POLITECNICO DI TORINO
Repository ISTITUZIONALE

TAPrec: Supporting the Composition of Trigger-Action Rules Through Dynamic Recommendations

Original
TAPrec: Supporting the Composition of Trigger-Action Rules Through Dynamic Recommendations / Corno, F., DE RUSSIS, L., MONGE ROFFARELLO, A.. - STAMPA. - (2020), pp. 579-588. (IUI '20: ACM International Conference on Intelligent User Interfaces Cagliari (Italy) 17-20 March, 2020) [10.1145/3377325.3377499].

Availability:
This version is available at: 11583/2779432 since: 2020-03-30T09:07:07Z

Publisher:
ACM

Published
DOI:10.1145/3377325.3377499

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)

05 November 2026

TAPrec: Supporting the Composition of Trigger-Action Rules Through Dynamic Recommendations

Fulvio Corno
fulvio.corno@polito.it
Politecnico di Torino
Torino, Italy

Luigi De Russis
luigi.derussis@polito.it
Politecnico di Torino
Torino, Italy

Alberto Monge Roffarello
alberto.monge@polito.it
Politecnico di Torino
Torino, Italy

ABSTRACT

Nowadays, users can personalize the joint behavior of their connected entities, i.e., smart devices and online service, by means of trigger-action rules. As the number of supported technologies grows, however, so does the design space, i.e., the combinations between different triggers and actions: without proper support, users often experience difficulties in discovering rules and their related functionality. In this paper, we introduce *TAPrec*, an End-User Development platform that supports the composition of trigger-action rules with dynamic recommendations. By exploiting a hybrid and semantic recommendation algorithm, TAPrec suggests, at composition time, either a) new rules to be used or b) actions for autocompleting a rule. Recommendations, in particular, are computed to follow the user's high-level intention, i.e., by focusing on the rules' final purpose rather than on low-level details like manufacturers and brands. We compared *TAPrec* with a widely used trigger-action programming platform in a study on 14 end users. Results show evidence that *TAPrec* is appreciated and can effectively simplify the personalization of connected entities: recommendations promoted creativity by helping users personalize new functionality that are not easily noticeable in existing platforms.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Human-centered computing** → **Human computer interaction (HCI)**; Ubiquitous and mobile devices; User studies; • **Computing methodologies** → *Knowledge representation and reasoning*.

KEYWORDS

Trigger-Action Programming, End-User Development, Recommender System, Semantic Web, Internet of Things

ACM Reference Format:
Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2020. TAPrec: Supporting the Composition of Trigger-Action Rules Through Dynamic Recommendations. In *Proceedings of 25th International Conference on Intelligent User Interfaces (IUI '20).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IUI '20, March 17–20, 2020, Cagliari, Italy
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

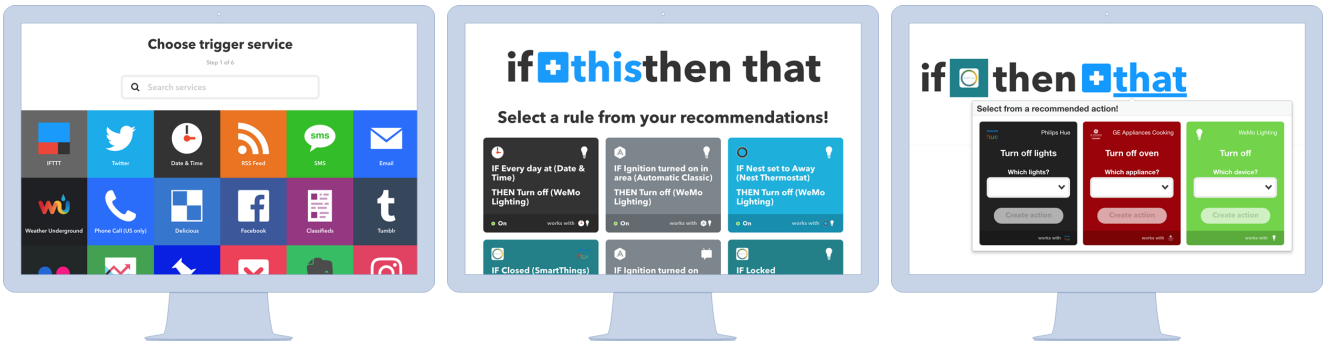
1 INTRODUCTION

People are nowadays surrounded by a multitude of smart devices, always connected to the Internet. With lamps, thermostats, and many other appliances, including fridges and ovens, that can be remotely controlled, homes are becoming "smart." Also other environments, ranging from workplaces to entire cities, are extensively leveraging on the Internet of Things (IoT) [6]. Besides physical devices, many different online services, ranging from social networks to news and messaging apps, are greatly used by almost everyone. The result is a complex network of *connected entities*, be they smart devices or online services, that can communicate with each other, with humans, and with the environment. In this scenario, end users can take advantage of visual programming platforms such as IFTTT[1] and Zapier[2] to personalize the joint behaviors of their own connected entities, without the need of writing any code. Most of these End-User Development (EUD) platforms are based on *trigger-action programming* [13], i.e., they allow the definition of IF-THEN rules such as *"if I publish a photo on Facebook, then upload it to my Google Drive"*, or *"if the security camera detects a movement, then blink the kitchen lamp."* With such an approach, users can express most of their desired behaviors [34]. By exploiting wizard-based procedures, in particular, users can define a rule by directly *composing* it, i.e., by linking a trigger and an action together, or they can *reuse* a rule composed and shared by another user.

Unfortunately, contemporary EUD platforms for trigger-action programming mainly model smart devices and online services on the basis of the underlying brand or manufacturer [7]: as the number of supported technologies grows, so do the design space, i.e., the combinations between different triggers and actions. To *compose* rules, for instance, IFTTT and Zapier are currently forcing users to browse large menus with more than 1,000 supported connected entities, each one with its own triggers and actions, displayed in a meaningless order [8]. Even finding a rule to be *reused* may be difficult: the number of rules publicly available on IFTTT exceeded 200,000 back on September, 2016 [35]. Therefore, without proper support, end users often experience difficulties in discovering rules and related functionality [35], while trigger-action programming becomes a complex task for them [21].

