Figaro: A Tabletop Authoring Environment for Human-Robot Interaction

David Porfirio
dporfirio@wisc.edu
University of Wisconsin–Madison
Madison, Wisconsin

Laura Stegner
stegner@wisc.edu
University of Wisconsin–Madison
Madison, Wisconsin

Maya Cakmak
mcakmak@cs.washington.edu
University of Washington
Seattle, Washington

Allison Sauppé
asauppe@uwlax.edu
University of Wisconsin–La Crosse
La Crosse, Wisconsin

Aws Albarghouthi
aws@cs.wisc.edu
University of Wisconsin–Madison
Madison, Wisconsin

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

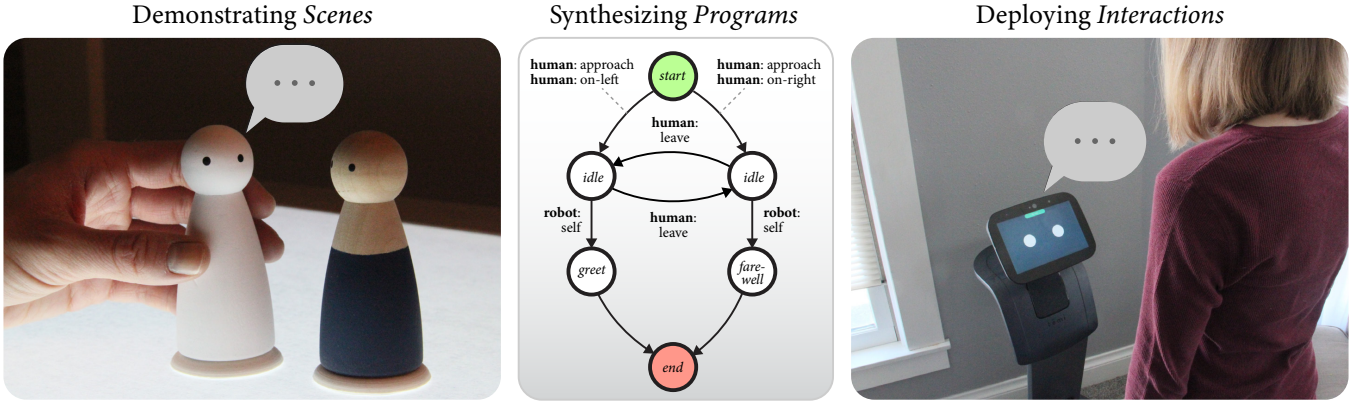


Figure 1: *Figaro* enables users to program robots through demonstrations via *scenes* (left) and constructs an interaction program from user demonstrations (center), which can be deployed on a robot (right).

ABSTRACT

Human-robot interaction designers and developers navigate a complex design space, which creates a need for tools that support intuitive design processes and harness the programming capacity of state-of-the-art authoring environments. We introduce *Figaro*, an expressive tabletop authoring environment for mobile robots, inspired by *shadow puppetry*, that provides designers with a natural, situated representation of human-robot interactions while exploiting the intuitiveness of tabletop and tangible programming interfaces. On the tabletop, Figaro projects a representation of an environment. Users demonstrate sequences of behaviors, or *scenes*, of an interaction by manipulating instrumented figurines that represent the robot and the human. During a scene, Figaro records the movement of figurines on the tabletop and narrations uttered by users. Subsequently, Figaro employs real-time program synthesis to assemble a complete robot program from all scenes provided. Through a user study, we demonstrate the ability of Figaro to support design exploration and development for human-robot interaction.

CCS CONCEPTS

• **Human-centered computing** → **Systems and tools for interaction design**; • **Computer systems organization** → **Robotics**.

KEYWORDS

Authoring environments; tabletop interfaces; shadow puppetry; program synthesis; human-robot interaction

ACM Reference Format:
David Porfirio, Laura Stegner, Maya Cakmak, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2021. Figaro: A Tabletop Authoring Environment for Human-Robot Interaction. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3411764.3446864

1 INTRODUCTION

Designing human-robot interactions (HRIs) for social applications entails carefully crafting both the overall flow of the interaction and the individual behaviors for the robot to perform. Furthermore, the

5th author's name in native alphabet: البرعوثي اوس.

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3446864

interaction design process must consider the context of the interaction, including the potential attitudes, preferences, and behavioral differences of humans interacting with the robot, in addition to the limitations of the physical environment within which the robot will be deployed. However, most existing programming approaches for social robotics require designers to work with highly abstract representations that are far removed from the context of the target interaction. On one extreme, traditional programming tools require designers to immerse themselves in low-level syntax and semantics, such as through traditional text-based programming interfaces (e.g., [6, 48]). In another extreme, interfaces that employ graphical representations of the interaction, such as finite automata, require designers to memorize symbolic representations of interaction logic (e.g., [21]). In both cases, designers must independently maintain mental models of how their abstract representations of the interaction map to situated interactions in real life. Although recent work has aimed to close this gap between the designer's vision and system behavior, e.g., by mapping designer demonstrations to robot behaviors [45], designers still lack representations of the situated resources, requirements, and constraints of the interaction. Without a clear mapping between abstract programming functionality used at *design* time and how the robot will behave in the target social environment at *execution* time, designers must rely heavily on simulation, extensive testing, and their own imaginations to predict how their designs will play out after deployment.

To address the gap introduced by programming abstractions between design and execution times, we have developed *Figaro* (Figure 1), a tabletop authoring environment for designing human-robot interaction flows that situates the interaction design process within a miniature representation of the target user environment in order to enable designers and developers of robot applications to focus exclusively on specifying system behavior. Figaro's design is inspired by "shadow puppetry," where a puppeteer can perform highly expressive scenes using a few static props and puppets with limited articulation through the use of motion and narration. The props and the stage provide the performer with just the right amount of context for situated play-acting, and simple motions of the puppets and the narration provide sufficient expressivity to convey social behavior and affect. As in shadow puppetry, using Figaro, designers who we refer to as *demonstrators* play out high-level sequences of robot and human behaviors by manipulating figurines representing human and robot agents on the tabletop and narrating verbal commands to specify speech for each figurine. We refer to a sequence of behaviors as a *scene*. A drawing of the target deployment environment is projected from underneath the tabletop onto a translucent surface, giving demonstrators a clear representation of the space where humans will interact with the robot. Demonstrator narration is combined with actions performed on the figurines and their position and movement within the deployment environment to create a complete model of each scene. Figaro automatically resolves the program semantics of each scene, freeing demonstrators from these low-level details. Given a collection of scenes, Figaro employs real-time program synthesis to construct a full interaction automata that can be deployed on a robot.

Despite detaching the demonstrator from direct control over the semantics of interactions, Figaro still maintains expressivity over critical aspects of the design process. First, demonstrators control the physical layout of the environment that is projected onto the tabletop, which specifies the physical space where the interaction is to take place. Prior to creating scenes, demonstrators specify the various objects that the robot can recognize and regions over which the robot can navigate. Second, Figaro enables detailed control over the robot's behaviors in the interaction through (1) tracking the relative position of the robot to the human and each object in the scene, (2) tracking the orientation of the robot with respect to these same components, (3) instrumenting the figurines so that the demonstrator may tangibly select behaviors that the robot should perform and (4) allowing the demonstrator to verbally specify speech utterances that the robot can emit or understand from a human.

In addition to providing an open-source software implementation of Figaro, the research contributions of our work include:

- A novel *tabletop programming interface* through which users can manipulate figurines to demonstrate the flow of human-robot interaction programs;
- A versatile *synthesis approach* that combines streams of user input into expressive HRI programs;
- An exploratory *user study* of Figaro that demonstrates its ability to support design exploration and programming.

2 RELATED WORK

Below we discuss prior work with regards to design considerations and methods for creating human-robot interactions, programming interfaces that enable this design to occur, and how these programs can be represented and automatically synthesized.

2.1 Designing Human-Robot Interactions

There are many aspects of human-robot interactions for designers to consider, many of which Figaro broadly intersects such as verbal and nonverbal communication [25, 54]. Locomotion in particular, which is a large focus of Figaro, requires designers to reason about the robot's physical proximity to its interaction partners [7] and situate the robot in a physical space [29, 53, 57]. Other navigation complexities to consider include humans and robots walking side-by-side [36] and ensuring that the trajectory taken by the robot reduces stress on the human [35].

The multi-faceted nature of human-robot interactions is easy for designers to use themselves *in situ* or recognize when incorrect, but is difficult for designers to articulate for *ad hoc* purposes. To help mitigate this issue, Wada et al. [64] implemented a system where the designer would first ideate, then tele-operate a robot to test their design, refining as needed. This immersive, hands-on experience gave the designers a better feel for the actual interactions they may encounter and how they might progress, however such a design methodology is not always practical. A more hands-off approach by Liu et al. [33] sought to use high-precision sensors to gather a large collection of human-human interactions in a public environment in order to automatically generate a human-robot interaction.

Figaro uses immersive design methods to help demonstrators in recognizing and articulating the needs of multiple interaction possibilities. For example, in scenario-based design, designers focus on envisioned use scenarios that guide the development of the system, which includes both the actual scenario flow as well as

personal motivations and histories for each actor in the scenario [50]. Drawing from this work, Figaro prompts demonstrators to create scenes of an interaction by thinking about the various ways that a robot might, or should be used, but does not prompt them to think about the background or motivations of the actors. Figaro also takes inspiration from design methods such as speed dating [18], bodystorming [41], and more traditional user enactments [40], all of which immerse designers in the context of an interaction through the use of role-playing and environmental similarity.

