Agreeing to Disagree: Translating Representations to Uncover a Unified Representation for Social Robot Actions

Saad Elbeleidy [1], **Jason R. Wilson** [2]

[1]Peerbots
[2]Franklin & Marshall College
saad@peerbots.org, jrw@fandm.edu

Abstract

Researchers and designers of social robots often approach robot control system design from a single perspective; such as designing autonomous robots, teleoperated robots, or robots programmed by an end-user. While each design approach presents a tradeoff between some advantages and limitations, there is an opportunity to integrate these approaches where people benefit from the best-fit approach for their use case. In this work, we propose integrating these seemingly distinct robot control approaches to uncover a common data representation of social *actions* defining social expression by a robot. We demonstrate the value of integrating an authoring system, teleoperation interface, and robot planning system by integrating instances of these systems for robot storytelling. By relying on an integrated system, domain experts can define behaviors through end-user interfaces that teleoperators and autonomous robot programmers can use directly thus providing a cohesive expert-driven robot system.

Introduction

Researchers and designers of social robots, or for the purpose of this work: robots that perform social expressions while interacting with people, make several choices in the design of these robots. Importantly, researchers must decide on how they control their robots, and they often focus on a single control mechanism. Since control approaches have historically been disconnected, researchers often make a single selection that results in a limiting setup that unnecessarily locks researchers into a collection of consequences related to their choice. In this paper, we demonstrate how different control mechanisms can be integrated so that researchers can rely on the appropriate tool based on their use case, and we use this integration to work towards defining a unified representation that can ease future integrations. Examining the tools used by different types of researchers and designers, we identify a robot's *action* as the most common concept and thus a necessary first step in defining a unified representation.

We demonstrate this integration through a robot storytelling use case, a common application for child-robot interaction since children can find robots engaging (Hubbard et al. 2021; Sun et al. 2017; Ligthart, Neerincx, and Hindriks 2020). Storytelling is a common activity that children find entertaining, enjoyable and potentially educational (Barton 1986). As we will discuss, robot storytelling is a task that can easily demonstrate how a single choice in robot control can result in restrictive consequences.

Design Approaches for Robot Control

In terms of how they control robots, researchers' perspectives often lead to a single choice of (1) autonomous programming (2) teleoperation or Wizard of Oz (WoZ), or (3) End-User Development. Each of these approaches has clear advantages and limitations.

Autonomous Programming

Developers of autonomous robots often rely on planning languages or behavior trees to describe, control, and execute robot actions. Behavior trees (BTs) handle complex robot controls using modularity, hierarchy, and feedback (Ögren and Sprague 2022). Similarly, hierarchical task networks (HTNs) use modular components organized in a hierarchical structure to plan out a robot's actions. The feedback mechanisms in BTs provide capabilities for execution, whereas planning approaches with HTNs often rely on a separate component for executing the plan. The hierarchical structures supported in these approaches provide programmers with an interface for specifying high-level goals or tasks to be accomplished. Components at the leaf level in BTs and HTNs are responsible for the specific actions the robot is to perform.

To manage the execution of actions on a robot platform, actionLib provides a common interface across robots supporting ROS. ROS actionlib works as a client-server architecture that focuses on sending a goal from the client to the server, which is responsible for managing the execution of the action on the robot platform (Santos et al. 2017). Since actionlib simply provides an interface and syntax for communicating actions, there is no commitment or constraints on how the action is handled, allowing developers autonomy in their implementation.

Autonomous robot programmers are able to rely on these structure to define limited scope interactions to be delivered by robots. Interactions such as storytelling can be highly successful examples of using an autonomous robot since the

task is of fairly limited scope and simple fallback mechanisms can be in place to ensure children's safety (Cagiltay et al. 2022). As children's expectations of the robots increase and robots' application scope increases, there will be a need for more human supervision (Elbeleidy, Mott, and Williams 2022). This could potentially be accomplished through interfaces for robot teleoperation.

Teleoperation

Teleoperation interfaces are commonly used as simple prototyping tools to verify the value or use of a robot. Researchers commonly use Wizard of Oz (Riek 2012) as a method to evaluate robots or their impact on interlocutors. However, in highly critical or sensitive scenarios, teleoperation can be even more appropriate than autonomous interaction and ought to be considered in its own right (Beer, Fisk, and Rogers 2014). By choosing to teleoperate robots, researchers must design a teleoperation interface and design for the teleoperators who use that interface.