In this paper, we introduce *TAPrec*, an EUD platform that supports the composition of trigger-action rules with dynamic recommendations that are continuously adapted in real-time to the current *high-level intention* of the user. An intention is defined as a goal-oriented activity that a user would like to be automatically executed in her environment of choice under given conditions, e.g.,

[1] https://ifttt.com/, last visited on September 26, 2019
[2] https://zapier.com/, last visited on September 26, 2019



(a) Rule Composition (b) Rule Recommendations (c) Action Recommendations

Figure 1: TAPrec is an EUD platform that supports the composition of trigger-action rules with dynamic recommendations. Users can compose IF-THEN rules in an IFTTT-like web application (a). By analyzing the rules saved by the users, TAPrec is able to recommend new rules to be used (b) and actions for auto-completing a the rule that is being composed (c).

personalizing the temperature and the lighting of some rooms depending on the time of the day. Since users often have abstract personalization goals that can be satisfied with multiple trigger-action rules [7, 9, 35], in particular, we aim at assisting users in the *current* personalization session: instead of taking into account the entire user's history, the high-level intention of the user is implicitly extracted by analyzing the rules that the user is composing, only, without considering her older rules. Consequently, users can change their intentions among sessions as the tool recomputes recommendations based on the rules defined in the current session, only. Figure 1 shows a sample usage scenario of TAPrec:

a) The user starts a personalization session by composing an IF-THEN rule in a web-based application modeled after IFTTT. When the first rule has been saved, TAPrec starts to analyze the rules the user is defining in the given session to compute recommendations that that follow the implicit user's high-level intention, i.e., by taking into account the final functionality of the rules rather than details like manufacturers of brands.
b) As long as the user start composing a new rule, in particular, TAPrec dynamically visualizes recommendations, if available. If interested, the user can select a recommendation to be directly used.
c) Furthermore, if the user compose a rule by selecting a trigger that is included in the current recommendation set, TAPrec recommend the related action to auto-complete the rule.

To compute recommendations, TAPrec exploits RecRules [10], a hybrid recommendation algorithm that addresses semantic-based information and collaborative user preferences in a graph-based setting to train learning-to-rank techniques. By leveraging a high-level ontological model of trigger-action programming [9], RecRules is able to abstract low-level details such as brands and manufacturers and uncover hidden connections between rules in terms of shared functionalities: a rule for turning on a lamp, for example, is *functionally similar* to a rule for opening the blinds, because they share a common final goal, i.e., to light up a place.

To the best of our knowledge, TAPrec is the first example of an EUD platform for trigger-action programming that dynamically recommends rules at composition time to assist users in satisfying their actual personalization needs. TAPrec is modeled after IFTTT, and it exploits the same metaphors and the same expressiveness of the existing platform. This choice was made deliberately, by considering the popularity of the platform [13], its ease of use and accuracy in the rule composition process [5], and the availability of real usage data [35], which we used to define available triggers and actions.

To understand whether and how TAPrec can simplify the end-user personalization of connected entities, we ran a controlled experiment with 14 end users with different education levels and backgrounds. In the study, we challenged participants in personalizing different scenarios with both TAPrec (i.e., with recommendations) and a IFTTT-like tool (the baseline, i.e., with no recommendations). Results shows that TAPrec is appreciated and can effectively simplify the personalization of connected entities. While composing a rule remains a fundamental mechanism for users that know *exactly* the personalization they want, participants stated that recommendations are useful to reduce the composition time and effort, and to discover new functionality that are otherwise "hidden" in the large and confused sets of supported entities, triggers, and actions. Compared to the IFTTT-like tool, in particular, TAPrec made users spend more time in the personalization tasks, and spurred participants to define more trigger-action rules that covered a larger number of smart devices and online services.

Summarizing, the main contributions of our work are the following:

- We show how recommendation systems could be adopted to help users define trigger-action rules for their connected entities.
- We introduce TAPrec, a visual recommender system for trigger-action programming that dynamically provides recommendations at composition time.
- In a study with 14 participants, we show that TAPrec simplifies the definition of trigger-action rules and promotes

creativity by helping users discover new functionality that are not easily noticeable in existing platforms.

2 RELATED WORKS

TAPrec lies at the intersection of research in two related areas: *(i)* trigger-action programming in the IoT, and *(ii)* recommender systems for the creation of software artifacts.

2.1 Trigger-Action Programming in the IoT

Lieberman et al. [25] defined End-User Development as *"a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact."* With the continuous growth of the IoT and the spread of new online services, EUD is becoming a fundamental paradigm to enable end users personalize their connected entities. Starting from iCAP [14], a visual rule-based system for PC to create context-aware applications, researchers extensively explored approaches and methodologies for "programming" connected entities in different contexts, e.g., mobile environments [31], smart homes [3, 12, 34], and web mashups [11, 33].

Meanwhile, IFTTT, Zapier, and several other commercial platforms for personalizing the *joint* behavior of smart devices and online services were born. Typically, such EUD platforms adopt a trigger-action programming approach: by defining IF-THEN rules, users can connect a pair of connected entities in such a way that, when an event (the *trigger*) is detected on one of them, an *action* is automatically executed on the second. Multiple studies, including empirical characterization of usage perfomances [30] and large-scale analysis of publicly shared rules [35], investigated different aspects of contemporary platforms like IFTTT. Despite their popularity [13] and apparent simplicity, using such trigger-action programming platform is often a complex task for users without any programming skills [21]. Indeed, while IF-THEN rules can express most of the behaviors desired by potential users [1, 34], users frequently misinterpret the behavior of trigger-action programs [4], often deviating from their actual semantics, and errors in trigger-action rules are very common [20].

Such a complexity is strictly related to the *technology-dependent* representation models adopted by contemporary EUD platforms, which require to manage separately every service and physical device [7]. Numerous recent works tried to overcome such an approach by exploring new visual paradigms for composing rules [13], or by adopting more abstract models that allow the definition of context-independent rules [9, 17]. In our work, we follow a different approach. As suggested by Ur et al. [35], the spread of new connected entities highlights the need to provide users with more support for *discovering* functionality. Rather than acting on the underlying paradigms and models, we therefore propose to adopt recommendation techniques to support users in defining trigger-action rules: recommender systems could be useful to help end users without programming skills to use EUD systems, and advances in EUD have expanded the opportunities for offering recommendations [18]. Since users often have abstract personalization goals that can be satisfied with multiple trigger-action rules [7, 9, 35], in particular, we aim at assisting users in the current personalization session: differently from traditional recommender systems, that typically take into account the entire user's history, TAPrec continuously recompute recommendations by considering the rules that the user is defining in a specific personalization session, only.

