How End Users Express Conditionals in Programming by Demonstration for Mobile Apps

Marissa Radensky
Computer Science Department
Amherst College
Amherst, MA
mradensky19@amherst.edu

Toby Jia-Jun Li
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA
tobyli@cs.cmu.edu

Brad A. Myers
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA
bam@cs.cmu.edu

***Abstract*—Though conditionals are an integral component of programming, providing an easy means of creating conditionals remains a challenge for programming-by-demonstration (PBD) systems for task automation. We hypothesize that a promising method for implementing conditionals in such systems is to incorporate the use of verbal instructions. Verbal instructions supplied concurrently with demonstrations have been shown to improve the generalizability of PBD. However, the challenge of supporting conditional creation using this multi-modal approach has not been addressed. In this extended abstract, we present our study on understanding how end users describe conditionals in natural language for mobile app tasks. We conducted a formative study of 56 participants asking them to verbally describe conditionals in different settings for 9 sample tasks and to invent conditional tasks. Participant responses were analyzed using open coding and revealed that, in the context of mobile apps, end users often omit desired *else* statements when explaining conditionals, sometimes use ambiguous concepts in expressing conditionals, and often desire to implement complex conditionals. Based on these findings, we discuss the implications for designing a multi-modal PBD interface to support the creation of conditionals.**

Keywords—conditionals, programming by demonstration, verbal instruction, end-user development, natural programming.

I. Introduction and Background

Script generalization continues to be the key challenge for programming-by-demonstration (PBD) systems for task automation [1],[2]. A PBD system should not only produce literal record-and-replay macros, but also understand end user intentions behind recordings and be able to perform similar tasks in different contexts [2]. Prior approaches of asking users to provide several examples from which AI algorithms can make generalizations using program synthesis approach, and having users supply the features needed for generalization have been shown to be infeasible due to users' limited ability to understand generalization options and provide sets of useful examples spanning the complete space for synthesizing the intended programming logic. Our research on SUGILITE [3], EPIDOSITE [4], and APPINITE [5] demonstrated that leveraging natural language instructions grounded by mobile apps' GUIs is a promising method to enable users to naturally express their intentions for generalizing PBD scripts. While these systems use natural language instructions to infer script parameterization and data descriptions for individual actions, none address the challenge of enabling users to create task-wide conditionals, an important aspect of generalization.

Evidenced in [6], non-programmers state conditionals using varying structures and levels of description. Understanding the different manners in which end user programmers construct conditionals and whether or not they provide the details necessary for an intelligent agent to comprehend their conditionals is crucial to building a PBD system that can interact with users to extract intended conditionals from verbal instructions. In this extended abstract, we summarize our study on how end users naturally describe conditionals in the context of mobile apps and discuss the implications for designing a multi-modal PBD interface that supports conditionals.

II. Methods

A. Formative Study

We conducted a formative study on Amazon Mechanical Turk with 56 participants (38 non-programmers; 38 men, 17 women, 1 non-binary person). 30 participants completed a 3-part survey, while 22 completed either Part 1 or 2, both followed by Part 3. The other 4 participants completed both versions of the survey. 11 of 104 utterances in Part 1, 10 of 62 in Part 2, and 19 of 65 in Part 3 were excluded from analysis due to question misunderstandings and blank responses. Each part included an example question and responses.

In Part 1, participants were given a description of a task for an intelligent agent to complete within a PBD system for mobile apps. The task had distinct associated situations, each of which led to the task being completed differently. The participants were assigned one of 9 tasks such as playing a type of music that depends on the time of day or going to a location with a mode of transportation that depends on how much time getting there by public transportation takes. They were asked what they would say to the agent so that it may understand the *difference* among the situations, and then for any alternative responses. To avoid biasing responses' wording, we used the Natural Programming Elicitation method [7], presenting pictures alongside limited text to describe the task and situations.

Part 2 differed in purpose from Part 1 in that it had participants express conditionals while looking at relevant phone screens. Participants were given a mobile app screenshot with yellow arrows pointing to the screen components containing information pertinent to the condition on which the task situation depended. If other components might have been

978-1-5386-4235-1/18/$31.00 ©2018 IEEE

confused with the correct ones, red arrows pointed them out. Participants were asked to explain to the agent how to locate and use the correct components to determine the situation at hand. Finally, Part 3 asked participants for another task for which an agent should perform differently in distinct situations.

B. *Open Coding*

The participants' responses were analyzed using open coding. For all 3 parts, a code identified conditionals with unambiguous versus ambiguous language. For Part 1, codes were used to identify conditionals without *else* statements, to categorize the implied necessity of omitted *else* statements, and to identify omitted *else* statements whose contents are implied. For Part 3, codes were used to identify conditionals with complex structures, those that use 2 or more apps, those initiated by automatic triggers, and those with automatic triggers based on information found in open APIs or app GUIs.

