End User Mobile Task Automation using Multimodal Programming by Demonstration

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

Abstract—**Conversational agents are often used to perform tasks on smartphones, but existing conversational agents are limited in capabilities and lack of customizability. My work explores using the programming-by-demonstration approach to enable end users to program new tasks for conversational agents by demonstrating using the familiar graphical user interfaces of third-party apps. I propose to use a multi-modal (demonstration and verbal instruction) interface to support generalization, editing, error handling as well as creating control structures in creating such smartphone automation.**

Keywords—programming by demonstration, end user development, task automation, SUGILITE

I. BACKGROUND

Smartphone users perform all kinds of tasks on mobile apps nowadays thanks to the wide range of apps available. Google Play store alone has more than 3 million apps. However, many tasks in smartphone apps are repetitive and tedious. A common daily task like ordering a cup of coffee using the Starbucks app can require as many as 18 clicks (and way more if you are not already registered). As a result, users would often like to use agents to automate these tasks and to perform them on the users' behalf. In our survey, 62.7% of the participants were interested in having a way to automate their repetitive mobile tasks [1]. Conversational agents can also enable the users to perform tasks by voice, which is useful in contexts where the user does not want or cannot touch on the phone.

Existing conversational agents like Apple Siri, Google Assistant, Amazon Alexa and Microsoft Cortana can perform various tasks, including device control, communication, web search and calendar management. However, these agents have limited functionalities. They can only invoke built-in apps (e.g., Phone, Calendar, Music, etc.) and a few integrated external apps and web services (e.g., Search, Weather, Wikipedia). While many of them provide APIs to enable third-party apps to integrate, so the apps can be invoked by the agents, only a small fraction of apps have been integrated, and only a subset of the most popular tasks in those apps have been supported due to the development costs and effort required for the integration. In particular, the "long tail" of apps and tasks are unlikely to ever get supported under this model.

Another limitation of the existing conversational agents is the lack of customizability and personalization. The users have little control over *how* the tasks are performed besides providing values for pre-defined parameters. Personalization is also often limited to preset values, like the home location of the user and the name of the user.

II. OUR APPROACH AND PROGRESS TO DATE

Currently, we have already designed and implemented a system named SUGILITE [1] [1]. It uses the *programming-by-demonstration* (PBD) approach [2], [3] to enable end users to extend the capabilities of conversational agents by creating automation scripts for their tasks. This approach leverages the resources of millions of available mobile apps and the users' knowledge about how to operate these apps by letting the users simply demonstrate the procedures of performing the tasks on the regular GUI of arbitrary third-party mobile apps. For example, if a user wishes to teach the agent the task "order a medium size cup of latte," she only needs to demonstrate ordering a medium size cup of latte using the app of her favorite coffee shop (e.g Starbucks). She can also include her personal preferences (e.g., any toppings, add-ins, sweetness) in the demonstration to create a personalized script for the task.

A major advantage of SUGILITE's approach is its applicability – since it uses the user interactions with the app interface for the demonstration, it can automate tasks using any third-party Android apps, as opposed to other task automation approaches that rely on the availability of an open API (e.g. [4]), a special framework or library (e.g. [5]), or the structure of web pages [6]. We have tested automating a wide range of tasks with different Android apps using SUGILITE.

Our work also focuses on the generalizability of the automation. When the user demonstrates a task, the agent should learn not only to perform the exact same task (as in [7]), but also to perform similar tasks of different varieties, or with different parameters. SUGILITE uses a multi-modal approach where the user provides both demonstration and verbal commands. With the help of the verbal commands, SUGILITE can identify parameters and their possible values in the task. In the previous example of ordering a coffee, after demonstrating ordering a medium size cup of latte, the user can then use SUGILITE to run another similar task like: "order a large size cup of cappuccino" to invoke the same coffee shop app without having to demonstrate again.

Good usability is another important characteristic of our approach. The learning barrier of SUGILITE is low for end users, because they can demonstrate the tasks directly using the

[1] A gemstone, and is short for "**S**martphone **U**sers **G**enerating **I**ntelligent **L**ikable **I**nterfaces **T**hrough **E**xamples."

This work was supported in part by Yahoo! InMind project.

familiar user interfaces of the third-party apps involved, unlike approaches where the users need to program the automation using a block programming language (e.g. [8]) or a scripting language (e.g. [9]). In a lab study, we recruited 19 participants with various levels of prior programming experience to try to program SUGILITE to perform four common tasks (requesting Uber, checking a sports score, ordering a coffee, and sending an email). Most participants were able to successfully program scripts for these tasks through SUGILITE regardless of their prior programming experience. In this study, 85.5% of the automation scripts created by the participants ran and performed the tasks successfully. There was no significant correlation between the participants' prior programming experience and the task completion rate or task completion time. SUGILITE was also found efficient to use in the task completion time, and easy to use from the participants' subjective feedback.

