Interaction Templates: A Data-Driven Approach for Authoring Robot Programs

David Porfirio [*1], **Maya Cakmak** [†2], **Allison Sauppé** [‡3], **Aws Albarghouthi** [§1] **and Bilge Mutlu** [¶1]

[1] *University of Wisconsin–Madison, Madison, WI*
[2] *University of Washington, Seattle, WA*
[3] *University of Wisconsin–La Crosse, La Crosse, WI*

Abstract

Socially interactive robots present numerous unique programming challenges for interaction developers. While modern authoring tools succeed at making the authoring experience approachable and convenient for developers from a wide variety of backgrounds, they are less successful at targeting assistance to developers based on the specific task or interaction being authored. We propose interaction templates, a data-driven solution for (1) matching in-progress robot programs to candidate task or interaction models and then (2) providing assistance to developers by using the matched models to generate modifications to in-progress programs. In this paper, we present the various dimensions that define first how interaction templates might be used, then how interaction templates may be represented, and finally how they might be collected.

Keywords: Human-robot interaction. Templates. Authoring. Robot programming.

1 Introduction

Authoring socially interactive robots [1] presents numerous unique challenges for human-robot interaction (HRI) developers, who may be professional software engineers or interaction designers creating robot applications for others to use, or end-user developers programming a robot for their own use. To begin, human-robot interactions can be exceptionally open-ended, thereby necessitating that developers consider numerous edge cases when crafting the high-level flow of a robot program. Even experienced developers may further struggle with low-level interaction details, such as behavior timing and speech volume, which must be correctly parameterized to adequately fit a robot program to an interaction context. And for end-user developers who use less precise development techniques, such as natural language programming, there exists less capacity for meticulous program specification and testing to ensure that authored programs are correct.

We propose *interaction templates* as a solution for the challenges associated with HRI authoring. Interaction templates are inspired by existing program synthesis work on program templates [2] and sketching [3]–[5] and build on prior HRI work on templates as generic, reusable program specifications that can be selected and instantiated by end users [6]. In our own investigation, we recast interaction templates as generic and reusable models of human-robot interactions derived from interaction data for the purpose of assisting developers specifying a robot program. Furthermore, in our own vision of interaction templates, a developer using an authoring tool to construct an in-progress robot program does not select the appropriate template to use, nor does the developer necessarily instantiate it. Rather, we envision that from a multitude of available interaction templates derived from interaction data, an authoring tool will proactively *match* an appropriate subset of templates to the in-progress program for the purpose of proposing or enforcing changes. When matched, any critical information present in a template that differs or is absent from the in-progress program becomes proposed. Additional uses of templates in prior work include robot motion planning and analysis [7], [8] and generating user interfaces [9].

*Email: dporfirio@wisc.edu
†Email: mcakmak@cs.washington.edu
‡Email: asauppe@uwlax.edu
§Email: aws@cs.wisc.edu
¶Email: bilge@cs.wisc.edu

