Understanding On-the-Fly End-User Robot Programming

Laura Stegner*
University of Wisconsin–Madison
Madison, Wisconsin, United States
stegner@cs.wisc.edu

Yuna Hwang*
University of Wisconsin–Madison
Madison, Wisconsin, United States
yunahwang@cs.wisc.edu

David Porfirio
U.S. Naval Research Laboratory
Washington, DC, United States
david.j.porfirio2.ctr@us.navy.mil

Bilge Mutlu
University of Wisconsin–Madison
Madison, Wisconsin, United States
bilge@cs.wisc.edu

ABSTRACT

Novel end-user programming (EUP) tools enable on-the-fly (i.e., spontaneous, easy, and rapid) creation of interactions with robotic systems. These tools are expected to empower users in determining system behavior, although very little is understood about how end users perceive, experience, and use these systems. In this paper, we seek to address this gap by investigating end-user experience with on-the-fly robot EUP. We trained 21 end users to use an existing on-the-fly EUP tool, asked them to create robot interactions for four scenarios, and assessed their overall experience. Our findings provide insight into how these systems should be designed to better support end-user experience with on-the-fly EUP, focusing on user interaction with an automatic program synthesizer that resolves imprecise user input, the use of multimodal inputs to express user intent, and the general process of programming a robot.

CCS CONCEPTS

• **Human-centered computing** → *Systems and tools for interaction design*; • **Software and its engineering** → Development frameworks and environments.

KEYWORDS

End-user Programming, Robot Programming, Service Robots, Programming Tools, User Study, Usage Patterns, User Experience

ACM Reference Format:
Laura Stegner, Yuna Hwang, David Porfirio, and Bilge Mutlu. 2024. Understanding On-the-Fly End-User Robot Programming. In *Designing Interactive Systems Conference (DIS '24), July 01–05, 2024, IT University of Copenhagen, Denmark.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3643834.3660721

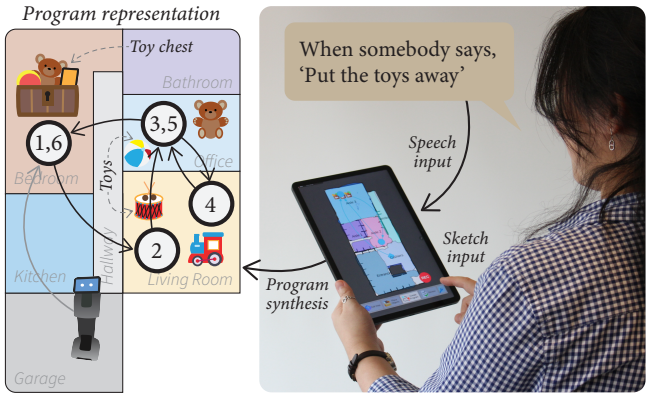

Figure 1: We investigate end-user experience with on-the-fly robot end-user programming using *Tabula*, a state-of-the-art open-source research prototype. *Right:* An experimenter using speech and touch input to program a robot to put toys away in a toy chest. *Left:* A visual representation of the generated program by a study participant (P5).

1 INTRODUCTION

Robots are increasingly being designed to aid *end users* in completing day-to-day tasks. These robots arrive with autonomous capabilities, yet they still require input from end users about which tasks must be completed and any contextual details surrounding the task. End users could include residents with robots in their private homes [45], shopkeepers with robot assistants to aid customers [60], caregivers with robots to assist in providing care to residents [63], and many more examples. In each of these scenarios, the end user may need to communicate to the robot a task for it to complete *on the fly*, *i.e.,* spontaneously, easily, and rapidly. To address this need, researchers have created various *end-user programming* (EUP) tools to allow end users to create interactions with robotic systems without extensive technical knowledge [1]. Specifically, EUP tools produce robot programs, which traditionally consist of sequences of actions for the robot to perform in order to complete a task.

Methods, techniques, and tools that facilitate rapid and intuitive robot EUP are rapidly proliferating (see Ajaykumar et al. [1] for a detailed review of robot end-user programming), including tools that better capture user intent [*e.g.,* 16], automatically synthesize programs given high-level user input [50], or contextualize programs within the user's environment [32]. EUP tools that incorporate multimodal inputs [*e.g.,* 24, 52, 53, 64] often combine various methods and techniques in an effort to create a more intuitive and natural on-the-fly EUP experience.

*Both authors contributed equally to this research.

DIS '24, July 01–05, 2024, IT University of Copenhagen, Denmark

ACM ISBN 979-8-4007-0583-0/24/07
https://doi.org/10.1145/3643834.3660721

Despite recent advances in EUP tools for robot programming, their full potential and impact remain unknown. A rich understanding of user experience with state-of-the-art EUP tools is missing, as these advanced EUP systems have yet to find real-world use and the research literature lacks deep understanding of use patterns, user experience, and limitations of these systems. As a result, very little is known about how these tools might be used by end users. Increasingly sophisticated methods and techniques that deviate from traditional programming paradigms require further exploratory user studies to contextualize the technical advances within end user needs and experiences.

Therefore, to help close this gap, we conducted an in-depth exploratory evaluation using a state-of-the-art on-the-fly EUP prototype called *Tabula* [52]. Tabula is an open-source EUP prototype tool that we developed previously (see Porfirio et al. [52]). It facilitates on-the-fly robot programming by combining multimodal input that enables end users to express task intent with a program synthesis technique that automatically completes missing elements of a program. Using Tabula as a medium for creating robot programs, we investigate the following research question.

- **RQ:** What are end users' experiences with on-the-fly robot programming?

To answer this question, we trained 21 participants to use Tabula and instructed each participant to create robot programs for three structured robot scenarios and one open-ended robot scenario. Specifically, we consider how end users approach the on-the-fly robot programming process through an in-depth exploratory evaluation of Tabula's key features, the multimodal inputs, and the program synthesizer. Therefore, we have the opportunity to probe usability aspects specific to Tabula's implementation, but also glean more widely applicable design insights.

This paper contributes to understanding how end users approach robot EUP through (1) a user study that evaluates a multimodal, on-the-fly EUP tool; (2) five themes that relate to both the usability and design of on-the-fly robot programming tools; and (3) a set of design guidelines that can inform future on-the-fly EUP tool design.

2 RELATED WORK

Our work builds on prior literature from software engineering, human-computer interaction (HCI), and human-robot interaction (HRI), focusing on how end users specify requests to interactive systems, approaches to end-user development and programming, and prior studies of end-user programming (EUP) tools.

2.1 Approaches to End-user Specification

Many programs written today do not rely entirely on professional programmers or roboticists [*e.g.,* 23, 30, 38, 47, 71]. Instead, end users with discrete domain expertise drive software development, specifically by contributing to obtaining a complete and consistent set of system requirements [37, 68]. Thus, seminal work in the software engineering field [*e.g.,* 7, 38] provides pointers on how to facilitate end-user specification, particularly at the exploratory phase [28] of the software lifecycle. *Dialogue* is an accessible paradigm for rapid prototyping based on its use in daily human communication [1]. Porfirio et al. [50] proposed an approach that utilized speech gathered from "role-playing" to synthesize human-robot interaction scenarios. Within the end-user specification frame, *visual programming interfaces* are frequently utilized. Flow-based visual interfaces allow users to conceptualize programs as processes [72]. In RoboFlow [2], edits to default programs can be easily made with the assistance of a flow-based visual expression.

Display of readily distinguishable domain-specific operation units to end users has proven successful when deployed on a visual interface. The system implemented and evaluated by Senft et al. [59] only exposes the graphical representation of the *task-level* (high-level) actions to the user, which in turn allowed effective teleoperation of users for individuals with varying levels of expertise. More recently, deep learning and large-language modeling (*LLM*) methods are gaining attention for "prompt-based prototyping" [*e.g.,* 5, 35]. ChatGPT (GPT-3.5 and GPT-4 [48]) and its related work [*e.g.,* 9, 49] serve as distinct use cases where the representation format of question-answer pairs closely resemble that of interpersonal communication, borrowing dynamics of turn-taking.

2.2 End-User Development and Programming

End-user development (EUD) encompasses tools and techniques that facilitate the creation of software systems by non-programmers [42]. Crucially, Lieberman et al. [42] distinguished "design-before-use" EUD as creating software artifacts prior to their execution versus "design-during-use" as modifying existing software already in use. *End-user programming* (EUP) is a type of EUD that typically occurs at the creation phase. Although both paradigms play important roles within robotics, the focus of programming tools for human-robot interaction is often on EUP, with these tools having distinct *authoring* phases involving the initial creation of a program [1].