Teleoperation interfaces rely on data representations that represent the robot's behavior as well as the user interface elements that ease teleoperation (Adamides et al. 2014). These interfaces must be designed at an appropriate level of abstraction to communicate to teleoperators all the possibilities for controlling the robot without overwhelming them and substantially increasing their cognitive load.

Teleoperation interfaces could be useful in controlling robots for storytelling. However, teleoperation interfaces alone are not sufficient. Teleoperation interfaces can rely on a low level of abstraction, allowing teleoperators precise control over the robot's actions but requiring preparation ahead of time or significant cognitive load during teleoperation (Elbeleidy et al. 2023). Previous research has found that a sole reliance on teleoperation leads to invisible labor performed by teleoperators to author content ahead of time (Elbeleidy, Reddy, and Williams 2023). Additionally, when domain specialists are involved in defining robot behaviors, they often struggle to use existing tools that relate to robotics (Elbeleidy et al. 2022).

End-User Development

End-User Development (EUD) is intended to specifically address the barrier in robotics-specific knowledge. End-User Development tools rely on simple interfaces, such as visual programming, to allow non-roboticists the ability to program robots (Coronado et al. 2020; Ajaykumar, Steele, and Huang 2021). End-User Development can encompass robot programming but goes beyond just that to include simply authoring the domain-specific knowledge for the robot. EUD interfaces can take on different approaches to simplifying behavior definition such as by using familiar metaphors or interfaces or using a high level abstraction to define a robot's actions. For example, the Polaris system allows authors to define behaviors by defining their desired goals for the robot to perform (Porfirio, Roberts, and Hiatt 2024).

While End-User Development tools can provide several advantages by interfacing with non-experts, they may require complex development efforts to support their creation. Robot systems ought to effectively integrate behavior authoring, robot teleoperation, and autonomous capabilities of robots in interfaces designed for various roles: author, teleoperator and supervisor, and expertise areas: domain specialists, roboticists.

Common Representation

Our effort to integrate tools that follow different design approaches reveals some of the similarities and differences in what the tools need to represent. Autonomous programming needs to be able to represent a command given to a robot, though a variety of terminology is used. Once that is determined, these tools need to be able to represent sequences of operations, the coordination between operations, and conditions for an operation. In contrast, teleoperation interfaces often rely on low level controls to move or modify individual parts of a robot, and the representations necessary are directly connected to the parts of the robot that can change. Additionally, teleoperation interfaces need to associate the robot controls to user interface elements for buttons, groupings, and labels. The teleoperation interface we use in this work has some support for low level controls but also provides buttons with a predefined grouping of speech, emotion, and color changes. End-User Development tools try to provide simplified and abstract descriptions of how to control the robot, but they also need to connect this to more concrete and perhaps more complex definition of the specific commands necessary to operate the robot.

Based on this knowledge, we propose that a shared representation that would support tool integration starts with a common need: representing what the robot is *to do*. Often there is a need to represent the robot's "doing" at various levels of abstraction, resulting in a variety of terms like "behavior", "action", "goal", and "task". We demonstrate that a common representation of an "action" allows an integration of tools for teleoperation, content authoring, and semi-automated authoring support. We demonstrate the efficacy of this system by applying it to robot storytelling using the Peerbots Platform [1] for content authoring and teleoperation, and the sarBehaviorGen (Wilson and Yang 2024) for autonomous generation of social robot behaviors.

Examining the representations used by these tools (see Figure 1) makes clear that there is no present unified representation that may be used to integrate the tools. Instead we identify the basis for a common representation is an *action*, defined as a fine-grained instruction to the robot. Defining a common action representation is a necessary step in integrating multiple components whether by sharing a unified representation or translating between representations. In order to translate across representations, we first need to identify the common element to be translated.

