Collecting Insights into How Novice Programmers Naturally Express Programs for Robots

Rajeswari Hita Kambhamettu[1], Michael Jae-Yoon Chung[2], Vinitha Ranganeni[3], and Patrícia Alves-Oliveira[3]

***Abstract*— End-user programming of robots holds promise to help users lacking specialized software development skills customize robot behaviors. To enable such novice control, it is essential to design an end-user robot programming system that best supports how users naturally express what they want the robots to do. In this paper, we conducted a two-part online user study with fifteen participants to analyze how novice programmers intuitively solve robot-related tasks (part 1) vs how they code robot behaviors using a static version of Cozmo Codelab, a visual block-based robot programming system (part 2).**

Our part 1 results show that novice programmers do not intuitively consider programming tasks using an imperative procedural programming paradigm, which is typical of current visual block-based robot programming systems. Additionally, we find that users who initially adopted an imperative procedural programming paradigm (in part 1) for task completion in part 2 wrote less accurate robot programs. The mismatch between natural tendencies and enforced coding structure has implications for the design of easily learned end-user robot programming interfaces and systems. It suggests that block-based visual imperative programming systems, though at present one of the most accessible ways to program robots, could benefit from new interfaces and targeted design enhancements.

I. Introduction

Robots are becoming increasingly ubiquitous across domains that require ongoing human interactions.[1] Programming such robots in human environments to be effective for every unique use case and environment remains a bottleneck given the complex interactive nature of desired robot behaviors [1] and individual user preferences [2].

Research in end-user robot programming [3]–[6] aims to create tools that enable users with little or no software development experience to write programs involving robots on their own through intuitive interfaces. While research in this field has progressed substantially, most non-academic robot programming systems targeting novice programmers are block-based visual programming systems, often wrapped around an imperative procedural programming language. Such systems also incorporate a robot-specific sensing and control library and a block-based visual programming interface. An example of a commonly used visual programming tool is Scratch [7], which introduces children and novice adult users to programming (and controlling a robot using this language). Though Scratch is among the best known and prominently used visual programming languages, others are surveyed in [8], [9].

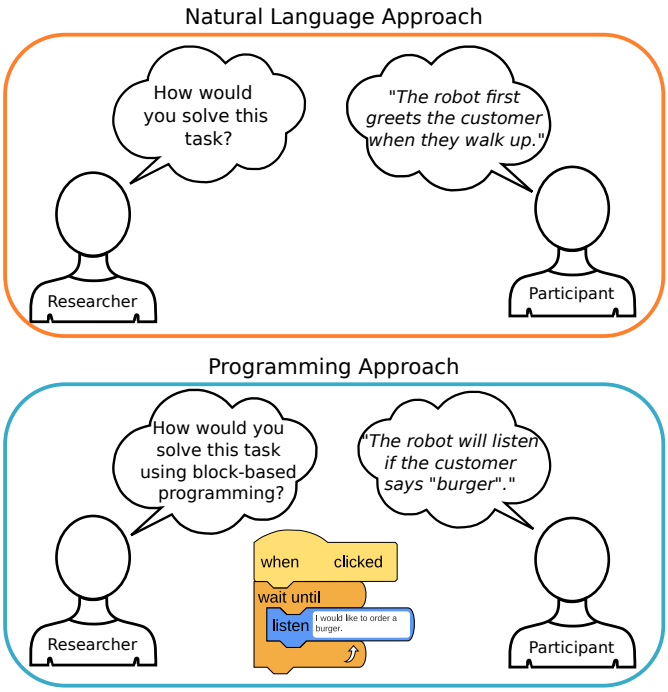

Fig. 1. We investigated how novice programmers think about programming robots through two tasks. (Top) **A natural language approach,** where participants verbally instruct the robot.(Bottom) **A block-based programming approach,** where participants implement the task using block-based programming.

There are known limitations with visual coding languages, such as challenges with expressing concurrency. Our key insight is that **one of the main challenges for non-expert users who want to program robots is cognitive dissonance caused by the mismatch between the programming paradigm they naturally adapt to express desired robot behaviors versus the programming paradigm supported by the widely used block-based robot programming system, i.e., imperative programming.** (See Section IV, Table 1 for definitions of the various programming paradigms we consider in this study.)

