A Visual Tool for Analysing IoT Trigger/Action Programming

Luca Corcella, Marco Manca, Fabio Paternò(✉), and Carmen Santoro

CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 56124 Pisa, Italy
{l.corcella, m.manca, f.paterno, c.santoro}@isti.cnr.it

Abstract. The Trigger-Action programming paradigm has been widely adopted in the last few years, especially in the Internet of Things (IoT) domain because it allows end users without programming experience to describe how their applications should react to the many events that can occur in such very dynamic contexts. Several end user development tools exist, in both the research and industrial fields, which aim to support the increasing need to specify such rules. Thus, it becomes important for application developers and domain experts to enrich such environments with functionalities able to monitor how users actually interact with such rule editors, and show useful information to analyse the end user activity. In this paper, we present a visual tool for monitoring and analysing how users interact with a trigger-action rule editor. The goal is to provide a tool useful to better understand what end users' personalization needs are, how they are expressed, how users actually specify rules, and whether users encounter any issues in interacting with the personalization features offered by the editors. The proposed solution supports the analysis through a dashboard and a set of timelines describing the actual use of the personalization tool, with the possibility to select specific events of interest. It moreover provides data useful for understanding the types of triggers, actions and rules actually composed by users, and whether they effectively exploit the personalization features offered.

Keywords: Trigger action programming · Visual analytics
Log user interaction · Internet of Things applications

1 Introduction

A consequence of the rapid spread of the Internet of Things (IoT) is that the environments where we live and act are increasingly characterized by the presence of a multitude of interactive devices and smart objects interconnected with each other. Since we interact with our applications in such very dynamic and unpredictable environments, it is not possible to foresee at design time how an application should react to all the possible contextual changes that can occur during its use. Only end users can know the most appropriate ways their applications should react to dynamic contextual events. For such reasons, in order to obtain applications able to adapt to the context of use in an effective way it becomes important to allow end users themselves to 'program' the behaviour of their applications.

Published by Springer Nature Switzerland AG 2019
C. Bogdan et al. (Eds.): HCSE 2018, LNCS 11262, pp. 189–206, 2019.
https://doi.org/10.1007/978-3-030-05909-5_11

In this trend, trigger-action programming has emerged as a useful and intuitive approach. Users can personalise the application behaviour through sets of rules indicating triggers and consequent effects. Triggers can be instantaneous events (corresponding to context changes) and/or conditions that, if satisfied, activate the execution of specific actions. This type of approach has stimulated several contributions both from the research [e.g. 2–5, 9] and industrial viewpoints (IFTTT, Tasker, Zapier, Resonance AI, …). This approach can be adopted in many domains that share the need for supporting tailoring of applications that would benefit from considering the occurrence of dynamic events (smart retail, remote elderly assistance, smart home, industry 4.0, …). Although seemingly intuitive overall, because it mainly asks users to indicate the relevant events and desired actions, sometimes identifying the relevant concepts and understanding how to specify them using the tool is not always clear to people without a programming background. For example, Huang and Cakmak [8] found that users may encounter difficulties interpreting the differences between events and conditions or between action types, and such misunderstandings can cause undesired behaviours.

In this area, IFTTT[1] (If This Then That) has been particularly successful. It has more than 320,000 automation scripts (called "applets") offered by more than 400 service providers. The applets have been installed more than 20 million times, and more than half of IFTTT services are IoT device-related [10]. One large repository of IFTTT rules is even publicly available [18]. Thus, we can foresee in the near future an increasing interest in environments allowing people to provide many rules to personalize their context-dependent applications.

In this perspective, the availability of tools able to analyse how users actually try to personalise their context-dependent applications with such approaches can become very useful, not only for developers of trigger-action authoring environments, but also for IoT application developers and domain experts. We have thus considered previous work in the area of analytic tools for Web site usability, which often log user interactions in order to support identification of potential usability problems. However, the application of such tools to analyse the use of trigger-action programming would not provide the most relevant information. Indeed, differently from existing tools that exploit log analysis for usability evaluation purposes, in this case the goal is not strictly to understand whether there is some bad user interface design, but rather to see how the personalization needs are expressed by users, and whether they have some conceptual problems in expressing them in terms of trigger-action rules.

In this paper, after discussion of related work, we introduce the trigger-action programming environment considered in this study. Then, we discuss the design requirements for the novel analytic system for IoT programming platforms, describe the functionalities supported by the current version, and report on a first user test. Lastly, we discuss the user test on this tool, draw some conclusions, and provide suggestions for future work.

[1] https://ifttt.com.

2 Related Work

Our work draws from research on trigger-action programming for IoT applications and tools for visualizing logs of Web interactions.

2.1 Trigger-Action Programming

Both in research and industrial fields there has been interest in the trigger-action programming to allow users to define their own adaptation rules. From the commercial point of view IFTTT is one of the most used application. It provides mechanisms to create rules composed of one trigger and one action. Triggers are events occurring in some connected applications, and cause the execution of associated actions in other applications. The possible applications are grouped according to their intended goal, i.e. environment control & monitoring, calendars & scheduling, news & information. Ur et al. [17] reported on a 226-participant usability test of trigger-action programming, finding that inexperienced users can quickly learn multiple triggers or actions obtained by extending the IFTTT language. Resonance AI[2] is a tool for developers that aims to automate and personalize applications. It provides contextual awareness services to enhance products and services with real-time understanding and reactions based on the current user's environment. Such data become actionable triggers that developers can use to automate or suggest actions in order to personalize apps and devices behaviour.

