Comparative Analysis of Composition Paradigms for Personalization Rules in IoT Settings

First Author[1][0000-1111-2222-3333] and Second Author[2][1111-2222-3333-4444]

[1] Princeton University, Princeton NJ 08544, USA
lncs@springer.com

Abstract. The rapid pervasive diffusion of Internet of Things technologies has opened up many opportunities for people to directly personalise the behaviour of surrounding objects and devices based on the dynamic events that can occur. To this end, several tailoring environments have been proposed supporting the end-user creation of trigger-action rules. Such tools can support different composition paradigms. In this paper we present a study that analyses three composition paradigms (graphical wizard, block-based, and conversational) to better understand how well they support rule creation activities. In order to make the analysis consistent we considered three implementations of such composition paradigms supporting the same set of triggers and actions. We have carried out a first user study in order to gather empirical feedback for substantiating our analysis, which provides indications of the pros and cons of each approach.

Keywords: Internet of Things, Trigger-action rules, Tailoring Environments.

1 Introduction

Over the past decade, we have witnessed an increasing diffusion of the Internet of Things (IoT) and related technologies, with the number of connected "things" (devices and physical objects) expected to reach 35 billion worldwide in 2021.Such a pervasive technological trend has opened up new possibilities in terms of end-user development. A variety of events can dynamically be detected in IoT settings and many connected objects and devices can be activated accordingly. This has stimulated growing interest in Trigger-Action Programming (TAP), where triggers can be events and/or conditions, and the actions indicate the functionalities to activate. Several commercial tools supporting this approach are available, such as IFTTT[1] and Zapier[2].

However, while creating personalization rules through such environments is relatively simple, often users either cannot specify the intended behaviour because of limitations of the language supported by the considered tool, or, even when they can, they obtain rules whose performance does not generate the expected behaviour. Brackenbury et al. [2] discuss a number of potential bugs in TAP, and highlight how often the

[1] https://ifttt.com/

[2] https://zapier.com/

temporal aspects of both triggers and actions is a crucial source of ambiguity. For example, the difference between events and conditions is often not noticed by users [10], also because many tailoring environments do not clearly highlight that they are two different temporal concepts. Indeed, one of the most known tools in this area is IFTTT, and it does not provide clear indications in this respect, and does not even support rules with multiple triggers. One further problem is that several tailoring environments do not provide the possibility of creating rules that are activated if some event does not occur in a given period of time (they do not support the NOT operator associated with a trigger). Overall, it seems that designing tailoring environments for TAP that are able to address such issues and offer satisfying usability is still an open issue, and deserves further investigation. One aspect that can be important to consider is the composition paradigm to support the development process of trigger-action rules. By composition paradigm we mean how the tailoring environments guide the rule development process: how they present the relevant concepts and interact with users.

For such reasons we have conducted a study that considers tailoring environments that on the one hand provide support for the problematic aspects mentioned above (clear distinction between events and conditions, support of NOT operator, possibility to compose multiple triggers and actions), and on the other hand support different composition paradigms. The goal is better understanding the pros and cons of each solution, and more generally to provide useful indications for designers and developers of tailoring environments for IoT settings. In particular, the composition paradigms considered in this paper are: graphical wizard, where users can create the rules through various selections performed on logically organised lists of options; block-based, where the puzzle metaphor is used to enable users to put together the elements composing the rules; and conversational, in which a chatbot in natural language asks user information, receives answers, and provides requests for clarification.

In the paper after discussion of related work, we introduce the three tailoring environments considered. Then, we report on a study carried out with a number of students without experience in programming, who created rules with different structures with the three considered tailoring environments. We then discuss the lessons learnt, and draw some conclusions and indications for future work.

2 Related Work

A first comparative study [12] analysed three Android apps (Tasker, Atooma, and Locale) for creating context-dependent behaviours in terms of expressiveness and usability, and found that the environment supporting the widest set of expressions for specifying rules (Tasker) was also the one that was found to be the most difficult to use (highest performance time, error numbers, and unsuccessful performance numbers). Then, a study [4] carried out a literature review on the design and evaluation of tools for smart home control, and an experimental study in which three tools (Tasker, IFTTT, Atooma) were compared in order to identify the interaction mechanisms that end users appreciate most. On the basis of the obtained results, a set of general design implica-

tions for the development of tools for smart home control and management were proposed. Another comparative user study between IFTTT and Atooma was reported in [3]. It involved students who used them in small groups. Participants observed that IFTTT provided more services for composition than Atooma, which was found easier to use and intuitive, and also able to combine multiple triggers. Overall, such studies mainly focused only on solutions following the graphical wizard approach.

More recently, a design space to compare tools for end-user development of IoT and/or robotic applications has been put forward [14]. It is composed of various logical dimensions that highlight the aspects to analyse and consider, such as the type of metaphor used for representing the relevant concepts and the programming style adopted for creating the desired behaviour. A comparison between composition paradigms for trigger-action rules was introduced in [8]. They considered two wizard-based tools (one with more guidance than the other) and one based on the graph metaphor. One of the results of the comparative study revealed that the visual data-flow paradigm downgraded both user performance and satisfaction. Thus, we decided to no longer consider such paradigm in our study, and expand our analysis to two rather different paradigms (conversational and block-based).