*This work was supported by the University of Washington Allen School Postdoc Research Award attributed to P. Alves-Oliveira.

[1]Rajeswari Hita Kambhamettu is affiliated with the School of Computer Science, Carnegie Mellon University, Pittsburgh, USA rkambham@andrew.cmu.edu

[2]Michael Jae-Yoon Chung is affiliated with Vicarious mchung@vicarious.com

[3]Patrícia Alves-Oliveira and Vinitha Ranganeni are affiliated with the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, USA patri,vinitha@cs.washington.edu

[1]IEEE Spectrum Article "Why Indoor Robots for Commercial Spaces Are the Next Big Thing in Robotics": https://spectrum.ieee.org/indoor-robots-for-commercial-spaces

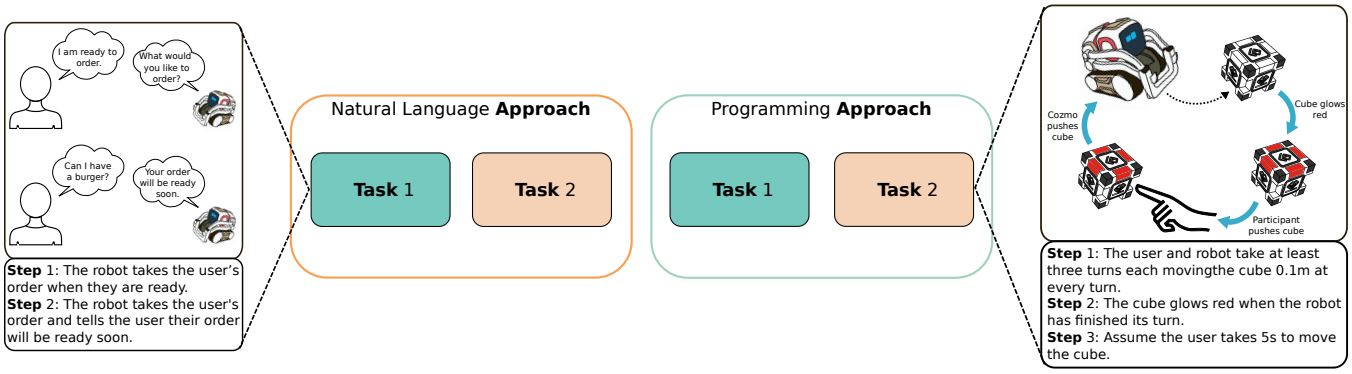

Fig. 2. Study Outline. We investigated two approaches, natural language and programming interface. For each **approach**, the participants completed the same two **tasks**. (*Left*) In task 1, the robot takes the user's order. (*Right*) In task 2, the robot and user collaborate to push a cube. Each task consists of several **steps**.

We aim to inform the development of new programming tools by first comparing these paradigms. Towards this goal, we conducted a two-part online pilot study to collect insights about how users naturally express what they want a robot to do vs how they actually program a robot, and the inherent challenges of a possible mismatch. In part 1, we asked participants to instruct the robot (using diagrams, language, flowcharts, etc.) to perform the two tasks we designed, and we analyzed their responses. We also studied the accuracy level of novice programmers, i.e., whether the robot correctly performed the tasks they specified. In part 2, we studied the effect of participants using a standard non-expert robot programming system that supports the imperative programming paradigm. We asked them to implement the same two tasks as in part 1, this time by using a block-based visual programming system. The system is based on Cozmo Codelab because our tasks were designed to be replicated with the Cozmo robot. [2].

Our work brings key insights into the mental models of novice programmers when 'thinking out loud' about programming a robot and how this can influence their programming performance. This work has a broader impact in the field of end-user programming for robots, especially on the design of the interface of end-user programming systems.