2.2 Recommender Systems for the Creation of Software Artifacts

Using recommendation approaches to support the creation of software artifacts, e.g., with feature recommendations [19, 23] or source code suggestions [29], is a long-standing topic in software engineering. Ye and Fisher [37], for instance, proposed *CodeBroker*, a system that promotes software reusing by visualizing different type of information into the current software development environment. Malheiros et al. [26], instead, developed a recommender system to solve change requests in source code. Other recent works focused on suggesting APIs [15, 16, 32]: Nguyen et al. [32], for example, presented *APIREC*, a novel API recommendation approach based on statistical learning, while D'Souza et al. [15] developed *PyReco*, an intelligent code completion system that recommends Python API.

While all the described recommendation systems are designed to support *professional* developers, recommendation opportunities have not yet been consistently explored to support *end-user* development, especially in the domain of trigger-action programming for personalizing connected entities. Contemporary platforms such as IFTTT and Zapier continue to offer limited types of suggestions, e.g., by promoting the most popular rules, and only few recent works started to address the problem of recommending new ways of personalizing the joint behavior of smart devices and online services [10, 36]. Yao et al. [36], in particular, developed a probabilistic framework to suggest relevant smart "things" to be personalized based on user preferences and interests. Corno et al. [10], instead, proposed *RecRules*[3], a semantic recommendation system that suggests trigger-action rules on the basis of content-based and collaborative information. None of such recommendation algorithms, however, has been tested with real users, and our understanding of whether and how recommendation techniques would assist end users in personalizing their connected entities remains limited.

In this work, we try to close this gap by integrating the RecRules algorithm in an EUD platform, and by reporting on the results of a user study with 14 end users.

3 RECOMMENDING FOR TRIGGER-ACTION PROGRAMMING

To compute recommendations, TAPrec exploits the RecRules algorithm [10], a hybrid and semantic recommender for suggesting IF-THEN rules to end users. Through a mixed content and collaborative approach, the goal of RecRules is to *recommend by functionality*, thus suggesting rules on the basis of the final intention of the user, e.g., personalizing the temperature and the lighting of some rooms. The algorithm, in particular, enriches trigger-action rules with semantic information, and it implements a semantic reasoning process to abstract low-level details such as involved technologies, brands or manufactures. As a result, RecRules can compute recommendations for yet unknown or rarely connected

[3] An implementation of the algorithm is available at https://git.elite.polito.it/public-projects/recrules, last visited on September 30, 2019

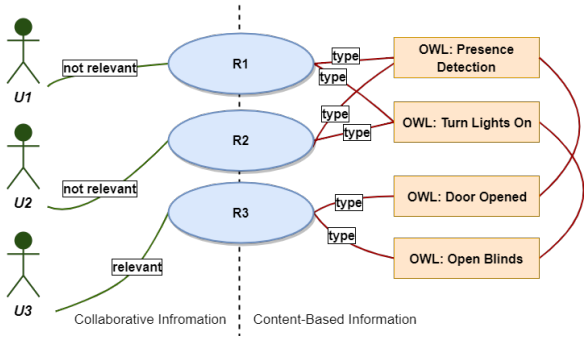


Figure 2: The knowledge graph built by *RecRules*. IF-THEN Rules are connected to users (collaborative information) and semantic classes (content-based information). From the graph, RecRules extracts path-based features able to characterize the interaction between users and rules, and it exploits learning to rank techniques to compute top-N recommendations.

entities, thus helping users to discover new functionality, starting from their actual needs.

Let us consider a user (**U1**) that has already defined a rule for turning on her *Philips Hue* lamp in the kitchen, for example:

R1 *"if the kitchen Nest Cam recognizes me, then turn on the kitchen Philips Hue."*

Besides other rules involving the same entities, i.e., *Nest Cam* and *Philips Hue*, RecRules is able to suggest to the user rules that have been defined by other users and that have identical functionality, even if they involve different technologies, e.g.:

R2 *"if the living room Homeboy Cam detects a movement, then turn on the bedroom LIFX lamp."*

Moreover, the algorithm can suggest rules that are *functionally similar* at high-level terms, i.e., "light up the room when I'm entering":

R3 *"if I open the SmartThings bedroom door, then open the Hunter Douglas blinds."*

RecRules is characterized by two main phases, i.e., Knowledge Graph Construction and Model Training. In the remainder of this section, we summarize these phases, by highlighting the choices we made in integrating the algorithm in TAPrec.

3.1 Knowledge Graph Construction

In the first phase of the algorithm, RecRules builds a knowledge graph that models content-based and collaborative information in a unique setting. Figure 2 shows a partial view of a knowledge graph that includes the user **U1**, her defined rule **R1**, and her potential recommendations, i.e., **R2** and **R3**. Trigger-action rules are first enriched with content-based information: they are linked through type-relationships to the EUPont model [9], an ontological semantic representation of trigger-action programming. EUPont classifies triggers and actions under OWL[4] classes, e.g., *"Presence Detection"* and *"Turn Lights On"*, that abstract the involved technologies, brands, manufacturers, and user context. By leveraging on this, rules that involve a trigger and an action classified under the same EUPont classes can be considered as functionally similar, as for **R1** and **R2** in the example above. Since TAPrec is based on the same metaphors and expressiveness of IFTTT, we exploit the instantiation of EUPont for IFTTT[5], that offers a hierarchical functionality representation of more than 500 triggers and actions supported by the popular platform.

After linking IF-THEN rules to semantic information, RecRules enrich the knowledge graph with collaborative information: rules are linked to users by means of relationships able to discriminate between *relevant* items, i.e., rules that are appreciated by the users, and *not relevant* items, i.e., rules that are not appreciated and therefore should not be recommended. In our integration, we exploited the dataset of IFTTT rules published by Ur et al. [35]. The dataset contains more than 250,000 publicly shared on IFTTT as of September, 2016, with the indication of how many times each rule has been reused by other users. We used this information to extract relevant/not relevant relationships. In particular, we followed the procedure described in the original RecRules paper [10], i.e., by normalizing the number of reuses through a graded-implicit feedback [24].