EUP tools capture user intent in a variety of different ways, often taking the form of traditional keyboard-and-mouse visual programming environments [*e.g.,* 2, 40, 58], demonstration [*e.g.,* 26, 34], and, more recently, *in situ* interfaces via mixed and augmented reality [*e.g.,* 15, 16]. Often, these interfaces require multimodal input from developers, such as *Figaro* [53], in which users paired spoken language statements with physical demonstrations through figurines. Due to the nature of programming, however, EUP systems often require meticulous and clear input from the user, which can be awkward for users of multimodal systems [50].

The focus of our work is to better understand how end users naturally approach programming using EUP tools. Natural input is often imprecise and rapid, a key observation of the *sloppy programming* paradigm [43]. Specifically, we focus on how end-user programmers might combine two historically popular EUP input modalities—*spoken language* and *sketching*. Spoken language has experienced widespread popularity for programming HRI systems in the collaborative [24] and service [69] domains. Sketching, too, has seen success within HRI EUP [44, 57], and has occasionally been paired with speech for robot control [22, 66]. Therefore, exploring how end users interact with these modalities within a working prototype will aid in the design of future EUD and EUP systems.

2.3 EUP Tool Usage

A critical aspect of EUP research in HRI and HCI is investigating how EUP tools could be used. Formative design studies are common

practice in EUP to investigate the potential use of tools that have not been built yet. Related to our work, Li et al. [41] investigated how a touchscreen interface can enhance spoken language, and found that multimodal input can reduce unclear or vague concepts in speech. Other work used formative studies to investigate the potential applications for which a hypothetical EUP tool might be used [20]. In addition to formative studies, Alves-Oliveira et al. [3] presented a myriad of case studies documenting how their EUP tool was used in real-world and open-ended deployments, including how end users applied the tool for robot personalization. Within the realm of general programming, Puig et al. [55] provided information on the kinds of programs users could create for robots to perform when provided with an open-ended development environment.

In our review of related EUP literature, we note that most work highlights technical over empirical contributions. Technical contributions are often still accompanied by usability measures [*e.g.,* 14, 46] and measures of whether study participants are able to meet predetermined task criteria successfully [*e.g.,* 34]. Most work that makes empirical contributions performs summative evaluations, including either quantitative scales, such as the System Usability Scale (SUS), or the Cognitive Dimensions of Notations (CDN) [*e.g.,* 13, 14], or open-ended, qualitative findings [*e.g.,* 53]. However, these empirical findings are often in service of validating the technical contributions of the work. In this work, we aim to add to the body of empirically focused EUP literature with a deeper understanding of user experience and use patterns with EUP tools and design guidelines derived from this understanding.

3 METHOD

We conducted a user study where we asked participants to use a multimodal EUP research prototype, *Tabula.*[1]

3.1 Participants

We recruited 21 individuals to participate in the study, aged 18–72 years (M = 25.19 years, SD = 11.53 years; 11 males, 10 females). While prior programming experience or exposure to robotic systems was not required, 12 participants reported previous programming experience (M = 3.92 years, SD = 2.94 years), and five of those participants also reported exposure to robotic systems ranging from using Lego robotics kits as a child to attending a Human-Computer Interaction summer school that included a robotics project. Participant backgrounds included 15 occupations or student majors from a variety of different fields which spanned science, engineering, math and statistics, medicine, and humanities.

3.2 Interface

For our study, we used an open-source, state-of-the-art research prototype tool called *Tabula* which we developed in previous work (see Porfirio et al. [52]). Tabula is a handheld EUP tool where given a 2-dimensional bird's-eye view of an environment, users can utilize multimodal speech and touch inputs to create custom robot programs [52]. The user can first optionally configure the environment by placing relevant objects (*e.g.,* toys, cabinets) with which the robot could interact. Then, the user creates the robot program by

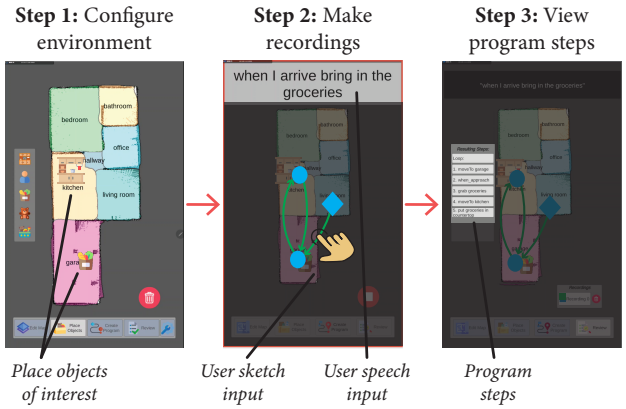

Figure 2: We used *Tabula*, a multimodal EUP tool that uses a combination of speech and sketching input to generate a robot program [52], to study end-user experiences with programming robots on the fly. *Left:* First, users configure the environment, including placing any objects for the robot to interact with. *Middle:* Second, users create recordings by first providing a speech utterance to instruct the robot what to do and subsequently creating a sketch by drawing a path of points of interest that the robot should visit. *Right:* Finally, inputs are combined by the program synthesizer, and users can view the resulting programming steps.

creating one or more *recordings*, which consist of a combination of a spoken command and a sketched path drawn on the interface. Their utterance is parsed into the core of what the robot has to achieve, including the base command (*e.g.,* put, move to) and any relevant parameters for that command (*e.g.,* objects or places within the environment). The drawn sketch includes a series of waypoints that represents locations the robot must visit during program execution. The system then contextualizes the *core* (the user's command and its parameters) within the drawn path, culminating in a program with waypoints from the sketch. Users are able to view the final program steps after creating recordings through a separate review panel. A high-level system operation is presented in Figure 2.

In the version of Tabula that we used, users do not meticulously specify *step-by-step* programs (*i.e.,* they do not specify commands and locations in the exact order to be performed), but rather supply the system with the core of the program and then contextualize that core with the sketch.[2] The command extracted from the utterance is not guaranteed to happen at any specific location, as the synthesizer will automatically decide where to place commands within the sketch. Therefore, we describe this input as *non-sequential.*

For example, the user may utter the speech "put the groceries in the kitchen" and draw a path to the garage, then to the kitchen. The system inferred that in order to 'put' the groceries, it first needs to 'grab' them. Since there are two actions, the system also infers that it should 'grab' at the first location and 'put' at the second. The system further has the constraint that the 'put' command requires a container to put the object in—in this case, putting the groceries

[1]All study materials, de-identified data, codebook, and supplementary video are available through the following OSF repository: https://osf.io/ps2fw/

[2]Tabula's implementation does not restrict users to providing the speech core and sketched path in any particular order, but its compilation to Android for this study imposes this restriction.

in the kitchen cabinets. While the utterance did not include the container parameter, the system infers which container based on a pre-configured dictionary of what containers are in different locations. The system would therefore interpret the given inputs such that the robot should travel to the garage, grab the groceries, travel to the kitchen, and put the groceries in the kitchen cabinet.

During a basic operation of Tabula, users create one recording which results in a basic robot program that accomplishes at most one goal based on the core. However, by creating multiple recordings, Tabula enables end users to specify more complex logic, *i.e.,* branching and looping. To create a branch, the user creates a second recording that starts from an existing waypoint and includes a *trigger* speech (*e.g.,* "when I arrive...") to indicate when the robot should opt to follow that branch. To create a loop, the user simply returns to a previously-visited waypoint within one recording. Specifying a loop's exit condition requires a second recording where the trigger speech indicates the desired exit condition, *e.g.,* "when I say stop..."

Tabula was selected due to its inclusion of state-of-the-art research concepts described above that have not yet been widely evaluated by end users. The key features include: (1) the combination of speech and touch input, (2) the automatic completion of an under-specified user input by searching for adequate entities to satisfy relevant preconditions, and (3) the embedding of programming logic (*e.g.,* loops) within the program to address task complexity. Porfirio et al. [52] specify that these features are intended to remove some of the users' burden in constructing comprehensive, end-to-end robot programs. However, as Porfirio et al. [52] do not include a user study that examines the usability of the system, our evaluation aims to better understand precisely how these features support end-user programming efforts.

3.3 Procedure

Participants were guided to a quiet room for the study. The experimenter briefly introduced what the study would entail and then the participants provided their informed consent before continuing. This study was reviewed and approved by the University of Wisconsin–Madison Institutional Review Board (IRB). The study consisted of the following five phases:

Tutorial & Training. Participants learned Tabula through 26 minutes of tutorial videos designed to help participants to familiarize themselves with the basic operations of the system. During the tutorial, the interface used a supermarket environment. The tutorial session was interactive, meaning the experimenter paused the video at pre-set points to prompt participants to try the examples from the tutorial. For example, participants practiced making recordings with the *speech* "say hello follow me to the sale" and a *sketch* of a path from anywhere on the map to the entrance of the store. The tutorial videos also asked participants questions to check their understanding of the key system rules. For example, the tutorial was designed deliberately to build upon previous concepts and raised questions on the difference between the new constraint and the previous constraint (*e.g.,* how does adding the new speech "if someone says yes" change how you program the robot?). Overall, the tutorial delivered the logistics of how to make a command with respect to the interface (*e.g.,* using which modality to specify a complete command) and provided examples of use cases where the system supports programming logic (*e.g.,* loops).