2.2 Programming Interfaces in Human-Robot Interaction

Prior work on robotic programming tools spans expert text-based frameworks (e.g., [6, 48]) to more user-friendly interfaces (e.g., [21, 26]). Some of these tools target general use (e.g., [48]), while others focus specifically on social robotics (e.g., [46, 47]). Previous work in user-friendly HRI authoring tools includes an extensive set of visual programming interfaces (VPE's) such as *Interaction Composer*, which similar to Figaro represents interaction programs as a collection of states and transitions [21]. However, Interaction Composer, in addition to other robot programming VPE's (e.g., [3, 46]), assumes that the designer possesses some programming knowledge. The authoring tool *Code3* addresses differences in designers' skills by providing different authoring layers suitable for different levels of expertise [26]. The assistance provided to designers takes many forms, from abstracting away low-level design details [54] to assisting designers by using methods such as formal verification [46]. Other VPE's have shown the efficacy of birds-eye-view maps where the user specifies movement paths and other logic elements [13, 16, 30, 32].

Given the challenges of programming robots with traditional text-based frameworks and VPEs, particularly for novice programmers, recent robot programming interfaces involve less traditional means of capturing user intent, such as through natural language [66], demonstration through tangible programming [56] and tabletop interfaces [20], or role-playing scenes of an interaction with a partner [45]. These interfaces seek to capture the results of a more immersive design process and automatically synthesize it into a program, addressing many of the issues novices face in traditional programming.

Creating a natural mapping between interface control and functionality is a key challenge in interface design [39]. This challenge is exacerbated for interfaces with functionality that does not necessarily map to a physical domain [19]. Prior work on bodystorming HRIs, which leverages the embodied nature of both the bodystorming design method and the interaction itself, shows that designers feel that the interaction can seem artificial, making it difficult to capture the actual nuances of an interaction [45]. To avoid the artificial nature of bodystorming, Figaro borrows from shadow puppetry to use a "stage," a set of figurines, and projected props to prototype an interaction in a situated, tangible way. Prior work has argued that shadow puppetry offers an appropriate framework for designing interactive media [42]. Building on the puppetry metaphor, Young et al. developed Puppet Master–a system for designing reactive virtual agents through demonstrations that express style, personality and emotions [67, 69]. The idea of the use of figurines for design expression is supported by concepts from experience prototyping [10] and sketching [12], which both suggest that using physical artifacts in the design process allows the designer to more intensely experience and understand the design scenario.

Tangible user interfaces (TUIs) leverage small objects as programming manipulatives to close the gap between how we conceptualize goals for programs and then communicate those goals to our program [27]. Given the physical nature of robots, robot programming is a natural application of TUIs. Prior work exploring children's use of TUIs shows that they offer several key advantages, namely lowering the barrier of entry for non-programmers [51]. Sapounidis et al. [52], found that compared with using a graphical user interface (GUI), children using a TUI to program robots were able to complete tasks more quickly and with fewer errors. Tools for programming robot manipulators have also used TUIs [20], significantly reducing training time compared to traditional programming interfaces.

Tabletop TUIs have been used in numerous applications: urban planning [63], theater stage planning [23], flight-clearing of naval aircrafts from a ship [15], augmenting physical objects to create tangible displays [17], and robot navigational tasks [31]. Across these different interfaces, a few common benefits emerge: it is easier to collaborate in the tabletop interface, and for spatially-focused tasks, the visualization offered a much more intuitive experience. Figaro expands the use of these tabletop TUIs in robotics programming by applying it to the novel domain of HRI.

Various HRI programming interfaces, tangible and non-tangible alike, offer similar interaction paradigms to Figaro, but unlike Figaro, are not intended for the demonstration and generalization of high-level interaction flows. Reactile, for instance, offers a tabletop interface but excels in programming the fine-grained coordination of swarm robots [60]. The "style by demonstration" approach coincides with Figaro's design of dyadic interactions but is used to craft physical reactions to stimuli [67–70]. At the level of specifying interaction logic, users of Picode demonstrate individual behaviors but must program flow textually [28]. At an even higher level, Magic Cards reasons about tasks in a physical environment but abstracts steps necessary to perform them [71]. Figaro's interface also bears striking similarity to educational interfaces such as Cellulo, where children interact with small robots in a simulated layout but do not program the robots themselves [43].

2.3 Synthesizing and Representing Programs

In facilitating the collection of scenes to serve as input to program synthesis, Figaro draws from *keyframe demonstrations* to program interactions, in which significant events in the interaction are demonstrated but the precise trajectory taken by the robot is left to the synthesizer [1]. Prior work has also explored program refinement after one or more demonstrations have been provided, such as by interacting with a visualization of the program [2] or by offering critiques after an initial program has been created from demonstration [5].

In our particular approach, Figaro uses inductive synthesis—learning a program from a small set of examples [22]—to construct HRI programs from designer input. An example of inductive synthesis is TRANSIT, which synthesizes protocols from execution traces [62]. Other prior work has generalized sets of execution

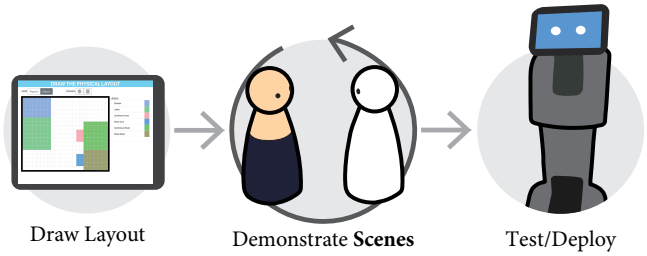


Figure 2: The design process that Figaro facilitates, from drawing a physical layout (left), leading to demonstrating scenes (center), and ending with testing/deployment (right).

traces into automata [38, 45]. In an inductive synthesis approach, counterexamples from the synthesized program can subsequently be used as input to further iterations of synthesis [58]. The goal of Figaro is similar to that of *automata learning*: constructing an automaton by querying whether or not traces are accepted by a target system [4, 49, 59]. In classical automata learning, the target system is already specified and the algorithm seeks to build a model of that system, but for Figaro, the target system is not yet specified, so the synthesis must both create and model the target system.

While Figaro creates interaction programs that solely dictate the robot's behavior based on sensory input, prior work has developed representations that include both human and robot actions in the execution of a human-robot collaboration plan [37]. Other prior work in programatically representing interactions reasons about timing between the robot's behaviors in order to create fluent interactions [14, 61], while Figaro reasons only about the ordering of discrete events.

3 FIGARO: DESIGN AND IMPLEMENTATION

In this section, we describe the technical details and implementation of Figaro,[1] beginning with the design pipeline that Figaro facilitates. We then introduce a running example of a potential design scenario, and subsequently describe each component of Figaro in detail using the running example for illustration.

3.1 Design Pipeline

Figure 2 depicts the design pipeline facilitated by Figaro. In general, demonstrators will draw the physical layout of the environment in which the robot will function and then iteratively demonstrate different scenes of how the robot should interact with people, objects, and regions within its environment. Between iterative demonstrations of scenes, Figaro will synthesize the full interaction program in the background and query demonstrators about ambiguities within their designs. At any point, demonstrators can compile these interaction programs onto a robot for testing/deployment.

The components of Figaro, shown in Figure 3, facilitate the design pipeline described above. The Figaro system includes (1) a tablet control interface through which demonstrators can enter various design *modes*; (2) two wooden peg doll figurines that represent the human and the robot; (3) a projected tabletop design interface used

[1] An open-source implementation of Figaro is available at https://github.com/Wisc-HCI/Figaro.

Figure 3: The Figaro system includes the tabletop interface connected to a computer, a tablet control interface, and a robot for deploying designs.

by demonstrators to manipulate figurines to demonstrate scenes; (4) an audio component that listens to and interprets demonstrators as they narrate human and robot speech; and (5) a program synthesizer for interpreting and building programs from collections of scenes.

3.2 Design Scenario

Consider the following toy example: A demonstrator wishes to program a household robot that can host video calls on a tablet, which we refer to as the *Call* robot. Upon request, the robot should approach a human. If the human is standing, the robot may need to adjust its screen to suit the height of the human. The human is then free to initiate a call by interacting with the tablet. Until a video call is requested, the robot will move around the house looking for people to interact with. Given these capabilities, the details of how the robot will complete these tasks are left to the demonstrator. In the following sections, we describe how each component of Figaro facilitates the design of this scenario.

3.3 Control Interface

The tablet interface provides access to various *modes* (Figure 4) as demonstrators progress through the design process, while also ensuring that demonstrators are not confined to the tabletop at all times. The primary modes of Figaro are (1) *draw* a physical layout, (2) *create* scenes, and (3) *resolve* ambiguities. From the home screen, a demonstrator can enter draw mode or create mode, and resolve mode is automatically triggered whenever a design ambiguity needs resolution. The tablet displays the current mode as well as options within that mode. We briefly introduce the modes below.

Draw a Physical Layout—Through the tablet, Figaro allows demonstrators to draw a coarse physical environment within which the robot will be placed (Figure 4, top-right). A drawn environment has two components: *regions* and *objects*. The physical layout is split into a grid that can be used to specify regions, which represent specific areas of the space recognizable to the robot, such as a room or a certain area within a room. Shapes can be drawn on top of the

physical layout grid to specify physical items in the regions. The demonstrator can also specify the names of regions and objects. Due to the low-fidelity nature of drawn environments, Figaro is best suited for cases in which only discrete locations need to be incorporated into the final programs, rather than the exact physical characteristics of objects and regions.

Create Scenes—Creating scenes encompasses the main part of the design process. Within the create mode (Figure 4, bottom-left), the demonstrator can start and stop scenes, remove existing scenes, and view details of previously created scenes.

Resolve Ambiguities—The resolve mode allows Figaro to query demonstrators for additional information or to reconcile conflicting information. As a demonstrator narrates a scene, they may verbally specify a behavior that Figaro cannot recognize, e.g., failed speech recognition or categorization. After the scene is complete, the tablet will prompt demonstrators to clarify their intent (Figure 4, bottom-right). Ambiguities are further discussed in *Audio Interface*.