Conversational interfaces seem a promising approach. Valtolina et al. [15] reported on a study evaluating the benefits of a chatbot in comparison to traditional GUI, specifically for users with poor aptitude in using technologies. They considered applications in the healthcare and smart home fields, and found that for the user experience the chatbot application appears to be better than the traditional one. Another relevant work exploring the use of chatbots is CAPIRCI [1], a multi-modal web application supporting end users to define tasks to be executed by collaborative robots. In general, the conversational interaction style has received limited attention for supporting creation of trigger-action rules. InstructableCrowd [9] is a framework which enables users to converse with the crowd through their phone and describe a problem. Then, it provides a graphical interface for crowd workers to both chat with the user, and compose a rule with a part connected to the user's phone sensors, and a part connected to the user's phone effectors in order to solve the problem. HeyTap [6] allows users to communicate their personalization intentions, which are used, along with related contextual and semantic information, to recommend rules that are able to map the abstract user needs to entities among those in the available dataset. Overall, such work has not focused on the use of chatbots for creating trigger-action rules since in IntructableCrowd the rules are created through a graphical interface, while HeyTap uses the chatbot to receive parameters for recommending rules from an existing dataset. An exploration of using chatbot to create trigger action rules is in [11] but it only considers single trigger – single action rules, and does not distinguish between events and conditions. Thus, in the paper we provide an original comparative analysis of three composition paradigms: one is based on a graphical wizard that aims to guide the users by selecting available options shown graphically (TAREME [5][3]), one follows the puzzle metaphor to support the composition activity (BlockComposer [13][4]), and the last one adopts a conversational style.

[3] https://tare.isti.cnr.it/RuleEditor/login

[4] https://giove.isti.cnr.it/demo/pat/

3 The Tailoring Environments

The three tailoring environments considered in the study were configured in such a way to support the same triggers and actions, and the same operators, according to the same language. In this way they were able to express the same rules, and thus we avoided that greater capabilities of one tool could influence the user perception of its usability and usefulness. The triggers considered refer to three contextual dimensions: user, technology, and environment; while the actions considered state changes of home appliances, or the generation of reminders, alarms, or the activation of Alexa services.

The tool supporting the graphical wizard composition (TAREME) is visually structured into two main parts, one for triggers and one for actions, and one sidebar providing feedback on the progress achieved in creating the rules. Both the trigger and action parts are organized in terms of main categories, which, when selected, unfold their subcategories, and it is possible to iterate until the basic elements with their attributes are visualised. The puzzle metaphor has been used in different environments (e.g. smartphones [7], robots [16]). In this case the tool considered (BlockComposer) is structured into a lateral panel from which the elements can be selected and dropped into the main area, while the composition operators are available in the bottom part. The available elements for composing the rules in the lateral panel are hierarchically organized as well. There is a block element associated with the rule that has to be completed in the composition process.

The tool supporting the conversational interface is RuleBot, which has been implemented with DialogFlow. It has been obtained by defining an intent for each possible trigger and action, and the corresponding entities allow the identification of the terms in the intents. Some contexts have been defined in order to direct the conversation in such a way that users enter the necessary information. After the chatbot welcome, the user can enter a possible trigger or action, and then the chatbot provides feedback, and asks for the remaining information to complete the rule (an example in Figure 1). When the user input corresponds to one of the defined intents, DialogFlow checks the presence of the necessary entities, and send them to a Node.js server, which will then generate the natural language description of the rule created. In the supported conversations users can delete the last item entered, and asks for a summary of the rule so far created.

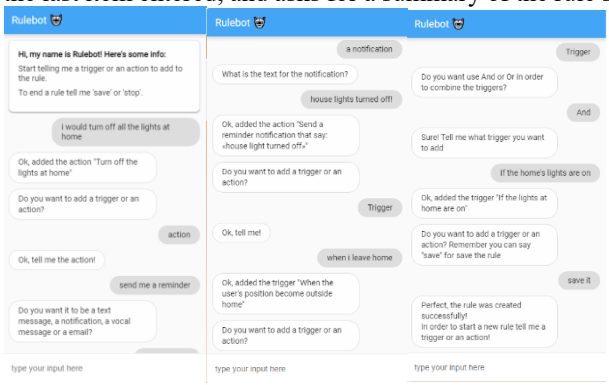

Fig. 1. Example of dialogue with the conversational interface.

4 User Study

We carried out a user test to better assess the strong and weak points of the three approaches proposed for programming automations via trigger-action rules by non-professional developers.

4.1 Participants

The participants for this test were recruited in a course of a Digital Humanities degree. Twenty-three users (55.6% females) with age ranging between 22 and 55 (mean=27.22, std. dev.= 8.17) were involved in the user study. Their knowledge of programming languages was categorized into five different levels from no knowledge to very good knowledge: 1: No knowledge in programming; 2: Little Knowledge (which means: knowledge of HTML, CSS, and basic knowledge of a programming language, such as JavaScript); 3: Medium Knowledge (knowledge of a programming language, basic knowledge of another language, such as PHP or Java or C++); 4: Good Knowledge (good knowledge of the known languages); 5: Very Good Knowledge (knowledge of development languages at a professional level). In our analysis, from the initial 27 participants, we excluded the data of 4 participants because they had good or very good knowledge of programming languages. As such, all the 23 participants had a knowledge of programming languages between no knowledge and medium knowledge. In particular, 5 of them (22%) declared no programming experience, 11 (48%) had little experience, 7 (30%) reported a medium level in programming experience.