Structured Scenarios. Participants were then prompted to work with four different design scenarios. Participants programmed a human-robot interaction using the interface. We then asked participants to *think aloud* while completing each scenario, which allowed the experimenter to notice any hesitancy from the participant and ask clarifying questions thereafter. All three structured scenarios commonly used a home environment. We deliberately used a different environment in the scenarios versus the tutorial to observe how participants were to operate with the system, without relying on their familiarity with a specific environment. The scenarios were designed around the idea that participants were hosting a party and wanted the robot to help prepare. For each scenario, participants were briefed on the context and given an objective of what the robot program should accomplish. The objectives encouraged participants to use a variety of Tabula's features, including a robot passing an object, carrying multiple objects, and acting in regard to varying responses from the end-users described in the scenario. The comprehensive list of objectives that the participants were asked to complete is as follows.

- *Scenario 1*: The robot should put away the toy
- *Scenario 2*: The robot should bring all of the groceries from the garage to the kitchen
- *Scenario 3*: The robot should respond to guests that approach it to either show them the kitchen or the bathroom

Open-Ended Scenario. After the participants completed the three structured scenarios, the experimenter then asked the participants to come up with their own scenarios. Because these scenarios were open-ended, participants were given a choice to use either the supermarket environment or the home environment.

Interview & Questionnaires. Following the open-ended scenario, the experimenter asked participants to respond to a usability questionnaire (the System Usability Scale (SUS) [12]) based on their experience across all scenarios. In the last portion of the study, the experimenter conducted a semi-structured interview and asked participants to respond to a demographic questionnaire. The interview questions include topics such as the perceived level of system flexibility (*e.g.,* if the participants deemed the rules of the system too rigid) and the dynamic participants experienced while utilizing both speech and touch (*e.g.,* if the order of operation of the speech first and sketch second was natural for them).

3.4 Measures and Analysis

We collected the following data: 10 items from the SUS questionnaire [12], screen recordings of tablet usage during the scenarios, audio recordings of the think aloud conducted during the scenarios, and audio recordings of responses to questions during the semi-structured interviews. The think aloud and interviews were transcribed and formatted into tables for analysis. Two coders reviewed the data and decided to split the analysis of the think alouds and interviews due to the additional context required to understand the think alouds, as that dialogue is tightly linked to participants' use of the interface. For the interview transcripts, one coder developed

Table 1: A summary of the themes developed in our analysis.

Summary of Findings
General — These themes relate to user experiences which may generalize to other on-the-fly EUP tools.
Theme 1: End users viewed program steps to better understand the system End users relied heavily on viewing program steps to shape their understanding of how the system works and to look ahead to their next actions.
Theme 2: End users have poor mental model of the input paradigm End users who naturally tended toward step-by-step instructions for the robot struggled with articulating their intent non-sequentially.
Theme 3: End users felt that the robot was a tool to use End users viewed creating the robot programs as a way to utilize a robot in a tool-like manner rather than as an independent, autonomous agent.
Usability — These themes relate to specific experiences based on Tabula's implementation of on-the-fly EUP.
Theme 4: End users had mixed experiences on interaction with the program synthesizer End users either appreciated that the program synthesizer provided human-like common sense support, or they disliked the assistance because they perceived it as a loss of control over the robot program.
Theme 5: There is more to using the system than understanding its basic functionality Even after learning Tabula's basic functionality, end users still faced a learning curve to become proficient with its use.

a codebook and conducted a thematic analysis following the guidelines of Braun and Clarke [11]. For the scenario data, one coder developed a codebook for the think aloud transcripts and a list of behaviors to code in the screen recordings, *e.g.,* started a recording, made a speech input, checked the review mode, *etc.* Screen recording data was coded using BORIS [25], an open-source event recording software. The think aloud and screen recording data was then chronologically organized and subsequently analyzed for frequency of co-occurring codes within a five-second window. Across all coding, the two coders had a high inter-rater reliability (Cohen's Kappa, $\kappa = 0.83$), which indicates an "almost perfect" agreement according to interpretation guidelines from Landis and Koch [39]. We present themes that emerged through the interview data codebook as well as the patterns that emerged from the scenario data.

4 FINDINGS

Participants overall had a positive experience with the interface, with a "good" [6] mean SUS score of 69.9 ($Median = 72.5$, $SD = 11.6$). From our qualitative analysis, we developed five themes about how end users perceived and interacted with various features of the on-the-fly end-user programming (EUP) tool that they used. The themes are summarized in Table 1. Themes 1–3 illustrate experiences with on-the-fly EUP on a more general level, while Themes 4–5 pertain to *usability* aspects of Tabula's specific implementation. For each theme, we present its definition and use participant quotes to provide support. Theme 4 is further organized into subthemes to more explicitly illustrate its different facets. Participant quotes are attributed by participant ID, with minimal edits made to ensure clarity while retaining meaning.

4.1 Theme 1: End users viewed program steps to better understand the system

The first theme captures how users reflect on system rules when viewing program steps (see Figure 2, Right), initiate revisions to the program after discovering an error, and proactively utilize the program steps to make programs incrementally. Reflections shared by five participants highlight the end users' heavy reliance on step visualization when understanding system operation.

Helping in recalling system rules. Five participants recalled key system rules as they connected those rules to shaping expectations and interpreting the final output of the program steps. Participants explicitly stated concepts such as "*recordings*" (P6), "*loops*" (P6), and triggers, *e.g.,* "*adding [my] stops*" (P15). Similarly to P6's comment on recordings, P10's comment on the number of recordings provides insight on how users were able to evaluate system output as they viewed program steps and remembered key concepts. P10 mentions, "*I think that was... yeah I don't know what the third [recording]'s supposed to be*" as they viewed the program steps. For the case of recalling how to specify program logic, P6 asked a critical question "*does it loop?*" as they meticulously viewed each program step.

Initiating revisions after discovering errors. Besides the phenomenon described in the previous paragraph, there were instances where users motivated themselves to match the program steps provided by the system to the steps they imagined and desired. When misaligned, end users took the initiative to redo the entire program or wanted to make a revision after they viewed the steps. Six participants felt they wanted to "*do it again*" (P12) as they examined the final steps. P9 displayed confidence as they noticed what output of the program steps were "*obviously [...] wrong*" and expressed the urge to redo it by saying "*because I know exactly why.*" P21 expressed a related sentiment and wanted to make partial revisions, as they cited "*so I go to just delete this recording.*" Participants were able to make these reflections and express their urge to revise the program because they viewed the detailed program output.

Incremental revisions. In addition to the more common ways participants interacted with the visualized program steps, P15 also used the program steps to create programs incrementally. Deliberately checking the review panel and the detailed output of the program steps, P15 planned for the next recording after citing "*okay*

let me see what it did here this is only one of my things." Additionally, revision of the "*next instruction*" was made as they have continued citing "*is not when someone says stop but when someone says go to when someone says where is [the] kitchen.*" This observation brings us insight into the importance of including detailed step visualizations within the system, rather than solely focusing on how to capture user intent with regards to system rules.

4.2 Theme 2: End users have poor mental model of the input paradigm

This theme reflects end users' experiences with the non-sequential, rapid specification of the robot programs. Instead of requiring *step-by-step* instructions, Tabula accepts *non-sequential* input, *i.e.,* end users need not instantiate commands and locations in the exact order to be performed. With Tabula's non-sequential input, users provide a verbal task hint and the sketch on the tablet interface, and then these inputs are synthesized into a list of program steps by the system. Twelve participants indicated that the way that the interface required them to provide input was unintuitive.

Preference for step-by-step inputs. Eleven participants articulated that they would have preferred the flexibility to interchange the speech and sketching inputs, especially to support specifying programs step by step rather than non-sequentially. For example, P16 felt that first giving the speech input was "*backwards*" when specifying a task for the robot to greet patrons at the front of the store because they want to "*first get [the robot] to the entrance and then give the command.*" P14 expressed similarly that for them, it was easier to draw out the path and then think of the speech because they first need to "*invite [their] mind to imagine that place*" then provide the speech command. The comments from these participants evoke the sense that they are thinking of the robot program in a step-by-step, linear sequence of actions, which contrasts the non-sequential input paradigm used by Tabula. Other participants were more explicit about their preference for "*step wise*" (P3) inputs and found it "*difficult*" (P19) to adapt to the non-sequential pattern. P17 explains how they would have preferred to create the robot program that brings the groceries to the kitchen, saying:

> "*The robot goes to the garage, and then [I'll] tell them 'Take the groceries.' I'll put the groceries to the kitchen, and [I] draw [a path] to the kitchen.*"

Overall, these participants seemed to struggle with the misalignment between their step-by-step mental model of the robot program and the non-sequential inputs they were asked to provide.