In a larger unified representation or in translating between representations most tools will ignore various parts of the representation, as they are not relevant to the functionality of that tool. For example, the teleoperation interface defines values that are not informative to the behavior generator. The teleoperation interface wraps actions with

[1] https://peerbots.org

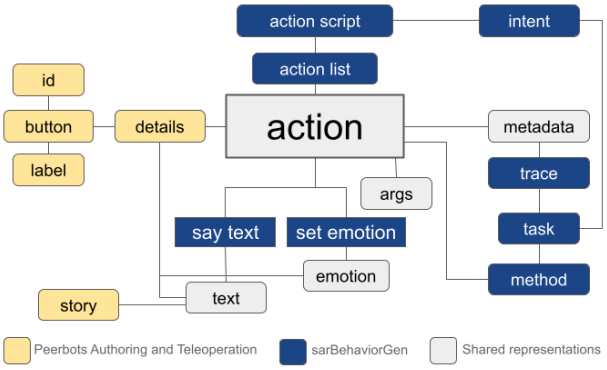

Figure 1: An *action* is the central representation across each of the tools. Shown here are the relationship between an action and the other representations used by each tool.

interface-specific presentation, hierarchical groupings, and user-specified metadata. The behavior generator requires intents defined and associated with each action as a way to integrate groups of actions together. Each approach provides its own abstractions on top of the base action that the various tools share. Additionally, each tool makes a choice in how to define synchronous behaviors or to perform them simultaneously. This decision heavily informs the representation chosen by the robot tool designers and resulted in our decision to translate rather than unify a representation. In fact, relying on different representations that can be translated between each other could result in even more flexibility and integration across different tools. This all begins with a common understanding of a robot action.

Tools & Representations

In this section, we describe each tool used, its data representation(s), target audience and level of abstraction of the interaction with robots. A summary of each tool and the input and output representations that it uses are presented in Figure 2.

Autonomous Behavior Generation

A behavior generator allows a robotocist to specify a robot's behavior at an abstract level and have the behavior automatically executed on the robot. We use the sarBehaviorGenerator, which provides an API that takes a verbalization and an *intent*, which are translated into an *action script* by a planner. The action script can then be sent to a robot middleware for execution. Alternatively, the action script may be returned through the programming interface.

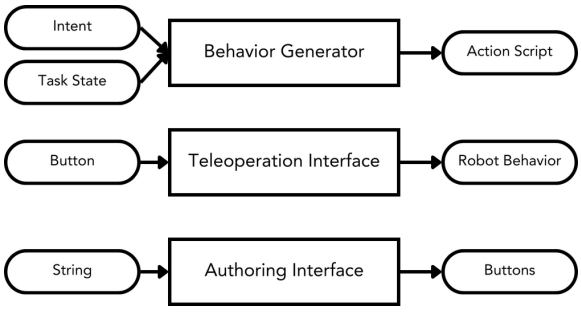

Figure 2: The data representations as input and output of each tool.

We define these terms in more detail as follows. An *intent* provides a high-level description of the communicative purpose for what the robot is saying (Yang et al. 2023). We use four intents: (1) **Inform** communicates factual information. (2) **Inquire** communicates a request for information. (3) **Instruct** provides information about the task. (4) **Social** uses social norms to facilitate a natural, positive, and smooth interaction. An intent is a high-level abstraction specifying the robot's behavior, which the behavior generator translates into a less abstract specification of the particular actions for the robot to perform. The behavior generator uses a planner, which uses planning models that represent a sequence of actions as a *method*. A method represents one approach to executing a higher level *task*. The intent provided to the behavior generator is given to the planner as a top-level task to plan for. The resulting actions are defined in an action script that contains of a list of actions.

An *action script* is a list of actions in the order that the robot is to perform them. Each action includes the name of the action, its arguments, and optional metadata. The metadata can include a trace of the reasoning that connects the provided intent to the resulting action. An example of a portion of the resulting actions script is the following.

Listing 1: Data representation of an action script.

```
{
   "intent": "Inquire",
   "description": "Example of Inquire",
   "actionList": [
      {
         "name": "LookInDirection",
         "args": ["center"],
         "metadata": {
            "trace": [["claim-look-0"],
               ["claim-look"],
               ["take-turn-default"],
               ["take-turn"],
               ["how-going"],
               ["inquire", "progress"]]
         }
      },
      {
         "name": "TiltHead",
         "args": ["left", "small"],
         "metadata": {
            "trace": [["
                tiltHeadandPause1"],
               ["tiltHeadandPause",
                  "left", "small"],
               ["how-going"],
               ["inquire", "progress"]]
         }
      }]
}
```