From the research perspective, we started our study from the TARE [5] trigger-action rule editor that provides the possibility to create rules more flexibly than IFTTT since they can be created as compositions of multiple triggers and actions. In this area, Desolda et al. [3] developed EFESTO, a visual environment that allows users to express rules for controlling smart objects. The followed paradigm is based on the 5W model, which defines some specification constructs (Which, What, When, Where, Why) to build rules coupling multiple events and conditions exposed by smart objects, and for defining temporal and spatial constraints on rule activation and actions execution. Coutaz et al. presented AppsGate [2], an EUD (End-User Development) environment designed to empower people with tools to augment and control their home. AppsGate aims to support different activities such as monitoring the home state and programming its behaviour in a context-dependent manner. Another similar approach is ImAtHome [4], an iOS application built over Apple HomeKit allowing home inhabitants without programming skills to control home automation by means of creating scenes and rules for defining the complex behaviour of a smart home. The approach proposed by [11] aims to address some specific issues of users when writing trigger-action programming rules by a tool, called TriGen, aimed at preventing errors due to too few triggers in the rules by statically analysing a rule's actions to determine which triggers are necessary. Another approach to address some of the issues that users encounter when they specify ECA (Event, Condition, Action) rules is reported in [16]. Still in this area Metaxas and Markopoulos [9] propose a context-range editor supporting end users formulate logical expressions regarding the context, define the

[2] https://www.resonance-ai.com.

concept of affinity regrouping heuristics, and present the mechanisms to apply it throughout the contextual ranges of the involved services. The semantic information that the services disclose lets the editor recognize this affinity and allows it to group terms in logical expressions when they refer to the same aspect (e.g. user's activity).

The monitoring and visualization method proposed in this paper can be useful to analyse the user interactions with these tools as well, since they still support trigger-action rules for IoT applications, and thus their use is characterised by similar user-generated events such as trigger selection, action selection, rule saved.

2.2 Web Analytics Tools

One typical use of the information contained in logs of user interactions is for usability studies [7]. Palmer [12] presents different metrics for measuring usability, and lists different types of methods to evaluate a user interface. UsaProxy [1] exploits a proxy-based solution to access remote Web pages: the proxy adds some JavaScript code to specify the listeners which log the user interaction with the concerned page(s). The output produced by the proxy is a simple list representing the IP address of the connected device, the visited pages, and some events' description, without any particular visualization able to support their analyses. MUSE [13] also exploits a proxy server in order to insert in the target Web pages some code to log user interactions. The logged events are shown in a timeline representation in which it is possible to compare a timeline representing the 'optimal' interaction with the one expressing the 'real' user interactions in order to help designers to discover some usability issues in the user interaction. WELFIT [14] is a tool to identify usage patterns based on client-side event logs and by presenting event stream composition characteristics. The system records usage data during real use, identifies usage patterns, and indicates potential user interface design problems. Harms and Grabowsky [6] proposed to transform the recorded user interaction in task trees that are then checked to identify usability issues. The goal of such contributions is to identify a method to record user interactions and then further analyse the logs in order to highlight usability problems. HistoryViewer [15] is a system that aims to support exploration of log data obtained from user interactions. In this case the goal is to support final users for communication purposes, and not usability evaluators, by describing the interactions that took place in a way they can recall and communicate their own discoveries about the data, not focused on the interaction mechanisms or on difficulties they may have encountered.

Differently from such proposals, in this work we focus on providing designers of trigger-action rule editors and IoT application developers with interactive visualisations supporting exploration and filtering of the logged relevant interaction data, so as to derive higher-level information such as the types of rules that users were interested in creating with the tool, the most popular trigger and action types used, and the types of usage patterns followed by users while interacting with the tool. The visualizations offered also allow them to rapidly identify whether the personalisation rules composition was straightforward or the users had to go through various possibilities before completing their tasks. Moreover, by analysing user logs it is possible to understand what the personalization needs are, how they are expressed by the users, whether their

rules actually support the desired results, if the personalization features offered by the editors are sufficient.

3 The Proposed Approach

In this section we first introduce the trigger-action programming environment that has been considered in this study, and then we report on the initial set of requirements that have been identified for the visualizations to provide. Lastly, we describe how it has been instrumented in order to obtain relevant and meaningful log file for further analysis.

3.1 The Trigger-Action Rule Editor

In this study we considered the TARE platform (Trigger Action Rule Environment) [5], which allows users to define their trigger action rules in an intuitive way. The tool is flexible in the order in which users can specify the rules (they can start either from triggers or from actions), they can also re-use a previously defined rule in order to create a new one. Moreover, they can combine multiple triggers by using the Boolean operators AND and OR. The Not operator is also supported to check whether some event has not occurred in a specific period of time.

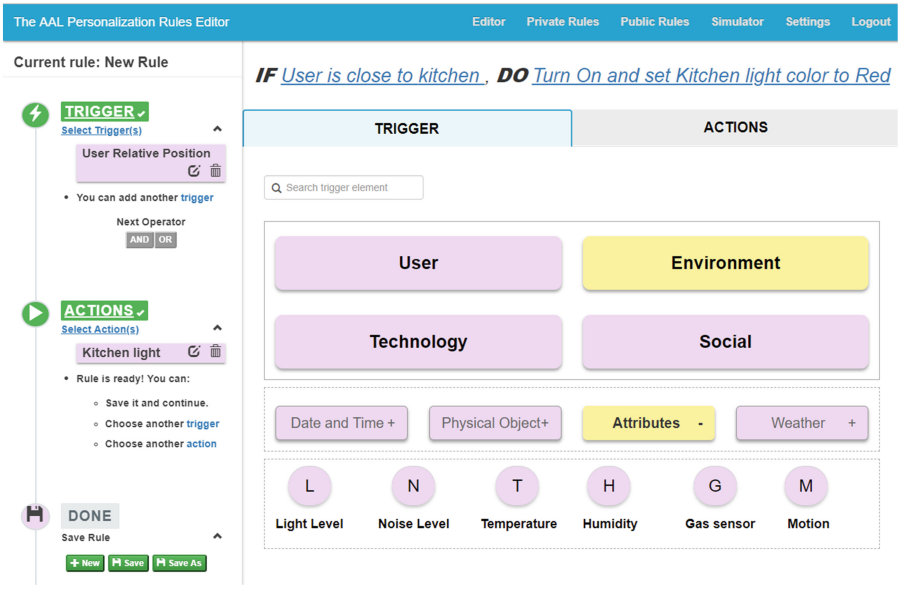

Fig. 1. The TARE editor user interface.