4.2 Test Organisation

The test was introduced in a course lesson given by teleconference because of the pandemic crisis. The students received a brief introduction to the study, the three tools, and the trigger-action rule structure, as well as an explanation of the main features of the three tools, and a short introductory video for each. The students were required first to login to the three tools and then familiarize themselves with each for some time. Then, the students had to write down (using natural language) five rules that had to follow a predefined structure. Each of these rules needed to be specified using each of the three tools. The rules had to comply with the following structures indicating the number of elements, whether events or conditions should be considered, and the type of contextual aspect and action to consider:

- Elementary trigger (technology) + elementary action (reminder)
- Elementary trigger (including the NOT operator) + composed action (alarm, light)
- Composed trigger (event+condition: environment+environment) + elementary action (reminder)
- Composed trigger (conditon+condition: environment+technology) + composed action (alarm, light)

- Composed Trigger (event+condition: user+technology) + composed action (light, reminder)

Such rules were presented to participants in the same order. The idea was to progressively increase the complexity of the rules, which was mainly connected with two aspects: a higher number of triggers and actions to specify implied a more complex rule. In addition, we assumed that a trigger that needs the inclusion of a NOT operator would be more difficult than a trigger that does not need it.

After receiving the introduction to the exercise, the students were free to complete the tasks autonomously. In the last phase they had to fill in an online questionnaire asking for some socio-demographic information (age, gender), and also indicating their expertise in programming. In addition, they also had to rate on a 1(min) to 5 (max) scale how much they agreed on statements concerning some aspects of the proposed environments, and provide observations on positive/negative aspects they noticed on the assessed systems, and recommendations for possible improvements. We asked them to provide honest feedback, saying that the test is an evaluation of the tools and not of their performance, and that any issue they could find would be useful to improve the tools. The final anonymous questionnaire was administered remotely, so the possibility of social pressure was very low. The students had to complete the test and the questionnaire remotely in a few days.

4.3 Questionnaire Results

Question 1 I found it easy to edit the "trigger" part of a rule using the user interface of the TAREME tool (Median=4; Min=2; Max= 5)

The vast majority of people liked the intuitiveness of the tool. In particular, its hierarchical representation was appreciated for its immediacy, and the support it offers in progressively refining the level of detail in the specification of the contained elements. In the composition process, the two main parts of a rule (the one dedicated to triggers and the one dedicated to actions) were judged easily distinguishable, and the layout was appreciated for its clarity. Two users pointed out that, after a minimal phase of familiarization, using the tool (i.e. for editing the triggers) is easy. Another user pointed out that, while editing a trigger, sometimes it can happen that its parent node is no longer visible, and in such cases the user might lose the context. Another highlighted that not all the trigger names are equally clear and self-explanatory. One user pointed out that the set of available triggers should be extended, while another found the use of IF vs. WHEN within the tool not immediately clear. Finally, one user pointed out that the natural language feedback appearing on the top, though useful, could be improved.

Question 2 I found it easy to edit the "trigger" part of a rule using the user interface of the Block Composer tool (Median=4; Min=2; Max=5)

A significant part of users (10) clearly expressed their appreciation towards Block Composer. They found it intuitive and usable, and especially liked the idea of using the jigsaw metaphor to guide the rule specification process. One of them particularly liked the availability of a lateral, compact menu that can be expanded as needed: this allows users to have a ‘clean’ view of the main panel (by hiding what is not currently relevant).

Another one found BlockComposer as very pleasant to use, also appreciating the sound that is rendered when a block gets composed. One user pointed out that the movement of the blocks is not always 'fluid' when using a desktop PC with a mouse, compared to touch-based tablets. One highlighted that the size of the characters used for rendering the names of the triggers, as well as the distance between the associated UI elements should be increased. However, one user did not find the jigsaw metaphor particularly intuitive: he said that sometimes the various pieces seem to go "out of control". One user, while appreciating the overall intuitiveness of the tool, pointed out that, when adding new blocks, they are placed in a position that is not always visible to users (i.e. in the top-left part), which forced her to scroll the main panel to find the new element.

Question 3 - I found it easy to edit the "trigger" part of a rule using the user interface of the RuleBot tool (Median=3; Min=1; Max= 5)

Six users explicitly said that RuleBot is easy to use even though some of them complained that it is not always able to correctly recognize user input. One user found it easy to edit the "trigger" part of a rule, while another reported it to be very easy and convenient to use the chatbot especially for those not very familiar with technology. A pair of users declared to have had some initial difficulties that disappeared after a quick familiarization with the tool. However, several users pointed out that, while on the one hand the interaction with the chatbot is quick, on the other hand the considered chatbot did not always correctly understand the input given by the user. One complained about the 'rigidity' with which it sometimes works (i.e. not being very flexible in the input it can recognize). A pair of users complained that the sentences to use for interacting with the chatbot need to be precise and that, in order to be understood, it is necessary for the user to properly 'tune' his language to the one used by the chatbot. In addition, another user reported that sometimes the chatbot asks questions that seem unrelated to the current conversational context, and which the user needs to answer to continue the interaction (this likely happens when the user is less aware of the current point of interaction and thus interprets the chatbot's questions as unrelated to the current context, which might not be the case).