II. Related Work

Research in end-user robot programming aims to achieve easy to use visual interfaces by designing an appropriate abstraction level for target use cases [4], [10]–[13]. For example, researchers investigated adopting flowcharts [10], [11], [14], state machines [13], behavior trees [15], block-based imperative programming languages [4], [16], and trigger-action rules [17] with relevant visualization interfaces. Robot programming systems have also been built by taking a human-centered approach that provides specialized ways to express frequently desired concurrent actions [12], [16]. More recently, researchers have started to evaluate a non-expert-friendly robot programming interface like CoBlox with a larger group of novice adult users (e.g., N=67) [18]. However, to our knowledge, no robot programming system has been built to support the target users' intuitive programming paradigm.

Beyond end-user programming of robots, researchers have investigated the mental models novices use with trigger-action systems to program smart home applications [19] and analyzed how users of a commercially available trigger-action programming system debugged programs, including smart home applications [20], with the goal of helping future programming systems better support end-user programmers. Researchers also investigated how a target user group (children) solves problems to design a new programming language [21].

III. Methods

We now describe the methodology we employed for our online study.

Our study results are consistent with findings in prior work, which showed the applicability of human-computer interaction techniques to programming tools [22].

Our exploratory study focused on the following two research questions:

RQ.1 How do paradigms that novice programmers apply to control a robot map onto different programming paradigms? (study part 1)

RQ.2 How does an imperative programming interface affect the programming paradigm that novices apply? (study part 2)

A. Participants

We conducted our user study with 15 participants (9 female). We also conducted studies with 3 additional people to test and refine the study; these individuals were not included in the main analysis. Our participants were between $20-36$ years old and worked in a variety of disciplines, such as finance, marketing, and biology. Self-reported ethnicity revealed that 13 were of Asian descent, 5 White, and 2 Hispanic. Participants were recruited by individually reaching

[2]Cozmo codelab: https://developer.anki.com/blog/news/cozmo-code-lab/index.html

Images of Cozmo and cubes in Fig.2 are adapted from https://www.kinvert.com/

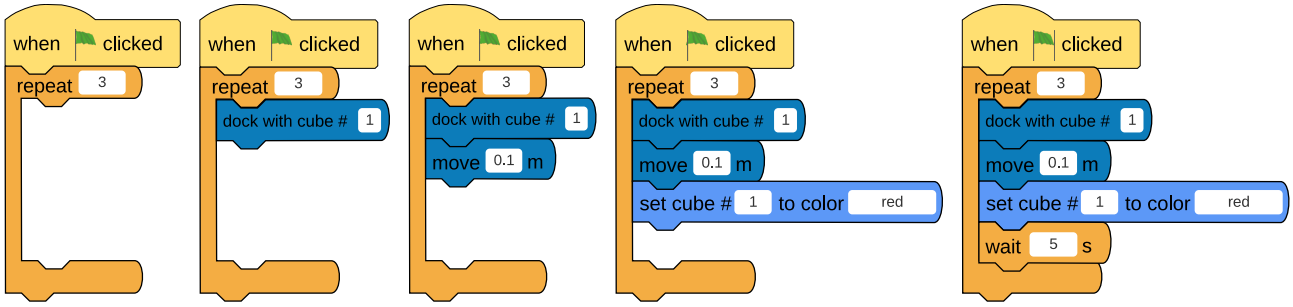

Fig. 3. An example of solving Task 2 using block-based programming. (a) When the program loop repeats for three iterations. (b) In every iteration, Cozmo navigates to the cube, (c) moves the cube 10cm, (d) sets the cube's color to red, and (e) waits five seconds for the user to push the cube.

out to peers who met the inclusion criterion (see below). We obtained IRB approval for the studies from the University of Washington. Participants gave informed consent and received a gift card of $10/hour for their study-related work.

Since this was a pilot study, our sole inclusion criteria was that the participants had taken at most one programming class/experience. Eight participants had never taken a programming class; four had experience in Scratch, a block-based programming language after which our programming tasks were modeled; and six had experience with robots. On a scale from one to five, seven participants self-reported that they had a level one programming proficiency, four self-reported a level two programming proficiency, and four self-reported a level three proficiency. We selected these participants from fields showing recent increases in robotics usage, i.e. healthcare, finance, and design.