Figure 1 shows the context hierarchy of the Rule Editor: users navigate by first selecting the context dimension to which the considered trigger belongs, and then they go through the associated context categories (and sub-categories), which group together logically related context elements. The leaves of the context hierarchy refer to context entities, and are used to specify triggers and their parameters. For instance, in Fig. 1 the trigger is "IF the user is close to the kitchen", and it involves the context entity "user". Also the possible actions are indicated through a classification based on the type of effects they allow users to achieve (changes to the user interface, changes to the state of some appliances, sending reminders or alarms, …). In addition, TARE provides users with the possibility to search for a specific trigger, by specifying a concept to search for in the hierarchy of the possible triggers. On the left side there is also an interactive panel that indicates the main steps in the workflow associated with rule editing, and continuously provides feedback on what has been done, indications of what can be done from the current state, and thus it can be used to control and activate the various parts of the rule editing.

3.2 The Requirements for IoT Trigger-Action Programming Visual Analytics

TARE has been validated in three different trials, which involved 58 users in total. During the tests, users, who had no prior programming experience, had to perform various tasks concerning the specification of personalization rules with different complexity in terms of number of triggers and actions.

In particular, the rule editor has been used in two interdisciplinary projects in the area of Ambient Assisted Living. Such projects involve different types of organizations: application developers, technology providers, medical institutions, elderly organizations. Thus, the projects' meetings and discussions were useful to identify possible requirements for the relevant visual analytics tool. In addition, the papers published in the area of trigger-action programming were considered as well, in particular for the part concerning how they have been empirically tested. Thus, by observing the results in the trials and considering previous work in the area, we identified some features required in an environment to support the analysis of how users define context-dependent personalization rules.

One important general requirement for a visual analyser is the availability of interactive data exploration: the tool should provide users with different zooming levels, as well as the possibility to select individual items and get specific details on demand. In our case, it should also provide different interactive features to enable users to focus on different aspects of trigger-action programming. For the type of tool considered in this study, relevant information includes: the most recurring/frequent context entities used in rules; the most recurring combinations of trigger types and action types used in created rules; the most recurring sequences of usage patterns logged, the most used operator to connect multiple triggers (AND/OR), etc. Indeed, this type of information can be useful to better understand also the aspects that users prefer to consider in the specification of their personalization rules.

Other useful information that the tool should provide, for each user and also across users, is:

- The rule part users prefer to start editing the rules from (triggers and then actions or vice versa), in order to provide more flexibility in the rule editing;
- The sequence of trigger/action dimensions and entities that users have passed through to reach the trigger/action leaf of interest, in order to assess whether the proposed logical organization of the contextual aspects is intuitive for end users;
- The time spent to create a rule (max, min, average);
- The time spent to modify a rule (max, min, average);
- The number of rules created in each session and in all sessions;
- The number of triggers/actions created in each session and in all sessions;
- The number of rules/triggers/actions that users started to edit without saving them;
- The number of simple/complex rules created by the users (simple rules are rules involving only one trigger, while complex rules are rules which involve a combination of multiple triggers).

We also judged it useful to provide designers with the possibility to filter the results to allow users to configure the list of events they want to focus on, as well as to provide further quantitative information, such as the context dimension and the trigger entity that have been most used in the defined triggers. Such summarized representations of the users' sessions are particularly useful when the number of events becomes very high and difficult to manage.

3.3 The Logs

To support the identified requirements, we had to identify the events that were meaningful to log. We decided to exclude some low-level events to log (such as mouse over, mouse out, blur) that were judged not particularly relevant for the type of planned analysis. We have focused only on the interactions with the trigger and the action hierarchy, and on the editor parts which manage the rules. The logging implementation was done by a JavaScript file which appends handlers to the relevant events supported in TARE and related to rule creation, editing, saving. In particular, we found it useful to log user's selections of:

- "New Rule", "Save Rule", "Save Rule as", "Edit Rule" and "Delete Rule" buttons (used to manage rules);
- "Triggers" and "Actions" buttons (used to go to the part of the tool dedicated respectively to trigger and action specification);
- "AND" and "OR" buttons (used to compose two triggers);
- Trigger/Context Dimension elements, to select one specific trigger dimension (User, Environment, Technology, Social);
- Action Dimension elements (Update/Distribute UI, Change Appliance State, Activate Functionalities, Alarm, Reminder), to select one specific action dimension;
- Trigger type, to select a specific type of trigger within the hierarchy of triggers;
- Action type, to select a specific type of actions within the hierarchy of actions;

- Trigger Operator (to select the operator involved in a trigger specification, e.g. equal, different, more, less);
- Action Operator (to select a specific type of action, e.g. turn on-off, open, close);
- Event/Condition (to specify whether the trigger specification refers to an event or to a condition);
- Entering specific Trigger parameters values (to specify the values associated involved in the specification of a trigger);
- Save/Update/Cancel Trigger or Save/Update/Cancel Action commands;
- Search Trigger Element (to search for a specific trigger in the hierarchy).

4 The Tool Visualizations

In this section we show and discuss the two types of visual information provided.

4.1 The Dashboard

When the tool is accessed, it shows a dashboard (see Fig. 2). On the left hand side, it is possible to select a specific user who interacted with the rule editor, then the dashboard shows an overview of the activities carried out by the considered user: the total rules, triggers and actions created in all interaction sessions, the number of rules which had been modified; the rules that have been specified (described in natural language), the context dimensions involved in the rule editing (see the pie chart on the right showing the percentage of triggers for each contextual dimension), the most used triggers and actions grouped by dimensions and the time of each working session. By "session" we mean the interval of time between a user's login and a user's logout from the system (or the system automatically does a logout after two hours of idle time).

