Toward a Better Understanding of End-User Debugging Strategies: A Pilot Study

Ren Manfredi [1*], Margherita Andrao [1,2], Francesco Greco[3] , Giuseppe Desolda[3] , Barbara Treccani[1] , Massimo Zancanaro[1,2]

[1] *Department of Psychology and Cognitive Science, University of Trento, Rovereto (Trento), Italy*
[2] *Fondazione Bruno Kessler – FBK, Trento, Italy*
[3] *University of Bari Aldo Moro, Bari, Italy*

Abstract

In this paper, we describe a pilot study aimed to explore strategies used by non-programmer users to test trigger-action rules for customizing an IoT device. The main goal of our research was to examine strategies used by participants to detect and solve errors. In the pilot study, we asked non-programmers to imagine testing a set of rules, some of which were bugged. The pilot study was meant to understand the feasibility of this approach to investigate users' mental models while performing this kind of task.

Keywords

End-User Development, End-User Programming, Debugging, Trigger-action programming, IoT.

1. Introduction

In this paper, we present a pilot study to explore the strategies of non-programmer users in debugging trigger-action (TA) rules to customize an IoT educational tool. In particular, we wanted to investigate how naive users approach debugging and the characteristics of the strategies they adopt.

In recent years, the rapid diffusion of IoT has brought end-users to the center of a complex ecosystem made of interconnected objects and web services, changing the way they live [1][2].

In this context, the End-User Development (EUD) paradigm has allowed non-technical users without programming skills to customize the behavior of their devices and applications [3], empowering users and letting them benefit from the potential of IoT. Specifically, through the trigger-action programming (TAP) approach, end-users have the possibility to create rules to automate the behavior of both hardware and software artifacts.

The relative simplicity and applicability of TAP to IoT have attracted a lot of interest [4]. However, this rule-based approach has limitations. Indeed, despite its ease of use, non-programmers still make numerous mistakes in composing TA rules, like loops, inconsistencies, and redundancies [5]. That is important because poor or conflicting rule settings can lead to unsatisfactory or even potentially dangerous behavior for the user.

Only very recently, a few studies have been carried out that focused on the problem of rule errors in EUD and explored debugging approaches to support end-users in customizing their IoT devices [6]. However, while many efforts have been directed towards debugging for mashup programming, spreadsheet, and rule analysis, little work has investigated debugging in TAP [3][5]. Some of these works, inspired by and extending the Interrogative Debugging paradigm of Ko and Myers [7], proposed

3rd International Workshop on Empowering People in Dealing with Internet of Things Ecosystems. Workshop co-located with AVI 2022, June 06, 2022, Frascati, Rome, Italy
* Corresponding author
EMAIL: ren.manfredi@unitn.it (R.Manfredi); margherita.andrao@unitn.it (M.Andrao); f.greco66@studenti.uniba.it (F.Greco); giuseppe.desolda@uniba.it (G.Desolda); barbara.treccani@unitn.it (B.Treccani); massimo.zancanaro@unitn.it (M.Zancanaro)
ORCID: 0000-0002-4584-3306 (R.Manfredi); 0000-0003-2245-9835 (M. Andrao); ; 0000-0003-2730-7697 (F. Greco) ; 0000-0001-9894-2116 (G.Desolda); 0000-0001-8028-0708 (B. Treccani); 0000-0002-1554-5703 (M. Zancanaro)

tools and approaches that allow end-users to simulate their own rules and identify errors [3][5][8]. Specifically, these pioneering works developed and tested different EUD interfaces able to simulate the rules created, detect potential errors, and return an explanation of those errors to the user to support them in correcting the rules. The results of these studies, although preliminary, seem to suggest that interfaces of this type can support the end-user in dealing with and better understanding errors in the composition of trigger-action rules.

However, several aspects remain to be clarified. In particular, we still do not know much about how end-users approach debugging and what strategies they adopt [3]. Knowing more about end users' debugging strategies is important to inform the design of better tools to support this important task [6].

The present study aimed to explore the strategies and approaches of naive users in debugging TA rules to outline possible and valid future research.

1.1. Tool and EUD interface

In this pilot study, a tangible educational IoT device prototype (Figure 1) was employed [9]. Such a device was designed to support mathematics learning in elementary school children. It has a rectangular surface with five slots for placing external pieces (digit, symbol, or operator tiles), and it is provided with a visual and acoustic feedback system. This tool makes it possible to implement different types of mathematical exercises such as comparing quantities, sorting numbers, and simple arithmetic operations.