B. Procedure

The study was exploratory in nature: its purpose was to elicit information about what programming paradigms novice programmers apply when programming social robots. Participants were asked to complete tasks using two different approaches, as shown in Fig.1: part 1, a descriptive approach, and part 2, a block-based programming approach. They were instructed to think aloud throughout each exercise and encouraged to talk through difficulties rather than asking questions. We designed our study to be completed in under one hour to limit participant fatigue.

The user study was conducted virtually through Zoom.[3] Participants were first asked to sign a consent form and complete a pre-questionnaire with questions about their demographic and programming skills. Next, in part 1, they were given two tasks to be completed using their preferred approach (e.g., diagrams or spoken instructions). While there was no physical robot involved in this study, the tasks were designed to be reproduced with the Cozmo robot, pictured in Fig.2. For the first task, participants were given the following prompt, as elucidated by (Fig.2):

> *In this task, the robot will be taking the order of a customer at a restaurant. The menu is as follows:*
> ***Foods:***
> *- Burger: $5*
> *- Pizza: $4*
> ***Drinks:***
> *- Soda: $2*
> *The robot should take the user's order once the user is ready. The robot will then take the user's order and tell them "your order will be ready soon."*

Participants were given five minutes to complete the task via any means they preferred (e.g., designing a diagram, speaking out loud, story-boarding). There was a time limit for every task to ensure the completion of all the tasks during the study session. All participants choose to speak aloud their method of directing the robot to complete this task.

For the second task, participants were given the following prompt, also to be completed in a non-specified approach, depicted in Fig.2:

> *You will be working with the robot to move cube 1. You must each take at least three turns moving the cube 0.1m at every turn. The cube should glow red once the robot is finished with its turn; assume that the user takes 5 seconds to move the cube.*

This completed part 1, where participants shared their unique approaches.

Next, for part 2, the researcher walked through the documentation of the blocks participants would use to complete the same tasks as in part 1 but using the block-based programming approach (see Fig.2). Participants were also given an example of how to use the blocks to complete small programming tasks involving the robot. The tasks remained the same for both study parts to achieve a fair comparison. The block-based programming interface, modeled after

[3]Zoom: http://zoom.us

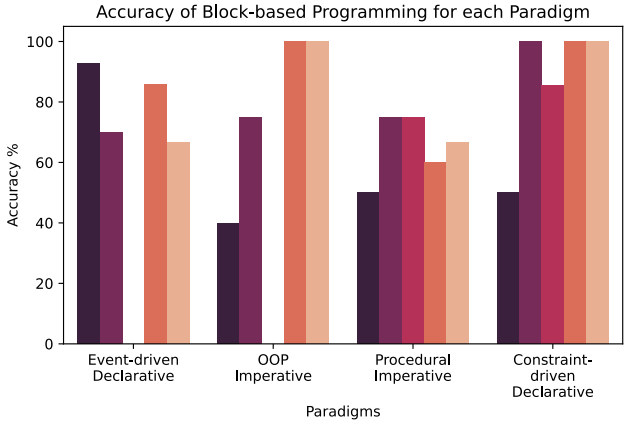

Fig. 4. Accuracy in accomplishing each part of the natural speaking tasks when various paradigms were used.
(—) Task 1, Step 1: The robot takes the user's order when user is ready.
(—) Task 1, Step 2: The robot takes the user's order and tells the user their order will be ready soon.
(—) Task 2, Step 1: The user and robot take at least three turns each, moving the cube 0.1m at every turn.
(—) Task 2, Step 2: The cube glows red when the robot has finished its turn.
(—) Task 2, Step 3: Assume the user takes 5s to move the cube.

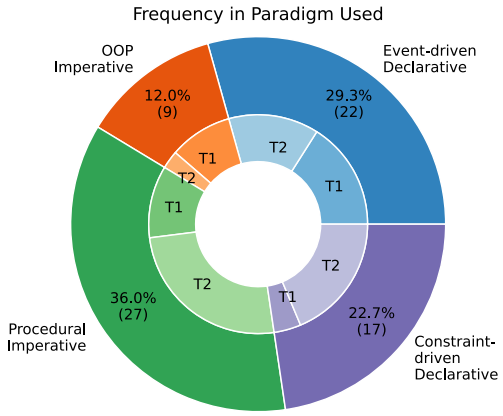

Fig. 5. Frequency with which a paradigm is used during the natural programming approach. T1 refers to task 1, and T2 refers to task 2. We observed that participants naturally adopt a wide array of programming paradigms, with the majority using a declarative approach when thinking naturally about programming the tasks.