In addition, the tool shows information about the number of rules that have been saved and not (see the bar charts visualised in the bottom part and clustered by session). The unsaved rules are those that the user started to edit and never saved. Such bars are interactive and the user can select each bar to get the details of the concerned rules (e.g. the names of the rules saved). For each trigger/action dimension there is a section which shows the name of the context entities and the number of times they have been used in all defined rules. The dashboard also provides indication of how long each session lasted overall and also how long it took to create and save a specific rule.

4.2 The Timelines

In order to visualize the sequence of relevant events logged during users' activities in a simple manner, we decided to use a dynamic timeline visualization (see Fig. 4), which provides a time-dependent overview of the relevant events that occurred. In particular, when the Timelines tab is selected, it is possible to see for the currently selected user a set of timelines, each one presenting the list of events recorded in an interactive session. In the timelines, each event is identified by a label describing it and including the name of the associated event and the corresponding value: such a value is shown only when this is applicable, e.g. for the leaves of the hierarchy. The timeline is thus a mono-dimensional

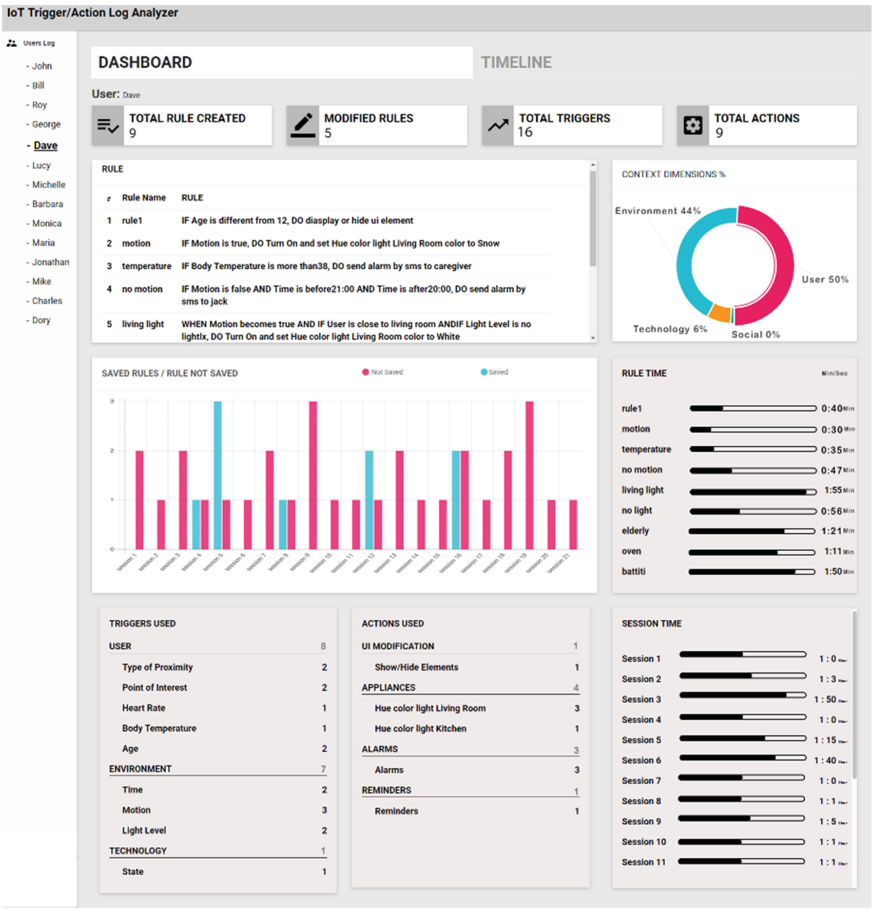

Fig. 2. The dashboard presenting summary information.

visualisation, in which the X axis represents time, according to which the events are ordered, thereby making it easy to see when they occurred. Each timeline corresponds to a user's session. It is worth noting that the events are automatically rearranged in a vertical manner by the library used[3] according to the current level of zoom and the number of labels to visualise, in order to avoid overlapping between labels.

However, due to the large number of events that can be recorded, the timeline could be difficult to interpret (see for example Fig. 4). For this reason, in addition to the possibility to zoom the timeline in and out, the tool provides users with a functionality that allows them to select only the events of interest, so that only specific types of events will be shown in the final visualization (e.g. the events that involve selecting a specific type of context dimension). In addition, in order to allow users to better perceive the differences between the various types of events, they can select a specific

[3] http://visjs.org/.

color to assign to each of them (see Fig. 3). Furthermore, it is worth noting that the list of triggers from which this selection is carried out only shows the actual triggers that occur in the timeline and not all the possible triggers that are potentially available in the system. Another option that was provided in the tool as filtering support is the possibility to save the current configuration of events that are of interest for the user, in order to be ready for later use, also allowing users to name such configuration in a meaningful manner. This is done to enable users to retrieve and load this configuration more quickly and effectively later on. Indeed, sometimes an effective filtering process could involve multiple iterations (e.g. the user progressively refines the current set of events of interest so as to better focus on the information s/he currently judges as important). In addition, different visualisations (obtained through different sets of filters) might have different goals. Therefore, this feature allows users to save time and have the intended visualisations ready for use, instead of doing this process over from scratch whenever they access the tool.