We have also extended SUGILITE into the domain of smart home automation though a follow up system named EPIDOSITE [10], which supports users to leverage smartphones as hubs for smart home automation, and to create automation for smart home devices by demonstrating the desired behaviors using the smartphone apps for the smart home devices. Using the EPIDOSITE extension, SUGILITE scripts can also control smart home devices, read the values and the status of smart home devices, and be triggered by events from smart home devices, by smartphone app usage context, or by external web services, allowing the creation of highly context-aware automations.

III. PROPOSED FUTURE WORK

A focus of the future work for SUGILITE is on further exploring its multi-modal interaction. In the current SUGILITE system, the user only gives a verbal command *before* the demonstration. Then SUGILITE tries to generalize the script from the demonstration by using the user's verbal command as the intention of the user and trying to identify parameters in the command. In future work, we want to go further and allow the users to give verbal instructions in multi-turn conversations with the agent and to perform demonstrations simultaneously while speaking.

We propose to support the following four activities through multi-modal interaction: generalization disambiguation, conditionals creation, procedure editing, and error handling. All of these are consider long-standing major challenges in end-user PBD tools, and we think leveraging the user's natural language instructions can be very powerful in supporting them.

The generalization disambiguation solves the *data description problem* in PBD [2], [3]. When the user demonstrates clicking on a UI element on the screen, it is difficult to determine what feature (e.g., text label, id, screen location, child elements, etc.) should be used for identifying the item to click on during future executions of the script. In a pilot study, we asked the users to give verbal instructions for their actions while demonstrating. We found that these instructions are often very helpful in disambiguating the features (e.g., the users said things like "click on the first item in the list", "click on the submit button", "choose the option with the lowest price"). We plan to use an *interaction proxy* [11] to allow the users to talk about the UI elements on the screen during the demonstration and specify the features to use for identifying the element to operate on. The proxy should support the user in interactively "discussing" and specifying the procedures of the automation without actually invoking the target app.

This simultaneous demonstration and verbal instruction inputs should also help with conditionals creation and procedure editing. The user should be able to simply say instructions like "choose iced coffee here if it's hot outside" to create conditionals, and "order a cup of cappuccino, but choose delivery instead of pickup" to quickly edit an automation. To achieve this, we will need to incorporate visualizations that represent the script structure, and design interaction models that leverage state-of-art NLP and AI techniques.

Another issue is error handling. Because SUGILITE scripts rely on the GUI of third-party apps, they may fail if (1) the app's GUI has changed due to an update or (2) an app gets into an unexpected or unknown state. We seek to design new multi-modal interactions so users can fix the script collaboratively with the system to add handlers for new situations while retaining the old script for the original situation if desired.

Beyond the lab study we have already done [1], we also plan to conduct a longitudinal field study to further understand how users use SUGILITE in real life contexts. We are particularly interested in knowing what tasks they choose to automate, what approach they choose for automating the tasks, whether different components of SUGILITE can support these real end user development activities as intended, and how much benefit SUGILITE can provide in real smartphone usage.

REFERENCES

[1] T. J.-J. Li, A. Azaria, and B. A. Myers, "SUGILITE: Creating Multimodal Smartphone Automation by Demonstration," in *Proceedings of CHI '17*, 2017, pp. 6038–6049.

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

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

[4] R. de A. Maués and S. D. J. Barbosa, "Keep Doing What I Just Did: Automating Smartphones by Demonstration," in *Proceedings of MobileHCI '13*, 2013, pp. 295–303.

[5] J.-H. Chen and D. S. Weld, "Recovering from Errors During Programming by Demonstration," in *Proceedings IUI '08*, 2008, pp. 159–168.

[6] G. Leshed, E. M. Haber, T. Matthews, and T. Lau, "CoScripter: Automating & Sharing How-to Knowledge in the Enterprise," in *Proceedings of CHI '08*, New York, NY, 2008, pp. 1719–1728.

[7] A. Rodrigues, "Breaking Barriers with Assistive Macros," in *Proceedings of ASSET '15*, New York, NY, 2015, pp. 351–352.

[8] "Automate · everyday automation for Android · LlamaLab." [Online]. Available: http://llamalab.com/automate/

[9] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using GUI Screenshots for Search and Automation," in *Proceedings of UIST '09*, 2009, pp. 183–192.

[10] 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 '17*, Eindhoven, the Netherlands, 2017.

[11] X. Zhang, A. S. Ross, A. Caspi, J. Fogarty, and J. O. Wobbrock, "Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility," in *Proceedings of CHI '17*, 2017, pp. 6024–6037.