Unintuitive to program robot remotely. One remaining participant articulated that they did not like creating a robot program when the robot was not in the same location. They expected to "*let the robot come to [them] first and then give [the robot] a task*" (P17) instead of using the tablet interface to do so remotely. While only one participant expressed such a differing model of creating programs for the robot, it highlights a different aspect to the input paradigm that was not widely explored in this study.

4.3 Theme 3: End users felt that the robot was a tool to use

End users viewed creating the robot programs as a way to utilize a robot like a tool rather than treating the robot as an independent, autonomous agent capable of reasoning about its environment. This theme is formed from seven participants' remarks, and it encapsulates a unique way in which they viewed the robot. Based on the demographic data, we see a potential relationship between experience with programming languages and how people perceived the robot—of the seven participants who reported no experience with programming languages, only one of these participants articulated the robot was a tool rather than an autonomous agent.

Learning to use the tool. Participants felt it was necessary to learn the specific rules to use Tabula because "*if you buy anything you want to use you have to read and use the manufacturer's manual to be able to understand how to use it*" (P20). This viewpoint emphasizes the robot's role as a product to purchase and use as a tool.

Prioritized the robot's capabilities over their own preference. Two participants built on the notion of learning the robot's specific rules by indicating that as they created their robot programs, they prioritized adapting their inputs based on their perception of the robot's abilities. P3 described that while "*it was easy enough for [them] to do one thing or the other,*" they opted to create robot programs based on "*whatever [they] thought it was easier to implement for the robot.*"

Need to ensure real world matches robot's world model. In a more extreme view, four participants felt they had a direct responsibility to ensure that the reality reflected the assumptions that the system made because the robot would not have the reasoning capabilities to troubleshoot deviation. P8 articulates this point clearly, saying:

> "*If [the robot] assumes that [a container is] going to be there, then it's your responsibility to make sure that [...] the containers [are] there to for the robot to put [the object] in.*"

This perspective shifts the responsibility onto the end user to ensure that the robot is able to succeed at its program, rather than expecting the robot to reason about the world autonomously.

4.4 Theme 4: End users had mixed experiences on interaction with the program synthesizer

End users either appreciated that the program synthesizer provided human-like common sense support, or they disliked the assistance because they perceived it as a loss of control over the robot program. Participants interacted with the assumptions made by the program synthesizer when it automatically inserted missing actions and objects. Twenty participants specifically commented on this aspect of the system, revealing a dichotomy of end users who either appreciate or reject the notion of the program synthesizer making automated assumptions and a small subset who had mixed perspectives. The two subthemes presented below illustrate the two prevalent, opposing viewpoints of the system. Based on the demographic data, we see the potential for experience with programming languages to impact how people perceived the interaction with the program synthesizer—of the seven participants who reported no

familiarity with any programming languages, only one participant appreciated the automated assumptions.

4.4.1 Subtheme 4a: Automated assumptions can offer support to end users. The 11 participants who spoke positively of the automated assumptions made by the program synthesizer expressed that it was "*natural*" (P12) and that it provided support during their programming experience.

Interactions were more natural/human-like. Participants specifically commented on the assumptions made about when to insert an object and when to insert actions, indicating that these assumptions offered a desirable level of human-like common sense from the robot that is "*helpful*" (P5). For example, P18 expressed that the actions automatically inserted by the program synthesizer simplified the process for them because "*the put action combined the grab and the move and stuff*" which meant that they did not need to take time to think through or add those actions–that burden was offloaded to the program synthesizer.

Desire for additional automated support. Four participants further indicated that the system could be more helpful by making additional assumptions based on user input. For example, P4 wished that the system would automatically generate a condition for "*exiting the loop,*" while P13 wanted the system to "*provide suggestions*" if the user made a mistake. P11 further envisioned that the system could make assumptions based on the robot's ability to interact with objects that it is close to in its environment, such as "*if you move [the robot] to the item, [the system] just infers that [the robot]'s supposed to pick it up.*" The automatic assumptions provided convenience to some end users, who felt support from the system for easing into the robot programming process.

4.4.2 Subtheme 4b: Automated assumptions can lead to loss of control. The 13 participants who commented negatively about the use of automated assumptions felt that these assumptions led to a loss of their ability to control how the robot would act. Eight of these participants focused comments on the automatic insertion of objects and items, while the remaining 5 participants expressed the desire for more control over the robot's precise movements within its environment, such as indicating specific regions to avoid.

Doubting robot's knowledge to automatically insert objects/actions. Eight participants focusing on the automatic insertion of objects and actions felt that it was "*unnatural*" (P2) and questioned whether the robot could or should have enough knowledge of the environment to make such assertions. For example, P2 felt that depending on the scenario, the user may or may not intend for an object to be placed inside of a container. P2 explains:

> "*Given certain use cases, I could imagine like if you have one of these robots moving gravel around a yard, you probably wouldn't have a container there, but uh in [the grocery delivery] scenario it felt right to assume that there would be a cabinet.*"

From P2's example, it may be difficult to infer when an object should be placed in a container or not. P6 similarly felt that it was reasonable for the end user to have to explicitly specify whether there is a container, saying "*It just makes sense if I have to tell it that there's a teddy bear in the middle of the floor that I should also have to tell it that there's a cabinet on the wall.*" P19 echoes the sentiment that they "*don't know exactly what [the robot]'s going to assume to do and especially with the assuming where it's going to put.*" Overall, participants who did not like the automatic insertion of objects and actions felt that they did not have as much control or understanding over how the system would behave.

Desire to control robot's location. Five participants viewed the abstraction of the environment into regions as opposed to exact coordinates as a negative assumption of the system—they wanted more control over the precise location or path the robot would travel within the space. The current system abstracted away precise coordinates in favor of general semantic regions such as "*kitchen*" or "*living room.*" P20 explains their desire, using the example that the kitchen is a "*big place [...] so maybe [in] the command there should be a way to specify where exactly in the kitchen you want the groceries to be placed.*" Building off of this sentiment, P5 and P13 both expressed that they may want the robot to "*avoid a certain area*" (P5), so the path that they draw for the robot is the precise one that it should follow. This group of participants includes the four participants who also spoke favorably about the automatic assumptions regarding actions and objects in Subtheme 4a, which indicates that there is a need to create a balance between easing the programming process and giving the users the desired level of precision over the robot's behaviors.

4.5 Theme 5: There is more to using the system than understanding its basic functionality

Even after understanding Tabula's basic functionality, end users still faced a learning curve to become proficient.

Translating rules into use. Eleven participants expressed that there were "*differences between understanding and doing*" (P12). P10 articulated that rules for creating robot programs led to instances where "*you have something in your mind but you don't know how to immediately put it in the system.*" This "*gap*" (P12) forced P15 to resort to "*taking different parts of the training and kind of consolidating it into doing a scenario.*"

Performance aspect to making recordings. In addition to conceptual difficulties with "*connecting the dots*" (P15) between various concepts, participants also noted that making the recordings created "*a performance aspect [...] to get it all in one go*" (P2). Once participants began a recording, they had to "*remember the vocabulary that the robot would understand*" (P10). If they made a mistake or if the system "*had a hard time*" (P4) discerning what participants said, then they had to delete the recording and start again.

Desire for editing support. While some participants seemed comfortable with the iterative process of creating, reviewing, deleting, and re-doing recordings, others wanted a different way to correct mistakes. Four participants wanted the ability to "*edit a recording afterwards*" (P7), which would ease the pressure of providing precisely correct speech and touch on the first attempt. Two participants wanted a quick way to "*erase if you messed up*" (P11) during a recording without having to "*restart*" (P8) the whole program.

5 DISCUSSION

We sought to better understand end user experience with on-the-fly robot programming through an in-depth assessment of the open-source EUP tool *Tabula*. Through our investigation, we uncovered themes that provide insight into various aspects of on-the-fly robot EUP. Some themes relate specifically to the implementation of Tabula, such as its use of multimodal inputs and the use of automated assistance in the form of a program synthesizer. However, combined with prior work, other themes point to broader implications regarding the concepts realized through Tabula, such as the reliance on the visualized program steps. Overall, we see the promise of on-the-fly EUP tools as a way to facilitate the use of robots to aid with day-to-day tasks, but these tools require further research and refinement before they will be sufficient. We encourage future researchers to conduct more in-depth user studies with existing or novel EUP tools so that we can build a better understanding of end user needs based on a variety of on-the-fly EUP tools.

