters and the rules within each cluster based on a number of relevant characteristics

At this point it can be useful to propose an exercise with the students concerning the various criteria to consider for determining the rules to recommend, how to present such recommendations, and the effectiveness of the different modalities for presenting such recommendations.

Another aspect where the intelligent support can be useful is in helping users to understand whether the rules created actually perform the desired behaviour. For example, when multiple rules are created for a given context of use it can happen that there are conflicts, which means that different rules at the same time indicate to perform in different ways a certain object. For this purpose, previous work [12] has proposed to pro-

[9] https://github.com/andrematt/trigger_action_rules

vide visual representations of the state of the relevant contextual aspects, and automatically generate explanations of why or why not a given rule can be performed in that context. In this context the XAI question bank proposed for explainable AI [10] seems relevant since it provides a set of important questions that indicate many aspects that users often need to understand when looking at some automatic intelligent system.

Lastly, intelligent support can be useful to provide serendipitous support in creating, monitoring, and modifying automations in daily environments. In this perspective, smartphone-based augmented reality can play an important role. Indeed, it would avoid using special devices that many do not have and provide the possibility of direct interaction with the object of interest, monitoring nearby automations while moving, and the ability to select a real object directly and know the automations that involve it, add new automations, or modify existing automations. For this purpose, it can be useful to exploit object detection, which is a computer vision technique that has undergone significant improvement over the last years with the development of Convolutional Neural Networks (CNN) for the estimation of the position and class of the various objects that are present in an image.

The last part of the course can be dedicated to discussing with the participants the various concepts and tools presented, and also to analyse together whether other design aspects should be considered, and it can be concluded with a short discussion of a research agenda for this area.

3 Practical Work

It is important to highlight that practical work is important in order to acquire and consolidate the proposed concepts. There can be at least two types of interactive exercises, and a final discussion session. In the first exercise participants in pairs write for example five examples of automations in natural language, with different complexity and structure:

- Elementary trigger + elementary action
- Elementary trigger (including the NOT operator) + composed action
- Composed trigger (event+condition) + elementary action
- Composed trigger (conditon+condition) + composed action
- Composed Trigger (event+condition) + composed action

During this exercise participants can discuss together the rules specified and how well they specify the desired behaviour in order to allow participants to better understand possible ambiguities and problems in their execution.

Once the tools for creating rules have been introduced, participants should be asked to specify four automations in the form event / condition / action, with at least two rules including at least two triggers, and each rule should refer to different contextual aspects, by using two different visual tools publicly available on the Web (e.g. IFTTT and TAREME). In this way they can take direct experience on this type of tools and discuss possible advantages and disadvantages.

A third exercise can be focused on the possible criteria to automatically recommend possible relevant automations. Thus, the students should think about and discuss what data should be used for this purpose, and which criteria should be applied to select the most relevant automations to recommend. They should also discuss and indicate possible effective ways of presenting such recommendations.

The final discussion should aim to summarise the main relevant concepts and receive feedback from the participants on to what extent they are able to apply them in designing automations for the contexts that are interesting for them.

4 Conclusions

In modern training of designers, engineers, AI specialists, it is fundamental to assimilate a number of concepts, methods, and tools that can be useful to empower people to control the many connected objects and devices that they can encounter at home, work, and while moving.

In this paper I indicate a set of concepts, methods and exercises that can be useful for this purpose. They have been derived from my teaching experience and research work. There are also indications of examples of publicly available tools that can be useful in the teaching activities.

Acknowledgments. This work is partially supported by the Italian Ministry of University and Research (MUR) under grant PRIN 2017 “EMPATHY: EMpowering People in deAling with internet of THings ecosYstems”.

References

1. W. Brackenbury, A. Deora, J. Ritchey, J. Vallee, W. He, G. Wang, M. L. Littman, B. Ur: How Users Interpret Bugs in Trigger-Action Programming. CHI 2019: 552
2. L Corcella, M Manca, F Paternò, Personalizing a student home behaviour, International Symposium on End User Development, 2017, 18-33
3. F. Corno, L. De Russis, A. Monge Roffarello: Empowering End Users in Debugging Trigger-Action Rules. CHI 2019: 388
4. F. Corno, L. De Russis, and A. Monge Roffarello. 2021. Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization. In End-User Development 2021, Springer International Publishing, Cham, 71–86
5. C. Elsden, T. Feltwell, S. W. Lawson, J. Vines: Recipes for Programmable Money. CHI 2019: 251
6. Gallo S., Manca M., Mattioli A., Paternò F., Santoro C. (2021) Comparative Analysis of Composition Paradigms for Personalization Rules in IoT Settings. End-User Development. IS-EUD 2021. Lecture Notes in Computer Science, vol 12724. Springer, https://doi.org/10.1007/978-3-030-79840-6_4
7. G. Ghiani, M. Manca, F. Paternò and C. Santoro (2017). Personalization of Context-dependent Applications through Trigger-Action Rules. ACM Transactions on Computer-Human Interaction, Vol.24, Issue 2, Article N.14, April 2017.

8. Huang, H., and Cakmak, M., 2015. Supporting mental model accuracy in trigger-action programming. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15), 215-225
9. N.Leonardi, M.Manca, F.Paternò, C.Santoro, Trigger-Action Programming for Personalising Humanoid Robot Behaviour, ACM Conference on Human Factors in Computing Systems (CHI'19), Glasgow, Paper 445. 1.
10. Q. Vera Liao, D. M. Gruen, S. Miller, Questioning the AI: Informing Design Practices for Explainable AI User Experiences, CHI 2020
11. M. Manca, F. Paternò, C. Santoro, Remote Monitoring of End-User Created Automations in Field Trials, Journal of Ambient Intelligence and Humanized Computing, 2021, https://doi.org/10.1007/s12652-021-03239-0
12. M. Manca, F.Paternò, C. Santoro, L Corcella, Supporting end-user debugging of trigger-action rules for IoT applications, International Journal of Human-Computer Studies, Vol.123, 56-69
13. P. Markopoulos, J. Nichols, F. Paternò and V. Pipek (2017). End-User Development for the Internet of Things, ACM Transactions on Computer-Human Interaction (TOCHI) Vol- 24 Issue 2, 9, May 2017
14. A. Mattioli and F. Paternò. 2020. A Visual Environment for End-User Creation of IoT Customization Rules with Recommendation Support. In International Conference on Advanced Visual Interfaces (AVI '20). https://doi.org/10.1145/3399715.3399833
15. A. Salovaara, A. Bellucci, A. Vianello, and G. Jacucci. 2021. Programmable Smart Home Toolkits Should Better Address Households' Social Needs. In CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan. ACM, New York, NY, USA, 14 pages.
16. V. Srinivasan, C. Koehler, and H. Jin. 2018. RuleSelector: Selecting Conditional Action Rules from User Behavior Patterns . Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2
17. B. Ur, E. McManus, MPY. Ho and ML. Littman (2014). Practical trigger-action programming in the smart home. CHI 2014: 803-812
18. R. Yang, M. W Newman Learning from a learning thermostat: lessons for intelligent systems for the home, 2013 ACM international joint conference on Pervasive and ubiquitous computing, pp. 93-102
19. L. Zhang, W. He, O. Morkved, V. Zhao, M. L. Littman, S. Lu, and B. Ur. 2020. Trace2TAP: Synthesizing Trigger-Action Programs from Traces of Behavior. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4, 3, Article 104 (September 2020), 26 pages. https://doi.org/10.1145/3411838