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

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

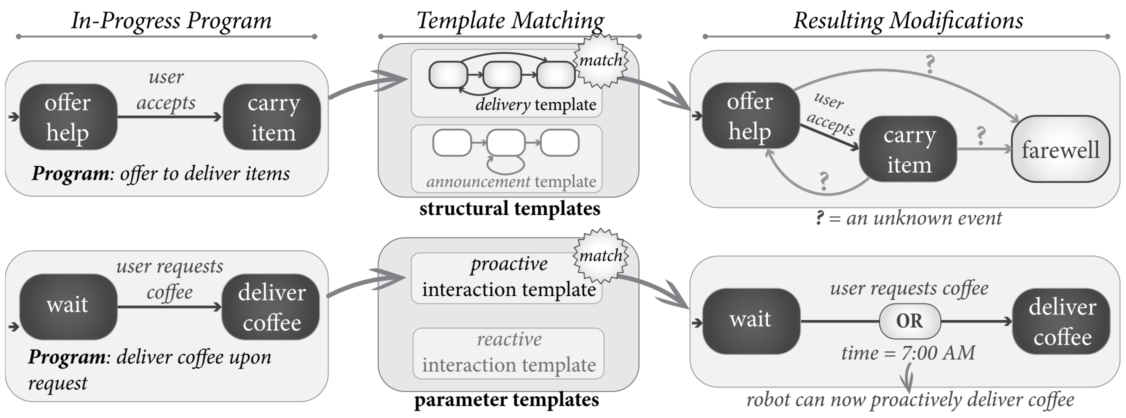

Figure 1. Examples of two delivery programs (left), hypothetically-matching templates (center), and proposed program modifications (right). The first example (top) shows a scenario in which helping a user carry items to a particular location is matched to a common delivery flow. The second example (bottom) shows a scenario in which delivering coffee to a user is matched to parameters that make the human-robot interaction *proactive* [10]. States represent actions and transitions are annotated with events that cause behavioral changes.

Our current progress with interaction templates is exploratory. In what follows, we (1) provide a preliminary vision of how templates may be used in authoring, (2) discuss different representations that templates may assume, and (3) provide insights about how templates might be collected.

2 Template Use

In this section, we explore preliminary use cases for templates. What does it look like to apply a template within an authoring tool? In which authoring paradigms would templates prove most useful? Do templates suggest, enforce, or notify developers of potential changes to a robot program?

For traditional keyboard-and-mouse authoring tools that support meticulous and iterative development, we envision templates to lead to an *auto-suggest* function that proposes, but does not enforce, changes. In flow-based (e.g., *Interaction Composer* [11]) and blockly-based (e.g., the *Opsoro* toolkit [12]) authoring systems, templates may serve to help add missing structure to an in-progress robot program, such as in the example workflow in Figure 1 (top). Here, a developer may have programmed a simple delivery interaction, causing the authoring tool to match the program to a delivery template and ultimately use the matched template to propose branching and looping modifications to the program. The template may be matched to the program using a simple "diff" heuristic, in which the best-matching template is decided based on how similar the flow of the template is to the flow of the original program. Furthermore, this particular template does not propose parameters to control the flow of the program, hence the question marks in Figure 1 (top-right). Other templates may propose changes to interaction parameters, such as in Figure 1 (bottom), involving an in-progress program of a robot that delivers coffee upon request. In this example, perhaps there exist *proactive* and *reactive* interaction templates based on the implicit interaction framework characterized by Ju [10]. The proactive template may match to in-progress programs based on prior knowledge that coffee delivery often occurs at the same time each day during the user's morning routine. The template may then provide suggestions to enable coffee deliveries to be made proactively, proposing a parameter change for the condition that guards the transition from "wait" to "deliver coffee."

While traditional authoring approaches benefit from developers providing meticulous attention to detail and can presumably present proposed modifications to the user in a way that is visually intuitive, various non-traditional interfaces also exist for programming robots, such as verbal interfaces [e.g., 13] and demonstration-based interfaces [e.g., 14]. These interfaces may provide fewer channels for feedback to developers and may also experience a significant degree of noise or uncertainty when sensing user input. Even for users of graphical interfaces that support meticulous, precise, and iterative development, the task at hand may require "quick-and-dirty" programming (e.g., for a delivery robot being programmed on the fly by an employee of a hotel [15]), leaving little room for checking the veracity of user input or for iterating on designs. Therefore, we suggest that for non-traditional

