A Visual Environment for End-User Creation of IoT Customization Rules with Recommendation Support

Andrea Mattioli & Fabio Paternò
Human Interfaces in Information Systems Laboratory
CNR-ISTI
Pisa, Italy
{andrea.mattioli, fabio.paterno}@isti.cnr.it

ABSTRACT

Personalization rules based on the trigger-action paradigm have recently garnered increasing interest in Internet of Things (IoT) applications. However, composing trigger-action rules can be a challenging task for end users, especially when the rules' complexity increases. Users have to decide about various aspects: which triggers and actions to select, how to combine multiple triggers or actions, and whether some previously defined rule can help in the composition process. We propose a visual environment, *Block Rule Composer*, to address these problems. It consists of a tool for creating rules based on visual blocks, integrated with recommendation techniques in order to provide intelligent support during rule creation. We also report on a first test which provided positive indications and suggestions for further design improvements.

CCS CONCEPTS

•Human-centered computing→User interface programming; •Information systems;

KEYWORDS

Trigger-action programming, Block-based programming, End user development, Recommendation systems, Internet of Things

ACM Reference format:

Andrea Mattioli, Fabio Paternò. 2020. A Visual Environment for End-User Creation of IoT Customization Rules with Recommendation Support. In *AVI '20: 2020 International Conference on Advanced Visual Interfaces, AVI '20, Sept 28-Oct 2, 2020, Salerno, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3399715.3399833

1 Introduction and motivation

IoT devices are increasingly common nowadays. End-User Development (EUD) is an approach that allows people to better control this wealth of devices. Through EUD they can modify a part of their applications to extend or change their possible behaviours. Empowering users to act on the objects and devices that populate their environment can be a step towards making them more effective to improve the quality of their lives. To make a shift towards a "culture of participation" [8] fully possible, designers should work to provide technical infrastructure which support users in the developing of appropriate critical thinking skills [7]. One relevant approach to allow users to customize their IoT environments is Trigger-Action Programming (TAP) [1, 9, 16]. TAP is an approach to EUD that allows people to define conditional rules to customize the behaviour of environments in which IoT sensors and actuators are present, possibly jointly with Web services. This abundance of possibilities can entice users to compose rules that go beyond the basic structure consisting of a trigger and an action.

There are many possible temporal evolutions in the rules' behaviour. A trigger can refer to the instant in which a change in the environment occurs, or to a specific Boolean condition lasting over a period of time. An action can be almost immediate (such as sending a message), can be extended over a longer timeframe (such as an utterance by a home assistant device), can achieve a change in the state of the environment that persists until it is restored (such as turning on a light or opening a window). Temporal aspects of triggers and actions are "a crucial source of ambiguity for TAP" [1]. For this reason, it is necessary to clearly define what these temporal aspects are, and how the components of the rules can be used. Even if other approaches (such as [5, 9]) allow the creation of compound rules, the adequacy of an environment based on blocks for this purpose has not been thoroughly investigated. Despite its wide application (for example in [15, 18, 19]), the block-based approach has not been particularly considered for TAP programming. Some examples are [10, 13, 14]. Further contributions in this area include [2], which is a Block based language, where the graphical aspects of blocks are used to support both the composition and the debugging phase. On the other hand, [4] relies on the puzzle metaphor to permit the creation of mobile applications, which may use IoT devices. With respect to these contributions, we follow a different approach, with less constraints on the editing of events and conditions, and with focus on the use of blocks in TAP rules composition.

Furthermore, Recommendation Systems (RS) are not used in TAP, except for a few cases [3, 12, 17]. An RS could be effective to provide users with support through example rules that can be relevant for their purposes. TAP systems can generate a vast

AVI '20, September 28-October 2, 2020, Salerno, Italy

ACM ISBN 978-1-4503-7535-1/20/09…$15.00
https://doi.org/10.1145/3399715.3399833

amount of data regarding users and their environments, which could be used to provide rule recommendations. We focus on data about rules and user preferences, instead of sensor data, to discuss the feasibility of an approach based on these features.

To summarize, the contributions of this work are our findings regarding two research questions: RQ1: identify a block-based visual representation that allows people to easily create structured rules while also limiting issues in temporal interpretations, RQ2: how to introduce a recommendation system for supporting rule creation in this type of environment.

2 Design of the Visual Environment

After analysing the relevant literature and previous work, the design phase began with the identification of a list of requirements for the new proposed environment.