3.2 Model Training

From the knowledge graph, RecRules extracts path-based features able to characterize the interaction between users and rules. Features are defined as acyclic paths in the graph between users and trigger-action rules. Paths can include both collaborative relationships and content-based relationships. In this way, the feature vector summarizes both the importance of a rule, i.e., how many times it has been reused, and its similarity with other rules in terms of final functionality. To finally compute top-N recommendations, a ranking model from training data is built using a learning to rank technique. In our integration, in particular, we use RecRules to calculate the top-10 recommendations by exploiting the Random Forest [2] algorithm, as suggested in in the original RecRules paper [10].

4 THE TAPREC SYSTEM

TAPrec is an EUD platform that integrates dynamic IF-THEN rule recommendations. Composing trigger-action rules is a convenient mechanism to empower users to combine the behavior of their apps and devices according to their situational needs [13]. TAPrec *supports* such a mechanism with recommendations that are updated in real-time according to the user interaction with the platform. Suggestions, in particular, are continuously adapted to the current *high-level intention* of the users: the goal is to assist them in discovering new functionality able to satisfy their actual needs, thus reducing the complexity introduced by the spread of new supported technologies. High-level intentions are modeled through the top-level OWL classes of the EUPont ontology.

We implemented it as a web application consisting of two main components, i.e., the TAP Server and the TAP GUI.

[4]https://www.w3.org/OWL/, last visited on September 30, 2019

[5]http://elite.polito.it/ontologies/eupont-ifttt.owl, last visited on September 30, 2019

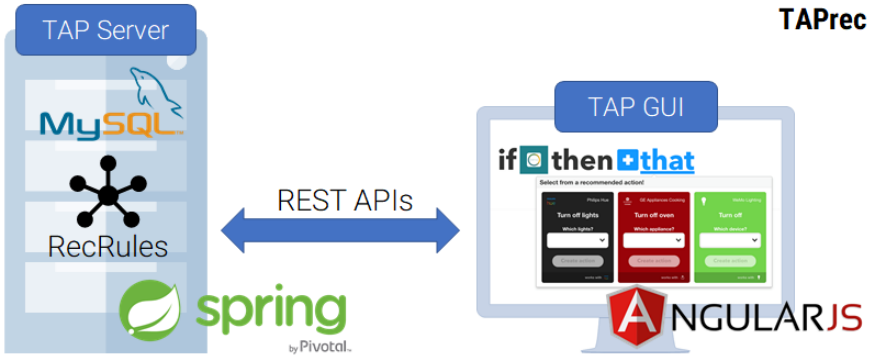


Figure 3: The TAPrec architecture. The user interface (TAP GUI) allows users to define trigger-action rules. It exploits, in particular, a set of RESTful APIs to communicate with the TAP Server, which is in charge of managing rules and computing recommendations through the RecRules algorithm.

4.1 TAP Server

The TAP Server has been implemented in Spring[6], an open source framework to develop web applications on top of the Java Enterprise Edition platform. It exposes a set of REST APIs that can be used by a user interface, e.g., the TAP GUI, to manage rules and get recommendations. By interacting with a MySQL database, in particular, the server offers the features needed to manage collections of trigger-action rules, i.e., to create, read, update, and delete rules. Furthermore, it integrates the RecRules algorithm to dynamically compute top-10 rule recommendations. To this end, whenever a rule is created, updated, or deleted, the server instantiates the following *recommendation process*:

(1) the underlying knowledge graph, modeling content-based and collaborative information, is updated in real-time;
(2) the updated graph is used to recompute the path-based features;
(3) the recomputed path-based features are used to train the Random Forest algorithm and recalculate recommendations.

The recommendation process is carried out in a separate thread: when new recommendations are available, the TAP Server saves them and makes them available through a dedicated REST API, without blocking the interaction with the user interface. The time needed to complete the recommendation process depends on the size of the underlying knowledge graph, mainly, i.e., how many users, rules, and connected entities are considered. In our evaluation, described in next session, it lasted 32 seconds on average ($SD = 13$).

4.2 TAP GUI

TAP GUI is the web-based user interface of TAPrec that interacts with the TAP Server through the provided REST APIs. We developed it with Angular[7], a TypeScript-based open-source web application framework.

To compose a rule, users can exploit an IFTTT-like interface to separately compose the trigger (*this*) and the action (*then*). They have to complete, in particular, the following *composition process*:

[6]https://spring.io, last visited on September 30, 2019
[7]https://angular.io, last visited on September 18, 2018

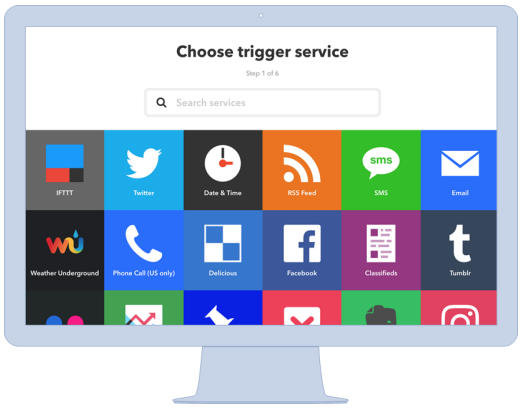


Figure 4: First step of the rule composition process: users have to select a supported connected entity on which instantiating the trigger.

(1) select a supported smart device or online service on which instantiating the trigger (Figure 4), e.g., Homeboy Cam;
(2) select the specific trigger to be used, e.g., "movement detected" for Homeboy security cams;
(3) fill in any additional information required by the trigger, e.g., which specific Homeboy cameras they want to use;
(4) repeat steps 1 to 3 for composing the action, e.g., to turn on a specific LIFX lamp in the bedroom.

When users start composing a rule, TAP GUI performs a REST request to the TAP Server to get the current top-10 recommendation set. If available, recommendations are dynamically displayed as shown in Figure 5: when selected, recommended rules can be directly completed and saved without the need of following the entire *composition process*.

Besides using a recommended rule, users can click on the "this" button to start a new composition process. If they select a trigger that is included in the current recommendation set, however, TAPrec recommends possible actions to auto-complete the rule, as shown in Figure 6.