In the paragraphs below, we provide general points of discussion of our findings, such as how participants perceived the role of the interface and how our findings relate to the Cognitive Dimensions of Notation (CDN) [31]. We reserve detailed discussion of implications for future design of EUP systems for §5.1, highlighting four key design recommendations.

Role of the interface. We note that participants had different assumptions about the role of the interface, with some thinking that its capabilities were limited to capturing user input on behalf of the robot and others attributing planning and reasoning capabilities to the interface itself. Specifically, some participants felt that the robot was directly generating the sequence of steps. For example, in Theme 3, P3 discussed "*If [the robot] assumes that [a container is] going to be there...*"), which attributes the automatic assumption made to the robot rather than to the interface's underlying synthesizer. Also in Theme 4-Subtheme 4b, P2 spoke as though the *robot* was assuming that there was a cabinet to place the groceries within the environment. Therefore, we believe that some participants attributed the automated decision making to the robot's autonomy rather than the features of the interface.

We find this connection particularly interesting, considering that there was no physical robot present during the study. The distinction between the EUP tool and the robot's autonomy is blurry, especially because the EUP tool may depend on specific robot capabilities. Given that participants did not necessarily separate the EUP tool from the robot's capabilities, we can consider the automated decision making of Tabula in close alignment with the autonomy of the robot for which it was being used to generate programs.

Relation to Cognitive Dimensions of Notation (CDN). CDN is a set of 14 design principles intended for evaluating programming languages, notations, and user interfaces [31]. Each principle illustrates one aspect of usability, intending to serve as a guide toward improving usability along specific dimensions. We found that two dimensions align particularly well with certain aspects of our themes, indicating that these dimensions are key to future robot EUP tools. While CDN is helpful in contextualizing usability aspects of Tabula, CDN does not explicitly discuss autonomy or perceptions surrounding interaction with automated decision making.

The first dimension, *progressive evaluation*, considers how easily users can evaluate and obtain feedback. This dimension connects well to Theme 1. Participants largely relied on the generated list of program steps as the mechanism for receiving feedback and updating their solutions accordingly, indicating that supporting progressive evaluation is critical.

The second dimension, *premature commitment*, considers both how strong the constraints are on using the system and also how users can easily change or correct decisions later on. With Theme 2, participants felt constrained by the strict order of speech and sketching inputs. Given that participants had varying notions on the best input order, avoiding premature commitment by providing more flexibility in speech and sketch inputs is crucial. Theme 5 also supports the need for premature commitment as participants wanted a way to edit the recordings after the fact instead of having to delete and re-do them.

5.1 Design Implications

Based on the themes discussed in §4, we present design implications and recommendations to inform future design of on-the-fly EUP tools. Each recommendation includes a general recommendation of how the implication could be applied generally to EUP tools, as well as a specific suggestion for modifications which would lead to "Tabula 2.0." The link between the findings, implications, and recommendations is visualized in Figure 3.

5.1.1 Design Implication 1: Feedback is critical for the successful use of on-the-fly EUP systems. Our findings highlight the importance of integrating feedback mechanisms within on-the-fly EUP systems like Tabula—in contrast to tools in which program flow is explicitly embedded within user input (*e.g.,* block-based programming tools [10, 20, 21]), Tabula users rely heavily on feedback (*i.e.,* program step visualizations) to understand system behavior and make program changes (Theme 1). Even without access to a way to deploy their programs to a simulated or physical robot, participants were still able to use the step visualizations as a pre-deployment check as a way to understand where an error occurred and how they could adjust the program flow to correct it.

The reliance on feedback echoes prior investigations of end-user developers interacting with a program synthesizer [50]. *How* feedback is applied is additionally crucial to human-AI systems in general [4], and our results suggest that purely *descriptive* (as opposed to *prescriptive* or *explanatory*) feedback can lead to a lengthy process of discovering system behavior. Specifically, because participants were only provided with the resulting program steps (descriptive feedback), they had to use their own judgment to discern if their program was correct and guess how to modify their inputs to Tabula in order to achieve the desired program output (Theme 1). In Theme 4-Subtheme 4a, participants specifically expressed that the system could provide additional automated support to further ease their programming efforts. This additional support included prescriptive measures such as the system detecting mistakes and offering corrective suggestions and preemptive measures such as the system generating conditions on their behalf (P4 and P13).

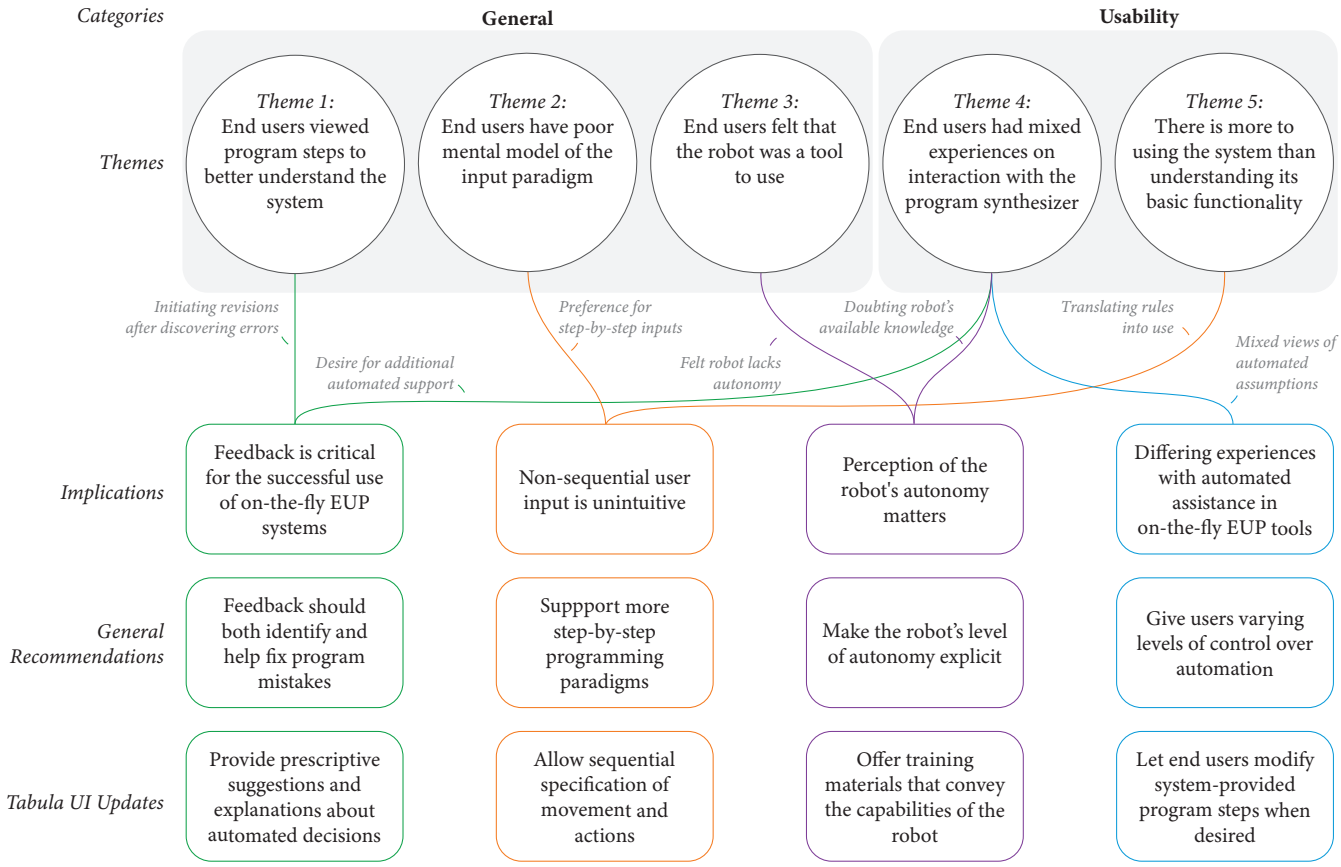

Figure 3: An overview of the connection between the findings and resulting design implications and recommendations.

Recommendation 1: For on-the-fly EUD tools, visual feedback should provide users not only with information on what is wrong with a program, but also with information on how to fix it or explanations for why the system behaved in a certain way. On-the-fly EUD should therefore draw from prescriptive approaches in formal methods, such as proposing repairs [19], and strengthen its descriptive approach through explainable AI techniques such as model reconciliation [18]. Approaches such as these could offer end users the ability to assess incomplete solutions, obtain feedback, and build programs based on the interface's suggestions.

Specifically, in realizing a "Tabula 2.0," we can clearly label steps provided directly by the end user and steps generated by the system. Then, the end user could select steps and ask a question such as "Why is this step before that step?" Using methods such as iterative planning as outlined by Smith [61] or Wang et al. [70], the system can interactively offer a rationale behind the decision and suggest new constraints to add or modifications to existing recordings which would alter the resulting steps.