Illustrating Design Flow with the Control Interface. The demonstrator for the *Call* robot must first specify the household's physical layout. They click *Draw Physical Layout* on the tablet home screen to enter draw mode. The robot should begin at its charging station, so the demonstrator specifies a *charge* region. The demonstrator draws other areas recognizable by the robot, such as the kitchen and a bedroom. When the demonstrator is satisfied with the drawn layout, they return to the home screen via the *Done* button. Then, the demonstrator presses the *Create* button to enter create mode. When the demonstrator wants to begin a scene, they press *Begin a scene*. Once the scene is complete, the demonstrator presses *Done* and the tablet returns to the create mode. From there, the demonstrator can choose to create additional scenes using the same process.

3.4 Demonstrating Scenes

When demonstrators initiate a scene, Figaro passes control from the tablet to the tabletop. We describe the three main components of the tabletop that facilitate scene demonstration: the tabletop design interface, the audio interface for natural language commands, and the instrumented figurines.

Figure 4: The tablet interface and its various modes, including the home screen (top-left), the draw mode screen (top-right), the create mode screen (bottom-left), and the resolve mode screen (bottom-right).

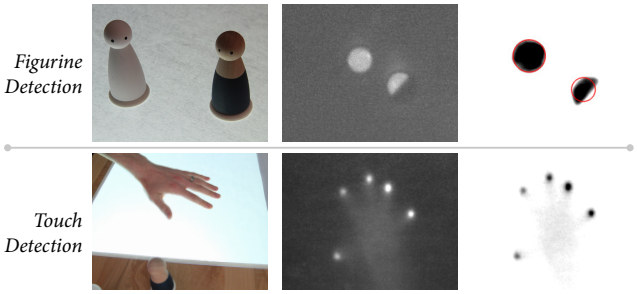
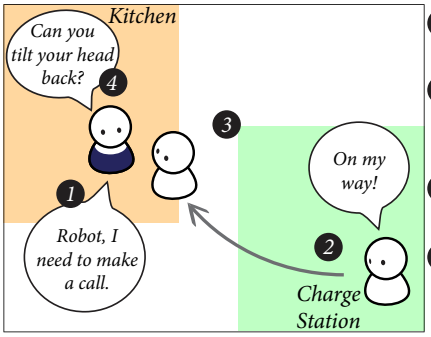


Figure 5: Figaro tracks the position and orientation of figurines using an infrared camera and performing blob and pattern detection.

Tabletop Design Interface. Within a scene, the demonstrator manipulates the positions and orientations of the figurines on the tabletop to demonstrate the behaviors of a robot or a human in the environment. As the demonstrator manipulates the figurines, Figaro (1) projects the drawn physical layout onto the tabletop from underneath its surface, (2) tracks the positions of figurines, (3) detects when a demonstrator touches the tabletop surface, and (4) offers visual feedback to demonstrators.

Hardware—The tabletop design interface, measuring approximately $0.94\ m \times 0.92\ m \times 0.61\ m$, is comprised of a rear projection setup similar to the *Wipe-Off* interface [44]. A projector from underneath the table shines an image of the physical layout to tracing paper attached to the underside of a clear sheet of acrylic, while an 850 nm infrared camera with a built-in infrared lamp tracks the position and orientation of figurines from under the table. We use OpenCV to detect the position of figurines by placing simple geometric shapes under each figure – a half-circle for the robot and a circle for the human – and using blob detection to isolate the shapes and pattern detection to assign them identities [8]. Furthermore, the orientation of the half-circle under the robot figurine is used to calculate its orientation. As Figaro tracks the movement of figurines across the board, it displays a trail of movement onto the projected display interface to provide the demonstrator with visual feedback that the figure is being tracked correctly. Figure 5 (top) depicts our approach to figurine detection.

Lastly, two 850 nm infrared LED light strips span the length of the perimeter of the acrylic to enable fingertip touch detection via frustrated total internal reflection. When demonstrators touch the board, Figaro highlights the area that was touched in a circle to provide visual feedback that the touch has been registered. While demonstrating scenes, basic touch capability allows the demonstrator to focus solely on the tabletop without needing to shift their focus to the tablet. Instances of the touch input capabilities include pausing/playing the scene creation or interacting with dialog boxes that Figaro displays (further discussed in *Audio Interface*). Figure 5 (bottom) depicts our approach to touch detection.



1. The human issues a *CallRequest*.
2. The robot responds and moves to the kitchen.
3. The robot arrives at the kitchen.
4. The human asks the robot to move its head back.

Figure 6: Part of a scene in which the human issues a request to make a video call on the robot and the robot travels to the human's location.

Audio Interface. In addition to manipulating the figurines, demonstrators must also specify the speech uttered by the human and robot. Thus, Figaro listens for *speech commands* as demonstrators perform scenes. We define a speech command as a specification for the speech that the robot should utter or the speech that it should be able to recognize at a particular moment in time.

Figaro uses Mycroft-precise [2] in order to quickly detect human or robot speech commands. If a demonstrator says "robot say" or "human say," Figaro knows to begin listening for robot speech or human speech, respectively. Upon detection of a hotword, Figaro will initiate real-time speech-to-text transcription.[3] As demonstrators speak a command, their speech is transcribed on the tabletop in real-time. Transcription concludes when Figaro hears a pause in the speech. The demonstrator can touch the displayed speech to flag it for later review if, for example, the uttered speech was incorrectly transcribed.

The transcribed utterance must then be classified. Figaro feeds the utterance into off-the-shelf intent recognition software to perform this categorization,[4] which is pre-trained to recognize a small set of general speech categories (e.g., greeting, farewell, thanking). Figaro marks any speech that either cannot be classified or was flagged by the demonstrator as *undefined*. After a scene is complete, Figaro will prompt the demonstrator to resolve undefined utterances within the resolve mode of the control interface (Figure 4, bottom-right). The demonstrator is queried to either classify the speech within one of the general categories, define a new category for speech if no suitable existing category exists and add additional speech examples to the new category, or discard the speech. Demonstrators may also edit the transcription. New speech categories can be re-used in future scenes by either stating the category name or a phrase corresponding to the classification verbatim.

Illustrating the Start of a Scene. The demonstrator wants to play out a scene in which a person in the kitchen (1) requests to make a video call with the robot and (2) asks that the robot tilt its head up to handle the case in which the human is standing and doing

[2]Hot-word detection is done using Mycroft-Precise, which can be found at https://github.com/MycroftAI/mycroft-precise
[3]Speech to text is performed using Google Cloud Text to Speech, found at https://cloud.google.com/text-to-speech/
[4]Intent recognition uses Dialogflow: https://dialogflow.com

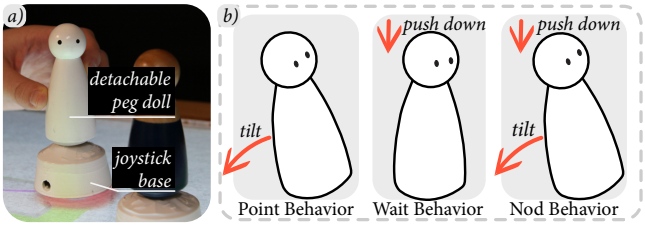


Figure 7: (a) The wooden peg doll figurines attach to joystick bases. (b) The three behaviors to which Figaro maps joystick movements – pointing, waiting, and nodding.

dishes. Before beginning the scene, the demonstrator places the human figurine in the kitchen region of the environment and the robot in the charge region. The demonstrator then uses the control interface to begin a scene. The demonstrator utters

Human say: Robot, I need to make a call.
Robot say: On my way.

For the purpose of illustration, we will assume that Figaro's speech classifier recognizes and classifies the phrase *"robot, I need to make a call"* as *CallRequest* and the phrase *"on my way"* as *Respond*. The demonstrator then moves the robot figurine to the kitchen and subsequently issues an additional utterance:

Human say: Can you tilt your head back?

Suppose that the human's speech phrase *"can you tilt your head back"* does not map well to any of the general speech categories recognizable by the robot. Figaro flags this speech as undefined, and will prompt the demonstrator to resolve it later by creating a new speech category on the tablet (Figure 4, bottom-right). The in-progress scene is illustrated in Figure 6.

Figurine Controllers. At the current point in the scene, the human has asked the robot to tilt its head back so that the human can have a better view of the video call. To finish the scene by responding to the human's request, the demonstrator will need to enact a head nod. While it is possible to verbally specify actions such as a head nod, as was done in an early iteration of Figaro, doing so increases the chance for speech misclassifications. Thus, Figaro enables demonstrators to enact robot actions directly on the figurine controllers, with the goal of decreasing speech classifications and providing a more direct mapping between demonstrator intent and robot behaviors. To enable this enactment, the figurines are affixed to a joystick controller base (Figure 7a).

The joystick has three output voltages – x and y axis of the joystick motion, and button displacement. These three measures are transmitted via Bluetooth Low Energy (BLE) [5] [6] to Figaro. We enclose the electronics in a 3D printed case affixed to each figurine.

Figaro then classifies combinations of voltage signals into one of three possible categories – a *pointing* behavior, a *head nod* behavior, or a *wait* behavior. Figure 7b depicts how each behavior can be achieved on the figurine. If the figurine is tilted far enough in a particular direction, Figaro recognizes a point behavior. While

[5]We use the Adafruit Feather NRF52840: https://www.adafruit.com/product/4062
[6]The microcontroller was programmed with CircuitPython, which can be found at: https://circuitpython.org using the Adafruit_CircuitPython_BLE library, which can be found at https://github.com/adafruit/Adafruit_CircuitPython_BLE

pointing, clicking the figurine causes Figaro to recognize a nod behavior, downward if tilted forward and upward if tilted backward. Clicking the figurine without tilting it causes Figaro to recognize a wait behavior, in which the robot idles.

Illustrating the End of a Scene. Simulating the robot responding to the human's request to tilt its head back, the demonstrator pushes down on the figurine and pulls it backward on the joystick, causing Figaro to recognize a nod behavior. The human may then complete their video call on the robot, which we consider to be an action native to the tablet interface on the robot rather than one of the robot's programmable behaviors. Finally, the demonstrator simulates the case where the human has finished their video call by uttering:

Human say: You can leave now.