Figure 5: Recommendations of new trigger-action rules to be used.



Figure 6: Recommendations of actions for auto-completing a rule.

5 USER STUDY

To understand to what extent TAPrec can simplify the end-user personalization of connected entities we performed a user-centered online evaluation by running a user study with 14 participants. Offline evaluations of recommender systems, in fact, may not give the same outcome as online evaluations [27]. Furthermore, even when testing online, higher accuracy does not always mean higher satisfaction [28]. To understand and improve the user experience of recommender systems, instead, it is necessary to conduct empirical evaluations that consider the entire process of how the user experience comes about [22]. This is even more important in the EUD domain for connected entities, where recommendation opportunities have not yet been consistently explored, and our understanding of whether and how recommendation techniques would assist end users in personalizing their connected entities is still limited.

5.1 Participants

The study involved 14 participants (7 females and 7 males) with a mean age of 25.43 years ($SD = 3.74$, $range : 19 - 32$). In particular, we balanced our sample population by recruiting a) 7 users with previous experience in computer science and programming (2 female and 5 males, *experts* group), and b) 7 users without any programming skills (5 females and 2 males, *non experts* group). To this end, we sent emails to students enrolled in different university courses and private messages to our social circles.

Expert participants included 3 PhD students and 4 master's degree students in Computer Engineering. On a Likert-scale from 1 (Very Low) to 5 (Very High), they stated their level of technological savviness ($M = 4.43$, $SD = 0.49$), programming skills ($M = 4.43$, $SD = 0.49$), and familiarity with the trigger-action programming approach ($M = 3.86$, $SD = 0.83$). One of them habitually used IFTTT, 3 of them used IFTTT a few times, sporadically, while the remaining 3 experts never used any EUD platform.

Non-expert participants, instead, included a medical student, 4 primary education science students, a middle school teacher, and a salesperson. Their level of technological savviness ($M = 3.14$, $SD = 0.99$), programming skills ($M = 1.71$, $SD = 0.69$), and familiarity with the trigger-action programming approach ($M = 1.14$, $SD = 0.34$) were substantially lower comparing to the experts group. Only a non-expert participant had used IFTTT a few times, while the remaining 6 had never used any EUD platform.

5.2 Procedure

We devised a within-subject study during which participants were requested to personalize 2 different scenarios, one with an IFTTT-like tool only (i.e., without recommendations), and one with TAPrec (i.e., with recommendations). Both tools shared the same user interface: the only difference was the absence or the presence of recommendations. We brought each participant to our lab for a 45-minute session using the 2 tools on a Macbook Pro connected to an external 22-inch monitor. At the beginning of the study, participants were introduced to the trigger-action programming for personalizing connected entities. Furthermore, before using a tool, i.e., the IFTTT-like interface or TAPrec, participants saw a demonstrative video that highlighted the main functionality of the given interface.

Each scenario described a list of 34 devices and online services that participants could personalize in a given context. While online services (like Facebook and Gmail) and personal devices (like smartphones) were in common, each scenario included other specific devices and systems:

Free-time. The first scenario included a set of smart devices and systems installed in the user's home, e.g., a washer, a refrigerator, an irrigation system, smart thermostats, and connected lamps. Participants were free to combine the joint behavior of their hypothetical online services, personal devices, and home devices and systems to personalize their free time.

Work-time. The second scenario included a set of smart devices and systems installed in the user's personal office, e.g., a coffemaker, an air purifier, a PC, a printer, smart heaters, and connected lamps. Participants were free to combine the joint behavior of their hypothetical online services, personal devices, and office devices and systems to personalize their work time.

To explore whether and how recommendations assisted users in defining personalizations, we deliberately designed a constraint-free procedure: participants could use all the connected entities described in a given scenario, and they were free to define as many trigger-action rules as they want. In particular, we explicitly told participants to stop using a given tool when they were satisfied with their defined trigger-action rules. Scenarios and evaluated tools were fully counterbalanced between participants. All the sessions were audio recorded.

5.3 Measures

To collect measures, we followed the framework proposed by Knijnenburg et al. [22], i.e., a user-centric approach to recommender system evaluation. According to the authors, it is important to distinguish the following *aspects*:

- Objective System Aspects (OSA), i.e., the recommendation algorithm;
- Subjective System Aspects (SSA), i.e., the users' perception of the objective system aspects;
- User Experience (EXP), i.e., users' evaluation of their interaction with the system; and,
- Interaction (INT), i.e., users' behaviors.

Since the exploited recommendation algorithm, i.e., RecRules, had already been evaluated offline, in terms of precision and recall [10] (OSA), we focused on SSA, EXP, and INT aspects. Table 1 describes the measures we collected during the study, with the indication of the related aspects, and the modality with which they have been collected. Participants answered Likert-scale questions immediately after using TAPrec, while the debriefing session was carried out at the end of the study. Logs, instead, were used to record the interaction between participants and both evaluated tools.

Some measures, e.g., the total number of defined rules and the time participants spend to personalize a given scenario, were available for both the evaluated tools: we used them to compare TAPrec with the IFTTT-like baseline tool. Other measures, e.g., PRQ and PRV, were instead specifically collected to evaluate the users' perception, experience, and interaction with TAPrec.

6 RESULTS AND DISCUSSION

Results are organized as follow. First, we investigate to what extent TAPrec can simplify the end-user personalization of connected entities with respect to contemporary EUD platforms. To this end, we compare our tool with the IFTTT-like tool, i.e., the baseline. Then, we analyze the measures we specifically collected for TAPrec to further understand whether and how recommendations help users define trigger-action rules.

6.1 TAPrec vs. the Baseline

To understand whether and how TAPrec can simplify the processes needed by end users to define trigger-action rules, we compared the usage of TAPrec and the IFTTT-like tool in the scenario personalizations. To this end, we analyzed the interaction between participants and the exploited tools, i.e., the first 4 measures reported in Table 1. Table 2 reports the average results of such a comparison.