Question 4. I found it easy to compose two "triggers" of a rule using the user interface of the TAREME tool (Median=4; Min=2; Max= 5)

On the one hand, the majority of participants (around 15 users) declared having found this task quite easy. On the other hand, practically all of them highlighted that the button dedicated to add the operators, as being placed in a lateral bar in the left-hand part of the window, results not particularly intuitive and visible. One of them further added that this becomes particularly challenging when the window is resized, since the lateral menu can even become hidden (in extreme cases); this same user highlighted that, even if one just focuses on the central panel of the main window, it is not clear how to include operators. One of them suggested placing a dedicated button in the central part of the window.

Question 5. I found easy composing two "triggers" of a rule by using the Block Composer (Median=3; Min=2; Max= 5)

Seven users explicitly declared to like the way in which BlockComposer supports the combination of triggers. Nevertheless, some aspects were judged as problematic. A user found this tool not intuitive to use when combinations of triggers are to be created,

as it is not immediately clear that another block should be added to specify the composition operator. Another user reported difficulties in moving the elements and in the fact that the operators are not very visible in the UI. To this regard, seven users complained about the limited visibility of operators within the UI (since – they noted – they are placed in the bottom part of the UI). However, at the same time most of them declared that this issue is ameliorated by the fact that an alert notification is given to users when it is necessary to include an operator to compose triggers. One user complained that the elements in the UI (i.e. the 'trigger' blocks) are rendered sometimes too close each other, which could create confusion in the layout. A pair of users reported that sometimes it seems that the tool did not correctly frame some pieces together.

Question 6 I found it easy to compose two triggers of a rule using the user interface of the Rule Bot tool (Median=4; Min=1; Max= 5)

The majority of users (16) appreciated the easiness with which it is possible to compose two triggers with the RuleBot. Many users highlighted that this is because it is directly the chatbot that suggests the user to combine a new trigger with a previously specified one, also highlighting the available operators. The remaining users complained that the chatbot needs specific statements to recognize correctly the user's input, thus showing limited flexibility. In this regard one user reported to have unsuccessfully tried to provide the chatbot with two triggers at once (not supported by this tool at the test time).

Question 7 I found it easy to edit a "trigger" that uses the "not" operator using the user interface of the TAREME tool (Median=4; Min=2; Max= 5)

The vast majority of users (around 20) found this very intuitive in TAREME. For the few remaining users, it seems that their difficulties were at a more conceptual level. For instance, some of them did not understand that, in order to verify that an event is not occurred, an associated interval of time should be specified to indicate the temporal period when it should not occur.

Question 8 I found it easy to edit a "trigger" that uses the "not" operator using the user interface of the BlockComposer tool (Median=4; Min=1; Max= 5)

The vast majority of users found intuitive to carry out this task using BlockComposer. As for the remaining users, one user complained that the checkbox for including the NOT operator was not very visible, another one had some difficulties with specifying the associated time interval.

Question 9 I found it easy to edit a "trigger" that uses the "not" operator using the user interface of the RuleBot tool (Median=4; Min=1; Max= 5)

15 users gave overall quite positive comments, not having encountered specific difficulties in carrying out this task. The rest of the users pointed out that the chatbot did not always immediately recognize the negation.

Question 10 I found the distinction between event and condition simple when editing the "triggers" of a rule using the user interface of the TAREME tool (Median=4; Min=2; Max= 5)

The vast majority of users (19) found this quite intuitive, using TAREME. As for the remaining ones, a pair of users noted that, while setting an event or a condition using TAREME is intuitive, the tool should better support users in understanding (at a conceptual level) the underlying concepts associated with events and conditions. A pair

of other users said that the difference between the two, as rendered just by the keywords "IF" and "WHEN" are not immediately understandable: one of them however added that having available explicitly in the UI the information about the kind of trigger associated with each keyword (conditions for "IF" and "events" for WHEN) was very helpful.

Question 11. I found the distinction between event and condition simple when editing the "triggers" of a rule using the user interface of the BlockComposer tool (Median=4; Min=2; Max= 5)

The vast majority of user (19) declared that the tool allows them to do such task in a quite easy manner, since every time the selection of the most suitable option (between events and conditions) is required by the user, the tool first explains the distinction (showing a short explanation), and then it allows the user to select the correct one. As for the remaining participants, one user declared that this distinction was clear only after having completed all the parts of the concerned rule, i.e. after having the possibility to read the whole rule.