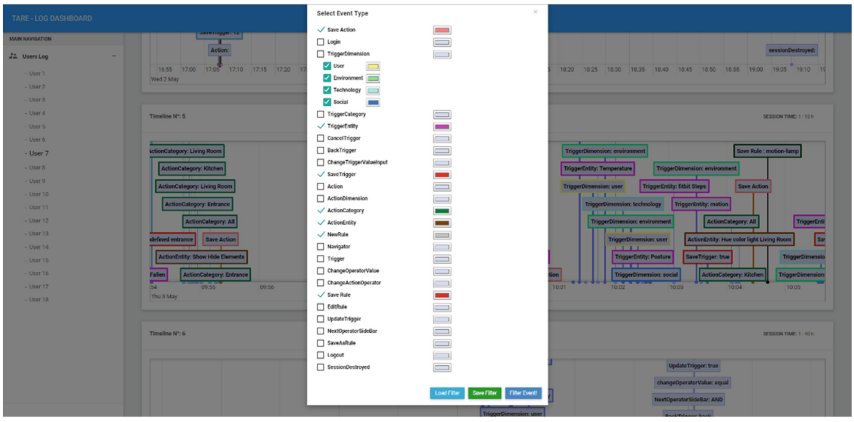

Fig. 3. The Log Filter: only the events of interest have been selected by the user.

Figure 4 focuses on a portion of a timeline obtained when the user created two simple rules: a first rule called “motion-lamp”, whose specification is “when there is motion, do turn on the light in the living room”, a second rule called “spd-temperature-alarm” whose specification is the following: “when body temperature is more than 30 °C, do send an alarm by sms to caregiver”. As you can see from Fig. 4, the associated timeline becomes very crowded without any filtering: this makes interpretation of the succession of events occurring in the timeline problematic. Nonetheless, in spite of this, the timeline shown in Fig. 4 is still able to provide some information that could be relevant for the analyzer. For instance, the timeline highlights the time interval in which the sequence of events associated with a specific rule composition occurred, by displaying an additional red line under the X axis of the timeline. This red line represents a useful cue because it can be used as a reference when performing zooming and filtering operations, and thereby allow users to better focus on how the creation of that rule was actually carried out. Other information that the timeline shown in Fig. 4

provides –although at a coarse grain– is the parts of the timeline in which the interactions were more (or less) close to each other, which could be relevant information for the analyzer. In the example shown in Fig. 4, closer interactions occur in the first part of rule building, whereas more distanced interactions occur in the vicinity of critical actions such as saving actions or saving rules (the user takes more time to think about the rule before actually saving it).

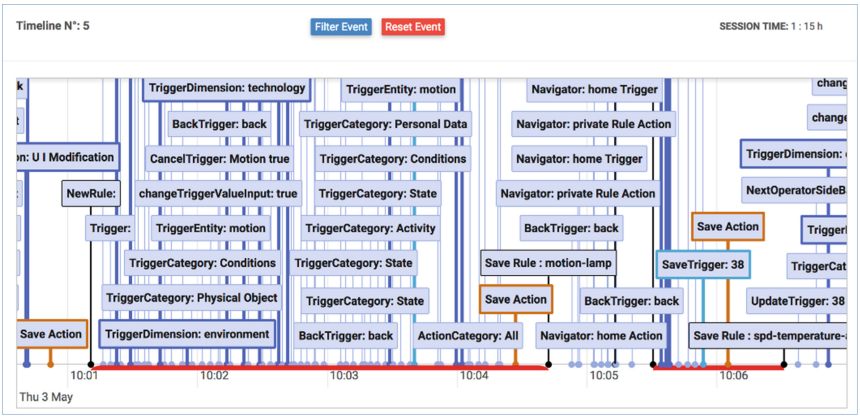

Fig. 4. The timeline associated with the considered example, as initially visualized.

However, in some cases the user may want to further investigate the succession of events occurred in a specific time interval, and observe them at a finer level of detail. Figure 5 shows a visualization obtained from the timeline shown in Fig. 4, by applying some filtering in order to hide events that were judged uninteresting for the user. In particular, the filtering done for obtaining the visualization shown in Fig. 5 only displays events associated with *TriggerDimension*, *TriggerEntity*, *SaveTrigger*, *ActionCategory*, *ActionEntity*, *SaveAction* and *SaveRule* events (which are also visualized in Fig. 3). This is to better understand the interactions occurring in the timeline visualized in Fig. 4 (in particular, for building the first saved rule named "motion lamp"). From Fig. 5 it is possible to see that after exploring several context dimensions without saving any trigger, the user focused on the "environment" dimension, and then

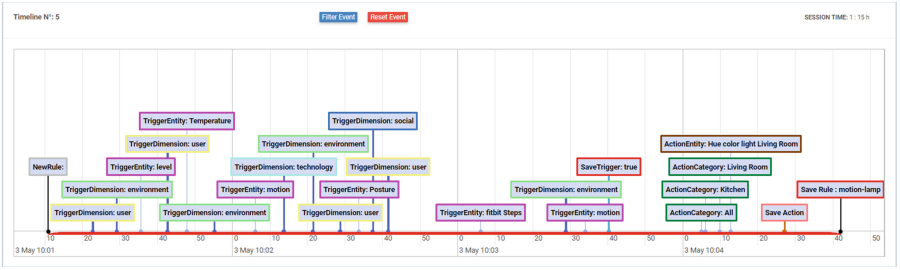

Fig. 5. Timeline obtained by applying filtering to the timeline shown in Fig. 4.

selected "motion". Then the user, after considering on various action categories ("All", "Kitchen"), focused on the action category "Living room". Within that category the user finally selected the action entity "Hue color light living room". This latter event was followed by a "Save action" event, and then a "SaveRule" event, meaning that the rule was actually saved.

Another interesting type of analysis that can be done concerns the use of the search functionality included in the rule editor. For example, frequent use of the search functionality would suggest that the user found it quicker to search than to explore the top-down hierarchy of triggers. Furthermore, it could be useful to compare what users specify in the "search" field and what they actually used afterwards in the rule specification; or analyse whether, after searching for an element, users actually found that element of interest and proceeded with the rule specification or they needed to repeat this process multiple times (and which terms they used in such repetitions). In addition, when a specific trigger element is repeatedly searched for by the same user instead of exploring the hierarchy, this could be a sign that the position of that concept within the hierarchy is not very logical according to the user's model and therefore not easy to locate.