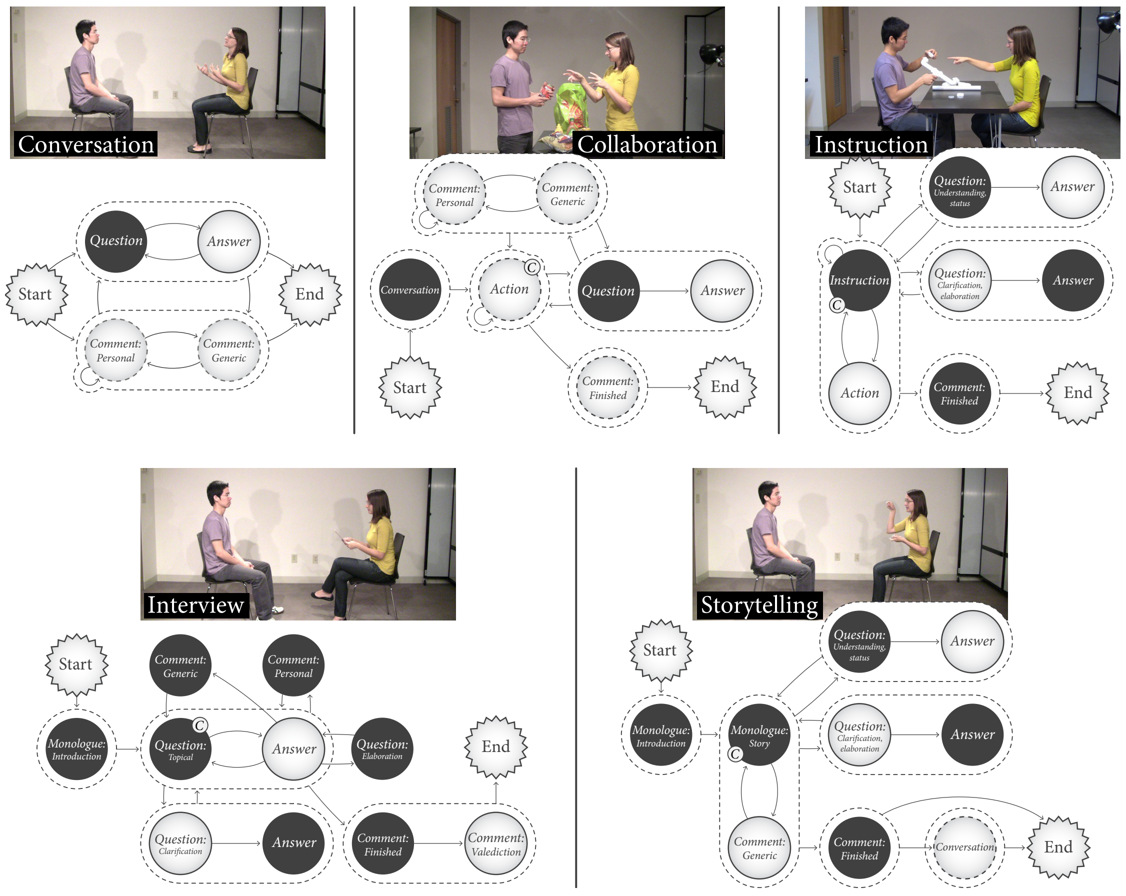

Figure 2. Five HRI models [18] derived from dyadic human-human interaction data. Each model contains start and end states. Dark and light gray states represent one or the other agent in the dyad. States with dotted outlines can be enacted by either agent. Larger dotted regions correspond to the patterns uncovered by Sauppé and Mutlu [17]. A "C" is placed over components that are core to the interaction scenario.

authoring interfaces and on-the-fly authoring pipelines that support fewer feedback channels, templates should largely enforce changes to a robot program. Fully enforcing changes without accepting further developer input, however, poses an increased risk that the resulting program will contain uncorrected erroneous robot behaviors. Therefore, some opportunity to provide feedback should still be offered to developers, and templates should only be matched to a program with high confidence.

In our own work, various authoring tools come close to using templates, but stop short. *RoVer* [16], in particular, checks in-progress programs during design time for social norm violations and presents developers with feedback and suggestions for improvement. Although the feedback mechanism is similar to that of our proposed interaction templates, automatically targeting feedback based on the interaction being programmed to different sets of social norm properties must consist of future work. Additional past work by the authors includes *Interaction Blocks* [17], which is based on a set of common interaction themes, or *design patterns* of interaction, observed in human-human interactions. Although design patterns can be treated as templates, *Interaction Blocks* does not use them as such.

