Authr: A Task Authoring Environment for Human-Robot Teams

Andrew Schoen, Curt Henrichs, Mathias Strohkirch, Bilge Mutlu
University of Wisconsin–Madison, Madison, Wisconsin, USA
{schoen,cdhenrichs,strohkirch,bilge}@cs.wisc.edu

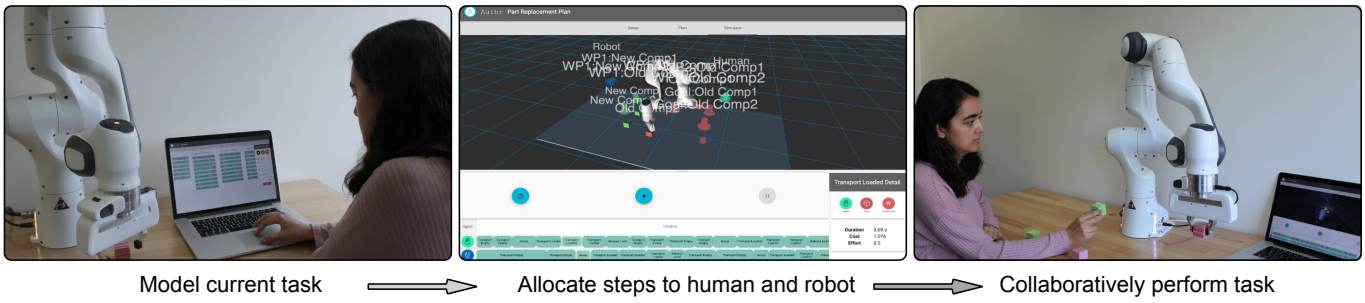

Figure 1. We present a novel workflow and a software environment, called *Authr*, that enable engineers to translate single-person, work-related tasks in domains ranging from manufacturing to logistics into tasks that can be performed by human-robot teams.

ABSTRACT

Collaborative robots promise to transform work across many industries and promote "human-robot teaming" as a novel paradigm. However, realizing this promise requires the understanding of how existing tasks, developed for and performed by *humans*, can be effectively translated into tasks that *robots* can singularly or *human-robot teams* can collaboratively perform. In the interest of developing tools that facilitate this process we present *Authr*, an end-to-end task authoring environment that assists engineers at manufacturing facilities in translating existing manual tasks into plans applicable for human-robot teams and simulates these plans as they would be performed by the human and robot. We evaluated *Authr* with two user studies, which demonstrate the usability and effectiveness of *Authr* as an interface and the benefits of assistive task allocation methods for designing complex tasks for human-robot teams. We discuss the implications of these findings for the design of software tools for authoring human-robot collaborative plans.

Author Keywords

Human-robot teaming; human-robot collaboration; authoring; robot programming; visual programming; task allocation

UIST '20, October 20–23, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7514-6/20/10 ...$15.00.
http://dx.doi.org/10.1145/3379337.3415872

CCS Concepts

•**Human-centered computing** → **Graphical user interfaces; Collaborative interaction; User interface programming;** •**Computing methodologies** → *Multi-agent planning; Robotic planning;* •**Computer systems organization** → *External interfaces for robotics;*

INTRODUCTION

The introduction of robots into industrial environments to automate physical work has been a paradigm shift for many industries, including manufacturing and logistics, increasing productivity and efficiency. To achieve these goals, highly capable, but inherently human-unsafe, robots had to be sequestered, separating work that can be automated with robots from manual work performed by people. The emergence of *collaborative robots* (cobots), *i.e.,* under-actuated robots designed to be human-safe and easy-to-use, is signaling another paradigm shift toward more flexible, collaborative workspaces with the potential to improve productivity and safety [36].

The prospect of *human-robot teaming*, where humans and robots collaboratively perform parts of the task that they are best suited to perform, holds considerable promise for improving industrial work, but significant hurdles still remain in capitalizing on that promise [10]. An ethnographic approach has indeed shown that there is a discrepancy between the traditional robot programming approaches used by developers and engineers who integrate robots into industrial environments and the needs of collaborative interaction design [29]. Put another way, the task of specifying *collaborative* tasks requires a different approach than what is afforded by standard non-interactive robot programming approaches. The result of this discrepancy is that most cobots today are still used as

traditional robots, but without the protective cages surrounding them. Overcoming this discrepancy requires answering a number of questions. For example, how can a user, *e.g.,* an engineer, take an existing task, *e.g.,* assembling a toy, and turn it into a human-robot plan? How can the user implement the robot's portion of the plan? How can the user know that the plan best utilizes the skills of human and robot workers? Although robot programming tools enable users to quickly program collaborative robots through demonstration, *e.g.,* the programming by demonstration (PbD) approach developed by Skoglund *et al.* [40], or using visual programming environments (VPEs), *e.g.,* CoSTAR [34], no tools exist to support users in the entire process of translating existing *human* tasks to those that *human-robot teams* can perform within the manufacturing context. In this paper, we outline the technical challenges involved in authoring human-robot plans and present our authoring environment, *Authr*, as a solution.