5 A First User Test

In order to assess the usability of the tool a first user test was organized. In particular, we wanted to investigate whether the tool was able to provide usable representations of the interactions carried out by end users with the TARE rule editor, in order to derive useful information about its usability and usefulness.

5.1 Participants

A set of 10 participants fully gender-balanced (5 females) with age ranging between 24 and 49 (mean = 30.4, median = 26, std. dev. = 7.6) were involved in the study. They were volunteers recruited through mailing lists. As for the education of participants, 1 user held a High School degree, 5 a Master Degree, 1 a PhD and 3 a Bachelor. Half users (50%) had already used a visual analytics tool before the test. They mentioned RescueTime, Google Fit, Strava, Elevate, WUP, Wireshark, Matlab (visual toolbox), Apache log viewer, Kibana, Google Fusion Tables, Google Analytics, TensorBoard.

Half of users (50%) had some familiarity with developing applications exploiting sensors. One user developed a system providing a geo-referenced visualization of on fine dust particles of polluting nature, and a system providing a visualization for data associated with a remotely operated submarine vehicle. A user developed a domotic application, another one mentioned both a system for motion detection exploiting a video camera, and an application for measuring light brightness (using a photodiode sensor). A cross-device application using a Kinect sensor and an application used for remotely monitoring the activity of elderly people were the applications mentioned by other two users. This type of expertise seems relevant because the visual analytics tool is oriented to analysers interested in investigating how IoT services have been personalized through trigger action rules by end users.

5.2 Test Organisation

The test was done in a laboratory. During the test, a moderator observed the participants interacting with the tool, annotating whether and how they carried out tasks and also any further relevant issue and remark. The test was organised in four phases: introduction and motivations, familiarisation, test execution, questionnaire. In the test the users were first introduced to the motivations and main functionalities of this work and of the trigger-action rule editor through a PowerPoint presentation, which also contained a video showing an example of use and application of the TARE tool, which was done to better support people unfamiliar with that tool. Next, users had to create a trigger-action rule of their choice (using the TARE tool) and then, through the log visualizer they had the possibility to get familiar with the corresponding timeline visualization.

Then, they had to access the visualizations related to a given user in a particular session. This was selected beforehand for the test and was the same for all users. Users had to accomplish a list of tasks involving the dashboard and the timelines associated with that particular user in that specific session. In particular, the selected session provided data of a user who initially explored the hierarchy of triggers and action without saving any rules and then the user was able to save two rules: the first one involved a time interval longer than the second one, which was characterised by interactions carried out in shorter time range. We judged it more relevant to provide users with relevant log data created by others, which would better reflect more realistic situations of designers/developers analysing the data generated by other users.

Tasks. The tasks to accomplish covered both the information provided by the dashboard (mainly, information about the most used triggers and actions, the most used context dimensions, the number of rules that was successfully saved by users, the time associated with the various user sessions, etc.). In particular, for the dashboard they had to:

- Task1: Provide the name of the session with the highest number of rules created
- Task2: Provide the name of the most used context dimension
- Task3: Provide the name of the most used action dimension
- Task4: Provide the name of the longest session

As for the *timelines*, users had to analyze the time-dependent visualization of the occurred interactions, even using some filtering to better focus only on the most meaningful information. In particular, regarding the *timelines* users had to:

- Task5: Set a filter to visualise only *newRule* and *saveRule* and then describe what they could derive from the visualised information
- Task6: Add further filters (also using different colours) involving *TriggerDimension*, *CancelTrigger*, *SaveTrigger*. Then users had to indicate whether and where in the session the user had problems and why.

The test moderator provided users with a PowerPoint slide in which all the tasks were visualised. Users were instructed to solve the tasks by answering the associated questions, by verbally communicating them to the moderator who wrote them down for

future analysis. At the end they had to fill in a questionnaire aiming to assess various aspects. They had to provide first some personal information, whether they had previous experiences with visual analytics tools and with applications using sensors. Then, on a 1 (worst) to 7 (best) Likert scale they had to assess usability and utility of the dashboard, the timelines, and the filtering feature. They also had the possibility to provide comments and suggestions, and indicate three positive and three negative aspects in the tool proposed.

The tasks submitted to users were identified in such a way to cover some typical information that could be useful for analysing user interactions with the personalisation editor. In particular, the first four tasks (Task1–Task4) involved the identification of very specific information which was available in the visualisations shown to users: in order to carry out such tasks users had to provide the moderator with an answer to the questions associated with the tasks (e.g. provide the name of the longest session). The last two tasks were more open-ended, and implied analysing and reasoning on the provided data in order to derive more general conclusions. Since Task 6 was mainly a refining of Task5, it was needed to be presented after Task5. For this reason, we decided to follow the same order for all users in presenting all the tasks to them.

5.3 Results

Tasks. All the participants were able to successfully provide the correct answer to the first four tasks associated with the information provided by the dashboard. Nonetheless, they provided further suggestions for improving the dashboard. For example, more than one user suggested to exploit a more consistent use of colours in the dashboard, another user suggested including a scrollbar in the panel showing the sessions associated with a specific user in order to avoid bringing about a too long page in case of a user having many sessions associated. Regarding Task5, all users except two were able to identify that there were some rules that were not saved at the beginning of the considered sessions. This was probably due to the fact that the visualisation showed a red line in correspondence with rules that were actually saved and therefore the two users just focused on this highlighted information. Regarding Task6, as a consequence of the further refining asked in this task, all the users were able to identify that the concerned user at the beginning of the session did not save any rule, while at the end of the session the user was able to save two rules. The most interesting part of this task was the different motivations users gave to that behaviour. On the one hand some of them interpreted this behaviour as an actual issue (e.g. "*the user did not find what she was looking for and therefore was not able to save any rule*"). On the other hand, other users interpreted this behaviour as just the user's need of first exploring the hierarchy for better familiarisation (even saving just some triggers), without necessarily saving any rule. One user also focused on the saved rules (instead of the unsaved ones, as most users did), especially noticing the different time slots needed for saving the two rules and the closer interactions occurring for creating the second rule, deriving that the second rule involved a quicker and more straightforward interaction.

