Collecting Insights about 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]

[1]*Carnegie Mellon University, Pittsburgh, PA*
[2]*Vicarious, Seattle, WA*
[3]*University of Washington, Seattle, WA*

Abstract

End-user programming of robots holds the promise to enable any user without specialized software development skills to create customized behaviors for their robot. To fully enable users to have control of robot programs, it is essential to design an end-user robot programming system that supports how users naturally think when expressing the desired robot behaviors. In this paper, we investigate how novice programmers naturally express robot programs by analyzing how they verbally solve a programming task and how they actually program robot behaviors using a typical visual robot programming system. Towards this goal, we conducted an online user study focusing on two approaches, natural and programming, with fifteen participants. We mapped how users naturally express robot programs to existing programming paradigms (e.g., declarative, imperative). Our results show that novice programmers do not naturally think about programming tasks in a procedural imperative programming paradigm, typical of current visual robot programming systems. This mismatch has implications for designing end-user robot programming systems. Additionally, our results show that users who initially used a procedural imperative programming paradigm wrote less accurate robot programs. This result seems to indicate that block-based visual imperative programming systems, while being one of the most accessible ways to program robots, might lead to more inaccurate robot programs.

Keywords: End-user programming. Human-robot interaction. Exploratory study.

1 Introduction

Robots are becoming increasingly ubiquitous across domains that require interacting with humans[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 the individual preferences of each user [2].

Research in end-user robot programming [3]–[6] aims to create tools that enable users with little or no experience with software development to write robot programs on their own through intuitive interfaces. While research on this field has substantially progressed, most available robot programming systems targeting non-experts programmers outside of academia are block-based visual programming systems, which often wrap a procedural imperative programming language. These block-based visual programming systems also function with 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]. Scratch is a visual programming tool used to introduce children and any user with no programming experience to programming (and controlling a robot using this language). While Scratch is one of the most well known and prominently used visual programming languages, there are more (see the systematic review for additional information on visual programming environments for end-users [8], [9]).

Though there are known limitations with visual programming languages, such as challenges with expressing concurrency, our key insight is that one of the main challenges for non-expert users that

*Email: rkambham@andrew.cmu.edu
†Email: mchung@vicarious.com
‡Email: vinitha@cs.washington.edu
§Email: patri@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

PLATEAU
12th Annual Workshop at the Intersection of PL and HCI

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

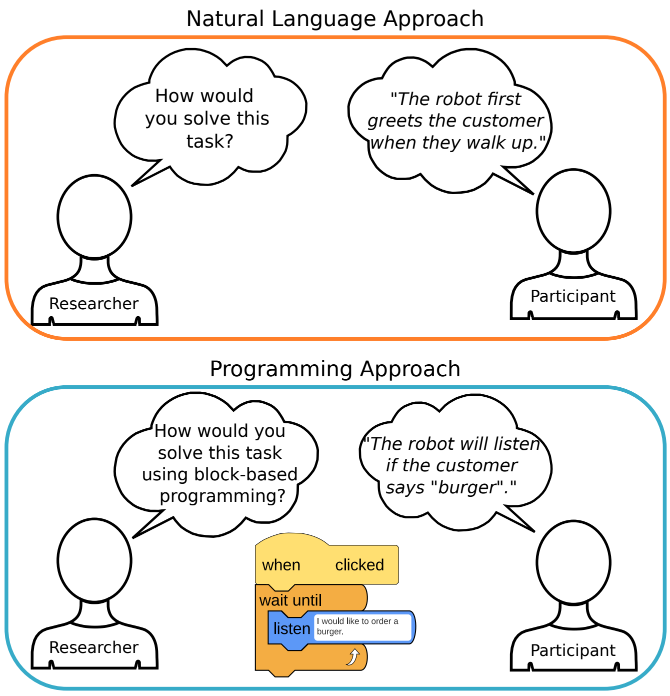

Natural Language Approach
How would you solve this task?
"The robot first greets the customer when they walk up."
Researcher
Participant
Programming Approach
How would you solve this task using block-based programming?
"The robot will listen if the customer says "burger"."
when clicked
wait until
listen
I would like to order a burger.
Researcher
Participant

Figure 1. We investigated how novice programmers think about programming robots through two tasks. *Upper Figure:* **Natural Language Approach** where participants verbally instruct the robot. *Lower Figure:* **Block-Based Programming Approach** where participants implement the task using block-based programming.

want to program robots may be the cognitive dissonance caused by the *mismatch between the programming paradigm (i.e., how non-expert users are thinking in when trying to express the desired robot behavior) versus the programming paradigm supported by the system (e.g., imperative programming)*.

