TAPping into Mental Models with Blocks

Daniel Rough
School of Computer Science, University of St Andrews
djr53@st-andrews.ac.uk

Aaron Quigley
School of Computer Science, University of St Andrews
aquigley@st-andrews.ac.uk

***Abstract*—Trigger-Action Programming (TAP) has been shown to support end-users' rule-based mental models of context-aware applications. However, when desired behaviours increase in complexity, this can lead to ambiguity that confuses events, states, and how they can be combined in meaningful ways. Blocks programming could provide a solution, through constrained editing of visual triggers, conditions and actions. We observed slips and mistakes by users performing TAP with Jeeves, our domain-specific blocks environment, and propose solutions.**

I. Introduction

With the increasing ubiquity of smartphones, wearable devices, and home automation systems in our everyday lives, a need has arisen to support end-users in customising the behaviour of their technology. Today, a range of commercial platforms aim to put the power of context-aware computing into the hands of non-programmers. A prevalent example is IFTTT [1], an app that allows users to customise their smartphones and external devices, with a growing user base that demonstrates a strong desire for such functionality.

Automation apps have succeeded in lowering the barriers to simple trigger-action programming (TAP) by end-users. For example, IFTTT allows a user to pick a trigger event, and an associated action, from a list of options. However, such simplicity necessitates a sacrifice in flexibility, and multiple actions or triggers cannot be combined. While this complex functionality is desired by users, previous work shows that they design trigger combinations with ambiguous interpretations [1], [2].

One solution to this problem is to actively disallow these ambiguous trigger combinations [2]. The syntax-directed editing of blocks supports this strategy. However, even if blocks support clearly defined triggers, can end-users make sense of correct combinations? Further, do blocks raise issues that text avoids? We conducted a usability study of Jeeves, our blocks environment [3] for creating experience sampling apps, recruiting 10 participants with limited or no programming experience. We briefly discuss pertinent issues they encountered in TAP, and suggest potential solutions.

II. Visual TAP Concepts

Previous work has shown that natural language is unsuitable for making the distinction between triggers (discrete events) and conditions (ongoing states) [4]. Even in a jigsaw-puzzle based composition environment, triggers and actions can be confused if they are not distinct [5]. Inspired by the Blockly editor [6] and similar environments, Jeeves blocks are separated into 'palettes'. An example of this distinction can be seen in Figure 1. The trigger represents an event - a user's location, acquired through their smartphone's GPS sensor, has **changed** to Home. The if-block then evaluates its condition, representing whether a user is **currently** Home. Finally, the action captures and stores data from the GPS sensor. While such an execution is nonsensical, it serves to illustrate the different uses of smartphone sensor functionality in Jeeves.

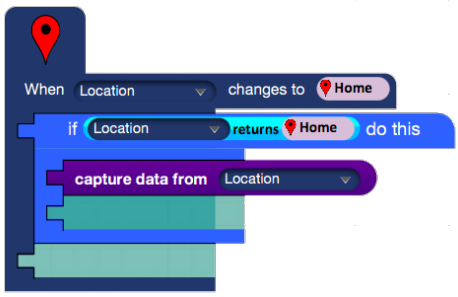

Fig. 1. A Location *trigger*, with an if-block containing a location *condition*, and an *action* to capture the current location

Unambiguous context-aware behaviours should be composed of one discrete *event* trigger, with an optional number of *state* conditions [1], [2]. As shown in Figure 1, Jeeves, like other blocks languages, supports these constraints through the visual affordance of correct connections. With no external 'notches', event triggers cannot be nested together. Multiple conditions can be nested, but only within an event trigger.

III. Slips and Mistakes

We observed that Jeeves blocks supported our participants in thinking about triggers, conditions and attributes correctly in task completion. However, errors from both *mistakes* and *slips* resulted in obscurity and damage of mental models [7].

A. From Slips To Falls

In part of a task, participants had to set a 'Do Not Disturb' attribute to be true. Figure 2 (Right) shows that a 'near miss' had a negative impact on their mental model of how blocks could be combined. One participant who made this slip, when later asked about her understanding of the 'set attribute' action, stated that she "*couldn't work out how to use it*", despite previously demonstrating a correct assumption of the action's behaviour. Participants learn that a block snapping into place

[1]https://ifttt.com/

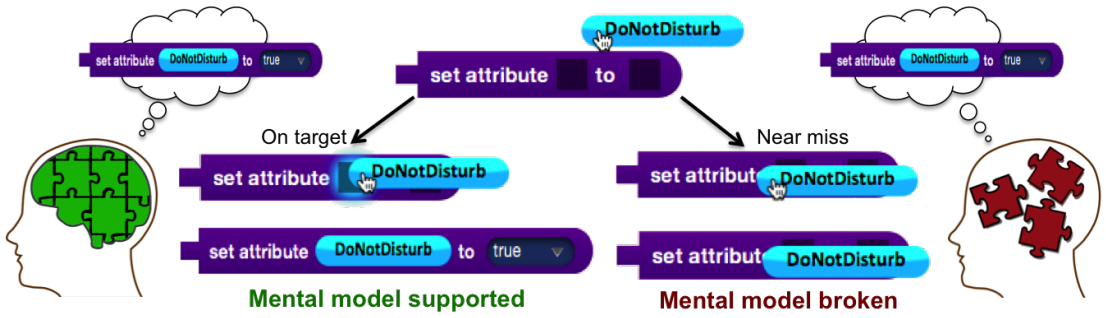

Fig. 2. Left: A correct snap supports a user's mental model. Right: A 'near miss' damages the mental model