Questionnaire. In the post-task feedback, participants provided positive feedback regarding the tool, and found it easy to perform the tasks. Across all questions, the median ratings were at or above 5.5 on a 7-point Likert-scale (7 = best), as it is possible to see from the following rated aspects:

- Usability of the dashboard (median: 6)
- Usability of the timelines (median: 5.5)
- Usefulness of information provided in dashboards (median: 7)
- Usefulness of information provided in the timelines (median: 6)
- Usability of the feature for setting filters (median: 7)
- Usefulness of the approach for understanding how users exploited the personalisation tool (median: 7)

An overview is presented in the following stacked bar chart (Fig. 6).

Moreover, participants also answered a series of open-ended questions, whose answers have been detailed in the following.

Do you have suggestion to improve the dashboard usability?
One user suggested grouping rules not by session but per day, in order to have more meaningful information for the user. In addition, another user suggested adding the possibility to hide/show some parts of the dashboard in order to allow users to better focus only on the most relevant information. The same user also suggested adding further information for better identifying the sessions (e.g. day, hour), which are currently just subsequently named (e.g. session1, session2).

Do you have suggestion to improve the timeline usability?
One user suggested putting the most recent timeline in the top part of the user interface (whereas now the timelines are visualized following a chronological order, with the most recent ones in the bottom part of the window). Three users complained about the way the labels were vertically visualized. Since the lines going from the labels to the X axis of the timeline are positioned in the center of the label, a user suggested moving such lines in the left-most side of the label (in a sort of a flag-based shape) as a way to better visualize the order of the events while reading the labels. Another user asked for more space dedicated to each single timeline (currently more than one timeline is visualized in the same window). This is to have a better overview of the timeline, while at the same time avoiding scrolling up the visualization vertically, with the risk of losing the overall context of the occurred interactions.

Do you have any suggestion to improve the dashboard utility?
Only a few users provided suggestions for improving this aspect. For instance, one user suggested that the information associated with triggers used should be put under the pie chart showing the percentages of used triggers, since it represents a more detailed refinement of the pie-chart-based visualization.

Do you have suggestion to improve the timeline utility?
Several users did not have any particular suggestions for improving this aspect. A couple of users questioned about the utility of showing some information in the labels, for instance the value that was set for a particular trigger (e.g. “Save Trigger 26”).

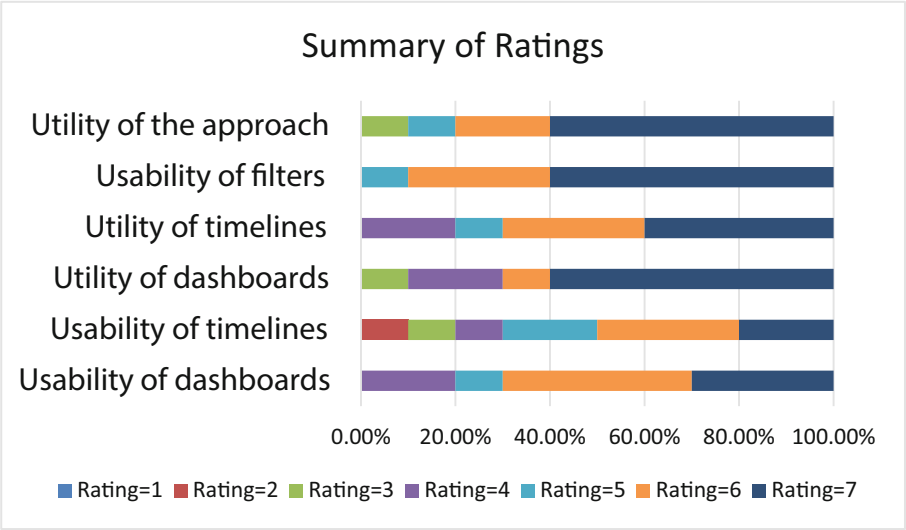

Fig. 6. Stacked bar chart showing ratings assigned to various aspects of the tool.

Do you have suggestion to improve the usability of the filtering features?
Most users appreciated this functionality. Just one user noted that it was not possible to move the window supporting the selection of filters, thus this window could cover the underneath timelines.

Is there any information that you did not find in the tool and you would have had?
One user suggested having further information about users (e.g. a picture) in order to better identify them. Another user said that it could be better adding the option to have the whole window dedicated to the visualization of a specific timeline of interest, so as to have more space for a more convenient visualization. Another user suggested adding in the dashboard a pie chart for visualizing the used actions, along the same line of the one dedicated to triggers used.

Please indicate three positive aspects you found while using the tool
Among the aspects that users mentioned as most appreciated there was the dashboard. It provides the users with an overall view of the interactions done by users, understand where the user found potential difficulties (e.g. in terms of longer time, higher number of interactions). Many users appreciated the possibility of filtering the timelines to better focus on a subset of events of interest.

Please indicate three negative aspects you found while using the tool
Some users complained about the clarity of some labels and how some data were presented. Another aspect regarded the limited space devoted to the visualization of the timeline of interest. Another aspect that was mentioned was the need of vertically scrolling the timelines to get all the relevant information. A user complained about a non-perfectly consistent use of colors (for instance the same color was used in different parts of the dashboard although no specific relationship was held between the associated information).