5.1.2 Design Implication 2: Non-sequential user input is unintuitive. We found across Themes 2 and 5 that participants struggled with the rules and input paradigms that Tabula enforces. Participants had to provide input following a strict pattern and adhere to usage rules, which they expressed resulted in feelings of frustration because they could not easily use the interface to express their intention. For instance, participants commented on the "*backwards*" input paradigm that Tabula enforced and conveyed their preference for "*step-wise*" inputs (P3, P14, and P16). This study included a fairly extensive tutorial which included many interactive examples, yet it is evident that additional training would be required for participants to achieve proficiency. The underlying representations of user input and resulting program steps appear to be critical to Tabula users' experience, a finding that aligns with prior work of user program comprehension—certain program representations may align better (*i.e.,* representations that facilitate forward-reasoning [67]) or worse (*i.e.,* the imperative programming paradigm [36]) with user intuition. Other representations are prone to misalignment between user mental models of program behavior (*i.e.,* trigger-action programming [33]) or may result in reduced user performance (*i.e.,* visualizations of data flow rather than control flow [29]). Fortunately, motivated by prior work that uses formative evaluation to inform product design [*e.g.,* 41], we believe that changes to the interface can improve user experience with non-sequential input, such as through the inclusion of the ability to edit recordings after they are created. Therefore, it will be important to balance efforts to create intuitive tools for end users with developing effective training protocols for introducing new paradigms and systems.

Recommendation 2: Find a way to design on-the-fly EUP tools that supports more step-by-step programming paradigms. Training to use on-the-fly EUP tools to create robot programs should remain essential, even if the training eventually becomes teaching end users about a robot's capabilities and limitations. However, interfaces can always be designed to be more intuitive, *e.g.,* by supporting more step-by-step paradigms where users can specify movement and actions sequentially, through methods such as participatory design and research through design.

In realizing "Tabula 2.0," we would remove the restriction imposed during our study of speech needing to occur before the sketch, update the synthesizer to allow end users to link utterances to specific waypoints, and add further support to accommodate multiple, separate speech utterances per recording. In making the above modifications, end users will have more flexibility and control to be able to specify step by step what the robot should do at which location. The result will be a system which would allow participants to interleave sketching and speech, similarly to how tools like *Figaro* [53] allow more sequential specification of movement and actions. Unlike Figaro, however, the system would still insert or complete missing or incomplete specifications.

5.1.3 Design Implication 3: Perception of the robot's autonomy can either limit or enhance the role of the robot as a collaborative entity. Theme 3 and Theme 4-Subtheme 4b together illustrated that a subset of participants perceived that the robot was not necessarily able to reason about its world. Participants from Theme 4-Subtheme 4b expressed this view through distrust of the automated assumptions of the program synthesizer (*e.g.,* P2 considered the automated assumptions to be "*unnatural*"), whereas participants in Theme 3 felt that the robot was merely a tool to use (*e.g.,* P20 felt there would have to be a "*manufacturer's manual*" such as the instruction booklets that come with other household tools). Misperceptions of robot capability [17] and the potential to view the robot as a "*tool*" (rather than having agency) [65] are known phenomena in human-robot interaction. Our interviews not only suggest that these phenomena translate to EUP, but also that, in participants' words, user perception of the robot's autonomy changes their behaviors and experiences. We further saw that in both of these themes, the participant's prior familiarity with programming languages may have been a factor impacting their current perception of the robot's autonomy. Given that a robot's level of autonomy may be set, it is important to think about how to communicate the robot's level of autonomy and precise role to the end user.

Recommendation 3: When designing an EUP tool, the robot's level of autonomy should be made explicit. Tools designed for autonomous robots who can reason about their world may differ from tools designed for using robots to extend human abilities. Tabula was designed with the intention that the system/robot could reason about the world, such as understanding when it may need to automatically insert steps or assume that certain objects would be present (*e.g.,* assuming the kitchen cabinets are there to put the groceries in). However, as some participants did not appreciate this level of autonomy of the system, they desired more low-level control over specifying exactly what to do at which locations. Future research should explore these differences, such as by investigating ways in which different levels of autonomy and agency can be communicated to end users. While it is not necessarily the case that each specific robot with varying autonomy levels would require a different EUP tool, different robot characteristics will likely indicate the need for more specific EUP tool features. Incorporating a conceptual framework such as the robot autonomy scale (see Beer et al. [8]) could create more transparency with regard to how much automated support is provided to the end user, and such integration will therefore be a necessary step for future EUP tool design.

For "Tabula 2.0," we can clearly situate Tabula's level of autonomy within the scale of robot autonomy [8] as *sharing control* (*e.g.,* the synthesizer can automatically complete the user's commands while the user has control on which commands to instantiate). We can specifically communicate the sensing, planning, and acting capabilities of the robot that is connected with Tabula through training materials that exhibit specific use cases of the synthesizer. Within the training materials, we will emphasize the exact capabilities the robot has, along with the extent to which the synthesizer makes assumptions about the user's intent. This training will be particularly critical for those who are not familiar with robots, although as familiarity with robots increases, the need for in-depth training will likely taper.

5.1.4 Design Implication 4: As demonstrated with Tabula, user experience with perceptions of automated assistance varies with on-the-fly EUP tools. Particularly with Theme 4, we observed that end users were split between appreciating the support of the automated assistance provided by the program synthesizer and wishing that they had more control over the programs generated. A handful of users expressed mixed opinions. These varying preferences may be due to our participant pool including a diverse background of programming, video game, and engineering experience. We note that robots in the home will similarly be used by individuals with varying backgrounds. Prior work in robot EUP acknowledges the need to cater to varying backgrounds by providing entry points for different types of developers [27, 34, 54]; our work suggests that in addition to providing multiple entry points, on-the-fly EUP tools will need to cater to a sliding scale of preferences regarding the level of user control versus automated assistance.

Recommendation 4: When incorporating automation, also give users varying levels of control. Ideally, anything that is handled by some form of automated assistance should also be directly controllable and/or modifiable for the end user. However, in reality, due to Tabula's non-sequential multimodal inputs, all aspects of its automated assistance may not be fully customizable. Therefore it is critical that EUP tools clearly communicate explanations behind their decisions and offer guidance to support end users in creating the programs that they desire.

In continuation of the discussion on robot autonomy and end-user control, "Tabula 2.0" can further be designed to allow end-users to specify their preferences with regard to the program synthesizer. Users should be able to adjust the level of control wielded by the synthesizer, in particular the degree to which the synthesizer involves the human in the loop. For instance, users who opt for high level of control and low level of robot autonomy should be able to prioritize or re-arrange the program steps suggested by the synthesizer.

5.2 Limitations and Future Work

Our work has a number of limitations that point to future work. We separate our limitations and future work into three categories.

Generalization to EUP. First, our investigation of end-user experience with on-the-fly robot EUP used one existing multimodal EUP tool with specific capabilities and constraints. Combined with the fact the on-the-fly development paradigm remains novel for EUP tools within HRI, the extent to which the behaviors we have observed and the perceptions we have documented will generalize to existing EUP tools is unclear. With that being said, the purpose of our investigation is less to understand existing EUP tools, but moreso to uncover guidelines for designing EUP tools within this novel paradigm. Future work must therefore apply our recommendations to the design and evaluation of a "Tabula 2.0." Future work must also extend our investigation to additional tools that represent different EUP approaches, including input methods, intent inference, and program generation methods so that we can understand what is unique about the multi-modal on-the-fly paradigm and what generalizes to robot EUP as a whole.

Learning Tabula. Second, although the EUP paradigm is intended to be accessible to non-experts in programming and robotics, effectively using any complex end-user tool requires learning and gaining comfort with its use, which put limits on how long our participants could explore the tool as well as the time needed to generate programs. Future work should include multi-day field studies with Tabula to investigate how real users learn to use, gain familiarity with, and generate or refine programs using on-the-fly EUP tools, similarly to the approach taken by Ranganeni et al. [56]. Additionally, recent work in tablet-based EUP suggests that end-user perceptions, experience, and success using EUP tools is tied to individual background [51]. Future work with Tabula should therefore investigate possible links between relevant user background characteristics and tool usage in order to better inform strategies for training Tabula end users.

Involvement of a Robot. Third, participants used the EUP tool in a sandbox and therefore did not see their programs being executed on an actual robot. Their programs were only represented on the EUP tool. The ability to see the robot behaviors that result from their programs might provide participants with stronger mental models of the capabilities and limitations of the robot platform they are working with, which might inform their programming choices.