Cozmo's CodeLab, was static so participants could not run their code. Further, they did not use a Cozmo robot in the study because of the virtual nature of the study.

After this introduction, participants were directed to create an account on LucidChart[4] to complete their tasks. They then completed the aforementioned two tasks in LucidChart by dragging and dropping blocks onto their workspace, as illustrated by Fig.3. They were given ten minutes to complete each task.

Following the completion of the block-based programming tasks, the participants completed a post-study questionnaire, which included the following questions:

- *Did you use the same strategy to think about the task with and without the blocks?*
- *What was your top priority in completing the task?*
- *How did the block-based programming align with how you would approach this task?*

These questions were intended to complement knowledge gained during the study from researcher observations. Upon completing the questionnaire, participants were thanked for their participation and time, and a gift card was sent to them.

Each participant took between forty-five minutes to about one hour to complete both study parts. We analyzed about sixteen hours of footage and manually scribed the users' cognitive process for each task in part 1 and part 2. We then categorized the users' responses into their subsequent paradigms.

IV. Results and Discussion

Our analysis is a combination of quantitative and qualitative data collected from the study and from the questionnaires that contained open-ended questions. The nature of our quantitative results is mostly descriptive since sample size for our study was small and did not allow for statistically robust comparisons. However, our results yield new insights about the mental models of novice programmers. Additionally, we bring a deeper understanding of the phenomena by combining descriptive statistics with qualitative results. Please contact study authors for a detailed transcript of study activities and analyses.

The results of the study address the research questions presented in Section III, and we divide results according to these questions. To code the programming paradigms participants applied, we used the coding scheme described in Table I [23]. We selected two widely used yet sufficiently distinctive imperative programming paradigms, i.e., procedural and object-oriented. We also selected two declarative programming paradigms based on either their popularity among novice users (e.g., event-driven programming) or their frequent adoption in robot and intelligent programming systems (e.g., constraint programming). We consistently use the terms 'approach,' 'task,' and 'steps' to describe our results; for a visual understanding of these terms, see Fig.2. In categorizing each participant's response, we first broke down each task into steps, or the most rudimentary part of the task. Then, we assigned each step to a category based on the participant's response and the rules of each paradigm, as shown in Table I.

> **RQ.1** How do paradigms that novice programmers adopt to control a robot map onto different programming paradigms? (study Part 1)

Participants tended to adopt a more declarative constraint or declarative event-centric programming approach. Data from the post-questionnaire revealed that 11 of the 15 participants indicated that their top priority in completing the tasks was thinking about all the combinations or possibilities the robot might encounter during the task while working within the constraints presented in the task steps. As shown in Fig.5, in **Task 1, about 40% of participants used**

[4]LucidChart, a free diagramming tool: https://lucid.app

TABLE I

TAXONOMY OF THE MOST COMMON PROGRAMMING PARADIGMS. WE USED THESE PARADIGMS AS THE CODING SCHEME FOR THIS STUDY [24].

PROGRAMMING PARADIGM	DEFINITION	EXAMPLE
Imperative	Describes the task in a sequence of commands.	*"Move 10 steps, listen to a customer's order, say your order will be ready soon"*
Imperative: Procedural	In the task description, repeated commands are referred to as sub-routines, and concepts like "goto" or "repeat" are used.	*"Repeat 3 times: Robot moves 10 steps and robot turns 90 degrees"*
Imperative: Object-Oriented Programming	Describes the task by using objects that are connected and manipulated by methods.	*"Robot - say 'hi' - human"* *"Robot - locate - cube"*
Declarative	Describes the task in terms of rules.	*"Whenever the robot sees a cube, it should move the cube forward"*
Declarative: Event-Driven	Describes the task as trigger and action pairs.	*"Once the user is finished talking, the robot will respond, saying 'Your order will be ready soon"'*
Declarative: Constraint	Describes the task in terms of constraints, e.g., whenever/wherever/if in certain situations, the robot should/should not take a certain action.	*"Whenever the customer orders something that is not on the menu, the robot should say the menu items again"*

a declarative event-driven approach, whereas 27% of participants used a procedural imperative approach, 23% used an object-oriented programming imperative approach, and 10% used a constraint-driven declarative approach. For **Task 2, 42% of participants used a procedural imperative approach**—most likely because the of the steps' sequential instruction. Additionally, 31% of participants used a declarative constraint-driven approach, 22% used a declarative event-driven approach, and 5% used an imperative object-oriented programming approach.