On average, the usage of TAPrec resulted in a higher scenario duration (M = 695.55 s, SD = 324.85 s) with respect to the IFTTT-like tool (M = 532.63 s, SD = 173.01 s). A paired t-test confirmed that this difference was statistically significant ($p < 0.05$). Such a behavior was more evident for non-expert participants: on average, they used TAPrec for 820.81 seconds (SD = 245.12), and the IFTTT-like tool for 551.17 seconds (SD = 158.21), with a difference in the scenario duration of 269.64 seconds. For expert participants, instead, such a difference was of 82.88 seconds, only: they used TAPrec for 601.61 seconds (SD = 359.90), and the IFTTT-like tool for 518.73 seconds (SD = 192.86). However, a two-way mixed ANOVA with a post-hoc analysis with Bonferroni correction did not reveal a significant effect of the participant group on the scenario duration ($p > 0.05$): as shown in Figure 7a, both experts and non experts spent more time in personalizing a scenario when they used TAPrec.

By spending more time with TAPrec, participants personalized the given scenario with a significantly higher number of rules ($p < 0.05$): on average, participants defined 6.36 rules when they can see and exploit recommendations (SD = 2.84), while they defined 4.85 rules (SD = 1.29) with the IFTTT-like tool. Also in this case, we did not find any significant differences between the participants' groups: as shown in Figure 7b, this behavior was common across experts and non experts users. When using TAPrec, in particular, participants selected and directly used 2.53 rules on average from the recommendations (SD = 1.26). None of the participants, instead, selected a recommended action to auto-complete a rule. This suggests that recommendations are more useful at the *beginning* of the composition process. When users deliberately decide to compose a rule, indeed, they already know the personalization they want: by analyzing the audio recordings of the studies, we found that, before starting to compose a rule, the majority of the participants reasoned out loud about the specific triggers and actions to be used.

To analyze whether and how the usage of TAPrec influenced the features of the defined rules, we analyzed the number of unique connected entities used in each scenario personalization. By defining more trigger-action rules, in particular, participants also covered a larger number of smart devices and online services: on average, they used 10.43 different entities with TAPrec (SD = 3.48), while they personalized 8.5 different entities with the IFTTT-like tool (SD = 1.65). Despite not significant ($p = 0.091$), Figure 7 (c) shows that such a trend characterized both experts and non experts participants. This suggests that TAPrec helped all the participants discover new functionality that were otherwise "hidden" in the menus characterizing the composition process.

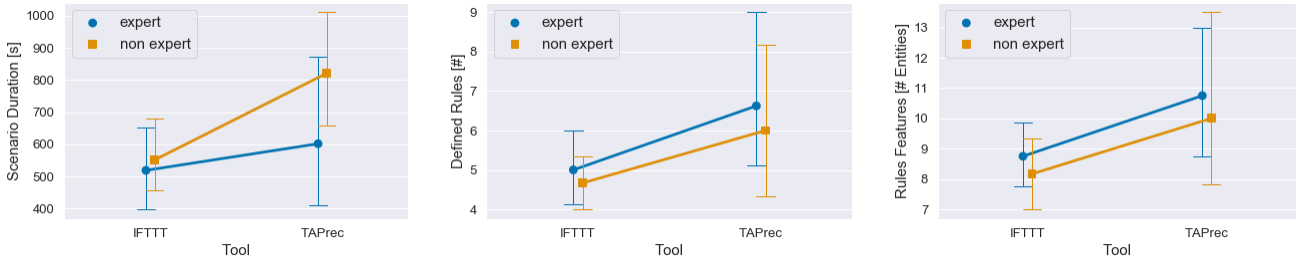
Key Takeaway: Compared with the baseline tool, TAPrec promoted creativity by making users spend more time in the personalization tasks. Recommendations, in particular, spurred participants to define more trigger-action rules that covered a larger number of smart devices and online services.

6.2 Recommendations Evaluation

To further understand whether and how recommendations helped participants in personalizing the scenarios, we investigated the participants' perception and experience with TAPrec, i.e., the last 4 measures reported in Table 1.

Table 1: The measures we collected during our user study. Through different logs, we recorded the interaction between participants and both the evaluated tools. Likert-scale questions and a final debriefing session were instead used to measure Subjective System Aspects (SSA) and User Experience (EXP) with TAPrec.

Measure	Description	Collection Type	IFTTT	TAPrec	Aspect
Rules Features	The entities, triggers, and actions involved in the defined rules	Logs	✓	✓	INT
Scenario Duration	The time participants spend to personalize a given scenario	Logs	✓	✓	INT
Defined Rules	The total number of defined rules	Logs	✓	✓	INT
Selected Rules	The number of rules selected from the recommendations	Logs	✗	✓	INT
PRQ	The Perceived Recommendation Quality of the proposed suggestions	Likert-scale questions	✗	✓	SSA
PRV	The Perceived Recommendation Variety of the proposed suggestions	Likert-scale questions	✗	✓	SSA
PEF	The Perceived Effectiveness and Fun in using the recommender system	Likert-scale questions	✗	✓	EXP
Usefulness	The perceived usefulness of the visualized recommendations	Debriefing	✗	✓	EXP



(a) Scenario Duration **(b) Defined Rules** **(c) Rules Features**

Figure 7: Independently of the participant groups, the usage of TAPrec increased (a) the time spent by the participants in personalizing a given scenario, (b) the total number of defined rules, and (c) the number of different connected entities involved in the define rules.

Table 2: Paired t-tests comparing TAPrec with the IFTTT-like tool in terms of scenario duration, defined rules, and rules features.

	IFTTT M (SD)	TAPrec M (SD)	p
Scenario Duration [s]	532.63 (173.01)	695.55 (324.85)	.049
Defined Rules	4.85 (1.29)	6.36 (2.84)	.034
Rules Features [# Entities]	8.5 (1.65)	10.43 (3.48)	.091

Table 3 reports the results of the analysis of the Likert-scale questions participants answered after personalizing a scenario with TAPrec.

On a Likert-scale from 1 (absolutely no) to 5 (absolutely yes), participants evaluated the perceived effectiveness and fun in using TAPrec (PEF measure) by stating that:

a) they would recommend the tool to other users ($M = 4.57$, $SD = 0.62$), and
b) having recommendations available was convenient ($M = 4.64$, $SD = 0.48$).