Figaro's classifier successfully recognizes this text as *ExitRequest*. The demonstrator then releases the figurine back to its resting position and slides the robot figurine back to its charging station.

3.5 Program Synthesis from Scenes

Figaro captures detailed raw information about the human and the robot figurine including figurine position and orientation, joystick manipulations, and timing for all behaviors. To synthesize a high-level program, our synthesis approach must abstract the relevant information from the multiple demonstrations that include such low-level data. The purpose of these abstractions is to balance including a sufficient amount of detail in the program with enabling our synthesizer to generalize full programs from a series of demonstrations. Our synthesis approach is as follows: we first convert each scene to an execution trace, then feed each trace into a synthesizer which constructs a program that accepts each trace.

Converting Scenes to Traces. The raw output from each scene consists of a set of parallel recording tracks captured from the tabletop, audio, and figurine input signals. Figaro first processes the recording so that it adheres to any parameters that are specific to the particular robot being used. For example, if a robot can only sense whether a human is close to it but cannot sense the human's exact location, the signal describing the human's position on the tabletop will be converted to a binary value indicating whether the human is close to or far from the robot.

Next, Figaro compresses the information within the set of modalities into into three categories—(1) the discrete *behaviors* that the robot exhibits, (2) any *event* triggers that the robot can notice, and (3) the state of the *environment* that the robot can perceive. We denote the set of behaviors that the robot can exhibit as B, the set of event triggers that the robot can recognize as E, and the set of different environmental states the robot can observe as V.

Figaro then bins the processed recordings by intervals of time. The start of a bin is denoted either by an event trigger, such as the human uttering speech to the robot, or by the first action taken by the robot. The end of a bin is denoted either by the next event trigger or the end of the recording. At the start of the bin, Figaro records the state of the environment. Within a bin exists any sequential and concurrent behaviors that the robot exhibits between triggers. If within a bin the robot exhibits two sequential behaviors of the same behavioral modality (e.g. one utterance followed by another different utterance), the bin encompassing those behaviors will

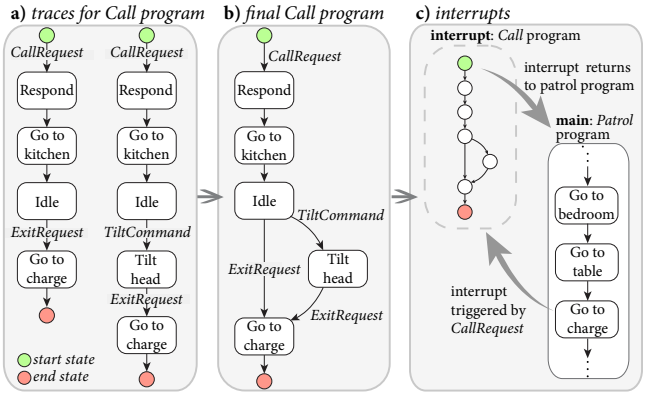


Figure 8: Our synthesis approach within Figaro, which takes (a) the set of demonstrator traces and (b) combines the traces into a program. For simplicity, environmental state has been removed from transitions, and unannotated arrows represent the *self* transition. (c) The *Call* procedure serves as an interrupt to the main *Patrol* procedure.

be split into two bins, in which the trigger for the second bin is considered to be *self*, indicating that the robot triggered its own behavioral change.

In the next step, Figaro converts each bin to a trigger-environment-behavior triple, denoted as $\xrightarrow{\langle e_i, v_i \rangle} b_i$, where $e_i \in V$ is the event that triggered bin i; $v_i \in \mathscr{P}(V)$ is the environmental state recorded at the start of bin i, denoted as an element of the powerset of V since Figaro observes multiple environmental states simultaneously (e.g. the robot may observe itself to be in a particular position of the room while at the same time observing that it is proximal to the human); and $b_i \in \mathscr{P}(B)$ comprises the behaviors observed within bin i, denoted as an element of the powerset of B since Figaro can observe multiple robot behaviors within a particular bin. If bin i is the first in the sequence, Figaro deems the event that triggered the bin to be *self* and the environmental state to be the state recorded at the start of the recording.

After all bins have been converted to trigger-environment-behavior triples, Figaro creates an accepting triple $\xrightarrow{\langle e_n, v_n \rangle} b_n$ in which $e_n = self$, v_n is the environmental state recorded at the end of bin $n-1$, and $b_n = \varnothing$. Finally, Figaro assembles an execution *trace* by linking consecutive trigger-environment-behavior triples. Formally, we represent a trace as a sequence of triples: $\xrightarrow{\langle e_0,\ v_0 \rangle} b_0 \xrightarrow{\langle e_1,\ v_1 \rangle} b_1 \xrightarrow{\langle e_2,\ v_2 \rangle} \ldots \xrightarrow{\langle e_n,\ v_n \rangle} b_n$.

Constructing a Program from Traces. Given a set of traces T, our task is to construct an interaction program I that accepts each trace. The program I is formalized as a tuple (S, s_0, s_f, f_T, f_B), where S is the set of states; $s_0 \in S$ is the initial state; s_f is an accepting state with no outgoing transitions; $f_T : S \times E \times \mathscr{P}(V) \to S$ is the transition relation, representing how the state changes in response to event triggers and depending on the environment; and $f_B : S \to \mathscr{P}(B)$ is a labelling function that assigns robot behaviors to states.

Synthesis algorithm—Our goal is to synthesize a program I that is small and generalizes to unseen scenes (traces). To that end, we

implement an approach similar to that used by Porfirio et al. [45] to efficiently search the space of programs I with the goal of accepting every trace $t \in T$. This task is made simpler by abstractions performed by Figaro, including, for instance, the conversion of movement trajectories into binary values indicating whether or not the robot is moving, and if so, the robot's target destination. Thus, multiple instances of movement to the same target destination, regardless of the specific trajectory taken by the robot, can be combined into the same behavioral state. A simple example of our synthesis algorithm is depicted in Figure 8, in which Figaro combines the set of traces collected from user demonstrations (Figure 8a) into a program (Figure 8b).

In a synthesized program, the robot will transition from behavior to behavior only if an appropriate event trigger is produced and the specific conditions of the transition's environmental state are met. However, it is often the case that the environmental state recorded by Figaro for a transition to occur is too strict. For instance, in the *Call* program, if the demonstrator always starts a scene with the robot in the living room, the resulting program would necessitate that the robot starts in the living room in order to successfully complete the interaction. Thus, Figaro loosens the preconditions for a robot behavior to be performed such that if a behavior has only one outgoing transition, Figaro will remove the environmental state from that transition.

Interrupts—Our synthesis approach supports proceduralization of behaviors through *interrupts*, or modularized interaction transition systems that can be executed at any point in time if a set of conditions are met [21]. When the robot executes its interaction program, it begins with a *main* procedure. The main procedure includes all traces that begin with a *self* action on the robot. Therefore, the main procedure involves the robot taking initiative to perform actions. If no such traces are provided, the main procedure can be automatically initialized as a simple wait loop in which the robot waits indefinitely. At any point during execution of the main procedure if the initial conditions for an interrupt are met, the robot will halt execution of the main procedure and execute the interrupt. Thus, demonstrators need only specify the robot's reaction to an event only once, rather than specifying the reaction in multiple demonstrations. For example, if the demonstrator wants the robot to react the same way every time the human says "thank you," the demonstrator need only create an interrupt triggered by a *thank* event. Once an interrupt finishes, it returns to the state from which it was called. Figaro's synthesis of the main procedure and interrupts do not differ. Thus, an interrupt beginning with an event $e \neq self$ consists of all traces beginning with e.

Illustrating Program Synthesis. In addition to demonstrating the scene in which the robot is asked to adjust its tablet, the demonstrator may also have provided an additional scene in which the human does not ask the robot to adjust its tablet. Figure 8a shows the traces created from these two scenes, and Figure 8b illustrates the synthesis of a complete *Call* procedure from these traces. Both traces are accepted by the final program.

Figure 8c demonstrates a potential application of an interrupt within the patrol robot design. Recall the design criteria for the *Call* robot that the robot should move around the house looking for interaction partners when it is not assisting the human with a video call. To demonstrate this behavior, the demonstrator records a *Patrol* scene in which the robot figurine simply moves between different regions in the drawn layout. The patrol scene results in the partial interaction trace shown in 8c. Since the *Patrol* procedure is initiated by the robot, it is designated as the main interaction. The *Call* program is therefore an interrupt that can be triggered at any point during the *Patrol*, regardless of what room the robot is in. When the interrupt finishes, it will return to the state in the main interaction before the interrupt was triggered.

3.6 Deployment

After creating scenes, the synthesized program is available to be tested on a robot. Currently, we deploy designs on the Temi[7] and virtual Pepper [11] platforms. Figure 9 illustrates what the execution of the *Call* program looks like on the Temi robot.

Lastly, we implemented each figurine behavior on the Temi. Because Temi does not have arms, to execute a point behavior the robot rotates its orientation towards its target, tilts its head down and back up, and then rotates back to its original orientation. To execute a head nod, the robot moves its screen up or down. In a wait behavior, the robot waits passively for input for an extended period of time.

4 EVALUATION

We evaluated Figaro with a user study in which participants used Figaro to complete a human-robot interaction design task. In our evaluation, we pay particular attention to design experiences and techniques exhibited by demonstrators, and assess the effectiveness of Figaro's physical and visual components in furthering demonstrators' experiences.

4.1 Study Procedure

Upon providing informed consent to participate in the study, participants were guided through an interactive tutorial on using Figaro. The tutorial included moving figurines on the tabletop, using the robot figurine to emit point, nod, and wait behaviors, and demonstrating scenes. Following the tutorial, we presented participants with a design scenario for a human-robot interaction at a museum with three exhibits, and included a pre-made physical layout for them to use. The scenario was purposefully left open-ended such that participants could prioritize a variety of different interaction paradigms—for instance, whether to keep the robot stationary and create a conversational interaction, or whether to have the robot guide visitors around the museum. Depending on the time remaining in the study after the completion of the tutorial, participants were allowed approximately 30 minutes to design their interactions. We then conducted a semi-structured interview and asked participants to fill out a questionnaire.