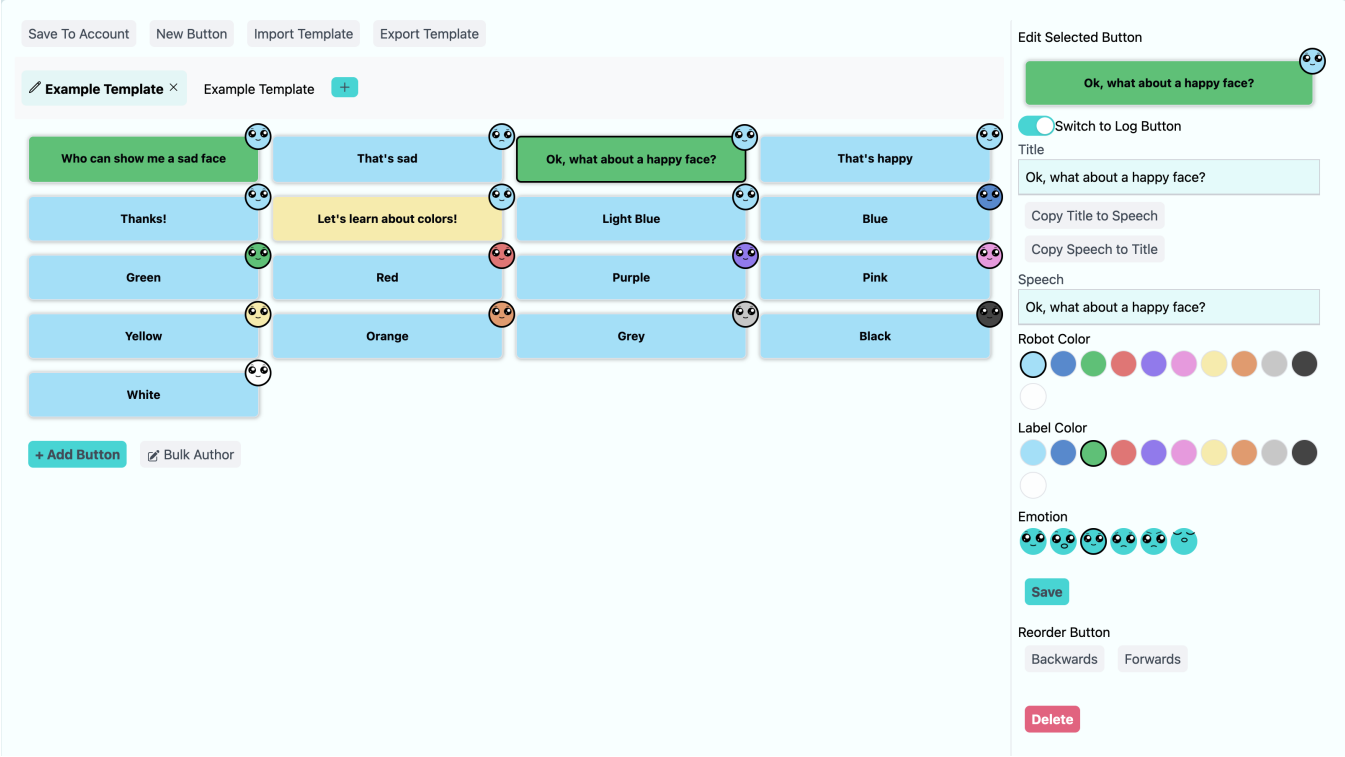

Figure 3: The Teleoperation Interface. Left: An example of how buttons are presented to a teleoperator. Each button transmits a social behavior to be expressed by the robot. Right: The button editing panel showing the button title, speech, robot color, label color, and emotion.

The order of the actions is explicitly defined by the action script and each action is assumed to run asynchronously. This requires Pause actions to coordinate the timing of speech and gestures but provides more flexibility for the robot to express in a variety of ways simultaneously. This choice is very distinct from the teleoperation interface's approach described below.

Teleoperation

We use the Peerbots platform as the teleoperation interface. The teleoperation interface provides a low level of abstraction, allowing teleoperators with minimal technical knowledge to directly control a robot's social expression. The interface (see Figure 3) contains tabs that allow teleoperators to move between different *collections* of *buttons* where each button initiates simultaneous *behaviors* upon selection. Pauses are implicitly defined between behaviors by the time it takes the teleoperator to perform a subsequent action. Note this distinction relative to the behavior generator, where pauses are explicitly specified.