3 Template Representation

What is the underlying model representing a template? A simple template may consist of a single, linear demonstration of how to perform a task or social interaction, such as each demonstration collected through the VirtualHome platform [19]. Alternatively, a template may arise from the combination of multiple demonstrations. Figure 2 depicts five individual interaction models from data qualitatively coded by Sauppé and Mutlu [17] that resemble nondeterministic finite automata and can serve as interaction templates.[1] These models, in particular, contain information about the high-level flow

1 The five interaction models are derived from data collected by Sauppé and Mutlu [17], later published in [18].

of an interaction rather than low-level interaction details such as timing, volume, and locomotion speed for mobile robots. Other templates may contain information about such low-level parameters. Templates may also contain probabilistic information derived from multiple demonstrations or from learning algorithms. State-of-the-art interaction adaptation approaches that employ reinforcement learning (RL) to automatically decide on the next optimal behavior for a robot to perform [e.g., 20], [21] can similarly be applied to authoring. A template may consist of a single RL model trained within a particular interaction context, and template matching may be performed by observing how well an in-progress robot program adheres to the model's optimal policy. A detriment of certain learning methods, however, is the sheer amount of data required to learn individual templates. As a result, we do not expect deep learning methods to prove as effective as methods that require less data.

A template's underlying model balances its ability to represent a wide variety of tasks and social interactions with its ability to make meaningful suggestions. Templates constructed from the models in Figure 2 can be applied to a wide variety of human-robot interactions. Due to their general nature and lack of low-level details, these templates may best be suited for generating *partial* suggestions, involving, for instance, a suggestion for the developer to add a loop between two states but not specifying the conditions for the loop to terminate. For on-the-fly programming interfaces, iterating with the developer to complete partial suggestions may cost too much time and be error prone. In these cases, templates may be better represented with a greater level of detail to enable the automatic resolution of suggestions generated by templates without requiring any further developer input. With greater detail per template, however, each individual template will represent a smaller range of possible robot programs. Collecting a greater number of templates may partially address this challenge.

4 Template Collection

There are many different ways to collect templates, each with a unique set of benefits and drawbacks. Recognizing that a vast amount of potential human-robot interaction scenarios fall under a small set of general categories, Sauppé and Mutlu [17] focused on collecting the small set of interaction models shown in Figure 2 within an in-person laboratory study.[1] The findings from various additional studies [e.g. 22]–[24] may prove similarly useful for creating further templates. However, such studies are time intensive and researchers must ensure that the data used to create templates represents the optimal design choices for an interaction, rather than representing, for instance, suboptimal designs that are suited specifically for laboratory settings.

Templates of common tasks may alternatively be collected from existing datasets of in-the-wild interactions, although many of these datasets are limited in scope as to the tasks or interactions they can represent. For human-robot conversations, the Loqui Human-Human Dialogue Corpus provides back-and-forth telephone conversations between patrons and employees at a service desk [25]. To overcome the potentially limited scope or scarcity of suitable in-the-wild data, datasets of freely available procedural knowledge on the web are viable alternatives [26]. A breadth of demonstrations per instruction, each representing a different way to perform a task or undergo an interaction, may also be useful, especially if one wishes to create templates that consist of automata or learned models. For multiple demonstrations per task, the VirtualHome dataset is appropriate, collected from multiple users of an online simulation of common household tasks and interactions that proved effective for collecting a large number of task models [19]. Planners may also be used to create artificial datasets of task instructions, as was done to create the ALFRED dataset [27].

5 Conclusion

We propose interaction templates as a technique for authoring robot programs and describe various considerations for their application. Further investigation is necessary to uncover the optimal collection, modelling, and usage strategies of templates given a particular programming task. The limitations of templates have also yet to be explored. In pursuing this future work, we can properly situate interaction templates within the toolkit of available techniques for authoring effective robot programs. We will also explore the use of interaction templates for autonomous agents in general, including, for instance, smart home agents, rather than limiting interaction templates specifically to robots.

References

[1] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[2] S. Srivastava, S. Gulwani, and J. S. Foster, “Template-based program verification and program synthesis,” *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5, pp. 497–518, 2013.

[3] A. Solar-Lezama, *Program synthesis by sketching*. University of California, Berkeley, 2008.