R1. *Design of a flexible environment not tied to a specific application domain.* Flexibility should be pursued in the rule composition, e. g. via suggesting through the user interface a cause/effect mechanism without imposing a rule composition order [5, 9].

R2. *Easily express rules involving more than one trigger and one action*. Many useful behaviours definable through TAP rules need greater expressiveness than that provided by a language limited to rules with one trigger and one action [1]. However, the definition of a language capable of expressing Boolean logic in a clear way to non-technical users has not found optimal solutions [6].

R3. *Use appropriate block-based graphical elements and fully exploit both dimensions in creating and editing rules.* The graphical representation of the rule is an important factor to provide control to the user during rule creation: composing elements of the rule should be clearly supported [5].

R4. *Mitigate the potential errors deriving from incorrect user interpretations through visual clues.* It has been reported [1] that users tend to create incorrect mental mappings of the temporal aspects of rules. The temporal features of events and states (systematized in [11]) have to be clearly distinguished to prevent errors.

R5. *Suggest rules which can be of interest during rule creation*. In this regard, some aspects of RS need to be investigated further, such as how to introduce support from a RS while creating the rule, how to generate recommendations, and how to present them to user.

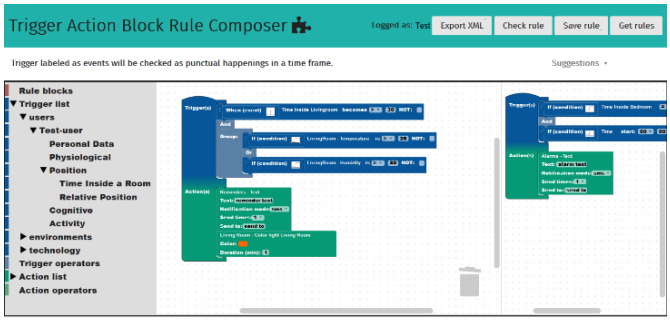

Figure 1: The main user interface of the editor.

One of the main aspects of the design phase was the definition of the concepts to convey via a block-based visual representation. The first concept was to express the inclusion relation between the various parts of a rule. This aspect has been expressed via the shape and the colour consistency of blocks, where the shape is used to immediately express what is contained in a single rule, and its components. Another important concept was modifiability, something that refers to the possibility of defining specific aspects of a block to better indicate its functionality. Given its optional nature, this aspect has been communicated via checkboxes which, when selected, change the shape of the associated block, adding or removing fields. The two dimensions of the user interface have been used to convey the distinction between sequential and parallel actions: sequential actions are assembled placing blocks vertically one after the other, parallel actions are represented by using a helper block to enable the horizontal arrangement of the corresponding blocks. To depict the logical composition of blocks, some operators are available: and/or for triggers, sequential/parallel for actions. The negation operator can be applied to a trigger using a checkbox. There is also the possibility to define a group element in order to compose elements according to the desired order in which the operators are to be applied to the various triggers. The distinction between triggers and actions is highlighted by colour. 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. The same colour distinction is used in the toolbox on the left (see Figure 1), from which the blocks are selected.

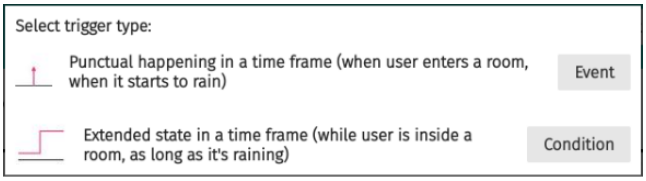

Figure 2: The modal window which explain the differences between event and state trigger.

The distinction between event and state type triggers is a main source of ambiguity for users. To mitigate this problem a modal window is shown every time a user selects a trigger type block. This window aims to illustrate as clearly as possible the event/state distinction. For each type, an appropriate icon (using the same representation proposed in [11]), a description of the temporality it models, and two possible cases are shown (see Figure 2). The different representations used for events and conditions in the modal window are then maintained on the trigger blocks in the workspace. When a user chooses "event" or "condition" from the presented modal window, the previously selected trigger block is added to the workspace, with the associated temporal block placed within it. The selection result is represented by text chunks added

dynamically on the trigger block, to create a quasi-natural language description of the trigger type, and by the associated icon (see example in Figure 3). The distinction between action types (immediate, extended, sustained) is indicated through tooltips on the corresponding blocks. Actions can also have specific behaviour when associated with a state trigger. Based on previous experiences (e.g. [1]), users may expect that, besides acting on a device upon activation of the rule, an action will act on it again at the end of the related state, restoring the device's previous state. This different type of action can only occur when state type trigger is selected, and the action provokes a change in the state of a device (sustained). To comply with this possible way of thinking, when these conditions are met, an icon representing two arrows is placed on the actions block (see the second "rule" block in Figure 3, which contains an action with the "revert" icon). This icon indicates that the action will behave as a "revert" action, e. g. turning on a light when the condition starts, tuning it off when it ends.