To represent the data for these behaviors, the teleoperation interface relies on a nested structure of *collections* containing *buttons* which contain the *behaviors*. Each button selection initiates multiple simultaneous behaviors at once: a verbalization, a color, and an emotional expression. The representation for a button contains the behaviors to initiate as well as teleoperator-facing values such as the label color, and metadata for logging purposes.

Listing 2: Data representation for a button.

```
"buttonID": {
    "type": "The type of button: Message
        or Log",
    "labelColor": "The color of the
        button",
    "details": "The robot behavior",
    "metadata": "Open ended metadata"
}
```

The robot behaviors (the details key in the button representation) are represented as follows:

Listing 3: Data representation of behaviors sent to a robot.

```
"details": {
    "title": "Title of the action",
    "speech": "The text to verbalise",
    "emotion": "The emotion to express",
    "color": "The color to present"
}
```

Note the distinction between these two representations since the teleoperation interface may contain additional information that is not sent to the robot such as information about how to present the button to the teleoperator.

A collection or visually, a selected tab is represented as a set of nodes and their organization where each node is a but-

ton. Button definitions are separate from their organization structure with button identifiers as keys under a ”nodes” object and their organization defined under a ”hierarchy” key.

Listing 4: Data representation of a collection of buttons.

```
"buttons": "An object of buttonIDs and
    their information",
"hierarchy": {
    "root": "A list of the root order of
        buttonIDs",
    "buttonID": "A list of buttonIDs of
        the descendents"
}
```

This data format allows the button collections to be easily represented as a graph which can be helpful for a variety of user interface representations such as behavior trees and to easily support features such as dragging and dropping to reorder buttons. From the teleoperator's perspective, button collections are a useful abstraction that allows teleoperators to organize behaviors that are similar, or commonly used together such as when expressed chronologically (Elbeleidy et al. 2021).

One important capability of teleoperation interfaces is to allow teleoperators to quickly edit robot behaviors. The teleoperation interface contains a side panel that allows teleoperators to edit an individual button's details, including both how the button appears in the interface and the behaviors the robot would perform when that button is selected. While these editing capabiltiies are essential, they are not sufficient for in depth authoring of content.

Authoring

Authoring interfaces allow authors to directly create a large amount of robot behaviors for a particular task and think about these tasks at a relevant level of abstraction. The authoring interface as part of the Peerbots platform contains a single text box, a submit button, and an option for the author to select whether they create a new button collection or append buttons to the currently selected collection. Upon submission, the authoring interface parses the provided string of text and converts it into the representation used by the teleoperation interface that is then presented as buttons.

The authoring interface allows authors to focus on the verbalizations to express and author them all at once in a distraction-free space. Authors need not consider the details of emotion and color presentation of the robot and get to focus on the dialogue of the interaction. However, this authoring interface still requires authors to enter all the verbalizations they expect the robot to verbalize.

Integration: Semi-Automated Authoring

We recognize the potential for integrating the autonomous behavior generator to support authors with working at a higher level of abstraction and rely on the behavior generator for the detailed definitions of each behavior to perform. Thus, we integrate these modules and provide semi-automated authoring capabilities. Rather than the behavior generator requiring a robotics expert to provide input, we integrate our existing authoring interface with the behavior generator so a non-roboticist author can specify their desired content. The data flow of the semi-automated authoring interface is shown in Figure 4.

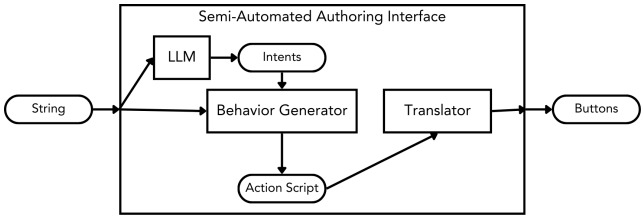

Figure 4: The data representation input and output to the semi-automated authoring interface including the ways in which the representation transforms along the way.