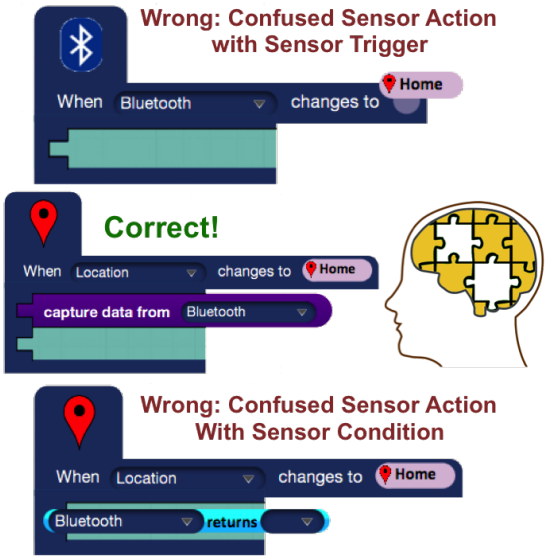

Fig. 3. Language ambiguity caused trigger, action and condition confusion

indicates correctness, so when this fails to happen, correct choices are discounted and mental models break down.

B. Blame The Name

Organising blocks by type into palettes supports *recognition over recall*. However, our participants' inexperience with TAP caused issues with recognition, thus they would hunt for a block by name, regardless of its type. In one task, we asked participants to turn on the Bluetooth sensor when a user returns home. Figure 3 shows two examples of participants failing to find the appropriate 'capture data' action - they instead tried adjusting the sensor trigger (top), or used a sensor condition in place of the action (bottom). When they found and used the correct action, they expressed doubt as to correctness.

IV. Rebuilding Blocks

How do we rebuild a damaged mental model? Based on feedback and observations of our study participants, we propose two potential solutions, which will be incorporated into Jeeves and evaluated in a further study.

A. Lock-On Blocks

Baudisch et al. [8] demonstrate their 'drag-and-pop' interaction, supporting drag-and-drop on large displays, whereby an icon's compatible drop locations are drawn towards it. We are currently considering a similar solution we call 'lock-on blocks', which will visually suggest compatible locations for dragged blocks, resolving certain mistakes, as well as reducing slips. As suggested by one participant, simple highlighting of correct locations for a dragged block may also be sufficient.

B. Palette Cleansing

The question "*Does that block live there?*" was raised by one of our participants, and echoed in various forms by others. Forgetting where to find correct blocks, many would search in the 'conditions' palette for an action. Conversely, they would sometimes look to actions for functionality provided by a condition. We propose that having the user click on a particular drop location would switch to the relevant palette, guiding their search. This 'palette cleansing' could reinforce how triggers, actions and conditions combine.

V. Conclusion

We propose that blocks can support TAP, but care must be taken to reinforce fragile assumptions when they are correct.

References

[1] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," in *Proc. 32nd Annu. ACM Conf. Human factors in computing systems*, 2014, pp. 803–812.

[2] J. Huang and M. Cakmak, "Supporting mental model accuracy in trigger-action programming," in *Proc. 2015 ACM Int. Joint Conf. Pervasive and Ubiquitous Computing*, 2015, pp. 215–225.

[3] D. Rough and A. Quigley, "Jeeves - A Visual Programming Environment for Mobile Experience Sampling," *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symp.*, pp. 121–129, 2015.

[4] M. García-Herranz, P. Haya, and X. Alamán, "Towards a Ubiquitous EndUser Programming System for Smart Spaces," *Journal of Universal Computer Science*, vol. 16, no. 12, pp. 1633–1649, 2010.

[5] Y. Dahl and R. M. Svendsen, "End-user composition interfaces for smart environments: A preliminary study of usability factors," in *Lecture Notes in Computer Science*, vol. 6770 LNCS, no. 2, 2011, pp. 118–127.

[6] N. Fraser, "Ten things we've learned from blockly," in *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, Oct 2015, pp. 49–50.

[7] D. A. Norman, "Design rules based on analyses of human error," *Commun. ACM*, vol. 26, no. 4, pp. 254–258, Apr. 1983.

[8] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, A. Zierlinger, and M. Push, "Drag-and-Pop and Drag-and-Pick: Techniques for accessing remote screen content on touch- and pen-operated systems," *Proc. INTERACT '03*, pp. 64–57, 2003.