Figure 3: Two rule blocks illustrating example of event and state trigger and the two different rule behaviours (standard and with revert action).

After a paper mock-up prototyping phase, design ideas were implemented via Blockly, a Javascript library to allow developers to implement visual editors for predefined or custom languages. The main page of the editor (see Figure 1) consists of:

- A toolbox: placed on the left of the screen, which contains the available trigger and action hierarchies, and the associated operators.
- The main workspace: the area where users compose and edit rules via drag-and-drop.
- A secondary workspace: another workspace which contains the rule suggested by the RS (as an example or accepted and moved to the main workspace). Over this area a dropdown list contains the links to generate a suggestion and to move the suggestion to the main workspace.
- A toolbar: it contains a list of utility buttons for the main operations possible on the editor (save and get saved rules, make syntactic check on the rule being edited, export it as XML), the name of the current logged user, and a text area, which updates at each click on a block, and shows information about how to use it.

The editor relies on a server from which it receives the JSON description of triggers and actions available, which is used to generate the graphical elements that populate the toolbox. In the tool version considered in this paper the trigger hierarchy is organized in dimensions (User, Environment, Technology), which are then structured into categories. For example, the User dimension contains, among others, the Physiological, Position and Cognitive categories. The Cognitive category contains the Self-Assessment Value, Emotional state, Cognitive state, Training result, Training time and Last Connection Time triggers.

3 Recommending rules

A hybrid collaborative-content based RS has been designed and implemented to generate suggestions for relevant rules. Rules, the "items" to suggest, are not unitary entities, but are composed of triggers, actions, and operators between them. Therefore, the suggestion of a complete rule based on other full rules is an approach that may not be completely relevant to support users during rule composition, since it is not really driven by what the user entered. In the proposed solution, recommendations are presented considering the input provided by the user during rule composition. The recommendation is generated after the click on the "get suggestion" button, and built starting from the first trigger inserted into the "rule" block by the user, hence from an incomplete rule. This input is used together with a graph structure drawn from the trigger part of the rules extracted from the available rules' dataset. The trigger part of these rules is particularly relevant because they are richer in information, hence potentially more exploitable to obtain fruitful information than the action parts, which in general consist of only one or two actions per rule. The recommendation consists of a phase of graph creation, one of graph traversing, and one of identification of a matching "action" part.

The graph creation phase begins by extracting those rules whose first trigger value element equals to the element just entered by the user. This node will also be the central starting node. To build the graph, the rules are represented as sequences of transitions from a trigger element (event/condition type, operators, trigger value) to another. The resulting structure is a graph enriched with weights, which are obtained by counting how many times a given transition occurs in the extracted rules. This graph structure is then traversed from the starting node following a preferential approach, to obtain the trigger recommendation. Afterwards, the path between triggers and operators resulted from graph traversing is matched with the saved rules, to obtain an "action" part for the recommendation. Three scores have been used as most relevant for identifying the rules from which select the action part for the recommendation: 1) the similarity between the user currently logged and all the other users in terms of rules created, using the Person correlation on the user/action matrix; 2) the similarity between the partial rule obtained with the previous "graph traversing" method and the trigger part of the other rules stored in the rule repository via a modified Jaccard index; 3) the number of adoptions of the action part of the stored rules. The action part of the rule that maximizes these three scores is joined to the trigger part obtained via graph

traversing and suggested as a complete rule to the user. It is displayed in the secondary workspace on the right, which can be used as a reference or moved into the main workspace.