When comparing participants with and without programming experience, the one difference in programming paradigm used concerns object-oriented programming (OOP). **Four of seven participants who had taken a programming course used imperative OOP for task 1**, but only one of eight non-programmers used OOP for task 1. **In task 2, only participants who had taken a programming course used the OOP paradigm to solve the programming tasks.** This suggests that object-oriented programming may not be an intuitive paradigm for non-programmers but is more comfortable for those with more formal programming experience.

Overall, the results of the natural programming portions of both scenarios indicated that:

> *An imperative procedural programming paradigm – be it procedural, or OOP – is not necessarily the most obvious or natural way that novice users express programs for the robot.*

The fact that most participants did not use an imperative procedural programming approach across both tasks suggests that *novice programmers tend to rely less on sequential or imperative paradigms*.

More participants used an imperative procedural approach to complete Task 2, likely because the step description was outlined sequentially. Nonetheless, the majority of participants did not naturally adopt that approach. As our analysis shows, six of 15 participants used an imperative procedural approach to solve one of the steps in Task 1, but only one participant used that approach to solve the entirety of Task 1.

Nine participants used imperative procedural programming to complete at least one step of the task, but only one participant naturally used imperative procedural programming to solve Task 2 as a whole. Both participants who naturally used this paradigm throughout their respective tasks indicated that they *"thought [of themselves] as the robot and went through how I would complete the task."* Perhaps because the participants completed the task *as* the robot rather than as the *controller* of the robot, they only considered the robot when completing the tasks and did not consider the robot's interactions with other entities, such as a cube or a customer. Additionally, one of these participants stated that they chose a *"systematic approach structured on whether or not the robot could do something."* This suggests that *novice programmers use a more declarative event-driven and declarative constraint-driven approach to program robots*. If a social robot programming language intends to model users' natural mental models, then it should include declarative elements in the programming language.

> **RQ.2** How does an imperative programming interface affect the programming paradigm that non-programmers adopt? (study part 2)

We found that the **approaches participants chose to implement the block-based programming steps did not consider the edge cases they mentioned in part 1** of the study, specifically when the robot acts in response to a user. Thus, 11 participants indicated that they did not use the same approach to complete the block-based programming tasks compared to the natural programming task. Our qualitative

results indicate that several of these participants noted that the block-based programming style had several limitations, including the fact that blocks allowed for less flexibility in programs, placed restraints on how to solve the task, and did not let them explore the more detailed approaches they envisioned in the natural programming section.

We evaluated the accuracy of the participants' block-based programs by dividing each task into the steps outlined in Fig.4, which served as test cases. We then indicated a binary score for each step in the task (1 for passing the test and 0 for not passing it) and evaluated the block-based programs based on these test cases. As shown in Fig.4, **participants who used an imperative procedural paradigm in their natural programming approach often achieved lower accuracy than those who chose other programming approaches.** For example, in Task 2, participants naturally using an imperative procedural approach had an average accuracy of $60-75\%$ per step, whereas those using declarative event-driven programming had an average accuracy of $67-85\%$ per step, those using declarative constraint-driven programming had an average accuracy of $85-100\%$ per step, and those using imperative OOP had an average accuracy of 100% per step.