Question 12. I found the distinction between event and condition simple when editing the "triggers" of a rule using the user interface of the Chatbot tool (Median=3; Min=1; Max= 5)

Eleven users reported that this part is sufficiently intuitive. Most of them said that they appreciated the fact that, if the user does not explicitly use 'IF' or 'WHEN' within a trigger when it is being created, the chatbot explicitly asks the user whether the newly added trigger should be considered as an event or a condition. As for the remaining users, one of them pointed out that this distinction is not sufficiently highlighted in the tool: in his view it seems somewhat "expected" that the user should know this difference. Other five users reported some difficulties in understanding this difference

Question 13. I found it easy to edit the action part of a rule using the user interface of the TAREME tool (Median=4; Min=3; Max= 5)

Almost the totality of users (20) said that the task is very easy and intuitive, as the tool offers a clear, simple and coherent UI for specifying the action part of a rule using TAREME. The other users did not report any specific difficulties: most of the time they just highlighted some aspects that they found not completely clear in the UI, for instance some of them did not understand the reason why the receiver of a message is required by the tool only for emails or sms and not for e.g. vocal notifications (this was due because in the latter case it was assumed that the person that is currently using the associated application -e.g. Alexa- will receive the notification).

Question 14. I found it easy to edit the action part of a rule using the user interface of the BlockComposer tool (Median=4; Min=2; Max= 5)

Also for BlockComposer, the part dedicated to specifying the action was judged intuitive to be used by practically all of the participants. One user added that using different colours for visualizing triggers and actions makes very easy to identify the action part.

Question 15. I found it easy to edit the action part of a rule using the user interface of the RuleBot tool (Median=4; Min=1; Max= 5)

Nine users found the interaction with the chatbot for editing rules sufficiently satisfactory. However, most of the remaining users complained that the chatbot does not

always understand the input provided by the user and often the user is forced to use as much as possible specific sentences for achieving better results. One user did not appreciate the way in which currently it is possible to edit a rule with this tool. Another user found not very clear the possibility to undo an interaction using the tool.

Question 16. I found it easy to compose two actions within a rule using the user interface of the TAREME tool (Median=4; Min=2; Max= 5)

The vast majority of users (19) found this task very easy using TAREME. As for the remaining users, one user highlighted that there is no specific operator to use for composing actions using this tool (differently from what is currently done with triggers). Alongside the same line, one user reported that it is not completely clear whether the tool considers (by default) the second action to be carried out just after the first one. A pair of users highlighted that the lateral menu on the left part of the tool might get hidden when the user significantly resizes the window.

Question 17. I found it easy to compose two actions within a rule using the user interface of the BlockComposer tool (Median=4; Min=3; Max= 5)

Six users reported that the element (a block) dedicated to specifying the composition of actions (through the "sequential" and "parallel" operators) could be better highlighted in the UI (currently it appears in the footer of the page). A pair of users reported some difficulties in realizing that, to compose two or more actions, a further block is needed. The remaining users reported that overall this task was easy to carry out using BlockComposer.

Question 18. I found it easy to compose two actions within a rule using the user interface of the RuleBot tool (Median=4; Min=1; Max= 5)

Almost all users appreciated the support offered by the tool for combining actions. Some of the others highlighted that the tool sometimes does not correctly understand the input provided by the user. One user pointed out that in the first place one could be tempted to provide the two actions at once (at the time of the test not supported by the tool, which asked them one by one).

Question 19 The way rules are built using the TAREME tool is fast / efficient (Median=4; Min=2; Max= 5)

Almost all the participants found this tool efficient, quick, and easy to use. Some of them appreciated the lateral bar in the left-hand part, however it was noted that it might get hidden in some cases. Some users especially appreciated the possibility to have the rule written in natural language on the top part of the window.

Question 20 The way rules are built using the BlockComposer tool is fast / efficient (Median=4; Min=2; Max= 5)

The majority of users found this tool easy to use. The jigsaw metaphor exploited was found very intuitive. However, some of them noted that sometimes the blocks are moved in a way that is not extremely ‘fluid’. One user suggested including the specification of the rule in natural language, beyond the block-based specification. Another user highlighted that the process of creating a rule using BlockComposer is a bit affected (in terms of efficiency) by the time needed to drag the various elements in the central, main panel.

Question 21 The way rules are built using the RuleBot tool is fast / efficient (Median=4; Min=1; Max= 5)

Several users expressed appreciation for this tool. However, many of them highlighted that by using this tool the process of creating rules is not very efficient as sometimes the chatbot is not able to understand the input provided by the user, therefore its flexibility in that should be increased. One user suggested better highlighting the terms according to which a user is more likely to have his input correctly recognized by the bot.

Question 22. Overall, I found the interaction with the user interface of the TAREME tool satisfactory (Median=4; Min=1; Max= 5)

The vast majority of the users liked this tool and found it structured, clear, and satisfactory in its use. The distinction between triggers and actions was found clearly visible and increased the clarity of the tool. Users also appreciated the availability of the natural language description of rules. One user suggested improving the lateral menu (in the left-hand side), to avoid that it gets hidden when the user resizes the window.

Question 23. Overall, I found the interaction with the user interface of the BlockComposer tool satisfactory (Median=4; Min=3; Max= 5)

This tool was found easy, intuitive and pleasant to use by many users, even "entertaining" by some of them. However, further improvements could be done in the support provided for editing rules. In addition, sometimes the various blocks do not immediately frame each other. A user said that a quick phase of familiarization might be needed to understand how to place the pieces to build a rule.