4 Discussion and Conclusions

The first user test involved 12 people, without previous experience with personalization systems and programming (3 with basic knowledge of HTML and CSS, 9 without any experience). The participants were introduced to the test goals, and the main functionalities of the editor were shown to them. A scenario was also described to them, to illustrate a possible use situation. The proposed scenario referred to the creation of personalization rules for an elderly relative who lives alone, to improve her life comfort. Overall, this introductory phase took about 20 minutes. Afterwards, they could freely try the platform without time constraints. Finally, they had to carry out five tasks:

T1) Compose a simple rule with an event and an action.
T2) Compose a rule with two triggers and one action.
T3) Compose a rule with a trigger and two sequential actions.
T4) Compose a rule that changes the state of a device as long as the condition lasts, then restore the initial condition.
T5) Start a rule freely, complete it by using the RS.

Some rules created during the execution of the tasks are:

1) When (event) daily steps become more than 500 => send notification to Elderly – send SMS to Caregiver
2) When (event) emotional state becomes Discouraged – and – if (condition) type of proximity is Inside Living room => start Living room Biorhythm Light
3) If (condition) Time is between 20:30 and 00:00 – and – When (event) Living room temperature becomes < 18 => send a notification to Elderly

Time recordings show that T1 (M = 73.9 s, std. dev. = 23.8 s) and T5 (M = 87.5 s, std. dev. = 30 s) were the fastest tasks, whilst the T2 and T3 were more problematic (M = 159.6 s, std. dev. = 91.1 s for the former and M = 116.3 s, std. dev. = 62 s with 2 errors for the latter). T4 yielded the worst performance (M = 176.8 s, std. dev = 79.2 s, with 5 errors). Further information was obtained by a post-test survey. It regarded quantitative assessments to be evaluated on a 5-points scale about specific aspects of the environment. Block metaphor, the creation of simple rules and the distinction between events and states were well received by users (4.5 on 5 on mean, standard deviation = 0.67 for all three). Also, the results of RS (M = 4.36, std. dev. = 0.67) and the distinction between sequential and parallel actions (M = 4.08, std. dev = 1) were appreciated. The use of logic operators for complex triggers was instead found a bit more problematic (M = 3.75, std. dev. = 1.29). Also, the distinction between standard actions and actions with a "revert" behaviour was not very easily understood (M = 3.5, std. dev = 0.9). Participants were positive how to use the editor autonomously after only one first session (M = 3.58, std. dev = 1.31). More insights emerged from observations of user behaviours during the test and from open questions. Users reported appreciating the specific composition aspect, the representation of blocks, and the support it provides:

- "You can see the construction of the rule as if you were doing it with Lego".
- "Speed of the block composer, possibility to rearrange the composition".
- "Blocks are intuitive".
- "I like blocks and their colours".
- "Simplicity of the association between trigger and action, ease of use of and/or/group connections".
- "The distinction between trigger and actions indicated by colours".
- "The distinction between triggers and actions is well organized".

Regarding the composition of structured rules, some critical aspects emerged from feedback and observations. The difference in the composition between the trigger and the action part was reported as a potentially problematic aspect by two participants, who expected to find the same operators for both sides of the rule. Two other participants had problems in correctly positioning a trigger operator. Even though these problems can be solved by simply adding a check on the block's connections, they indicate potential issues with how users perceive these concepts. Also, the task regarding the distinction between "standard" and "revert" actions was reported as difficult by six participants. A different and more explicit representation of this behaviour can be useful.

Regarding the use of RS in tasks where it was not required, three users reported having used it for checking which triggers and which actions to use together with the ones already chosen, two to check how to compose a structured rule, one to have feedback on the created rule. The RS has been therefore used as a "discovery" as well as a "support" tool. A more integrated RS in the environment may be useful to further speed up its use, and better integrate results into the work area. A different way to display results can be shown in the suggestion workspace: the single blocks that can be used at a given time in the rule construction (instead of the full rule at request). This can be obtained via a background call to the RS which can generate a suggestion concerning the last trigger inserted. Then, instead of a full rule, only the blocks relevant to this last trigger should be shown.

In conclusion, our research has confirmed the block paradigm as appropriate for TAP. Regarding the first research question, using explicit different indications to express the relevant concepts resulted to be a valid method for helping users to understand the event/condition difference. Furthermore, a colour distinction appears to be an appropriate enhancement to point out the trigger-action distinction in the editor. However, a visual representation suitable to effectively describe complex rules has to be better defined. Regarding the second research question, a methodology to implement a RS for the trigger-actions rules in the IoT context has been presented, and received positive feedback from first user tests.

Acknowledgements. This work has been supported by the PRIN 2017 EMPATHY Project, http://www.empathy-project.eu/.