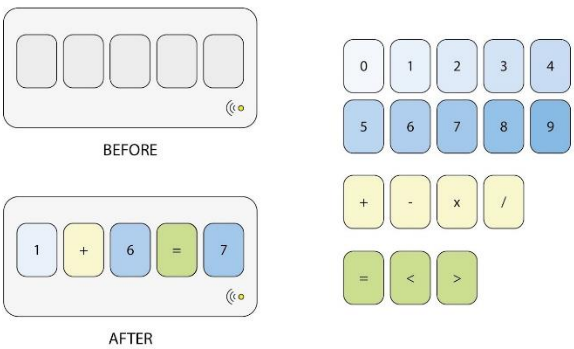

***Figure 1**. Schematic of the tangible tool (left) and tiles (right).*

Teachers could customize the tool's functioning through a set of language primitives for actions, states, and events implemented as part of an existing authoring interface (see Figure 2). The primitives for the actions consist of commands to control visual (lights) and acoustic feedback. At the same time, states and events describe respectively the operations that teachers and children can perform on the tool (i.e., insertion and removal of tiles) and the tool configuration at that moment. An example of an event description is "WHEN a digit tile is inserted", while an example of a state is "WHILE the position to the right of the inserted tile is empty". Finally, "Turn on blue LED SMARTER" is an example of action.

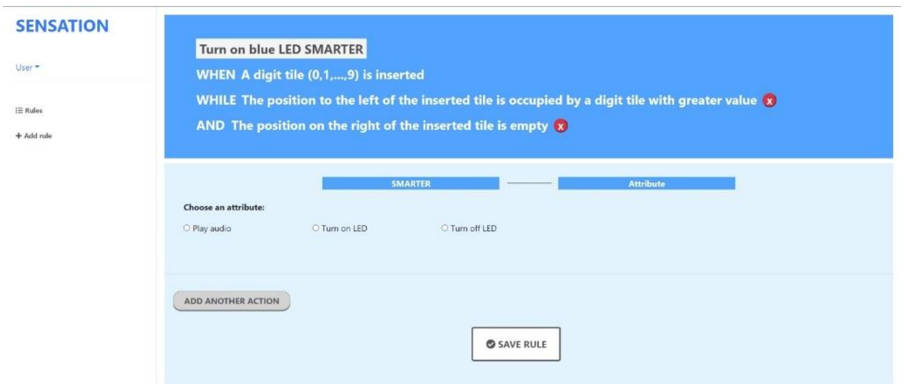

***Figure 2**. A screenshot of the authoring interface.*

2. The pilot study

Five (5) primary school math teachers with no previous experience in programming were recruited. The study consisted of individual semi-structured interviews conducted remotely (90 minutes).

After describing the tool and the interface through videos and explanatory images, the participants were familiarized with the interface and a set of rules already prepared and related to a simple number ordering game (from the smallest number to the largest). In the first phase, participants were then asked to discuss how they would evaluate the correctness of the rules and then to indicate three specific actions by which they would test the program's functioning (debugging task).

In the second phase of the study, participants were given an error-finding task on a different set of rules in which two types of errors were intentionally included. Specifically, both inconsistent and redundant rules were introduced, that is, rules providing for two conflicting actions for the same trigger and different rules producing the same outcome, respectively.

A verbal reporting procedure was used to explore participants' strategies and mental models during the familiarization and error-finding tasks.

2.1. Results

As a first approach to the debugging task, all participants imagined testing the rules in the field with a hands-on approach, i.e., in the classroom with their students. Some of them proposed a trial-and-error strategy ("I would try to make a rule if I realized something was wrong, I would go back and change it"), while others (participants B and C) suggested testing directly with the children using a more functional and child-focused approach.

Specifically, some participants (such as B, D, E) proposed a "step-by-step" strategy, e.g., proposing to insert the first tile, then a correct tile, and finally an incorrect one, checking the outcomes each time a single tile was inserted. In contrast, Participant A took a different approach, preferring to place all the tiles first to observe the result. Interestingly, such an approach, focused on obtaining a direct solution, has already been observed in novice programmers and has been associated with a misunderstanding of how the system works [10][11].

Finally, participants A and C proposed the placement of a symbol tile to test the alternative case. Also, participant E evaluated this approach, but she argued it was not fundamental considering the task goal of sorting numbers progressively. Debugging task results are summarized in Table 1.