4.2 Measurement and Analysis

We conducted a reflexive Thematic Analysis (TA) on the interview transcriptions following the guidelines by Braun et al. [9] and McDonald et al. [34]. Two authors who facilitated the study and transcription process worked individually to generate potential

[7] https://www.robotemi.com/

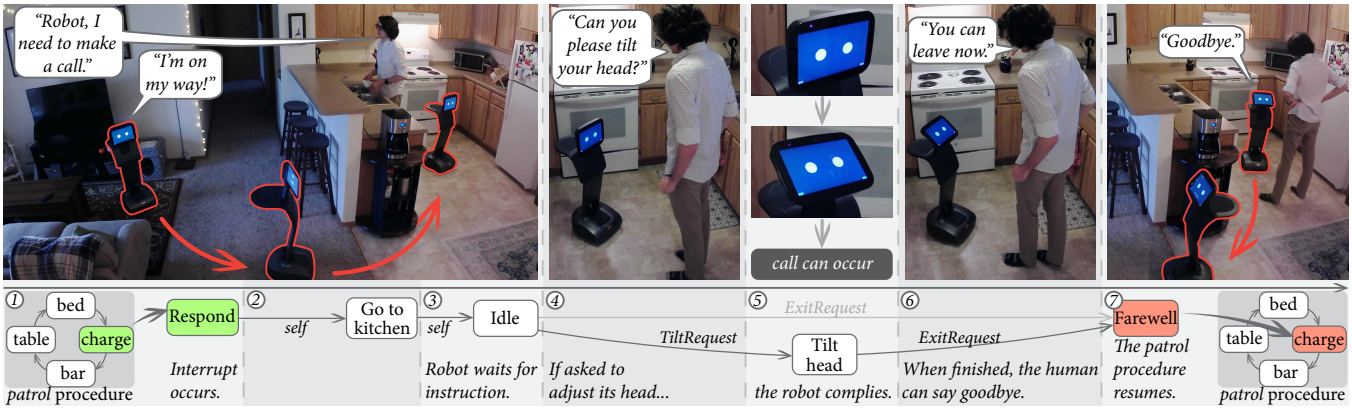


Figure 9: An example deployment of the *Call* program on the Temi robot, split into seven steps. Detail has been removed from the *Patrol* procedure for simplicity.

codes, and then regularly discussed candidate themes. Then the authors revised and combined candidate themes until a final set was established.

Lastly, we measured the effectiveness of various aspects of Figaro through a twenty-one item questionnaire. Four items measured the perceived effectiveness of scene *demonstration* as the primary design method ($\alpha = 0.71$), five items measured the perceived effectiveness of Figaro as a tool for individual usage as opposed to working with a partner ($\alpha = 0.85$), and three items measured the accessibility of Figaro's interface ($\alpha = 0.64$). Of the remaining nine items, we extracted a single item as a measure of the perceived importance of physicality in Figaro's interface and an additional single item as a measure of Figaro's ability to immerse demonstrators into the environment simulated by the design scenario. The remaining seven items could not be grouped into coherent categories with sufficient consistency.

4.3 Participants

We recruited 10 participants (6 males, 4 females) to participate in our study, with an average self-reported programming experience of3.25 ($SD = 2.97$) years. Five participants reported a background in Computer Science, two in Industrial and Systems Engineering, and one each in Mechanical Engineering, Integrative Biology, and Speech Language Pathology. Four participants self-described themselves as having little experience programming robots, while the remainder described themselves as having none. One participant self-reported themselves as having design experience, while three others described themselves as having "some," one as having "little," and five as having no design experience at all.

4.4 Results

In our analysis of participant interviews, we identified four main themes that emerged from the use of Figaro: (1) Expressing ideas tangibly, (2) Methods of idea generation, (3) Usability of the system, and (4) System feedback and limitations. Within each theme we include quotes from the participant interviews, which are attributed using both participant ID and line number in the interview transcripts (e.g. P4.12 means "participant 4, line 12"). Within our thematic analysis we also uncovered various subthemes, but we reserve these subthemes for our discussion since they contain our interpretations of participant feedback.

Theme 1: Expressing ideas tangibly. Seven of the ten participants discussed the tangibility of the interface in-depth. Particularly, P9 referred to the expressiveness of gestures:

> *Um, pointing is basically without saying you can direct [...] the human where to go. It's* ***answering by doing rather than by speaking****. So, I think it's [...]* ***very expressive****. (P9.34)*

Furthermore, participants described how tangibility affected organizing and expressing their understanding of the interaction:

> *And so by doing it physically [...] you're able to do it,* ***it's simulated exactly they way you'd want to do it*** *in real life. Because it's,* ***you're limited by physicality****. (P10.91)*

Furthermore, several participants discussed what it might be like to use input modalities other than the tangible figurines:

> *...if it [...] were to be a little bit more concrete* ***where I could have total faith in the movements of the figurines****, um, then maybe I'd be [...] more inclined to use it, because* ***it's also just kind of fun****. (P10.72)*

Theme 2: Methods of idea generation. Nine of the ten participants discussed how the interface affected their ideation process. Several participants commented on how Figaro affected their use of imagination:

> *I think it's pretty easy to be like,* ***I can imagine a robot being in the middle of the floor****, so I'll put him in the middle of the floor and I'll walk up to him,* ***or imagine him standing next to me*** *and ask me a question about what I know more <unintelligible> painting. (P5.42)*

Participants also remarked on the effect of Figaro as a visual aid on their imagination:

> *And because there was visual aid [...] to support it, [...]* ***it was easier for me to imagine****. So maybe if I didn't have this and if I were to kind of imagine and, you know, program, maybe some other scenarios I would have* ***not even been able to come up with****. (P7.29)*

Furthermore, participants commented on Figaro's ability to connect their designs to the real world:

> *Okay, so, I thought personally that was very useful. I like to see things visually [...] So **having those physical things** in front of me and **being able to move them around** with my own hand and like pointing, that like kind of **makes it more realistic to me, and more human-like**. (P8.24)*

Several participants remarked on the physical limits imposed by the figurines on their designs:

> *And so by doing it physically, [...] you're able to do it, it's **simulated exactly the way you'd want to do it** in real life. Because [...] you're **limited by physicality**. (P10.91)*

Participants such as P4 also remarked how the visual and tangible nature of Figaro affected their creativity:

> *Uhm it was good to see like the **visual layout** of everything. I felt like it definitely **made me more creative** (P4.50)*

Additionally, participants commented how acting out the scenarios affected how they viewed their interactions unfold:

> *[...] I felt like [...] I was able to [...] **enact the actual scenario**, so, it kind of **made it easier to program** [...] so when you program it this way, maybe [...] the interaction between the human and the robot could improve. (P7.21)*

Theme 3: Usability of the system. Nine out of ten participants referred to the usability of the system. One participant, for example, references usability while describing how they corrected a mistake in their speech during a scene:

> *But, I think there was one scenario wherein I had to like **reword the phrase**. So because I had that option **it was easy**. (P7.26)*

Four participants also remarked on the learnability of Figaro, stating that it was difficult to pick up quickly or that they forgot how to perform certain tasks. P1 in particular remarked that they had to adjust to using the system:

> *Yeah I think that kind of goes back to like, [...] **getting more comfortable with it I think over time**, where at first like saying it out loud was kind of weird, but then as I got through it, I was like oh, okay, like, **this isn't that bad, this isn't that weird**. (P1.26)*

Other participants, however, commented positively on Figaro's approachability. Some suggested that this would be a suitable system for non-programmers, and P4 mentioned that Figaro seems approachable for children:

> *It seems like something you could use to you know maybe like to **elementary school kids**, [...] **people who are new to coding**. (P4.4)*

Theme 4: System feedback and limitations. Seven of the ten participants referred to the feedback or limitations of the system. Participants expressed that the feedback helped them understand what was happening on the tabletop. Specifically, P8 comments on how well the feedback worked:

> *Um, yeah that was really good, like **wherever I pointed the robot was able to point an arrow out and give the name of the region**, so that was really good. (P8.35)*

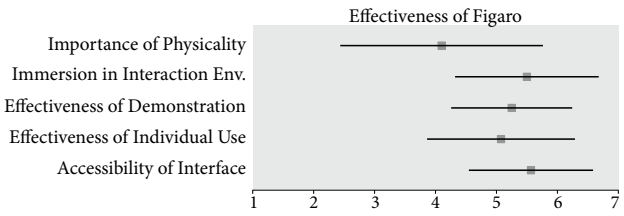


Figure 10: Results from the survey measuring the effectiveness of Figaro. Error bars represent standard deviation.

Participants sometimes expressed confusion, either that Figaro did not respond to an input that they expected, or that they were unsure how Figaro would handle certain situations they might create:

> *[...] I also **don't totally understand how they sort of stack up against each other**, so like... when is he asking a question vs if I did do patrol when is he patrolling? You know, there's some logic I guess that he executes situation A if it's 3 o'clock (P5.32)*

Effectiveness of Figaro. Figure 10 shows the results of the effectiveness survey. On average, participants rated the importance of physicality in Figaro to be 4.1 (SD = 1.66) and their immersion in the interaction environment to be 5.5 (SD = 1.18) on a 7-point Likert scale. Participants also rated their perceived effectiveness of demonstrations to be 5.25 (SD = 0.99), their perceived effectiveness of Figaro as a tool for individual use as 5.08 (SD = 1.21), and the accessibility of Figaro as 5.57 (SD = 1.02).

4.5 Usage Patterns

Below, we include descriptive results from observing participants use Figaro and the programs that they created.