[4] ——, “The sketching approach to program synthesis,” in *Asian Symposium on Programming Languages and Systems*, Springer, 2009, pp. 4–13.

[5] ——, “Program sketching,” *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5, pp. 475–495, 2013.

[6] P. Ferrarelli, M. T. Lázaro, and L. Iocchi, “Design of robot teaching assistants through multi-modal human-robot interactions,” in *International Conference on Robotics and Education RiE 2017*, Springer, 2017, pp. 274–286.

[7] J. Motes, R. Sandström, W. Adams, T. Ogunyale, S. Thomas, and N. M. Amato, “Interaction templates for multi-robot systems,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2926–2933, 2019.

[8] S. Waldherr, R. Romero, and S. Thrun, “A gesture based interface for human-robot interaction,” *Autonomous Robots*, vol. 9, no. 2, pp. 151–173, 2000.

[9] J. Nichols, B. A. Myers, and K. Litwack, “Improving automatic interface generation with smart templates,” in *Proceedings of the 9th international conference on Intelligent user interfaces*, 2004, pp. 286–288.

[10] W. Ju, “The design of implicit interactions,” *Synthesis Lectures on Human-Centered Informatics*, vol. 8, no. 2, pp. 1–93, 2015.

[11] D. F. Glas, T. Kanda, and H. Ishiguro, “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, 2016, pp. 303–310.

[12] A. D. Frederiks, J. R. Octavia, C. Vandevelde, and J. Saldien, “Towards participatory design of social robots,” in *IFIP Conference on Human-Computer Interaction*, Springer, 2019, pp. 527–535.

[13] M. Forbes, R. P. Rao, L. Zettlemoyer, and M. Cakmak, “Robot programming by demonstration with situated spatial language understanding,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2014–2020.

[14] H. Knight and R. Simmons, “An intelligent design interface for dancers to teach robots,” in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2017, pp. 1344–1350.

[15] J. Huang, T. Lau, and M. Cakmak, “Design and evaluation of a rapid programming system for service robots,” in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2016, pp. 295–302.

[16] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, “Authoring and verifying human-robot interactions,” in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 75–86.

[17] A. Sauppé and B. Mutlu, “Design patterns for exploring and prototyping human-robot interactions,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 1439–1448.

[18] A. V. Sauppe, “Designing effective communication strategies for human-robot collaboration,” Ph.D. dissertation, The University of Wisconsin-Madison, 2015.

[19] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, “Virtualhome: Simulating household activities via programs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[20] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, “Robot gains social intelligence through multimodal deep reinforcement learning,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 745–751.

[21] K. Weber, H. Ritschel, I. Aslan, F. Lingenfelser, and E. André, “How to shape the humor of a robot-social behavior adaptation based on reinforcement learning,” in *Proceedings of the 20th ACM international conference on multimodal interaction*, 2018, pp. 154–162.

[22] P. H. Kahn, N. G. Freier, T. Kanda, H. Ishiguro, J. H. Ruckert, R. L. Severson, and S. K. Kane, “Design patterns for sociality in human-robot interaction,” in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, 2008, pp. 97–104.

[23] Y. Mohammad, Y. Xu, K. Matsumura, and T. Nishida, “The h 3 r explanation corpus human-human and base human-robot interaction dataset,” in *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, IEEE, 2008, pp. 201–206.

[24] D. B. Jayagopi, S. Sheiki, D. Klotz, J. Wienke, J.-M. Odobez, S. Wrede, V. Khalidov, L. Nyugen, B. Wrede, and D. Gatica-Perez, “The vernissage corpus: A conversational human-robot-interaction dataset,” in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2013, pp. 149–150.

[25] R. Passonneau and E. Sachar, *Loqui human-human dialogue corpus (transcriptions and annotations)*, Available from https://academiccommons.columbia.edu/doi/10.7916/D82R3PW9, 2014.

[26] P. Pareti, B. Testu, R. Ichise, E. Klein, and A. Barker, “Integrating know-how into the linked data cloud,” in *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2014, pp. 385–396.

[27] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, “Alfred: A benchmark for interpreting grounded instructions for everyday tasks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.