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 do little in the way of 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 different dimensions that define first how interaction templates can be used, then how interaction templates can be represented, and finally how they might be collected.

Keywords: Human-robot interaction. Templates. Authoring.

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. Developers often must possess some level of programming expertise in order to leverage multiple robot actuators to create even simple behaviors and asynchronously interpret raw input from the robot's multi-modal sensing capabilities to create higher-level meaning. Even for experienced developers, a robot can fail in its task or interaction if not suited to the interaction context within which it is placed. The physical embodiment of the robot exacerbates these development challenges, because traditional keyboard-and-mouse programming tools may be unavailable if development must be performed on-the-fly. When in the field, often less precise development methods, such as natural language programming, must be used. Especially for on-the-fly development which offers reduced opportunity for iterative design and testing, it is imperative that authored programs are provably 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 a human-robot interaction derived from interaction data for the purpose of assisting developers specifying an interaction design. Furthermore, in our own vision of interaction templates, a developer using an authoring tool to construct an in-progress interaction design 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, the authoring tool will proactively *match* an appropriate subset of templates to the in-progress design 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 design becomes

*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

DOI: 10.35699/1983-3652.yyyy.nnnnn

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

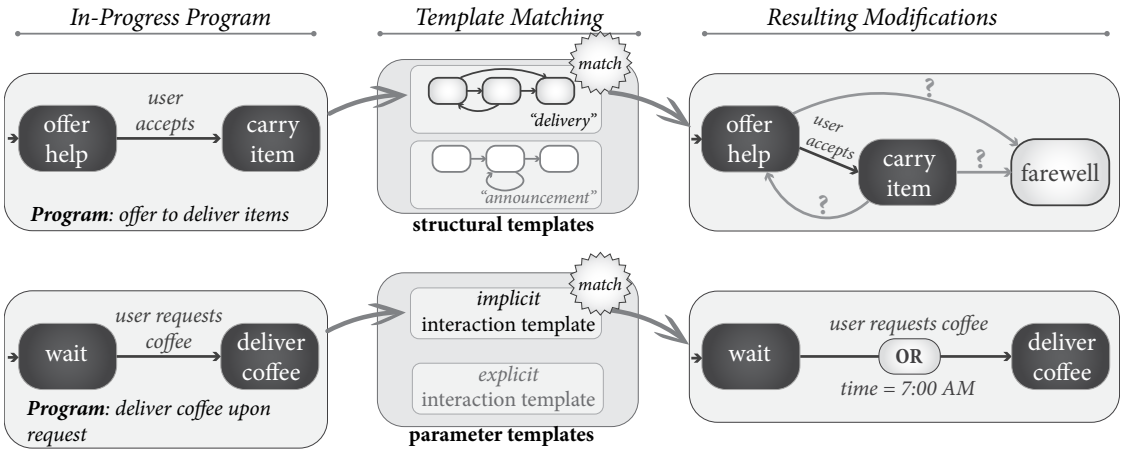

Figure 1. Examples of two in-progress "delivery" programs (left), matching templates (center), and proposed program modifications (right). The first example (top) demonstrates a hypothetical scenario in which the action of helping a user carry items to a particular location may be matched to a common delivery flow. The second example (bottom) demonstrates another hypothetical scenario in which the action of delivering coffee to a user is matched to a common set of parameters that make the interaction *implicit* [10]. States represent robot behaviors and transitions are annotated with events that cause changes in behaviors. Question marks represent unknowns.

proposed. Additional uses of templates in prior work include robot motion planning and analysis [7], [8] and generating user interfaces [9]. Figure 1 depicts two hypothetical sets of templates and how they might be matched to in-progress robot programs and used to propose changes.

In order to provide as concrete of a discussion of templates as possible, we begin with our vision of how templates may be used in authoring tools. Subsequently, we will discuss the many different representations that templates can take. Lastly, we provide insights about how templates can be collected. We supplement each discussion with our own experiences in designing human-robot interactions and additional prior work that has been done.