Interface Usage. We examined the study video footage to reveal usage patterns of Figaro that emerged in the user study. Our analysis encompasses all scenes created by participants, even scenes that were not ultimately included in participants' final interaction designs. We found that six participants included dialogue and movement-based scenes; three participants created dialogue-based scenes, but the robot was moved in between scenes; and only one participant created a scenario purely consisting of dialogue. Ten participants utilized the point gesture on the robot figurine, three used the nod behavior, and five used the wait behavior. Six participants performed a gesture simultaneously with speech, and one of those participants also used a gesture simultaneously with movement. This combination was most often pointing and speaking. A variety of scene structures were used. Three participants demonstrated scenes *in-series* with each other, in which rather than each scene containing a categorical beginning and ending, each scene picks up where the previous one left off. Five others made *independent* scenes where they would repeat similar situations with different outcomes or were simply unrelated to each other, and three used a mix of the two methods. Some features of Figaro were not emphasized in Figaro's tutorial and were therefore not widely used by participants. Only two participants interacted with the touch table, three deleted scenes, and three added additional objects or regions to the physical layout.

User-Created Programs. In order to characterize the participant programs created with Figaro, we select a small set of participant programs to describe in detail.

Participant 1—This participant demonstrated seven scenes. Rather than representing each scene as an individual, disjoint interaction between the human and robot, participant 1 crafted each scene in-sequence as part of a larger interaction story.

> *Scene 1:* The robot greets the human, and the human greets back. The robot then offers assistance.
>
> *Scene 2:* The human asks where exhibit 1 is. The robot moves to the exhibit, states that the exhibit is "over here," and then beckons the human to "follow me." The human then moves to the robot's location.
>
> *Scene 3:* Starting from exhibit 1, the human asks for information about the exhibit, and the robot responds accordingly.
>
> *Scene 4:* Starting from exhibit 1, the robot prompts the human to ask for more information about the exhibit. The human asks for more information, and the robot answers accordingly.
>
> *Scene 5:* Starting from exhibit 1, the human thanks the robot, and the robot offers to direct the human to exhibit 2. The robot then says "Follow me" and moves to exhibit 2.
>
> *Scene 6:* Starting from exhibit 2, the robot beckons the human to approach the exhibit. The human approaches the robot in exhibit 2, and states that they do not want to go to exhibit 2. The robot asks "Why not?"
>
> *Scene 7:* Starting in exhibit 2, the human asks the robot to point the way to exhibit 3. The robot moves to exhibit 3 and answers that exhibit 3 is "to my left."

As Figaro's synthesis approach does not connect scenes end-to-end, it is not possible to reproduce an uninterrupted sequence of scenes, beginning from the greeting in scene 1 and ending with the approach to exhibit 3 in scene 7, within a single execution trace of the synthesized program. Instead, all seven scenes are compiled into four program procedures. The main procedure includes scenes 1, 4, and 6, in which the robot will by default either greet the human, answer a question about exhibit 1, or beckon them to exhibit 2. The procedure generalized from these scenes contains program paths not specified by the participant, such as when the robot prompts the human to ask them more information about exhibit 1 (trace 4). If the human deviates from trace 4 and does not respond to the robot's prompt, the robot will end the interaction. Additionally, interrupts allow participant 1's scenes to execute repeatedly—at any point in the interaction can the robot handle being thanked within the interrupt that is triggered with scene 5.

Participant 1, in addition to many other participants, overgeneralized certain robot behaviors by using broad speech classifications. For instance, participant 1 categorized the phrases "Where may I direct you?" (scene 1) and "Why not?" (scene 6) both under the *Question* category. Therefore, in the main procedure, the synthesizer combined both distinct phrases within the same behavioral state, which can lead to nonsensical behaviors such as after an exchange of greetings the robot says "Why not?" rather than asking where it can direct the human.

Participant 10—This participant demonstrated five unique scenes. In the process, participant 10 demonstrated *conflicting* scenes that, if inserted into the interaction program, would result in nondeterminism. At the time of the user study, Figaro was not specified to handle conflicts.

> *Scene 1:* The robot waits for an event, then makes an announcement that it is available to be asked about the exhibits.
>
> *Scene 2:* The human approaches the robot, and the robot asks if the human would like to know about the exhibits.
>
> *Scene 3:* The human approaches the robot and asks who created exhibit 1. The robot answers accordingly.
>
> *Scene 4:* The human approaches the robot and asks for the location of exhibit 2. The robot points to exhibit 2 and then states "it is right over there."
>
> *Scene 5:* The human approaches the robot and asks for the location of exhibit 3. The robot verbally states the location of exhibit 3 and then points to the exhibit.

Scene 2 specifies that the robot should ask a question when the human approaches it, while scenes 3, 4, and 5 specify that the robot should remain idle. Thus, scene 2 conflicts with scenes 3, 4 and 5. Our description of participant 10's program considers the case in which only scene 2 is removed.

The four non-conflicting scenes are compiled into two program procedures. In contrast to participant 1, these scenes are independent of each other. The *main* procedure includes the exact execution trace created from scene 1, with no branching or looping. Thus, by default, the main procedure will cause the robot to idle, and then make an announcement after idling. A negative outcome of this program is the inability of the robot to idle indefinitely in the main procedure. Thus, after idling and making an announcement, the robot terminates the program and ends any further chance for a human to ask the robot a question. At any point in the main procedure in which the human approaches the robot, a *Question* interrupt will trigger that contains scenes 3, 4, and 5. The interrupt contains three branches, each involving the linear paths specified in scenes 3, 4, and 5 with no further branching or looping.

5 EXTENSIONS OF FIGARO

In addition to the use of Figaro as a design tool for individuals demonstrating the logic of interactions, we envision a number of extensions to Figaro that make it suitable for collaboration and the design of low-level interaction details. Below, we describe how Figaro currently supports collaboration and low-level design and how these capabilities can be further extended.

5.1 Figaro as a Collaborative Tool

Given the potential for multiple demonstrators to use Figaro at the same time, the authors of this paper underwent an exploratory collaborative design scenario for an *airport guide robot* who escorts people from the security checkpoint to their gate. The design team demonstrated three scenes. In the first scene, the human asks the robot for directions, and the robot successfully escorts the human to their gate. In the second scene, the human asks the robot for directions, and the robot attempts to escort the human to their gate

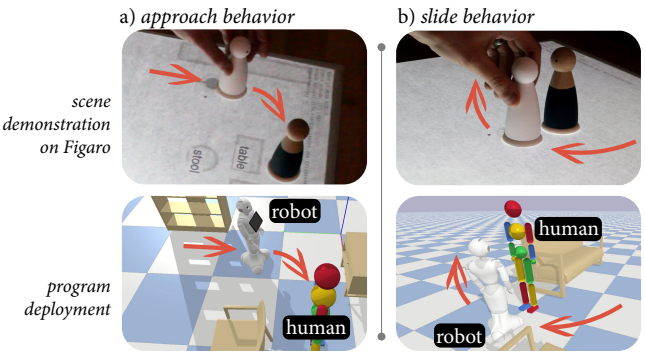


Figure 11: Two motion trajectories demonstrated where the robot approaches (left) and slides past (right) the human.

but the human is separated from the robot before arriving with the robot at the gate. In the third scene, the human stops the robot mid-escort to ask for the location of a different gate. The robot points in the direction of the other gate, and subsequently resumes escorting the human.

We observed two characteristics of this design session that differ from the individual design sessions in the evaluation of Figaro: (1) the design team discussed ideas before executing them, and (2) each member of the design team assigned themselves a particular role during the demonstration of scenes. One demonstrator performed the movements and speech of the robot figurine, while the other demonstrator did the same for the human figurine and controlled Figaro with the tablet. In our evaluation but not included in our thematic analysis, participants expressed thoughts on these characteristics, generally aligning in three different ways. First, having a partner could support design ideation:

> *Uhm well you'd definitely be able to have more input or more uh creativity on how different scenarios come... (P2)*

Second, having a partner could help coordinate the different components of Figaro:

> *Maybe one person will work, will take the uh, the robot part and then another will take the human part, and then let it interact freely like that way... (P9)*

Third, having a partner could cause the design process to become unwieldy:

> *Um, obviously the difficulty would come, you just have to be, you know, very clear communication in once you actually start a scene. (P10)*

Similar to participants' expectations, the authors found discussing scenes beforehand to be beneficial. However, a lack of coordination within the design team often caused mistakes. In one instance, the demonstrator controlling the robot prematurely ended a scene on the tablet. In another instance, the demonstrator controlling the robot paused a demonstration mid-scene to correct a mistake made by the other demonstrator in a previous scene to prevent the same mistake from happening in the current scene.

5.2 Figaro for Creating Motion Trajectories

Interaction designers may want precise control over a robot's position and rotation of its body to address certain design challenges, such as how a robot should approach a human [29, 53] or how a robot should navigate narrow spaces [57]. In each of these cases, the robot's exact position and orientation is important, which Figaro can record as motion trajectories. When creating trajectories with Figaro, each state in the resulting execution trace corresponds to incremental movement by the robot. Due to their high precision, multiple motion trajectories would not generalize well to a full program under our current synthesis approach.

We implemented and deployed two motion trajectory scenarios—an *approach* scenario shown in Figure 11a and a *slide scenario* shown in 11b. In the approach scenario, the robot figurine was moved slowly across the tabletop towards the human figurine. In the slide scenario the robot figurine was moved slowly towards the human, and then was rotated to slide past the human, simulating movement through a narrow space. We were able to replicate both behaviors on the virtual Pepper using Figaro.

6 DISCUSSION

Below, we discuss insights from our evaluation of Figaro, including our interpretations within each theme of our thematic analysis, which take form in *subthemes*. Following our discussion of subthemes, we synthesize all findings from Figaro and then discuss the limitations of Figaro and plans for future work.