Question 24. Overall, I found the interaction with the user interface of the RuleBot tool satisfactory (Median=3; Min=1; Max= 5)

Almost half of users expressed overall appreciation for the tool, and found it easy to use, even though a quick phase of familiarization might be needed. One user suggested improving the tool in handling possible user errors (e.g. adding the possibility of undoing the last action), or better handling situations in which the user needs to edit a rule. However, most of the complaints regarded the limited ability of the bot in understanding the inputs by the user. In addition, some users reported that they sometimes preferred to re-build a rule from scratch instead of modifying it.

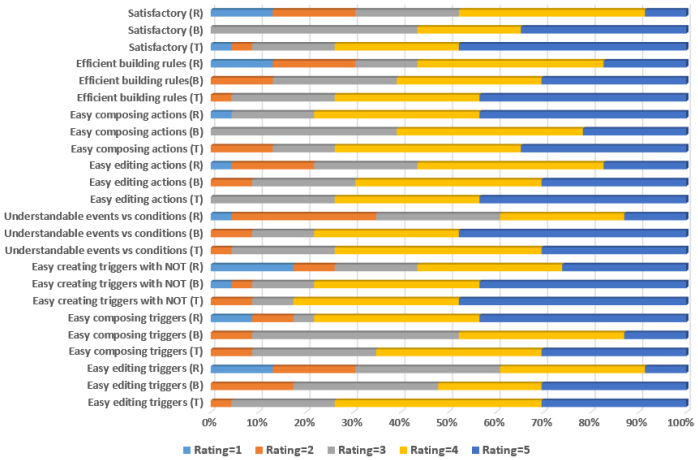

Fig. 2. Stacked bar chart summarizing the results on these questions.

Question 25 Which of the 3 composition styles did you find most suitable for editing rules in this context, and why? (wizard=13, BlockComposer=9, RuleBot=1)

TAREME was found the most intuitive, efficient and easy to use especially when considering unskilled users. Its ease of use was due to its clear structure, and the simplicity with which it is possible to edit the various parts of a rule. Also the lateral bar was appreciated in that it provides a convenient and helpful summary of the current point reached in the process of creating a rule. BlockComposer was appreciated for the intuitiveness of the metaphor used, which allows users to mentally 'visualize' the rule that is being currently created, one user reported. Another especially liked the pleasant effect of concretely moving and connecting the various blocks. One liked the fact that BlockComposer guides the user in the creation of rules, and it is also easy to fix errors. Just one user selected the RuleBot as the most suitable tool for creating rules, saying that this system was the one that allowed him to create the rules in the quickest manner.

Question 26. Which of the 3 composition styles did you find least suitable for editing rules in this context, and why? (TAREME=2, BlockComposer=6, RuleBot=15;)

The answers associated with this question highlighted that users did not much like the composition style exploited in the chatbot, which was judged as the least suitable one among the three tools. The chatbot was found as the least suitable one for various reasons: somewhat imprecise, requiring more interactions, thus not very efficient; with some features requiring improvements (e.g. the possibility to edit a rule); sometimes it does not understand the input provided, or it just partially understand it or such input needs to be provided according to specific syntactical 'guidelines' that were received as rather 'rigid'. Finally, the support aimed to handling possible user errors needs improvements, the chatbot sometimes just limits its assistance to just repeating the same question. A pair of users judged TAREME as the most unsuitable one, as it requires the highest number of steps to define a rule. BlockComposer was found to be a bit too complicated in its interactions, and overall not very intuitive.

4.4 Analysis of Rules Correctness

The rules generated by the participants were analysed, with the purpose of better grasping the strong and weak points of each composition paradigm. This investigation was based on the discrepancies between the behaviours that participants expected from the rules as expressed in their natural language description, and the actual behaviours implemented with the rules created in the tailoring environments. Indeed, before starting the rule composition using the editors, participants were asked to provide a natural language description of the behaviour that they wanted to obtain. Such natural language descriptions were subsequently manually translated into a non-ambiguous structure. For example, from a rule explained in natural language as "When the user's footsteps count is less than 1999 from 12:00 and 18:00, send a notification and start an activating light scene in the living room", we obtained the non-ambiguous form "not Event (steps) with time constrain (start time - end time), Action (notification), Action (living room activating light)". This was done to facilitate the later comparison with the rules obtained by the students with the tools.

Four categories of errors were identified and counted and involved: definitions of triggers and actions; association of a trigger to event vs condition; applications of the negation operator; use of the trigger composition operators (and/or). For each error, a weight was assigned following a severity scheme previously defined (see Table 1), where 1 point indicates a severe error and 0.5 a moderate one. Classifying an error as "severe" or "moderate" was made depending on its consequences. For example, using two events composed by a logical AND operator is considered a severe error because the rule will never be triggered. Using two conditions in a rule instead of an event and a condition was counted as moderate, because the rule will partially fulfil the intended behaviour. All the errors that involved an incorrect application of the logical operators between triggers were considered severe.

Error type	Trigger / action	Event / condition	Negation	Trigger operators
Severe	Missing or completely different rule element	The wrong specification makes the rule completely different/ nonfunctioning	Missing operator or applied to a wrong trigger	Wrong operator
Moderate	Different but related rule element	The behaviour of the rule has partial overlap with the intended one	Incorrect time specification	/

Table 1. Criteria adopted for computing the error scores.