Following our integration, the previously mentioned authoring interface's user input is provided to an intermediate component which passes appropriate information to the behavior generator. This component splits user-provided input into distinct verbalizations (i.e., sentences) and infers how each expression ought to be performed. The first part of this inference is classifying each verbalization into one of the previously mentioned intents. Using Claude AI, we prompt it with each of the intent definitions and ask it to classify each verbalization. The second part uses the behavior generator to produce an action script that details the specific sequence of actions for the robot to perform based on the provided intents and verbalizations. The authoring interface then converts action scripts to the teleoperation interface's representation to be previewed to the user. This integration is possible by translating the behavior generator outputs to the teleoperation interface's representation.

Application: Robot Storytelling

We use robot storytelling as an example use case that demonstrates a data pipeline to communicate between autonomous and teleoperation systems. An author can begin in the authoring interface and enter the contents of the story. Once submitted, the author is presented with two preview visualizations. First, a table with the input broken up into verbalizations with intent classifications for each. Second, a table with the proposed robot behaviors that would be expressed while telling the story. Upon acceptance, the new content is loaded in the teleoperation interface and authors can edit it to their liking. An author with no robotics expertise can now benefit from the capabilities of the autonomous behavior generator which previously required some robotics and programming knowledge. This is only possible thanks to these modules being able to translate their representations and arrive at a common understanding for their base actions.

Discussion

Unified Representation vs Translation Interfaces

We identified *actions* as the most essential common representation across tools for teleoperation, end-user develop-

ment, and autonomous robots. However, the syntactic and semantic representation of an action differed across these tools. Given the differences in syntax and semantics around one of the most essential representations, it is unclear how to move forward towards a unified representation. Alternatively, tool integration could rely on translation interfaces.

A remaining challenge in developing these interfaces is understanding the different semantics of commonly used terms like "action". We see this in our integration of Peerbots and the sarBehaviorGen. A behavior in the teleoperation interface describes a small collection of effects produced by different parts of the robot and are to be done nearly simultaneously. Conversely, the autonomous behavior generator's outputs actions that focus on a single effect (e.g., moving the arm to point) and achieves synchronous effects by adding pauses between otherwise simultaneous actions. The choice in representing complex behaviors at once is especially important when autonomous robot perception is involved.

Semi-Autonomous Behavior

To integrate teleoperation interfaces, end-user development interfaces, and autonomous programs for robots, we narrowed our scope to the overlapping needs. Since perception is performed by the teleoperator when teleoperation is used, we consequently deemed any data representations for perception or integration with perception out of scope. This choice was intentional as it allows us to think of representations for domain knowledge in a simpler state. However, we ought to consider perception in the future in ways that can enable semi-autonomous behavior where appropriate.

For semi-autonomous behavior, we consider two human-in-the-loop cases: (1) teleoperator tells the robot to continue autonomously, and (2) robot functioning autonomously decides whether to continue or notify the teleoperator that human intervention is needed. Achieving semi-autonomous behavior will require integrating with an autonomous robot architecture, many of which use goals and actions to specify the robot's behavior (e.g., (Lemaignan et al. 2017; Scheutz et al. 2019)). The robot architecture is needed for perceiving the user and reasoning about how and when to proceed. Assuming that the robot architecture can provide a goal of communicating the next portion of the story, then our solution described above is able to assign an intent and generate a plan for how the robot will communicate. Since sarBehaviorGen integrates into two social robot platforms (Wilson and Yang 2024), the plan can be automatically executed with the provided robot middleware.

Conclusion

Across autonomous robots, teleoperation, and end-user development for robots, we find different opinionated approaches to simplify robot control to the relevant audience. However, there is a common need to specify and represent an "action", or more generally a description of what to tell the robot to do. We demonstrated an example integration across these systems and found that identifying these similarities and differences is a critical step towards defining a shared representation, or at least a shared understanding, that may lead to better integration.

References

Adamides, G.; Christou, G.; Katsanos, C.; Xenos, M.; and Hadzilacos, T. 2014. Usability guidelines for the design of robot teleoperation: A taxonomy. *IEEE Transactions on human-machine systems*, 45(2): 256–262.

Ajaykumar, G.; Steele, M.; and Huang, C.-M. 2021. A survey on end-user robot programming. *ACM Computing Surveys (CSUR)*, 54(8): 1–36.