Table 1
Debugging task results

Strategy	Strategy description	Participants
Step-by-step	Placement of one tile at a time, checking the result at each step	B, D, E
Direct solution	Placement of all tiles, with only one final check	A
Alternative case	Placement of symbol tiles to test alternative cases	A, C

In the error-finding phase, participants exhibited two different approaches. Some assumed a rule-by-rule approach (participants B and D), directing the error search to individual rules and ignoring their relationships. Participants who adopted this approach failed to identify any errors present in the ruleset. They also state that the errors were due to the excessive number of rules, in their opinion, unnecessary and confusing. Conversely, participants who extended their error search to include the relationships between rules (participants A, C, E) could identify some errors. Eventually, only one participant was able to identify all errors correctly. Results of the error-finding task are summarized in Table 2.

Consistent with the literature [5], the most commonly identified error was inconsistency, while redundancy was the most difficult error to identify.

Table 2
Error-finding task results

Participant	Approach adopted	Identified errors
A	Error search extended to rule relationship	Inconsistency
B	Error search limited to individual rules	None
C	Error search extended to rule relationship	Inconsistency
D	Error search limited to individual rules	None
E	Error search extended to rule relationship	Inconsistency, Redundancy

3. Conclusions

In this pilot study, we explored the strategies adopted by non-programmer users for debugging a TA rule set and for finding inconsistent and redundant rules. While the extant literature on end-users debugging TA rules focuses on tools to support bug identification, in our approach, we tried to focus on the mental models users initially assume in facing a debugging task.

Three different debugging strategies implemented by participants emerged from the results. These strategies were partly similar to those already observed in novice programmers [10][11][12]. Nevertheless, to our knowledge, they have not been discussed in the context of EUD with TA rules. Specifically, naive users might have more problems considering the entire set of rules and their relationships. We believe that extending this approach with a more robust study will help to design more effective tools (along the lines, for example, of the work done by Corno et al. [5]). In our future work, we plan to refine the study's design by involving the users in actual debugging with the real tool.

4. Acknowledgments

This work has been supported by the Italian Ministry of Education, University and Research (MIUR) under grant PRIN 2017 "EMPATHY: EMpowering People in deAling with internet of THings ecosYstems" (Progetti di Rilevante Interesse Nazionale – Bando 2017, Grant 2017MX9T7H).

5. References

[1] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. Personalization of context-dependent applications through trigger-action rules. ACM Transactions on Computer-Human Interaction 24, 2 (2017), 1–33. DOI:http://dx.doi.org/10.1145/3057861

[2] Barbara Rita Barricelli and Stefano Valtolina. 2015. Designing for end-user development in the internet of things. End-User Development (2015), 9–24. DOI:http://dx.doi.org/10.1007/978-3-319-18425-8_2

[3] Luigi De Russis and Alberto Monge Roffarello. 2018. A debugging approach for trigger-action programming. Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (2018). DOI:http:// dx.doi.org/10.1145/3170427.3188641

[4] Paternò, F., & Santoro, C. (2019). End-user development for personalizing applications, things, and robots. International Journal of Human-Computer Studies, 131, 120-130.

[5] Corno, F., De Russis, L., & Monge Roffarello, A. 2019, May. Empowering end users in debugging trigger-action rules. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (pp. 1-13).

[6] Grigoreanu, V., Burnett, M., Wiedenbeck, S., Cao, J., Rector, K., & Kwan, I. 2012. End-user debugging strategies: A sensemaking perspective. ACM Transactions on Computer-Human Interaction (TOCHI), 19(1), 1-28.

[7] Ko & Myers 2004 (Ko, A. and Myers, B. 2004. Designing the whyline: A debugging interface for asking questions about program behavior. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 151–158)
[8] Manca, M., Paternò, F., Santoro, C., & Corcella, L. 2019. Supporting end-user debugging of trigger-action rules for IoT applications. International Journal of Human-Computer Studies, 123, 56-69.
[9] Andrao, M., Desolda, G., Greco, F., Manfredi, R., Treccani, B., Zancanaro, M. 2022, June . SMARTER: an IoT learning game to teach math. In International Conference on Advanced Visual Interfaces (AVI 2022), June 6–10, 2022, Frascati, Rome, Italy.
[10] Chen Li, Emily Chan, Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2019. Towards a framework for teaching debugging. Proceedings of the Twenty-First Australasian Computing Education Conference on - ACE '19 (2019). DOI:http://dx.doi.org/10.1145/3286960.3286970
[11] Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L. and Zander, C., 2008. Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. Computer Science Education, 18(2), pp.93-116.
[12] S. Fitzgerald, R. McCauley, B. Hanks, L. Murphy, B. Simon and C. Zander, "Debugging From the Student Perspective," in IEEE Transactions on Education, vol. 53, no. 3, pp. 390-396, Aug. 2010, doi: 10.1109/TE.2009.2025266.