Do you have further suggestion to improve the tool?
Auser suggested adding further interactivity in the information visualized in the dashboard, so that it is possible for the user to refine and interactively explore the information at various levels of detail. Another user suggested improving the timeline visualization, even grouping together some data for the goal of obtaining lighter and clearer representations. Another user mentioned providing users with the possibility to order some information according to various criteria. For instance, each session could be ordered in terms of duration but also in terms of number of rules saved, etc.

Discussion. The results of the user test were overall encouraging and promising. Users appreciated the type of information provided by the tool and the supported features, and gave us useful and relevant feedback for further improving them.

In general, the users were able to successfully carry out the submitted tasks, showing that the tool is easy to learn since they used it for the first time and the learning phase was minimal. In addition, it was interesting to note how different users explained in slightly different manners the same information represented in the timelines, especially when carrying out Task6, which was the most open-ended task. Some users did not interpret the unsaved rules as a usability issue but just as a sign of users' need to get familiar with the tool and the hierarchies elements, or that some users preferred to restart from scratch instead of editing the elements appearing in the current rule state. Others, instead, interpret such data as a sign of users not finding what they were looking for in the hierarchies. In this regard, the filtering features seem very useful in effectively supporting further exploration and investigation of the information of interest.

6 Conclusions

In this paper we present a method and the features of a supporting tool for analysing the users' behaviour when interacting with a trigger-action rule editor for personalising their IoT context-dependent applications. We discuss the most relevant features for this analysis, provide example visualizations that can be supported, and report on an initial user test.

While in this work we applied the approach to a specific tool (TARE), the type of analysis of the users' behaviour presented can be easily extended and applied to other tools supporting trigger-action programming of IoT context-dependent applications. Indeed, all such tools share a number of key logical concepts on which the visualisations are centred (triggers, actions, rules, trigger operators). Regarding the timelines, by logging the relevant user actions it is possible to derive the usage patterns exploited by users while interacting with the tool.

Future work will be dedicated to extending the functionalities of the visual analytics tool taking into account the feedback from its use, and carrying out further, more longitudinal, empirical validations.

References

1. Atterer, R., Wnuk, M., Schmidt, A.: Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In: Proceedings of the 15th International Conference on World Wide Web, pp. 203–212 (2006)
2. Coutaz, J., Crowley, J.L.: A first-person experience with end-user development for smart homes. IEEE Pervasive Comput. **15**(2), 26–39 (2016)
3. Desolda, G., Ardito, C., Matera, M.: End-user development for the internet of things: EFESTO and the 5W composition paradigm. In: Daniel, F., Gaedke, M. (eds.) RMC 2016. CCIS, vol. 696, pp. 74–93. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53174-8_5
4. Fogli, D., Peroni, M., Stefini, C.: ImAtHome: making trigger-action programming easy and fun. J. Vis. Lang. Comput. **42**, 60–75 (2017)
5. Ghiani, G., Manca, M., Paternò, F., Santoro, C.: Personalization of context-dependent applications through trigger-action rules. ACM Trans. Comput. Hum. Interact. **24**(2), 3 (2017). https://doi.org/10.1145/3057861. Article 14
6. Harms, P., Grabowski, J.: Usage-based automatic detection of usability smells. In: Sauer, S., Bogdan, C., Forbrig, P., Bernhaupt, R., Winckler, M. (eds.) HCSE 2014. LNCS, vol. 8742, pp. 217–234. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44811-3_13
7. Hilbert, D.M., Redmiles, D.F.: Extracting usability information from user interface events. ACM Comput. Surv. **32**(4), 384–421 (2000)
8. Huang, J., Cakmak, M.: Supporting mental model accuracy in trigger-action programming. In Proceedings of UbiComp 2015, pp. 215–225. ACM, New York (2015). https://doi.org/10.1145/2750858.2805830
9. Metaxas, G., Markopoulos, P.: Natural contextual reasoning for end users. ACM Trans. Comput. Hum. Interact. **24**(2), 36 (2017). https://doi.org/10.1145/3057860. Article 13
10. Mi, X., Qian, F., Zhang, Y., Wang, X.: An empirical characterization of IFTTT: ecosystem, usage, and performance. In: IMC 2017, pp. 398–404 (2017)
11. Nandi, C., Ernst, M.D.: Automatic trigger generation for rule-based smart homes. In: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (PLAS 2016), pp. 97–102. ACM, New York. https://doi.org/10.1145/2993600.2993601
12. Palmer, J.W.: Web site usability, design, and performance metrics. Inf. Syst. Res. **13**(2), 151–167 (2002)
13. Paternò, F., Schiavone, A.G., Conte, A.: Customizable automatic detection of bad usability smells in mobile accessed web applications. In: Proceedings Mobile HCI 2017, Article No. 42, Vienna. ACM Press, September 2017
14. Santana, V.F., Calani Baranauskas, M.C.: WELFIT: a remote evaluation tool for identifying web usage patterns through client-side logging. Int. J. Hum Comput Stud. **76**(C), 40–49 (2015)
15. Segura, V.C.V.B., Barbosa, S.D.J.: HistoryViewer: Instrumenting a Visual Analytics Application to Support Revisiting a Session of Interactive Data Analysis. In: PACMHCI. EICS, vol. 1, pp. 11:1–11:18 (2017)
16. Terrier, L., Demeure, A., Caffiau, S.: CCBL: a language for better supporting context centered programming in the smart home. In: Proceedings of ACM Human-Computer Interaction, vol. 1, EICS 2017, Article 14, 18 p., June 2017. https://doi.org/10.1145/3099584
17. Ur, B., McManus, E., Ho, M.P.Y., Littman, M.L.: Practical trigger-action programming in the smart home. In: Proceedings of CHI 2014, pp. 803–812. ACM, New York https://doi.org/10.1145/2556288.2557420
18. Ur, B., et al.: 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 2016), pp. 3227–3231. ACM, New York (2016). https://doi.org/10.1145/2858036.2858556