We also compared participants with and without programming experience. For Task 1, participants with no prior programming experience had an average of 71% accuracy, while those who had taken one programming course had an average accuracy of 75%, a nominal difference. However, in Task 2, participants with no prior programming experience had an average of 70% accuracy, while participants who had taken one programming course had an average accuracy of 97%. This difference in accuracy is potentially attributable to the human-robot interaction component of the task and its many constraints. Upon further analysis of the studies, we found that seven of the eight non-programmers forgot to account for at least one of the constraints included in the task. The step in Task 2 that participants most often did not accurately complete was step 1 (refer to Fig.2 for the step description). Three of eight non-programmers (37.5%) used constraint-driven programming to solve this step, while four of seven participants (57%) with prior programming experiences used constraint-driven programming. Perhaps because this task included constraints, participants with prior programming experience thought to employ a constraint-driven approach while those with no prior experience did not modify their paradigm to align with the task. Only one participant scored 100% accuracy across both tasks. This participant has taken an introductory programming class before and used an OOP approach for both tasks. In Task 1, the participant used the 'menu items,' 'user,' and 'robot' as objects that they modified using methods such as adding food to the order. In Task 2, the participant used the 'robot,','user,' and 'cube' as entities with methods to move the cube. The high accuracy could be attributed to the OOP paradigm the participant originally used, which allowed them to consider all constraints of the task while also ensuring the actions were robustly defined. Nine participants made the mistake of executing an action without considering an initial constraint. For example, in Task 1, participants would program the robot to say "What would you like to order?" before checking if the customer was ready.

This result suggests that *if novice programmers naturally think in an imperative or imperative procedural way, they are less likely to develop accurate programs*. If user accuracy is the principal objective of a social robot programming language, developers should consider employing a declarative programming paradigm to better align with novice programmers' natural programming tendencies.

In sum, when analyzing the performance of participants based on their natural programming approaches, we found that:

> *Participants who naturally used an imperative procedural style had less accurate block-based programs than those who used different paradigms.*

V. Limitations

This study has the following limitations:

- Because this was a pilot study, we recruited a small sample size of mostly undergraduate students in non-technical STEM majors rather than a more representative group. Because our sample does not reflect a broader set of novice programmers, we may have overlooked insights. To make more solid claims, we want to recruit a larger and more diverse sample size.
- We considered only two main categories of paradigms and two subcategories of these paradigms. However, in practice, there are many more programming paradigms that programmers use. Heavily constraining our assumptions to consider only imperative and declarative programming could potentially bias our results.
- Though all of our tasks can be reproduced with the Cozmo robot, in the interest of the study brevity and considering the remote study setting, we did not instruct participants to use a robot when programming their tasks. However, including the robot in the study itself would have better reflected the intended setting for participants to program.
- The way we instructed study participants may have affected their approach. In the pilot study, we wanted to keep the the task description as simple as possible so participants felt comfortable verbally (or otherwise) expressing their responses. We also included elements of concurrency in each task since it is a known limitation of visual block-based programming. However, we noticed that the task description potentially altered the paradigm employed by the participants to solve the tasks. In future work, we plan to ensure that our task descriptions do not influence participants by iterating over small test samples to address this issue.

VI. Conclusion

In this paper, we investigated how novice users most accurately express intent with respect to robot behavior. We

conducted a two-part pilot study with fifteen participants who were first asked to (1) naturally express how they would program a robot to accomplish predefined tasks *(all participants chose verbal expression)*, and then asked to (2) *program* the robot to accomplish the same predefined tasks. Based on our analysis, we share the following main insights:

- **Insight 1:** *Novice users apply different paradigms when they naturally think through robot control tasks vs when they actually develop programs for robots.* They adapt their programming style to the end-user programming system they are faced with, which might not be the optimal way for the user to interface with a programming system and could hinder the development of desirable behaviors for the robot due to the mismatch of mental models.

- **Insight 2:** *Novice users who naturally express their programs in non-imperative paradigms tend to write more accurate programs.* Those who use imperative paradigms write programs that are less accurate than those who approach programs with more natural paradigms. Thus, an imperative-style end-user programming system might prevent users from programming more accurately.

We believe these two primary insights, coupled with other findings reported in the study, could inform the development of new end-user programming tools that are more aligned with how novices naturally think about specifying robot behavior.

REFERENCES

[1] P. Tsarouchi, S. Makris, and G. Chryssolouris, "Human–robot interaction review and challenges on task planning and programming," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, 2016.