Participants, in particular, perceived the visualized recommendations as accurate (PRQ metric): on a Likert-scale from 1 (absolutely no) to 5 (absolutely yes), they liked the proposed suggestions ($M = 4.07$, $SD = 0.96$), and they found that the recommended rules and actions fitted their preferences in most cases ($M = 3.86$, $SD = 0.74$). Furthermore, they agreed that, in many cases, recommendations contained a lot of variety ($M = 3.86$, $SD = 0.74$, PRV metric).

Table 3: The analysis of the Likert-scale questions participants answered after personalizing a scenario with TAPrec. Results show that TAPrec was appreciated in terms of Perceived Effectiveness and Fun (PEF), Perceived Recommendation Quality (PRQ), and Perceived Recommendation Variety (PRV).

	M (SD)	Measure
I would recommend TAPrec to others	4.57 (0.62)	PEF
TAPrec is convenient	4.64 (0.48)	PEF
I liked the recommendations	4.07 (0.96)	PRQ
Recommendations fitted my preferences	3.86 (0.74)	PRQ
Recommendations contained a lot of variety	3.71 (1.03)	PRV

In the debriefing session, we measured the usefulness of TAPrec. 7 out of 14 (50%) participants pointed out that the major advantage of having recommendations is related to the time needed for defining trigger-action rules. P6, for example, said that *"TAPrec made you save time by proposing to you the most common rules"*, while

P2 asserted that, with recommendations, *"you don't have to think about anything else, just if you need the rule."*

6 participants (42.85%), instead, recognized that TAPrec reduces the effort needed to discover new functionality. P1 and P4, for example, said:

> *"I found recommendations useful to personalize the scenario: the tool showed to me rules I hadn't thought of, and also rules that I didn't know could be done. I never thought I'd use my car to detect that I got home!"* (P1)

> *"Yes, recommendations were useful because they made me think of new possible connections between triggers and actions."* (P4)

Besides helping users in discovering new functionality, the recommendations provided by TAPrec were also useful for "refining" the rules already defined by the users:

> *"In one case, I have already composed a rule to automatically turn on the kitchen lamp when I was near home. Then, I received a recommendation of a rule for the same behavior, but with a more precise trigger that detected that I was entering the kitchen. So I think recommendations are also useful to improve your rules."* (P1)

Participants, in particular, liked the novel characteristic of TAPrec, i.e., helping users in the current personalization session, with recommendations that are updated in real-time to follow the user's high-level intention. P5, for example, noted that *"recommendations fitted my personalization needs"*, while P13 acknowledged that *"the tool analyzed my preferences and then recommended proper trigger-action rules."* Furthermore, P9 explicitly said that *"recommendations focused on the specific situation I was trying to personalize."*

Finally, P13 confirmed that, despite recommendations have proven to be extremely useful, composing a rule remains a fundamental mechanism, especially when users know *exactly* the personalization they want:

> *"Yes, recommendations can help you save time, but they often don't reflect exactly your needs. I prefer to commit myself in composing rules."* (P13)

Key Takeaway: Participants appreciated the recommendations provided by TAPrec, and they liked the novel characteristics of the tool by stating that suggestions reflected their current personalization needs. They pointed out that recommendations were useful to reduce the time needed for defining trigger-action rules, and to reduce the effort needed to discover new functionality.

7 LIMITATIONS

The main limitation of our study is that it involved the definition of trigger-action rules in a lab setting; a more ecologically-valid study would be to deploy and test TAPrec in-the-wild, where users could define and execute trigger-action rules on their (real) smart devices and online services. As such, our results suggests that recommending trigger-action rules at composition time is a viable way of simplifying trigger-action programming.

8 CONCLUSIONS AND FUTURE WORKS

Given the spread of connected entities, be they novel smart devices or online services, trigger-action programming becomes a complex task for end users. In this paper, we presented TAPrec, an end-user development tool that supports the composition of trigger-action rules with dynamic recommendations of new rules to be used or actions to auto-complete a defined trigger. By exploiting a hybrid and semantic recommendation algorithm, the tool assist the user in the current personalization session, with suggestions that are updated in real-time to follow the high-level intention of the user, e.g., personalizing the temperature and the lighting of her rooms. Results of a user study with 14 participants highlight that TAPrec can effectively simplify the personalization of connected entities. Participants found recommendations useful to reduce the time for defining trigger-action rules and to discover new functionality. Furthermore, compared to an IFTTT-like tool (i.e., a tool with the same interface but without recommendations), TAPrec promoted creativity by making participants to a) spend more time in personalizing scenarios tasks, and b) define more trigger-action rules that covered a larger number of smart devices and online services.

Future works will include an in-the-wild study of the proposed system involving real devices and online services, with the aim of further understanding how much the computed recommendations effectively map users' intentions over a long period of usage.

REFERENCES

[1] B. R. Barricelli and S. Valtolina. 2015. *End-User Development: 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings.* Springer International Publishing, Cham, Germany, Chapter Designing for End-User Development in the Internet of Things, 9–24. https://doi.org/10.1007/978-3-319-18425-8_2

[2] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (Oct. 2001), 5–32. https://doi.org/10.1023/A:1010933404324

[3] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring End User Programming Needs in Home Automation. *ACM Transaction on Computer-Human Interaction* 24, 2, Article 11 (April 2017), 35 pages. https://doi.org/10.1145/3057858

[4] A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. 2011. Home Automation in the Wild: Challenges and Opportunities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).* ACM, New York, NY, USA, 2115–2124. https://doi.org/10.1145/1978942.1979249

[5] Danilo Caivano, Daniela Fogli, Rosa Lanzilotti, Antonio Piccinno, and Fabio Cassano. 2018. Supporting end users to control their smart home: design implications from a literature review and an empirical investigation. *Journal of Systems and Software* 144 (2018), 295–313. https://doi.org/10.1016/j.jss.2018.06.035

[6] Vint Cerf and Max Senges. 2016. Taking the Internet to the Next Physical Level. *IEEE Computer* 49, 2 (Feb 2016), 80–86. https://doi.org/10.1109/MC.2016.51

[7] F. Corno, L. De Russis, and A. Monge Roffarello. 2017. A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT. *Computer* 50, 11 (November 2017), 18–24. https://doi.org/10.1109/MC.2017.4041355

[8] F. Corno, L. De Russis, and A. Monge Roffarello. 2019. EUDoptimizer: Assisting End Users in Composing IF-THEN Rules Through Optimization. *IEEE Access* 7 (2019), 37950–37960. https://doi.org/10.1109/ACCESS.2019.2905619

[9] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. A high-level semantic approach to End-User Development in the Internet of Things. *International Journal of Human-Computer Studies* 125 (2019), 41 – 54. https://doi.org/10.1016/j.ijhcs.2018.12.008

[10] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. RecRules: Recommending IF-THEN Rules for End-User Development. *ACM Trans. Intell. Syst. Technol.* 10, 5, Article 58 (Sept. 2019), 27 pages. https://doi.org/10.1145/3344211

[11] Florian Daniel and Maristella Matera. 2014. *Mashups: Concepts, Models and Architectures.* Springer Publishing Company, Incorporated.