Aiming to inform the development of new programming tools targeting end-users, our paper sets to investigate the challenges between how non-expert robot programmers think about programming robots with the existing visual programming environment. Towards this goal, we conducted a study that aims to collect insights about how users naturally express programs and how they actually program a robot, and the inherent challenges of a possible mismatch. Therefore, we first studied how non-experts naturally describe robot tasks and their underlying programming paradigm by asking participants to verbally instruct the robot to perform the two tasks we designed and analyzed their responses. We also studied the accuracy level of novice programmers robot programs in relation to the programming paradigm used.

Additionally, we studied the effect of using a typical non-expert robot programming system that supports the imperative programming paradigm by asking participants to implement the same two tasks in both using a natural programming setting and using a Block-based visual programming system. Our visual block-based programming is based off Cozmo Codelab[2].

Our work brings key-insights into the mental models that novice programmers have while thinking 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, specially on the design of the interface of end-user programming systems.

2 Related Work

Research in end-user robot programming aims to achieve ease-of-use by designing for an adequate abstraction level for the target use cases with intuitive visual interface [4], [10]–[13]. For example, researchers investigated adopting flowcharts [10], [11], [14], state machines [13], behavior trees [15],

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/

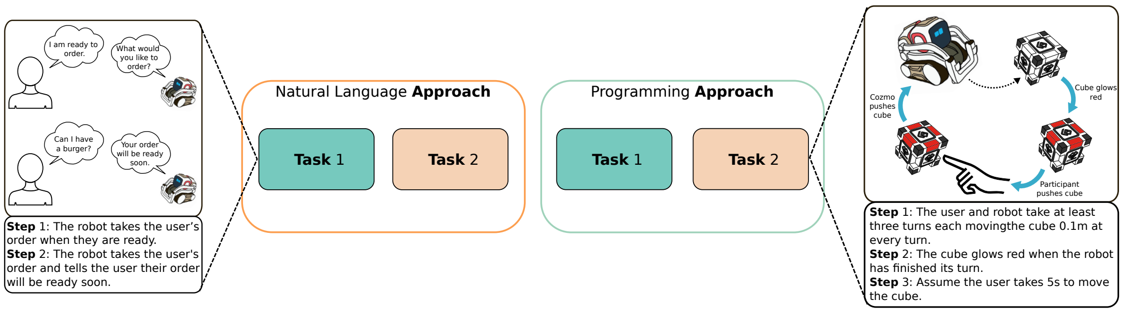

I am ready to order.
What would you like to order?
Can I have a burger?
Your order will be ready soon.
Step 1: The robot takes the user's order when they are ready.
Step 2: The robot takes the user's order and tells the user their order will be ready soon.
Natural Language **Approach**
Task 1
Task 2
Programming **Approach**
Task 1
Task 2
Cube glows red
Cozmo pushes cube
Participant pushes cube
Step 1: The user and robot take at least three turns each movingthe cube 0.1m at every turn.
Step 2: The cube glows red when the robot has finished its turn.
Step 3: Assume the user takes 5s to move the cube.

Figure 2. Outline of the study. We investigated two approaches: Natural Language Approach and Programming Approach. For each approach, the participants complete two tasks. Note, we use the same tasks for each approach. (*Left*) In task 1 the robot must take the user's order. (*Right*) In task 2 the robot and user must collaborate to push a cube.

block-based imperative programming languages [4], [16], and trigger-action rules [17] with relevant visualization interface. Robot programming systems also have been built by taking a human-centered approach that provides specialized ways for expressing frequently desired concurrent actions [12], [16]. However, up to our knowledge, there has not been a robot programming system built to support the target users' natural programming paradigm.

Beyond the end-user programming of robots, researchers investigated the mental models of end-users when using trigger-action to program smart home applications [18] and analyzed how users of a commercially available trigger-action programming system debugged programs including smart home applications [19] with the goal of helping future programming systems better supporting end-user programmers.

3 Method

In this section, we detail our study conducted to explore the topic of programming paradigms expressed by novice users while programming robots. This study is in consonance with prior work that showed the applicability of human-computer interaction techniques to programming tools [20]. To investigate how non-experts—novice or non-programmers—approach programming human-robot interactions, we designed an exploratory user study focusing on the following research questions:

RQ.1 How do programming paradigms that novice programmers use when they program a robot map onto different programming paradigms?

RQ.2 How does an imperative programming interface affect the programming paradigm that the non-programmers follow?

3.1 Participants

We conducted our user study with 18 participants (9 female), from which 3 were part of the pilot study and not included in the main analysis. Our participants are aged between $20 - 36$ and come from a variety of fields 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 out to peers who met the inclusion criterion. 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 time spent in the study.