Exploration of Specific Application Domains. Finally, our evaluation engaged the general population of our campus community. While this population could represent home robot users, it does not represent specific domain experts such as shop keepers or healthcare workers. Each domain may have its own needs which dictate how end users would perceive Tabula. We are currently exploring applications of on-the-fly EUP in the context of care assistance to address caregivers' extremely variable and often hectic workflows, motivated by our past work on needs of caregivers [62]. Further exploration into different domains, such as by evaluating a "Tabula 2.0" with specific domain experts, will lead to a better understanding of when and how multi-modal on-the-fly EUP is best utilized.

6 CONCLUSION

This work contributes to a growing body of robot end-user programming (EUP) research by investigating end user experience with on-the-fly robot EUP. Using an open-source multimodal EUP prototype, we asked participants to create robot programs for structured and open-ended scenarios, then we interviewed them about their experiences. Our findings, which consist of five themes, illustrate user experiences which may generalize to other on-the-fly EUP tools, as well specific experiences based on Tabula's implementation. Contextualizing our findings within prior work, we develop design implications that can offer insights to inform the future design of on-the-fly EUP tools for robots.

The evaluation of Tabula presented in this paper contributes to the larger cause of democratization of robots. As robots are becoming more prevalent in day-to-day life, it is even more critical for non-roboticists and non-programmers to be able to effectively utilize them. Domain experts in fields such as healthcare, education, hospitality, and service are primed to receive assistance from robots, and yet we do not yet have the intuitive, natural interfaces necessary for them to use emerging robotics tools and platforms. This work is a foundational step toward these interfaces as it is an initial exploration into a promising new EUP paradigm. Over time, as more novel interfaces are built and paradigms are evaluated within the research and industry communities, a pool of knowledge will accumulate to help future designers understand what kind of interface to create for a specific robot application and the user population. These tools will ultimately help more people engage with robots to create solutions that address their unique needs.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation (NSF) award IIS-1925043 and an NSF Graduate Research Fellowship under Grant No. DGE-1747503. DP's contributions occurred while supported as an NRC Postdoctoral Research Associate at the U.S. Naval Research Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or the U.S. Navy.

REFERENCES

[1] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. 2021. A survey on end-user robot programming. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36. https://doi.org/10.1145/3466819

[2] Sonya Alexandrova, Zachary Tatlock, and Maya Cakmak. 2015. RoboFlow: A flow-based visual programming language for mobile manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5537–5544. https://doi.org/10.1109/ICRA.2015.7139973

[3] Patricia Alves-Oliveira, Kai Mihata, Raida Karim, Elin A Bjorling, and Maya Cakmak. 2022. FLEX-SDK: An Open-Source Software Development Kit for Creating Social Robots. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–10. https://doi.org/10.1145/3526113.3545707

[4] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing systems*. 1–13. https://doi.org/10.1145/3290605.3300233

[5] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena Glassman. 2023. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. *arXiv preprint arXiv:2309.09128* (2023). https://doi.org/10.48550/arXiv.2309.09128

[6] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies* 4, 3 (may 2009), 114–123. https://doi.org/10.5555/2835587.2835589

[7] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. 2019. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software* 149 (2019), 101–137. https://doi.org/10.1016/j.jss.2018.11.041

[8] Jenay M Beer, Arthur D Fisk, and Wendy A Rogers. 2014. Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of human-robot interaction* 3, 2 (2014), 74. https://doi.org/10.5898/JHRI.3.2.Beer

[9] William L Benzon. 2023. Discursive Competence in ChatGPT, Part 1: Talking with Dragons. (2023). https://doi.org/10.2139/ssrn.4318832

[10] Sara Beschi, Daniela Fogli, and Fabio Tampalini. 2019. CAPIRCI: a multi-modal system for collaborative robot programming. In *End-User Development: 7th International Symposium, IS-EUD 2019, Hatfield, UK, July 10–12, 2019, Proceedings 7*. Springer, 51–66. https://doi.org/10.1007/978-3-030-24781-2_4

[11] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[12] John Brooke. 1996. SUS: A 'Quick and Dirty' Usability Scale. *Usability Evaluation in Industry* 189, 3 (1996), 189–194. https://doi.org/10.1201/9781498710411

[13] Nina Buchina, Sherin Kamel, and Emilia Barakova. 2016. Design and evaluation of an end-user friendly tool for robot programming. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 185–191. https://doi.org/10.1109/ROMAN.2016.7745109

[14] Nina G Buchina, Paula Sterkenburg, Tino Lourens, and Emilia I Barakova. 2019. Natural language interface for programming sensory-enabled scenarios for human-robot interaction. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1–8. https://doi.org/10.1109/RO-MAN46459.2019.8956248

[15] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 521–534. https://doi.org/10.1145/3332165.3347902

[16] Yuanzhi Cao, Zhuangying Xu, Fan Li, Wentao Zhong, Ke Huo, and Karthik Ramani. 2019. V. ra: An in-situ visual authoring system for robot-iot task planning with augmented reality. In *Proceedings of the 2019 on designing interactive systems conference*. 1059–1070. https://doi.org/10.1145/3322276.3322278

[17] Elizabeth Cha, Anca D Dragan, and Siddhartha S Srinivasa. 2015. Perceived robot capability. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 541–548. https://doi.org/10.1109/ROMAN.2015.7333656

[18] Tathagata Chakraborti, Sarath Sreedharan, Sachin Grover, and Subbarao Kambhampati. 2019. Plan explanations as model reconciliation–an empirical study. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 258–266. https://doi.org/10.1109/hri.2019.8673193

[19] Michael Jae-Yoon Chung and Maya Cakmak. 2020. Iterative Repair of Social Robot Programs from Implicit User Feedback via Bayesian Inference. In *Proceedings of Robotics: Science and Systems*. Corvalis, Oregon, USA. https://doi.org/10.15607/RSS.2020.XVI.028

[20] Michael Jae-Yoon Chung, Justin Huang, Leila Takayama, Tessa Lau, and Maya Cakmak. 2016. Iterative design of a system for programming socially interactive service robots. In *Social Robotics: 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016 Proceedings 8*. Springer, 919–929. https://doi.org/10.1007/978-3-319-47437-3_90

[21] Enrique Coronado, Dominique Deuff, Pamela Carreno-Medrano, Leimin Tian, Dana Kulić, Shanti Sumartojo, Fulvio Mastrogiovanni, and Gentiane Venture. 2021. Towards a modular and distributed end-user development framework for human-robot interaction. *IEEE Access* 9 (2021), 12675–12692. https://doi.org/10.1109/ACCESS.2021.3051605

[22] Andrew Correa, Matthew R Walter, Luke Fletcher, Jim Glass, Seth Teller, and Randall Davis. 2010. Multimodal interaction with an autonomous forklift. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 243–250. https://doi.org/10.1109/HRI.2010.5453188

[23] 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*. 2109–2114. https://doi.org/10.1145/2702613.2732795

[24] Maxwell Forbes, Rajesh PN Rao, Luke Zettlemoyer, and Maya Cakmak. 2015. Robot programming by demonstration with situated spatial language understanding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014–2020. https://doi.org/10.1109/ICRA.2015.7139462

[25] Olivier Friard and Marco Gamba. 2016. BORIS: a free, versatile open-source event-logging software for video/audio coding and live observations. *Methods in ecology and evolution* 7, 11 (2016), 1325–1330. https://doi.org/10.1111/2041-210X.12584

[26] Yuxiang Gao and Chien-Ming Huang. 2019. PATI: a projection-based augmented table-top interface for robot programming. In *Proceedings of the 24th international conference on intelligent user interfaces*. 345–355. https://doi.org/10.1145/3301275.3302326

[27] Dylan Glas, Satoru Satake, Takayuki Kanda, and Norihiro Hagita. 2011. An Interaction Design Framework for Social Robots. In *Proceedings of Robotics: Science and Systems*. Los Angeles, CA, USA. https://doi.org/10.15607/RSS.2011.VII.014

[28] Hassan Gomaa and Douglas BH Scott. 1981. Prototyping as a tool in the specification of user requirements. In *Proceedings of the 5th international conference on Software engineering*. 333–342. https://dl.acm.org/doi/10.5555/800078.802546

[29] Judith Good. 1999. VPLs and novice program comprehension: How do different languages compare?. In *Proceedings 1999 IEEE Symposium on Visual Languages*. IEEE, 262–269. https://doi.org/10.1109/VL.1999.795912

[30] Javi F Gorostiza and Miguel A Salichs. 2011. End-user programming of a social robot by dialog. *Robotics and Autonomous Systems* 59, 12 (2011), 1102–1114. https://doi.org/10.1016/j.robot.2011.07.009

[31] Thomas R. G. Green and Marian Petre. 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131–174. https://doi.org/10.1006/jvlc.1996.0009

[32] Gaoping Huang, Pawan S Rao, Meng-Han Wu, Xun Qian, Shimon Y Nof, Karthik Ramani, and Alexander J Quinn. 2020. Vipo: Spatial-visual programming with functions for robot-IoT workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3313831.3376670

[33] 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* (Osaka, Japan) *(UbiComp '15)*. Association for Computing Machinery, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

[34] Justin Huang and Maya Cakmak. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. 453–462. https://doi.org/10.1145/2909824.3020215

[35] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. Promptmaker: Prompt-based prototyping with large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–8. https://doi.org/10.1145/3491101.3503564

[36] Rajeswari Hita Kambhamettu, Michael Jae-Yoon Chung, Vinitha Ranganeni, and Patrícia Alves-Oliveira. 2021. Collecting Insights about How Novice Programmers Naturally Express Programs for Robots. Plateau Workshop. https://doi.org/10.1184/R1/19799197.v1

[37] Neil W Kassel and Brian A Malloy. 2003. An approach to automate requirements elicitation and specification. In *Proc. of the 7th Int. Conf. on Software Engineering and Applications*. Citeseer, 3–5.