III. Preliminary Results and Implications

A. *Omission of Else Statements*

In Part 1, though only conditionals with *else* statements were given as example responses, 56% of the 39 participants who completed Part 1 provided at least one response without an *else* statement. Of those participants, 45% omitted an *else* statement even though it was not clear whether it would be needed or not. As an example, "*Whenever I go to bed past 11 p.m. set 3 alarms*" may or may not require an alternative such as setting 1 alarm. Furthermore, 18% omitted an *else* statement when it was definitely necessary. "*Default to upbeat music until 8pm every day*," for instance, requires an alternative for other times. This finding suggests that end users will often omit the appropriate *else* statement in their natural language instructions. Additionally, merely 33% of participants expressed conditionals that implied the required alternative when it was omitted and possibly or definitely necessary (e.g. "*If a public transportation access point is more than half a mile away, then order an Uber*" implies an alternative of finding a public transportation route). PBD must thus be designed to detect omitted *else* statements in natural language and guide users to resolve ambiguity in conditional alternatives.

B. *Ambiguous Concepts in Conditions*

6 of the 9 tasks' descriptions deliberately referred to conditions incorporating ambiguous concepts such as "*cold*" and "*daytime.*" To the last 27 participants, only unambiguous example responses were shown to try to guide them away from using ambiguous concepts. 10 of them completed Part 1 for one of the 6 tasks just mentioned. 40% of the 10 participants *still* supplied an ambiguous condition, such as "*When I am going to outside at chance of rain I will take umbrella ... .*" An agent should be able to use multi-turn dialogue to ask users to clarify ambiguous concepts like "*chance of rain*."

With or without seeing exclusively unambiguous example responses, 25 participants completed Part 2 for one of the 6 potentially ambiguous tasks. Interestingly, in this part in which participants were provided an app screenshot displaying specific information relevant to their task's condition, all 25 participants provided clear definitions such as "*longer than an hour*" and "*past 8:00 pm*" for ambiguous concepts. However, 15 participants who were given all unambiguous example responses completed Part 3, in which participants invented their own conditional tasks, and 20% of these 15 participants expressed conditions that contained ambiguous concepts. These results suggest that users might eliminate ambiguity from their conditions by describing them while looking at the relevant mobile app GUIs. If users still use ambiguous concepts, they may be guided to disambiguate their conditions by prompts to explain the ambiguous concepts in the context of the GUIs.

C. *Desired Conditionals*

Many participants desired conditionals that were complex in some manner. 55% of the 44 invented conditionals use more than 1 app, and 9% use more than 2 apps. 14% of the conditionals, such as the switch statement "*if it is day X, order food Y,*" have a more complicated structure than just "If ... else ... ." Also, automatic triggers instead of voice commands must initiate 55% of the conditionals, and 58% of these triggers are not simple triggers like a notification but rather information found in open APIs or app GUIs. For instance, "*Turn the light on in the room if I'm at home at sunset or when I arrive home after sunset"* has a trigger involving the user's location, time of sunset, and current time, all information in open APIs. These results motivate our PBD system, which allows users to develop scripts for cross-app tasks more complex and personalized than common pre-programmed ones.

We are now researching how to augment SUGILITE [3] and APPINITE [5] to have all the indicated functionalities.

Acknowledgment

This research was supported in part by Oath through the InMind project and in part by NSF grants CCF-1560137 and CCF-1814826. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

[1] H. Lieberman, *Your wish is my command: Programming by example*. Morgan Kaufmann, 2001.

[2] A. Cypher and D. C. Halbert, *Watch what I do: programming by demonstration*. MIT press, 1993.

[3] T. J.-J. Li, A. Azaria, and B. A. Myers, "SUGILITE: Creating Multimodal Smartphone Automation by Demonstration," in *Proceedings of CHI 2017*.

[4] T. J.-J. Li, Y. Li, F. Chen, and B. A. Myers, "Programming IoT Devices by Demonstration Using Mobile Apps," in *Proceedings of IS-EUD 2017*.

[5] T. J.-J. Li *et al.*, "APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Natural Language Instructions," in *Proceedings of VL/HCC 2018*.

[6] J. F. Pane, B. A. Myers, and others, "Studying the language and structure in non-programmers' solutions to programming problems," *Int. J. Hum.-Comput. Stud.*, vol. 54, no. 2, pp. 237–264, 2001.

[7] Brad A. Myers, Andrew J. Ko, Thomas D. LaToza and YoungSeok Yoon. "Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools," *IEEE Computer*. 2016. vol. 49, no. 7. pp. 44-52.