2 Template Use

In this section, we explore answers to several questions regarding how templates may be used. What does it look like to apply a template to an authoring tool? In which authoring pipelines would templates be most useful? Furthermore, do templates suggest, enforce, or notify the developer of potential changes to a robot program?

For traditional keyboard-and-mouse authoring tools that support meticulous and iterative design and development, we envision templates to best serve as both *auto-complete* and *auto-correct* functions that suggest, but do 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 interaction design, such as in Figure 1 (top), where a simple delivery program matches to a delivery template that results in suggestions for branching and loops. In this particular example, the template does not propose parameters to control the flow of the program, therefore resulting in the "holes" denoted by question marks. Other templates may propose changes to interaction parameters, such as in Figure 1 (bottom), involving an in-progress interaction design of a robot that delivers coffee upon request. In this example, perhaps there exist *explicit* and *implicit* interaction templates based on the implicit interaction framework characterized by Ju [10]. The implicit template may match to in-progress designs 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 delivery interactions to exist in the background, namely that the robot should proactively take action without waiting for an explicit command from the user. Thus, the template in this example proposes a change to a parameter of the interaction, namely the condition that guards the transition from "wait" to "deliver coffee."

Various non-traditional interfaces also exist for programming robots, such as verbal interfaces [e.g.,

Figure 2. Five hand-crafted HRI templates derived from dyadic human-human interaction data. Each template contains a start and an end state. Dark and light gray states represent one or other agent in the dyad, respectively. States with dotted outlines can be enacted by either agent. Larger dotted regions correspond to the patterns uncovered by Sauppé & Mutlu [17].

13] and demonstration-based interfaces [e.g., 14], that 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 programming 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 authoring interfaces and on-the-fly authoring pipelines that support fewer feedback channels, templates should enforce changes to an interaction design, rather than suggest them.

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 the 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 simulator [18]. Alternatively, a template may arise from the combination of multiple demonstrations. Figure 2 depicts five individual interaction templates manually constructed from the data collected by Sauppé & Mutlu [17] that resemble nondeterministic finite automata. These templates, in particular, contain information about the high-level flow 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., 19], [20] 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 interaction design 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 possible tasks and social interactions with its ability to make meaningful suggestions. The templates 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 specifications of the conditions for the loop to terminate. For on-the-fly programming interfaces, iterating with the developer to complete partial suggestions may require 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 input from the developer. With greater detail per template, however, each individual template is less likely to represent or match-to such a wide variety of possible interaction designs. To address this challenge, more templates should be collected.

4 Template Collection

There are many different ways to collect templates, each with a unique set of benefits and drawbacks. Sauppé & Mutlu [17] collected the templates in Figure 2 within an in-person laboratory study, and the findings from various additional studies [e.g. 21]–[23] may prove useful for creating further templates. However, such studies are time-intensive and creating a comprehensive set of templates may be infeasible. Furthermore, researchers must ensure that the data used to create templates represents the optimal design choices for an interaction, rather than representing, for instance, a particular sub-optimal design that was used for laboratory testing purposes.

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 dataset provides back-and-forth telephone conversations between patrons and employees at a service desk [24]. To overcome the potentially limited scope or scarcity of existing in-the-wild datasets, datasets of human annotations are viable alternatives, such as the Web of Know-How dataset of WikiHow instructions [25], which contains a breadth of hand-crafted instructions for both tasks and social interactions. 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 Unity simulation of common household tasks and interactions [18] that proved effective for collecting a large number of templates. Planners may also be used to create artificial datasets of task instructions, as done to create the Alfred dataset [26].

5 Conclusion

We propose interaction templates as a technique for authoring robot programs and describe various considerations for their application. Further investigation is necessary uncover the optimal collection, modelling, and usage strategies of templates given a particular interaction design 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.

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] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language," *Robotics and Autonomous Systems*, vol. 38, no. 3-4, pp. 171–181, 2002.

[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] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.