Barton, B. 1986. *Tell me another: Storytelling and reading aloud at home, at school and in the community.* ERIC.

Beer, J. M.; Fisk, A. D.; and Rogers, W. A. 2014. Toward a framework for levels of robot autonomy in human-robot interaction. *J. Hum.-Robot Interact.*, 3(2): 74–99.

Cagiltay, B.; White, N. T.; Ibtasar, R.; Mutlu, B.; and Michaelis, J. 2022. Understanding Factors that Shape Children's Long Term Engagement with an In-Home Learning Companion Robot. In *Proceedings of the 21st Annual ACM Interaction Design and Children Conference*, 362–373.

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

Elbeleidy, S.; Mott, T.; Liu, D.; Do, E.; Reddy, E.; and Williams, T. 2023. Beyond the Session: Centering Teleoperators in Robot-Assisted Therapy Reveals the Bigger Picture. In *Proceedings of the ACM Conference On Computer-Supported Cooperative Work And Social Computing*.

Elbeleidy, S.; Mott, T.; Liu, D.; and Williams, T. 2022. Practical Considerations for Deploying Robot Teleoperation in Therapy and Telehealth. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 977–984. IEEE.

Elbeleidy, S.; Mott, T.; and Williams, T. 2022. Practical, Ethical, and Overlooked: Teleoperated Socially Assistive Robots in the Quest for Autonomy. In *Companion Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction (alt.HRI)*.

Elbeleidy, S.; Reddy, E.; and Williams, T. 2023. The Invisible Labor of Authoring Dialogue for Teleoperated Socially Assistive Robots. In *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*.

Elbeleidy, S.; Rosen, D.; Liu, D.; Shick, A.; and Williams, T. 2021. Analyzing Teleoperation Interface Usage of Robots in Therapy for Children with Autism. In *Proceedings of the ACM Interaction Design and Children Conference*.

Hubbard, L. J.; Chen, Y.; Colunga, E.; Kim, P.; and Yeh, T. 2021. Child-robot interaction to integrate reflective storytelling into creative play. In *Proceedings of the 13th Conference on Creativity and Cognition*, 1–8.

Lemaignan, S.; Warnier, M.; Sisbot, E. A.; Clodic, A.; and Alami, R. 2017. Artificial cognition for social human–robot interaction: An implementation. *Artificial Intelligence*, 247: 45–69.

Ligthart, M. E.; Neerincx, M. A.; and Hindriks, K. V. 2020. Design patterns for an interactive storytelling robot to support children's engagement and agency. In *Proceedings of*

the 2020 ACM/IEEE international conference on human-robot interaction, 409–418.

Ögren, P.; and Sprague, C. I. 2022. Behavior trees in robot control systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1): 81–107.

Porfirio, D.; Roberts, M.; and Hiatt, L. M. 2024. Goal-Oriented End-User Programming of Robots. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, 582–591. New York, NY, USA: Association for Computing Machinery. ISBN 9798400703225.

Riek, L. D. 2012. Wizard of Oz studies in HRI: a systematic review and new reporting guidelines. *J. Hum.-Robot Interact.*, 1(1): 119–136.

Santos, H. B.; Teixeira, M. A. S.; de Oliveira, A. S.; de Arruda, L. V. R.; and Neves, F. 2017. Control of mobile robots using actionlib. *Robot Operating System (ROS) The Complete Reference (Volume 2)*, 161–189.

Scheutz, M.; Williams, T.; Krause, E.; Oosterveld, B.; Sarathy, V.; and Frasca, T. 2019. An overview of the distributed integrated cognition affect and reflection diarc architecture. *Cognitive architectures*, 165–193.

Sun, M.; Leite, I.; Lehman, J. F.; and Li, B. 2017. Collaborative storytelling between robot and child: A feasibility study. In *Proceedings of the 2017 Conference on Interaction Design and Children*, 205–214.

Wilson, J.; and Yang, Y. 2024. Software Architecture to Generate Assistive Behaviors for Social Robots. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 1119–1123.

Yang, Y.; Langer, A.; Howard, L.; Marshall, P. J.; and Wilson, J. R. 2023. Towards an Ontology for Generating Behaviors for Socially Assistive Robots Helping Young Children. In *Proceedings of the AAAI Symposium Series*, volume 2, 213–218.