REFERENCES

[1] Will Brackenbury, Abhimanyu Deora, Jillian Ritchey, Jason Vallee, Weijia He, Guan Wang, Michael L. Littman, and Blase Ur. 2019. How Users Interpret Bugs in Trigger-Action Programming. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Article Paper 552, 12 pages. https://doi.org/10.1145/3290605.3300782

[2] Fulvio Corno, Luigi De Russis, and Alberto Roffarello. 2019. My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor. 18–33. https://doi.org/10.1007/978-3-030-24781-2_2

[3] Fulvio Corno, Luigi De Russis, and Alberto Roffarello. 2019. RecRules: Recommending IF-THEN Rules for End-User Development. ACM Transactions on Intelligent Systems and Technology 10 (09 2019), 1–27. https://doi.org/10.1145/3344211

[4] Jose Danado and Fabio Paternò. 2012. Puzzle: A Visual-Based Environment for End User Development in Touch-Based Mobile Phones. In Human-Centered Software Engineering, Marco Winckler, Peter Forbrig, and Regina Bernhaupt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 199–216.

[5] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Empowering End Users to Customize Their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools. ACM Trans. Comput.-Hum. Interact. 24, 2, Article Article 12 (April 2017), 52 pages. https://doi.org/10.1145/3057859

[6] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Specification of Complex Logical Expressions for Task Automation: An EUD Approach. 108–116. https://doi.org/10.1007/978-3-319-58735-6_8

[7] Gerhard Fischer, Daniela Fogli, Anders Mørch, Antonio Piccinno & Stefano Valtolina (2020) Design trade-offs in cultures of participation: empowering end users to improve their quality of life, Behaviour & Information Technology, 39:1, 1-4, DOI: 10.1080/0144929X.2020.1691346

[8] Gerhard Fischer. 2011. Understanding, fostering, and supporting cultures of participation. interactions 18, 3 (May 2011), 42–53. DOI: https://doi.org/10.1145/1962438.1962450

[9] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. ACM Transactions on Computer-Human Interaction 24 (04 2017), 1–33. https://doi.org/10.1145/3057861

[10] T. L. Guilly, J. H. Smedegård, T. Pedersen, and A. Skou. 2015. To Do and Not to Do: Constrained Scenarios for Safe Smart House. In 2015 International Conference on Intelligent Environments. 17–24. https://doi.org/10.1109/IE.2015.11

[11] Justin Huang and Maya Cakmak. 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). Association for Computing Machinery, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

[12] Hanjo Jeong, Byeonghwa Park, Minwoo Park, Ki-Bong Kim, and Kiseok Choi. 2019. Big data and rule-based recommendation system in Internet of Things. Cluster Computing 22, 1 (01 Jan 2019), 1837–1846. https://doi.org/10.1007/s10586-017-1078-y

[13] Bak Nayeon, Byeong-mo Chang, and Kwanghoon Choi. 2018. Smart Block: A Visual Programming Environment for SmartThings. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 32–37. https://doi.org/10.1109/COMPSAC.2018.10199

[14] Daniel Rough and Aaron Quigley. 2017. Overcoming mental blocks: A blocks-based approach to experience sampling studies. 2017 IEEE Blocks and Beyond Workshop (B&B), 45–48. https://doi.org/10.1109/BLOCKS.2017.8120409

[15] Andrew Stratton, Chris Bates, and Andy Dearden. 2017. Quando: Enabling Museum and Art Gallery Practitioners to Develop Interactive Digital Exhibits. In End-User Development, Simone Barbosa, Panos Markopoulos, Fabio Paternò, Simone Stumpf, and Stefano Valtolina (Eds.). Springer International Publishing, Cham, 100–107.

[16] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). Association for Computing Machinery, New York, NY, USA, 3227–3231. https://doi.org/10.1145/2858036.2858556

[17] Beidou Wang, Xin Guo, Martin Ester, Ziyu Guan, Bandeep Singh, Yu Zhu, Jiajun Bu, and Deng Cai. 2018. Device-Aware Rule Recommendation for the Internet of Things. InProceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18) . Association for Computing Machinery, New York, NY, USA, 2037–2045. https://doi.org/10.1145/3269206.3272009

[18] David Weintrop. 2019. Block-Based Programming in Computer Science Education. Commun. ACM 62, 8 (July 2019), 22–25. https://doi.org/10.1145/3341221

[19] David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David Shepherd, and Diana Franklin. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. 1–12. https://doi.org/10.1145/3173574.3173940