We analysed the 345 rules produced by the participants (5 tasks with each of the 3 editors). The obtained average error score considering all the tasks performed per participant is 0.72 (std. dev. 0.81) for the Wizard-based editor, 1.26 (std. dev. 1.38) for the Block-based and 1.02 (std. dev 0.85) for the Conversational-based. The Wizard-based editor obtained a total error score of 16.5, the Block-based of 29, and the Conversational-based of 23.5. Looking at the score per task graph (Figure 3, right side) we see a higher score for the Block-based editor, especially on the first 2 tasks.

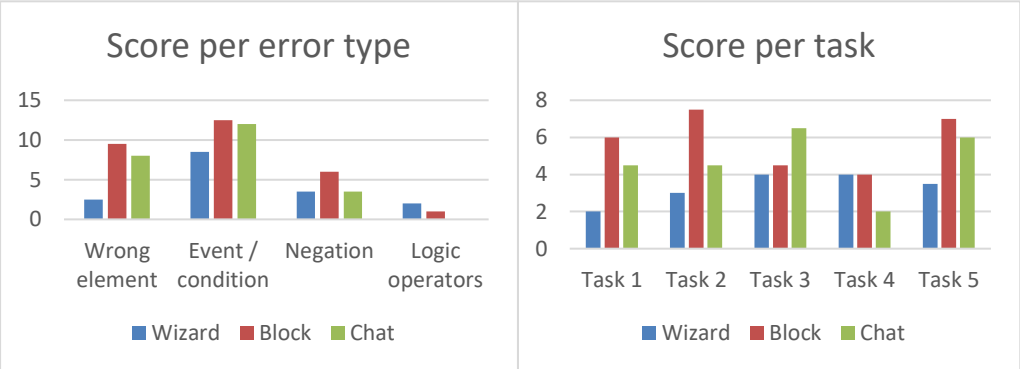

Fig. 3. Comparison of the error scores by error type and by task

From a comparison of the error scores with respect to the different paradigms (Figure 3, left side), we can obtain some preliminary insights. Logical operators between triggers were well applied by participants, and errors were found only in 3 occurrences out

of 207 rules which included these operators. Errors in the choice between event and condition were the most common errors in all the paradigms. Errors of this kind can be found in 51 out of 345 total rules, with a compound error score (the sum of the scores associated with the considered errors) across all editors of 33. The choice of the correct triggers and actions appears to be more problematic with the conversational and the block paradigms. This type of error is present in 25 out of 345 rules, with a compound error score of 20. Many of these errors consist in the choice of a related rule element, e.g., bathroom lights instead of bedroom lights. Lastly, some issues in the application of the negation operator are present in all the composition paradigms, but especially in the block-based editor. This class of error affected 21 out of 69 rules that required this behaviour, with a compound error score of 13. However, most of the errors of this type (16 out of 21) are of the moderate type, hence related to the timing aspects of the negation (i.e. not inserting the required timing constraint, or indicating a rule that will be activated when the selected *trigger condition* becomes different from the specified one in a time interval instead of when the *trigger event* does not occur in that time interval, or vice versa).

The wizard composition style was more effective in particular from the point of view of the selection of the correct elements. This could be because in this style the composition is precisely guided, therefore when users select an element it is used only for the current rule. This is not always true for BlockComposer where it is possible to move several blocks in the work area, use some of them for one rule, and then reuse them for the next rule. This, on the one hand, can be convenient for those familiar with the tool, but it may have led to some confusion and use of wrong blocks. As for the conversational approach, the difficulties in selection can be consequent to the user not always having clear the state of the rule under construction, and to the difficulties of interpreting the natural language. One consideration regarding the block environment is the mismatch between the perception of understanding the concepts of events and conditions (very good, see answers question 11), and the errors of this type later made (more than in other editors). It could be an indication that the method used to present this difference (modal window with explanation and examples) is not very effective in conveying this concept (although in this case it would be a problem in the explanation presentation, and not in the style of composition).

5 Discussion

The study has been useful to indicate some pros and cons of the various approaches. We considered users who were not initially familiar with the set of triggers and actions proposed. One issue that emerged was that with users unaware of the available triggers and actions provided by the tailoring tools, having a visual representation that directly offers the possibility of navigating across the available options is more immediately useful than a conversational interface, which can also be rendered vocally, and therefore does not immediately present all the available options also to avoid becoming tedious. On the other hand, when it is clear what should be done, then the conversational interface can be more efficient than the other composition paradigms, especially with simple

rules (one trigger, one action), whose description can be more easily recognised by the chatbot. When users have to build more complex rules, then the visual approaches tend to be more effective in guiding them, as they provide more interaction control just using their visual features, thereby limiting the possibility of error. The wizard seeks to have the user enter the missing info, and the block-based editor shows in the puzzle elements the parts that still need to be connected with a missing piece. BlockComposer was a bit affected (in terms of efficiency) by the time needed to drag the various elements in the central, main panel. Overall, the participants appreciated that the relevant concepts are clearly indicated in all the approaches. Some issues were still detected concerning exact understanding of some of them. For example, some users did not immediately understand that a trigger specified using a NOT operator needs to be accompanied by the definition of a time interval, which is essential to indicate when the lacking event should be checked. Likewise, explicitly using different terms for indicating events and conditions was useful to make users aware that they are different concepts, but in some cases such distinction was not yet completely understood.