6.1 Expressing Ideas Tangibly

The findings of our thematic analysis reveal two subthemes that provide insight into participants' experiences with tangibility. Within our first subtheme, we found that *Tangibility is Helpful*, as evidenced by participants commenting on the helpful nature of the gestures (P9.34, P10.91). For these participants, tangibility increased their perceived expressivity of Figaro (P9.34) and helped them organize and understand their interaction designs. In contrast, we found evidence for a second subtheme, *Hesitation in Tangibility*, due to participants expressing a lack of confidence when using the figurines (P10.72). While participants generally found tangibility to be enjoyable and helpful, some prefer less tangible input. Further exploration is needed to determine if this hesitation is due to tangibility in general or its implementation within the figurines.

6.2 Figaro as a Means of Idea Generation

In our thematic analysis, we split this theme into three subthemes that reveal ideation strategies used by participants. First, we found that Figaro fostered *Use of Imagination* within participants, which is evidenced by participants noting their use of imagination (P5.42) and stating that Figaro as a visual aid helped foster the exploration of ideas by facilitating their imagination (P7.29). Additional participant feedback led us to uncover a second subtheme, *Mapping to Reality*, which pertains to the physical nature of Figaro grounding participants in the real world. Participants stated that the physical nature increased the realism of the design interface (P8.24) and also helped impose physical restrictions on participants' designs (P10.91). Lastly, we determined our third subtheme, *Inspired by the Interface*, due to participants stating how Figaro affected their design process by fostering creativity (P4.50) or highlighting aspects of their interaction designs to improve (P7.21).

6.3 Usability of the System

Opinions on Figaro's usability were mixed, possibly due to the variety of backgrounds of participants in the study. Many comments align with our first subtheme, *Easy & Intuitive*, which is supported by positive participant experiences in tasks such as fixing mistakes (P7.26). Despite the intuitiveness of Figaro for many participants, we determined our second subtheme to be *Steep Learning Curve*. In particular, some participants viewed Figaro as difficult to pick up quickly, even if they were able to ultimately learn the system (P1.26). In contrast, we also found much evidence for a third subtheme pertaining to how *Approachable* Figaro is for newcomers. Several participants viewed Figaro as a highly approachable system, even for non-experts and children (P4.4).

6.4 System Feedback and Limitations

Participants expressed mixed views on the feedback offered by Figaro, or lack thereof. Some participants were satisfied by feedback given by Figaro on speech, gestures, and movement (P8.35). While Figaro's feedback on gestures and motion seemed to help participants grasp how to complete those actions, participants still expressed hesitation and confusion that they did not know how Figaro would interpret more complex actions or logic, such as what would happen if they initiated a point behavior and moved the robot at the same time (P5.32).

6.5 Synthesis of Findings

Despite mixed to positive feedback on the learnability of Figaro, the feedback that it provides to participants, and some hesitation with using its tangible components, our results support past findings surrounding the benefits of tangible and tabletop user interfaces. The helpfulness of tangibility is supported by the widespread usage of point, nod, or wait behaviors on the robot, in addition to participants moving the figurines around the tabletop during or in-between scenes. The tabletop additionally serves as an effective virtual space to represent the physicality of interactions, as indicated by our idea generation subthemes. Finally, the scenes demonstrated and programs created using Figaro reveal its ability to create complex interaction flows. As with shadow puppetry, this complexity is achieved with limited articulation of the figurines.

6.6 Limitations and Future Work

From its limitations, Figaro presents many opportunities for future work. In particular, Figaro's emphasis on high-level flows leaves much potential to incorporate lower-level design features into the design pipeline, such as navigating a robot within a detailed physical layout. Future work must then explore how exact positions, orientations, and velocities can be integrated into program synthesis. Designing precise localization will also necessitate handling deviations between the physical world and virtual tabletop environments, such as if furniture in the physical environment is moved. For these deviations, the in-situ capabilities of Figaro can be expanded upon, such as with augmented reality (e.g. [24, 65]).

Figaro can gain further low-level expressivity by mapping simple figurine movements to more complex robot motions, extending from our *point* behavior, in which tilting the figurine causes the robot to rotate its body, tilt its head down and then back up, and then rotate back to its original position. Extensions of *point* can allow demonstrators to select and parameterize even more complex behaviors in a similar way to how joysticks enable complex motions on video game characters. For a robot with arms, rotating the robot while pointing forward may induce a *presenting* behavior so that the robot may refer to multiple objects in front of it [55]. The speed of the behavior and objects that the robot refers to could be parameterized by characteristics of the figurine movement.

Furthermore, although Figaro's synthesizer takes various steps to generalize programs from scenes, it cannot produce a program that incorporates unseen regions in the physical layout, nor can it produce a program that contains individual states or transitions not present in the scenes provided by the demonstrator. Future work can improve Figaro's ability to generalize programs using learning techniques to further incorporate unseen paths into the interaction program and querying demonstrators for clarification if Figaro deems any interaction parameters underspecified.

Aside from Figaro's capability as a design tool, the workflow supported by Figaro is also limited in that demonstrators receive little support for iterating on their designs after deployment, and little debugging support that could guide the quality of their designs. Future work should expand Figaro's workflow by drawing on existing testing and debugging approaches similar to those performed in Porfirio et al. (2019) [45] in which program simulation is an active component of the design pipeline and demonstrators are prompted to review and edit execution traces after each demonstration. While building on testing and debugging, the role of the tablet interface in facilitating mobile workflows such as monitoring deployment can be explored.

Other limitations include that Figaro currently only supports dyadic interactions, whereas many interesting design scenarios exist with multiple interaction partners. And finally, the scope of our evaluation is limited in the participants recruited to use Figaro. In making the improvements to Figaro stated above, future work should ensure that Figaro is evaluated with its target users, including professional interaction designers.

7 CONCLUSION

In this paper, we present *Figaro*, a tabletop authoring environment for mobile robots, inspired by shadow puppetry, that exploits the intuitiveness of tangible and tabletop interfaces to provide designers with a natural, situated representation of human-robot interactions. Figaro projects a drawing of a physical layout on the tabletop while users manipulate instrumented figurines that represent a human and the robot to demonstrate scenes of an interaction. After demonstrating scenes, Figaro employs real-time program synthesis to assemble a complete robot program. Our user study with Figaro demonstrates the effects that tangibility and tabletop design have on specifying the flow of human-robot interactions.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) awards 1651129 and 1925043 and an NSF Graduate Research Fellowship. We also thank Kevin Ponto for providing guiding insights into tabletop interface design.

REFERENCES

[1] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. 2012. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. 391–398. https://doi.org/10.1145/2157689.2157815

[2] Sonya Alexandrova, Maya Cakmak, Kaijen Hsiao, and Leila Takayama. 2014. Robot programming by demonstration with interactive action visualizations.. In *Robotics: science and systems*. Citeseer.

[3] 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

[4] Dana Angluin. 1987. Learning regular sets from queries and counterexamples. *Information and computation* 75, 2 (1987), 87–106. https://doi.org/10.1016/0890-5401(87)90052-6

[5] Brenna Argall, Brett Browning, and Manuela Veloso. 2007. Learning by demonstration with critique from a human teacher. In *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 57–64. https://doi.org/10.1145/1228716.1228725

[6] J-C Baillie. 2005. Urbi: Towards a universal robotic low-level programming language. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 820–825. https://doi.org/10.1109/IROS.2005.1545467

[7] Dan Bohus, Sean Andrist, and Eric Horvitz. 2017. A study in scene shaping: Adjusting F-formations in the wild. In *Proceedings of the 2017 AAAI Fall Symposium: Natural Communication for Human-Robot Collaboration*.

[8] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[9] Virginia Braun, Victoria Clarke, Nikki Hayfield, and Gareth Terry. 2019. Thematic analysis. *Handbook of Research Methods in Health Social Sciences* (2019), 843–860. https://doi.org/10.1007/978-981-10-2779-6_103-1

[10] Marion Buchenau and Jane Fulton Suri. 2000. Experience prototyping. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, 424–433. https://doi.org/10.1145/347642.347802

[11] Maxime Busy and Maxime Caniot. 2019. qiBullet, a Bullet-based simulator for the Pepper and NAO robots. *arXiv preprint arXiv:1909.00779* (2019).

[12] Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann.

[13] 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* (San Diego, CA, USA) *(DIS '19)*. Association for Computing Machinery, New York, NY, USA, 1059–1070. https://doi.org/10.1145/3322276.3322278

[14] Crystal Chao and Andrea L Thomaz. 2012. Timing in multimodal turn-taking interactions: Control and analysis using timed petri nets. *Journal of Human-Robot Interaction* 1, 1 (2012), 4–25. https://doi.org/10.5898/JHRI.1.1.Chao

[15] Chu. 2011. Clearing the decks. https://news.mit.edu/2011/automated-flight-decks-0802.