[38] Amy J Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, et al. 2011. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)* 43, 3 (2011), 1–44. https://doi.org/10.1145/1922649.1922658

[39] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174. https://doi.org/10.2307/2529310

[40] Nicola Leonardi, Marco Manca, Fabio Paternò, and Carmen Santoro. 2019. Trigger-action programming for personalising humanoid robot behaviour. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3290605.3300675

[41] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 577–589. https://doi.org/10.1145/3332165.3347899

[42] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-user development: An emerging paradigm. In *End user development*. Springer, 1–8. https://doi.org/10.1007/1-4020-5386-X_1

[43] Greg Little, Robert C Miller, Victoria H Chou, Michael Bernstein, Tessa Lau, and Allen Cypher. 2010. Sloppy programming. In *No Code Required*. Elsevier, 289–307. https://doi.org/10.1016/B978-0-12-381541-5.00015-8

[44] Kexi Liu, Daisuke Sakamoto, Masahiko Inami, and Takeo Igarashi. 2011. Roboshop: multi-layered sketching interface for robot housework assignment and management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 647–656. https://doi.org/10.1145/1978942.1979035

[45] Fei Lu, Xinran Wang, and Guohui Tian. 2012. The structure and application of intelligent space system oriented to home service robot. In *2012 IEEE International Conference on Information and Automation*. 289–294. https://doi.org/10.1109/ICInfA.2012.6246820

[46] Gabriella Lucci and Fabio Paternò. 2014. Understanding end-user development of context-dependent applications in smartphones. In *Human-Centered Software Engineering: 5th IFIP WG 13.2 International Conference, HCSE 2014, Paderborn, Germany, September 16-18, 2014. Proceedings 5*. Springer, 182–198. https://doi.org/10.1007/978-3-662-44811-3_11

[47] Matt MacLaurin. 2009. Kodu: end-user programming and design for games. In *Proceedings of the 4th international conference on foundations of digital games*. xviii–xix. https://doi.org/10.1145/1536513.1536516

[48] OpenAI. 2023. GPT-4 Technical Report. https://doi.org/10.48550/arXiv.2303.08774 arXiv:2303.08774 [cs.CL]

[49] Lidiia Ostyakova, Kseniia PetukhovaO, Veronika Smilga, and Dilyara ZharikovaO. 2023. Linguistic Annotation Generation with ChatGPT: a Synthetic Dataset of Speech Functions for Discourse Annotation of Casual Conversations. In *Proceedings of the International Conference "Dialogue*, Vol. 2023. https://doi.org/10.28995/2075-7182-2023-22-386-403

[50] David Porfirio, Evan Fisher, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2019. Bodystorming human-robot interactions. In *proceedings of the 32nd annual ACM symposium on user Interface software and technology*. 479–491. https://doi.org/10.1145/3332165.3347957

[51] David Porfirio, Mark Roberts, and Laura M. Hiatt. 2024. Goal-Oriented End-User Programming of Robots. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction* (Boulder, CO, USA) *(HRI '24)*. Association for Computing Machinery, New York, NY, USA, 582–591. https://doi.org/10.1145/3610977.3634974

[52] David Porfirio, Laura Stegner, Maya Cakmak, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2023. Sketching Robot Programs On the Fly. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. 584–593. https://doi.org/10.1145/3568162.3576991

[53] David J Porfirio, Laura Stegner, Maya Cakmak, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2021. Figaro: A tabletop authoring environment for human-robot interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15. https://doi.org/10.1145/3411764.3446864

[54] Emmanuel Pot, Jérôme Monceaux, Rodolphe Gelin, and Bruno Maisonnier. 2009. Choregraphe: a graphical tool for humanoid robot programming. In *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 46–51. https://doi.org/10.1109/ROMAN.2009.5326209

[55] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8494–8502. https://doi.org/10.48550/arXiv.1806.07011

[56] Vinitha Ranganeni, Vy Nguyen, Henry Evans, Jane Evans, Julian Mehu, Samuel Olatunji, Wendy Rogers, Aaron Edsinger, Charles Kemp, and Maya Cakmak. 2024. Robots for Humanity: In-Home Deployment of Stretch RE2. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 1299–1301. https://doi.org/10.1145/3610978.3641114

[57] Daisuke Sakamoto, Koichiro Honda, Masahiko Inami, and Takeo Igarashi. 2009. Sketch and run: a stroke-based interface for home robots. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 197–200. https://doi.org/10.1145/1518701.1518733

[58] Andrew Schoen, Nathan White, Curt Henrichs, Amanda Siebert-Evenstone, David Shaffer, and Bilge Mutlu. 2022. CoFrame: A System for Training Novice Cabot Programmers. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 185–194. https://doi.org/10.1109/HRI53351.2022.9889345

[59] Emmanuel Senft, Michael Hagenow, Kevin Welsh, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021. Task-level authoring for remote robot teleoperation. *Frontiers in Robotics and AI* 8 (2021), 707149. https://doi.org/10.3389/frobt.2021.707149

[60] Emmanuel Senft, Satoru Satake, and Takayuki Kanda. 2020. Would You Mind Me if I Pass by You? Socially-Appropriate Behaviour for an Omni-based Social Robot in Narrow Environment. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. 539–547. https://doi.org/10.1145/3319502.3374812

[61] David Smith. 2012. Planning as an iterative process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 26. 2180–2185. https://doi.org/10.1609/aaai.v26i1.8449

[62] Laura Stegner and Bilge Mutlu. 2022. Designing for Caregiving: Integrating Robotic Assistance in Senior Living Communities. In *Designing Interactive Systems Conference* (Virtual Event, Australia) *(DIS '22)*. Association for Computing Machinery, New York, NY, USA, 1934–1947. https://doi.org/10.1145/3532106.3533536

[63] Laura Stegner, Emmanuel Senft, and Bilge Mutlu. 2023. Situated participatory design: A method for in situ design of robotic interaction with older adults. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15. https://doi.org/10.1145/3544548.3580893

[64] Maj Stenmark, Mathias Haage, and Elin Anna Topp. 2017. Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. 463–472. https://doi.org/10.1145/2909824.3020227

[65] Leila Takayama. 2012. Perspectives on agency interacting with and through personal robots. In *Human-computer interaction: the agency perspective*. Springer, 195–214. https://doi.org/10.1007/978-3-642-25691-2_8

[66] Seth Teller, Matthew R Walter, Matthew Antone, Andrew Correa, Randall Davis, Luke Fletcher, Emilio Frazzoli, Jim Glass, Jonathan P How, Albert S Huang, et al. 2010. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 526–533. https://doi.org/10.1109/ROBOT.2010.5509238

[67] John Gregory Trafton and Brian J Reiser. 1991. Providing natural representations to facilitate novices' understanding in a new domain: Forward and backward reasoning in programming. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates, Inc., 923–927.

[68] Jim Van Buren and David Cook. 1998. Experiences in the adoption of requirements engineering technologies. *Crosstalk-The Journal of Defense Software Engineering* 11, 12 (1998), 3–10.

[69] Nick Walker, Yu-Tang Peng, and Maya Cakmak. 2019. Neural semantic parsing with anonymization for command understanding in general-purpose service robots. In *Robot World Cup*. Springer, 337–350. https://doi.org/10.1007/978-3-030-35699-6_26

[70] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian (Shawn) Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. 36 (2023), 34153–34189. https://proceedings.neurips.cc/paper_files/paper/2023/file/6b8dfb8c0c12e6fafc6c256cb08a5ca7-Paper-Conference.pdf

[71] Jeffrey Wong and Jason I Hong. 2007. Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1435–1444. https://doi.org/10.1145/1240624.1240842

[72] Bahram Zarrin and Hubert Baumeister. 2015. Towards separation of concerns in flow-based programming. In *Companion Proceedings of the 14th International Conference on Modularity*. 58–63. https://doi.org/10.1145/2735386.2736752