To summarise we can say that TAREME was found intuitive, fast and easy to use, and it leads to making few mistakes, also because the user interface contains various support widgets (natural language feedback, rule sidebar). However, more steps are required to compose the rules than in other editors, thus it could be tedious for a frequent user. BlockComposer supports an intuitive metaphor, which leads to a clear view of the rule, pleasant interaction, a feeling of "hooking" the pieces, but the drag-and-drop interaction was not always easy, especially with small blocks, and it led to more errors. RuleBot is fast in rule creation, is able to clearly distinguish between events and conditions but it is necessary for the user to "align" with the way of understanding of the chatbot, and often it is necessary to repeat misunderstood phrases. Moreover, it needs extensive training to interpret the various possible ways of expressing the rules, and lacks a continuous feedback of the status of the rule during its composition.

6 Conclusions and Future Work

The study presented based on the user feedback and an analysis of the rules actually created through three different tools provides some useful indications for when and how a composition paradigm can be more effective. For example, the use of a chatbot as a vocal assistant is certainly efficient and immediate for short personalization rules. When rules become more complex, and require user to enter precisely various aspects, a visual tool can be more effective because it can more easily provide contextual information about the most relevant options.

As future work, we are considering the development of a new version of the chatbot that is also able to understand complete rules entered in one single interaction, supported by a relevant training, in order to see whether this can improve the user experience. We also plan to carry out further empirical studies, also with direct observation of users while interacting with the tailoring tools, to identify and provide further indications regarding the design and use of different compositional paradigms.

References

1. Beschi S, Fogli D, Tampalini F, CAPIRCI: a multi-modal system for collaborative robot programming, International Symposium on End User Development, Springer Verlag, LNCS 11553, pp. 51–66, 2019.
2. Brackenbury W., Deora A., Ritchey J., Vallee J., He W., Wang G., Littman M., and Ur B. 2019. How Users Interpret Bugs in Trigger-Action Programming. In Proceedings of the 2019 CHI Conference. ACM, New York, NY, USA, Article Paper 552, 12 pages
3. Cabitza F., Fogli D., Lanzilotti R., and Piccinno A.. 2017. Rule-based tools for the configuration of ambient intelligence systems: a comparative user study. Multimedia Tools and Applications 76, no. 4 (2017): 5221-5241.
4. Caivano D, Fogli D, Lanzilotti R, Piccinno A, Cassano F, Supporting end users to control their smart home: design implications from a literature review and an empirical investigation, Journal of Systems and Software 144, 295-313
5. Corcella L., Manca M., Nordvik JE., Paternò F, Sanders AM, Santoro C, Enabling personalisation of remote elderly assistance, Multimedia Tools and Applications, Volume 78, Issue 15, pp 21557–21583, Springer US
6. Corno F., De Russis L., and Monge Roffarello A., 2020. HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation. In Proceedings Advanced Visual Interfaces AVI '20, Salerno, Italy. ACM, New York, NY, USA, 9 pages
7. Danado J, Paternò F, Puzzle: a visual-based environment for end user development in touch-based mobile phones, International Conference on Human-Centred Software Engineering, HCSE 2012, LNCS 7623, pp. 199–216, 2012.
8. Desolda G., Ardito C., and Matera M. 2017. Empowering end users to customize their smart environments: Model, composition paradigms and domain-specific tools. ACM Trans. Comput.-Hum. Interact. 24, 2, Article 12 (April 2017), 52 pages
9. Huang THK, Azaria A, Bigham JP, Instructablecrowd: Creating if-then rules via conversations with the crowd, Proceedings of the 2016 CHI Conference Extended Abstracts, (CHI EA '16). ACM, New York, NY, USA, 1555–1562.
10. Huang J. 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). ACM, New York, pp.215–225.
11. Lago A., Dias J., Ferreira H., Managing Non-Trivial Internet-of-Things Systems with Conversational Assistants: A Prototype and a Feasibility Experiment, Journal of Computational Science, 2021
12. Lucci G., Paternò F.: Understanding End-User Development of Context-Dependent Applications in Smartphones. Proceedings HCSE 2014, Paderborn, Germany, Springer Verlag LNCS 8742, pp.182-198
13. Mattioli A., Paternò F., A Visual Environment for End-User Creation of IoT Customization Rules with Recommendation Support. AVI '20: Proceedings of the International Conference on Advanced Visual Interfaces, September 2020 Article No.: 44 Pages 1–5
14. Paternò F, Santoro C, End-User Development for Personalizing Applications, Things, and Robots, International Journal of Human-Computer Studies, Volume 131, 2019, Pp. 120-13
15. Valtolina S., Barricelli B.R. & Di Gaetano S. (2020), Communicability of traditional interfaces VS chatbots in healthcare and smart home domains, Behaviour & Information Technology, 39:1, 108-132
16. Weintrop, D., Afzal A., Salac J., Francis P., Li B., Shepherd D. C., and Franklin D.. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. Proceedings of the 2018 CHI Conference. ACM, Paper 366, 12 pages.