[12] Luigi De Russis and Fulvio Corno. 2015. HomeRules: A Tangible End-User Programming Interface for Smart Homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15).* ACM, New York, NY, USA, 2109–2114. https://doi.org/10.1145/2702613.2732795

[13] G. Desolda, C. Ardito, and M. Matera. 2017. Empowering End Users to Customize Their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools. *ACM Transaction on Computer-Human Interaction (TOCHI)* 24, 2, Article 12 (April 2017), 52 pages. https://doi.org/10.1145/3057859

[14] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive Prototyping of Context-aware Applications. In *Proceedings of the 4th International Conference on Pervasive Computing (PERVASIVE'06)*. Springer-Verlag, Berlin, Heidelberg, 254–271. https://doi.org/10.1007/11748625_16

[15] A. R. D'Souza, D. Yang, and C. V. Lopes. 2016. Collective Intelligence for Smarter API Recommendations in Python. In *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 51–60. https://doi.org/10.1109/SCAM.2016.22

[16] Ekwa Duala-Ekoko and Martin P. Robillard. 2011. *Using Structure-Based Recommendations to Facilitate Discoverability in APIs*. Springer Berlin Heidelberg, Berlin, Heidelberg, 79–104. https://doi.org/10.1007/978-3-642-22655-7_5

[17] G. Ghiani, M. Manca, F. Paternò, and C. Santoro. 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2, Article 14 (April 2017), 33 pages. https://doi.org/10.1145/3057861

[18] Will Haines, Melinda Gervasio, Aaron Spaulding, and Bart Peintner. 2010. Recommendations for End-User Development. In *Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI)*.

[19] Mostafa Hamza and Robert J. Walker. 2015. Recommending Features and Feature Relationships from Requirements Documents for Software Product Lines. In *Proceedings of the Fourth International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE '15)*. IEEE Press, Piscataway, NJ, USA, 25–31.

[20] Justin Huang and Maya Cakmak. 2015. Supporting Mental Model Accuracy in Trigger-action Programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. ACM, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

[21] Ting-Hao K. Huang, A. Azaria, and J. P. Bigham. 2016. InstructableCrowd: Creating IF-THEN Rules via Conversations with the Crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 1555–1562. https://doi.org/10.1145/2851581.2892502

[22] Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* 22, 4 (01 Oct 2012), 441–504. https://doi.org/10.1007/s11257-011-9118-4

[23] Jacob Krüger. 2018. When to Extract Features: Towards a Recommender System. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings (ICSE '18)*. ACM, New York, NY, USA, 518–520. https://doi.org/10.1145/3183440.3190328

[24] Lukas Lerche and Dietmar Jannach. 2014. Using Graded Implicit Feedback for Bayesian Personalized Ranking. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 353–356. https://doi.org/10.1145/2645710.2645759

[25] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. *End User Development*. Springer Netherlands, Dordrecht, Netherlands, Chapter End-User Development: An Emerging Paradigm, 1–8. https://doi.org/10.1007/1-4020-5386-X_1

[26] Y. Malheiros, A. Moraes, C. Trindade, and S. Meira. 2012. A Source Code Recommender System to Support Newcomers. In *2012 IEEE 36th Annual Computer Software and Applications Conference*. 19–24. https://doi.org/10.1109/COMPSAC.2012.11

[27] Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. 2002. On the Recommending of Citations for Research Papers. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW '02)*. ACM, New York, NY, USA, 116–125. https://doi.org/10.1145/587078.587096

[28] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. ACM, New York, NY, USA, 1097–1101. https://doi.org/10.1145/1125451.1125659

[29] Kim Mens and Angela Lozano. 2014. *Source Code-Based Recommendation Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 93–130. https://doi.org/10.1007/978-3-642-45135-5_5

[30] Xianghang Mi, Feng Qian, Ying Zhang, and XiaoFeng Wang. 2017. An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance. In *Proceedings of the 2017 Internet Measurement Conference (IMC '17)*. ACM, New York, NY, USA, 398–404. https://doi.org/10.1145/3131365.3131369

[31] A. Namoun, A. Daskalopoulou, N. Mehandjiev, and Z. Xun. 2016. Exploring Mobile End User Development: Existing Use and Design Factors. *IEEE Transactions on Software Engineering* 42, 10 (Oct 2016), 960–976. https://doi.org/10.1109/TSE.2016.2532873

[32] Anh Tuan Nguyen, Michael Hilton, Mihai Codoban, Hoan Anh Nguyen, Lily Mast, Eli Rademacher, Tien N. Nguyen, and Danny Dig. 2016. API Code Recommendation Using Statistical Learning from Fine-grained Changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 511–522. https://doi.org/10.1145/2950290.2950333

[33] K. T. Stolee and S. Elbaum. 2013. Identification, Impact, and Refactoring of Smells in Pipe-Like Web Mashups. *IEEE Transactions on Software Engineering* 39, 12 (Dec 2013), 1654–1679. https://doi.org/10.1109/TSE.2013.42

[34] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical Trigger-action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 803–812. https://doi.org/10.1145/2556288.2557420

[35] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3227–3231. https://doi.org/10.1145/2858036.2858556

[36] Lina Yao, Quan Z. Sheng, Anne H.H. Ngu, Helen Ashman, and Xue Li. 2014. Exploring Recommendations in Internet of Things. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 855–858. https://doi.org/10.1145/2600428.2609458

[37] Y. Ye and G. Fischer. 2002. Supporting reuse by delivering task-relevant and personalized information. In *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*. 513–523. https://doi.org/10.1109/ICSE.2002.1007995