[2] G. Ajaykumar and C.-M. Huang, "User needs and design opportunities in end-user robot programming," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 93–95.

[3] D. F. Glas, T. Kanda, and H. Ishiguro, "Human-robot interaction design using interaction composer eight years of lessons learned," in *International Conference on Human-Robot Interaction*. ACM/IEEE, 2016, pp. 303–310.

[4] J. Huang, T. Lau, and M. Cakmak, "Design and evaluation of a rapid programming system for service robots," in *ACM/IEEE International Conference on Human Robot Interaction (HRI)*, 2016, pp. 295–302.

[5] A. Kubota, E. I. Peterson, V. Rajendren, H. Kress-Gazit, and L. D. Riek, "Jessie: Synthesizing social robot behaviors for personalized neurorehabilitation and beyond," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2020, pp. 121–130.

[6] D. Porfirio, E. Fisher, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Bodystorming human-robot interactions," in *Symposium on User Interface Software and Technology*, 2019.

[7] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[8] E. Coronado, F. Mastrogiovanni, B. Indurkhya, and G. Venture, "Visual programming environments for end-user development of intelligent and social robots, a systematic review," *Journal of Computer Languages*, vol. 58, p. 100970, 2020.

[9] M. A. Kuhail, S. Farooq, R. Hammad, and M. Bahja, "Characterizing visual programming approaches for end-user developers: A systematic review," *IEEE Access*, 2021.

[10] D. Glas, S. Satake, T. Kanda, and N. Hagita, "An interaction design framework for social robots," in *Robotics: Science and Systems*, vol. 7, 2012, p. 89.

[11] S. Alexandrova, Z. Tatlock, and M. Cakmak, "Roboflow: A flow-based visual programming language for mobile manipulation tasks," in *International Conference on Robotics and Automation*. IEEE, 2015, pp. 5537–5544.

[12] J. Diprose, B. MacDonald, J. Hosking, and B. Plimmer, "Designing an api at an appropriate abstraction level for programming social robot applications," *Journal of Visual Languages & Computing*, vol. 39, pp. 22–40, 2017.

[13] F. Steinmetz, A. Wollschläger, and R. Weitschat, "Razer-a human-robot interface for visual task-level programming and intuitive skill parameterization," *Robotics and Automation Letters*, vol. 3, no. 3, pp. 1362–1369, 2018.

[14] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choregraphe: a graphical tool for humanoid robot programming," in *The International Symposium on Robot and Human Interactive Communication*. IEEE, 2009, pp. 46–51.

[15] C. Paxton, F. Jonathan, A. Hundt, B. Mutlu, and G. D. Hager, "Evaluating methods for end-user creation of robot task plans," in *International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 6086–6092.

[16] M. J.-Y. Chung, J. Huang, L. Takayama, T. Lau, and M. Cakmak, "Iterative design of a system for programming socially interactive service robots," in *International Conference on Social Robotics*, 2016, pp. 919–929.

[17] N. Leonardi, M. Manca, F. Paternò, and C. Santoro, "Trigger-action programming for personalising humanoid robot behaviour," in *Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.

[18] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[19] J. Huang and M. Cakmak, "Supporting mental model accuracy in trigger-action programming," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 215–225.

[20] W. Brackenbury, A. Deora, J. Ritchey, J. Vallee, W. He, G. Wang, M. L. Littman, and B. Ur, "How users interpret bugs in trigger-action programming," in *Proceedings of the 2019 CHI conference on human factors in computing systems*, 2019, pp. 1–12.

[21] J. F. Pane, *A programming system for children that is designed for usability*. Carnegie Mellon University, 2002.

[22] B. A. Myers, A. J. Ko, T. D. LaToza, and Y. Yoon, "Programmers are users too: Human-centered methods for improving programming tools," *Computer*, vol. 49, no. 7, pp. 44–52, 2016.

[23] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Comput. Surv.*, vol. 37, no. 2, p. 83–137, jun 2005. [Online]. Available: https://doi.org/10.1145/1089733.1089734

[24] M. Gabbrielli and S. Martini, *Programming Languages: Principles and Paradigms*, 1st ed. Springer Publishing Company, Incorporated, 2010.