Our sole inclusion criteria was that the participants had at most one programming class/experience. About ten of the participants had taken one programming class, and four of the participants had experience in Scratch, a block-based programming language after which our programming tasks were modeled. On a scale from one to five, seven of the participants self-reported that they had the level one of programming proficiency, four users self-reported the level two programming proficiency, and four of the users self-reported the level three of programming proficiency.

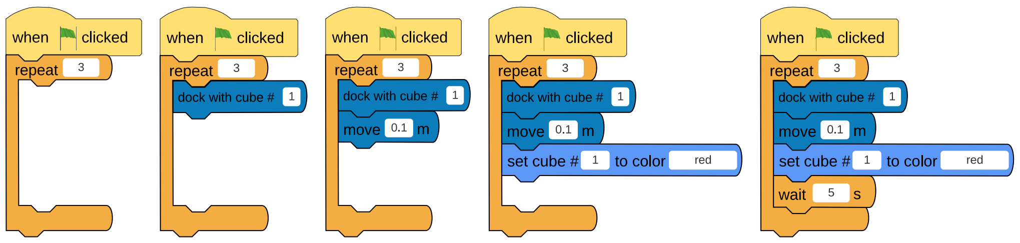


Figure 3. An example of solving Task 2 using block-based programming. a When the program a loop repeats for three iterations. b In every iteration Cozmo navigates to the cube, c moves the cube 10cm, d sets the cube's color to red, and e waits five seconds for the user to push the cube.

3.2 Procedure

The study was exploratory in nature: its purpose was to inform about what programming paradigms novice programmers gravitate towards when programming with social robots. Participants were asked to complete tasks in two different approaches, as shown in Fig.1: one was a natural programming approach and the other was using a block-based programming approach. Participants 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 an hour in order to make it easier for us to limit participant fatigue.

The user study was conducted virtually Zoom[3]. Participants were first asked to sign a consent form and complete a pre-questionnaire with questions about their demographic and their programming skills. Next, they were given two tasks to be completed in a natural programming approach. 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

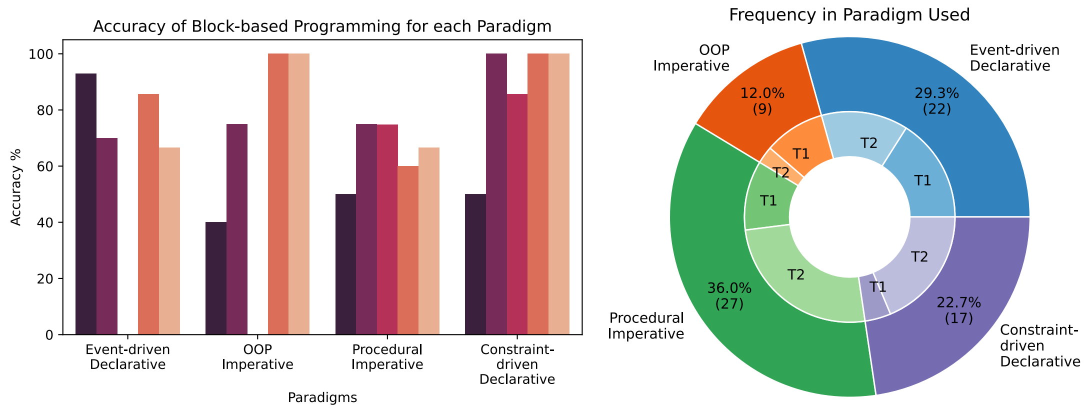
Drinks:

- 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 in 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 to programming the robot to complete this task. For the second task, participants were given the following prompt also to be completed in a natural programming approach, depicted in Fig.2:

3 Zoom: http://zoom.us



(a) (b)

Figure 4. (a) Accuracy in accomplishing each part of the natural speaking tasks when various paradigms are used.
(—) Task 1, Step 1: The robot takes the user's order when they are 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.
(b) Frequency in 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 use a wide array of programming paradigms, with a majority of participants used a declarative approach when thinking naturally about programming the tasks.

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 the section where participants shared their natural programming approaches. Next, the researcher walked through the documentation of the blocks participants would use to complete the same tasks using the block-based programming approach. Participants were also given an example of how to use the blocks to complete the small programming tasks involving the robot. The tasks that participants were asked to complete were the same Tasks 1 and 2 (see Fig.2). The tasks remained the same for a fair comparison of the programming paradigms adopted by the participants. The block-based programming interface that the participants used was modeled after Cozmo's CodeLab, however, the participants did not use the Cozmo robot in the study.