[16] Stéphane Conversy, Jeremie Garcia, Guilhem Buisan, Mathieu Cousy, Mathieu Poirier, Nicolas Saporito, Damiano Taurino, Giuseppe Frau, and Johan Debattista. 2018. Vizir: A Domain-Specific Graphical Language for Authoring and Operating Airport Automations. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 261–273. https://doi.org/10.1145/3242587.3242623

[17] Peter Dalsgaard and Kim Halskov. 2012. Tangible 3D tabletops: combining tangible tabletop interaction and 3D projection. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*. 109–118. https://doi.org/10.1145/2399016.2399033

[18] Scott Davidoff, Min Kyung Lee, Anind K Dey, and John Zimmerman. 2007. Rapidly exploring application design through speed dating. In *International Conference on Ubiquitous Computing*. Springer, 429–446. https://doi.org/10.1007/978-3-540-74853-3_25

[19] Tom Djajadiningrat, Stephan Wensveen, Joep Frens, and Kees Overbeeke. 2004. Tangible products: redressing the balance between appearance and action. *Personal and Ubiquitous Computing* 8, 5 (2004), 294–309. https://doi.org/10.1007/s00779-004-0293-8

[20] 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

[21] Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. 2016. Human-robot interaction design using Interaction Composer eight years of lessons learned. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 303–310. https://doi.org/10.1109/HRI.2016.7451766

[22] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. 2017. Program synthesis. *Foundations and Trends® in Programming Languages* 4, 1-2 (2017), 1–119. https://doi.org/10.1561/2500000010

[23] Yosuke Horiuchi, Tomoo Inoue, and Ken-ichi Okada. 2012. Virtual stage linked with a physical miniature stage to support multiple users in planning theatrical productions. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. 109–118. https://doi.org/10.1145/2166966.2166989

[24] Baichuan Huang, Deniz Bayazit, Daniel Ullman, Nakul Gopalan, and Stefanie Tellex. 2019. Flight, camera, action! using natural language and mixed reality to control a drone. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 6949–6956. https://doi.org/10.1109/ICRA.2019.8794200

[25] Chien-Ming Huang and Bilge Mutlu. 2012. Robot behavior toolkit: generating effective social behaviors for robots. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 25–32. https://doi.org/10.1145/2157689.2157694

[26] Justin Huang and Maya Cakmak. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 453–462. https://doi.org/10.1145/2909824.3020215

[27] Hiroshi Ishii. 2008. The tangible user interface and its evolution. *Commun. ACM* 51, 6 (2008), 32–36. https://doi.org/10.1145/1349026.1349034

[28] Jun Kato, Daisuke Sakamoto, and Takeo Igarashi. 2013. Picode: inline photos representing posture data in source code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3097–3100. https://doi.org/10.1145/2470654.2466422

[29] Yusuke Kato, Takayuki Kanda, and Hiroshi Ishiguro. 2015. May I help you?-Design of human-like polite approaching behavior. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 35–42. https://doi.org/10.1145/2696454.2696463

[30] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 395–405. https://doi.org/10.1145/2642918.2647375

[31] Paul Lapides, Ehud Sharlin, and Mario Costa Sousa. 2008. Three dimensional tangible user interface for controlling a robotic team. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. 343–350. https://doi.org/10.1145/1349822.1349867

[32] 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* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 647–656. https://doi.org/10.1145/1978942.1979035

[33] Phoebe Liu, Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. 2016. Data-driven HRI: Learning social behaviors by example from human–human interaction. *IEEE Transactions on Robotics* 32, 4 (2016), 988–1008. https://doi.org/10.1109/TRO.2016.2588880

[34] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 72 (Nov. 2019), 23 pages. https://doi.org/10.1145/3359174

[35] Eric Meisner, Volkan Isler, and Jeff Trinkle. 2008. Controller design for human-robot interaction. *Autonomous Robots* 24, 2 (2008), 123–134. https://doi.org/10.1007/s10514-007-9054-7

[36] Ryo Murakami, Luis Yoichi Morales Saiki, Satoru Satake, Takayuki Kanda, and Hiroshi Ishiguro. 2014. Destination unknown: walking side-by-side without knowing the goal. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. 471–478. https://doi.org/10.1145/2559636.2559665

[37] Lorenzo Nardi and Luca Iocchi. 2014. Representation and execution of social plans through human-robot collaboration. In *International Conference on Social Robotics*. Springer, 266–275. https://doi.org/10.1007/978-3-319-11973-1_27

[38] Daniel Neider. 2014. *Applications of automata learning in verification and synthesis*. Ph.D. Dissertation. Hochschulbibliothek der Rheinisch-Westfälischen Technischen Hochschule Aachen.

[39] Don Norman. 2013. *The design of everyday things: Revised and expanded edition*. Basic books.

[40] William Odom, John Zimmerman, Scott Davidoff, Jodi Forlizzi, Anind K Dey, and Min Kyung Lee. 2012. A fieldwork of the future with user enactments. In *Proceedings of the Designing Interactive Systems Conference*. 338–347. https://doi.org/10.1145/2317956.2318008

[41] Antti Oulasvirta, Esko Kurvinen, and Tomi Kankainen. 2003. Understanding contexts by being there: case studies in bodystorming. *Personal and ubiquitous computing* 7, 2 (2003), 125–134. https://doi.org/10.1007/s00779-003-0238-7

[42] Oğuzhan Özcan. 2002. Cultures, the traditional shadow play, and interactive media design. *Design Issues* 18, 3 (2002), 18–26. https://doi.org/10.1162/074793602320223262

[43] Ayberk Özgür, Séverin Lemaignan, Wafa Johal, Maria Beltran, Manon Briod, Léa Pereyre, Francesco Mondada, and Pierre Dillenbourg. 2017. Cellulo: Versatile handheld robots for education. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 119–127. https://doi.org/10.1145/2909824.

3020247

[44] Kevin Ponto, Maurizio Seracini, and Falko Kuester. 2009. Wipe-Off: An Intuitive Interface for Exploring Ultra-Large Multi-Spectral Data Sets for Cultural Heritage Diagnostics. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 2291–2301. https://doi.org/10.1111/j.1467-8659.2009.01532.x

[45] 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

[46] David Porfirio, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2018. Authoring and verifying human-robot interactions. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 75–86. https://doi.org/10.1145/3242587.3242634

[47] E. Pot, J. Monceaux, R. Gelin, and B. 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*. 46–51. https://doi.org/10.1109/ROMAN.2009.5326209

[48] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, Japan, 5.

[49] Harald Raffelt, Bernhard Steffen, and Therese Berg. 2005. Learnlib: A library for automata learning and experimentation. In *Proceedings of the 10th international workshop on Formal methods for industrial critical systems*. 62–71. https://doi.org/10.1145/1081180.1081189

[50] Mary Beth Rosson and John M Carroll. 2009. Scenario-based design. In *Human-computer interaction*. CRC Press, 161–180.

[51] Theodosios Sapounidis and Stavros Demetriadis. 2016. Educational robots driven by tangible programming languages: A review on the field. In *International Conference EduRobotics 2016*. Springer, 205–214. https://doi.org/10.1007/978-3-319-55553-9_16

[52] Theodosios Sapounidis, Stavros Demetriadis, and Ioannis Stamelos. 2015. Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing* 19, 1 (2015), 225–237. https://doi.org/10.1007/s00779-014-0774-3

[53] Satoru Satake, Takayuki Kanda, Dylan F Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. 2009. How to approach humans? Strategies for social robots to initiate interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*. 109–116. https://doi.org/10.1145/1514095.1514117

[54] Allison Sauppé and Bilge Mutlu. 2014. Design patterns for exploring and prototyping human-robot interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1439–1448. https://doi.org/10.1145/2556288.2557057

[55] Allison Sauppé and Bilge Mutlu. 2014. Robot deictics: How gesture and context shape referential communication. In *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 342–349. https://doi.org/10.1145/2559636.2559657

[56] Yasaman S Sefidgar, Prerna Agarwal, and Maya Cakmak. 2017. Situated tangible robot programming. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 473–482. https://doi.org/10.1145/2909824.3020240

[57] 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

[58] Armando Solar-Lezama. 2009. The sketching approach to program synthesis. In *Asian Symposium on Programming Languages and Systems*. Springer, 4–13. https://doi.org/10.1007/978-3-642-10672-9_3

[59] Bernhard Steffen, Falk Howar, and Maik Merten. 2011. Introduction to active automata learning from a practical perspective. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer, 256–296. https://doi.org/10.1007/978-3-642-21455-4_8

[60] Ryo Suzuki, Jun Kato, Mark D Gross, and Tom Yeh. 2018. Reactile: Programming swarm user interfaces through direct physical manipulation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3173574.3173773

[61] Andrea L Thomaz and Crystal Chao. 2011. Turn-taking based on information flow for fluent human-robot interaction. *AI Magazine* 32, 4 (2011), 53–63. https://doi.org/10.1609/aimag.v32i4.2379

[62] Abhishek Udupa, Arun Raghavan, Jyotirmoy V Deshmukh, Sela Mador-Haim, Milo MK Martin, and Rajeev Alur. 2013. TRANSIT: specifying protocols with concolic snippets. *ACM SIGPLAN Notices* 48, 6 (2013), 287–296. https://doi.org/10.1145/2499370.2462174

[63] John Underkoffler and Hiroshi Ishii. 1999. Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 386–393. https://doi.org/10.1145/302979.303114

[64] Kanae Wada, Dylan F Glas, Masahiro Shiomi, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2015. Capturing expertise: developing interaction content for a robot through teleoperation by domain experts. *International Journal of Social Robotics* 7, 5 (2015), 653–672. https://doi.org/10.1007/s12369-015-0288-9

[65] Michael Walker, Hooman Hedayati, Jennifer Lee, and Daniel Szafir. 2018. Communicating robot motion intent with augmented reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. 316–324. https://doi.org/10.1145/3171221.3171253

[66] Nick Walker, Yu-Tang Peng, and Maya Cakmak. 2019. Neural Semantic Parsing with Anonymization for Command Understanding in General-Purpose Service Robots. In *RoboCup Symposium*. Springer, 337–350. https://doi.org/10.1007/978-3-030-35699-6_26

[67] James Young, Takeo Igarashi, and Ehud Sharlin. 2008. *Puppet master: Designing reactive character behavior by demonstration*. Technical Report. University of Calgary. https://doi.org/10.11575/PRISM/31028

[68] James Young, Kentaro Ishii, Takeo Igarashi, and Ehud Sharlin. 2012. Style by demonstration: teaching interactive movement style to robots. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. 41–50. https://doi.org/10.1145/2166966.2166976

[69] James Young, Daisuke Sakamoto, Takeo Igarashi, and Ehud Sharlin. 2009. Puppet master: Designing reactive character behavior by demonstration. In *Human-Agent Interaction (HAI)*.

[70] James E Young, Ehud Sharlin, and Takeo Igarashi. 2013. Teaching robots style: designing and evaluating style-by-demonstration for interactive robotic locomotion. *Human–Computer Interaction* 28, 5 (2013), 379–416. https://doi.org/10.1080/07370024.2012.697046

[71] Shengdong Zhao, Koichi Nakamura, Kentaro Ishii, and Takeo Igarashi. 2009. Magic cards: a paper tag interface for implicit robot control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 173–182. https://doi.org/10.1145/1518701.1518730