After this introduction, the participants were directed to create an account on LucidChart[4] in which they were to complete their tasks. The participants 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 the final post-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. Upon completing the questionnaire, participants were thanked for their participation and time, and a gift card was sent to them.

Table 1. Taxonomy of the most common programming paradigms. These paradigms were used as the coding scheme for this study.

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, the 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"*

4 Results and Discussion

The results of the study addressed each of the research questions present in Section 3. Therefore, we divided the results according to the research questions. To code the programming paradigms used by the participants, we used the coding scheme described in Table 1. We selected the two widely used yet distinctive enough imperative programming paradigms, i.e., procedural and object-oriented. We also selected two declarative programming paradigms based off of their popularity among novice users (e.g., event-driven programming paradigm) or the frequent adoption in robot and intelligent systems programming systems (e.g., constraint programming paradigm).

RQ.1 How do programming paradigms that novice programmers use when they program a robot map onto different programming paradigms?

Participants had a tendency to use a more constraint or event-centric programming approach. Data from the post-questionnaire revealed that eleven out of fifteen 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.4b, in Task 1, about 40% of participants used an event-driven declarative approach, whereas 27% of participants used a procedural imperative approach, 23% of participants used an object-oriented programming imperative approach, and 10% of participants 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 constraint-driven declarative approach, 22% of participants used an event-driven declarative approach, and 5% of participants used an object-oriented imperative programming approach.

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

A procedural imperative programming paradigm is not necessarily the most obvious or natural way that users express programs for the robot.

The wide range of programming approaches with a majority of participants not using a procedural imperative programming approach across both tasks suggests that *novice programmers tend to use less sequential paradigms*.

More participants in Task 2 used a procedural imperative approach, likely because the step description was outlined sequentially. Nonetheless, a majority of participants did not naturally use a procedural imperative approach. As we can see from our analysis, six participants used a procedural imperative programming approach to solve one of the steps in Task 1, while only one participant used procedural imperative programming to solve the entirety of Task 1.

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

RQ.2 How does an imperative programming interface affect the programming paradigm that the non-programmers follow?

We found that the approaches participants took to implement the block-based programming steps did not consider edge cases that they had mentioned in their natural programming, specifically when the robot is acting in response to a user. Thus, eleven participants indicated that they did not use

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

the same approach when completing the block-based programming tasks compared to the natural programming task. Our qualitative results indicate that several of these participants referred that the block-based programming style had several limitations, some of which being the fact that blocks allowed for less flexibility in programs, placed restraints on how to solve the task, and did not let them explore more detailed approaches they started with in the natural programming section.

We evaluated the accuracy of the participants' block-based programs by dividing each task into steps outlined in Fig.4a and evaluated the block-based programs based on these test cases. As shown in Fig.4a, participants that used a procedural imperative programming paradigm in their natural programming approach often achieved lower accuracy than those who chose to use other programming approaches. For example, in Task 2, participants who naturally used a procedural imperative approach had an average accuracy of $60 - 75\%$ per Step, whereas those who used event-driven declarative programming had an average accuracy of $67 - 85\%$ per Step, those who used constraint-driven declarative programming had an average accuracy of $85 - 100\%$ per Step, and those who used object-oriented imperative programming had an average accuracy of 100% per Step.

Only one participant scored 100% accuracy across both tasks. This participant used object-oriented programming to approach both tasks. In Task 1, the participant used the 'menu items', 'user', and 'robot' as objects which were modified by 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 a cube. The high accuracy could be attributed to the object oriented programming paradigm the participant originally used which allowed them to consider all constraints of the task while also ensuring the actions were robustly defined. One mistake nine participants made was 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 a sequential or procedural imperative way, they are less likely to develop accurate programs*. If user accuracy is the principal objective of a social robot programming language, perhaps developers should employ 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 a procedural imperative style had less accurate block-based programs than those who used different paradigms.

5 Conclusion

In this paper, we investigated how novice users naturally express programs by conducting a user study with fifteen participants where they were asked to (1) *verbally express (naturally express)* how would they program a robot to accomplish pre-defined tasks; (2) *program the task* for a robot to accomplish the same pre-defined tasks. Based on our analysis, we share the following main insights:

- **Insight 1:** *Novice users use different programming paradigms when they naturally talk about programs and when they actually develop programs for robots.* This result seems to show that novice users 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 might 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.* This result seems to indicate that novice users who naturally think in accordance to an imperative paradigm write programs that are less accurate than those who naturally approach programs with different paradigms. Thus, an imperative-style end-user programming system might not allow users program more accurately.

We believe these two insights and other findings reported in the study could inform the development of new end-users programming tools more aligned with how we naturally think about robot programming.

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. 100 970, 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] 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.

[19] 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.

[20] 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.