Our work addresses four key technical challenges involved in human-robot teaming: (1) *representation:* representing work for both human interpretation and robot execution; (2) *task-skill matching:* creating human-robot plans that match task elements with worker skills while achieving task goals; (3) *robot programming:* implementing task elements for collaborative robots in a way that supports exploration of task plans across robot platforms; and (4) *authoring pipeline:* facilitating intuitive and effective translation of manual work into human-robot plans. Building on methods and tools from ergonomics, robotics, and human-computer interaction, we address these challenges by (a) formalizing a task- and action-level representation that is human-interpretable and robot-executable, (b) utilizing a multi-agent allocation algorithm that generates plans that match worker skills to task elements within task constraints, (c) developing a software stack which converts plans into robot-executable actions built on an extendable Robot Operating System (ROS) [37] infrastructure, and (d) designing an intuitive software environment that enables users to effectively create human-robot plans.

In the remainder of the paper, we discuss these technical challenges in more detail, describe our solutions for each challenge, present the system design and implementation of *Authr*, describe two user studies that evaluated different facets of human-robot teaming using *Authr*, and discuss our findings and their implications for the design of tools which support the authoring of human-robot collaborative plans. The contributions of this work include:

- A novel *workflow* to translate manual human tasks to human-robot tasks;
- Novel *representations* and formalizations for modeling, planning, simulation, and implementation;
- The design of an *authoring environment* that supports users in following this approach;
- An *open-source implementation* of the environment for public use and further development;[1]
- *Empirical evaluations* of the approach and the authoring environment through a series of user studies.

[1] https://github.com/Wisc-HCI/authr

RELATED WORK

A great deal of prior work has focused on the development of visual programming environments (VPEs) to enable easy programming of tasks. A primary example is the student-oriented *Scratch* interface, which uses a block design to indicate conventional programming constructs [27]. This approach has inspired a number of VPEs such as *Hammer*, a robotics-focused, android-based programming tool allowing novice users to design programs for robot arm movement and tool use [28], and *Code3*, a drag-and-drop system built for the PR2 robot [18], among others. Another influential VPE, *LEGO Mindstorms NXT Programming Environment*, focused on education and robotics [21, 22]. Flow designs have also been investigated: Roboflow embeds pre- and post-conditions into flow structures, focusing on a low-level specification of behaviors for robotics [2]; and *ROSco* (ROS Commander), a tool created for the PR2 Robot, uses hierarchical finite state machines and low-level building blocks to specify spatially-situated actions [33]. While all these interfaces contributed substantially in a variety of ways, they generally focused on specifying *robot behavior*, as opposed to *human-robot collaboration*.

The space of human-robot collaboration specification is still quite new. The ROBO-PARTNER project has helped by articulating the needs and requirements of such systems, namely user-friendly interfaces, planners that allow the creation of efficient human-robot collaboration task plans, robot instruction libraries that allow for easy generation and modification of robot programs, and continual attention to safety concerns [30, 31]. In an attempt to begin addressing these requirements, the *CoSTAR* system was developed, which integrates perception and reasoning into behavior trees [34, 35]. While the interface was successful in allowing users to specify complex programs, users had difficulties understanding the types and intentions of the robots' actions. *RAZER* was designed for task-level programming to allow shop-floor operators to leverage lower-level actions developed by experts, and it was later extended to support programming by demonstration [43, 42]. They compared their solution with systems such as CoSTAR and Scratch, finding *RAZER* to be easier to understand by non-experts. Graphical representation of the workspace to assist users in creating task graphs have also been explored [38].

In an effort to improve the efficiency of human-robot plans, research into multi-agent task planning has been explored with works such as *Tercio* [14] and *multi-abstraction search approach* (*MASA*) [48]. *Tercio* takes inspiration from real-time processor scheduling for multi-robot hierarchical problems. The objective is to assign tasks to agents and schedule tasks with the goal of minimizing change in agent assignment and minimize number of spatial interfaces between tasks assigned to different robots. *MASA* uses a multi-level optimization approach with three phases: finding an initial solution for agent placement, hill-climbing to minimize maximum makespan, and finally refinement to the solution. While both approaches work well for optimizing agent allocation, there is a relatively high planning time. Another consideration is prioritization of various goals such as maximal efficiency and minimal strain, as shown by Pearce *et al.* [36]. They found that tasks that benefited most from the goal of minimizing time and ergonomic

strain were ones which enabled parallel work, were repetitive, and utilized robot-performable actions.

Researchers have started to address how to leverage agent allocation in human-robot-collaborative authoring environments with systems such as *Sharedo* [19] and *WeBuild* [12]. *Sharedo* functions as a structured to-do list for daily tasks where multiple agents (human, robots, virtual-assistants) coordinate based on their capabilities. *WeBuild* provides allocation of tasks for multiple humans with varying capabilities in order to offload group coordination. Our work, drawing from the related literature, addresses the challenges of authoring human-robot collaboration within the manufacturing context.

TECHNICAL APPROACH

Technical Challenges

Translating manual tasks into human-robot task plans involves a number of technical challenges. We discuss these challenges in this section and detail our solutions in the next section.

1. Representing tasks for humans and robots

Translating tasks that are currently performed manually by human workers into human-robot plans requires representing them in a way that is both interpretable by a human, so that they can be trained on the task and their performance can be assessed, and executable by a robot. Tasks describing manual work in manufacturing settings are generally represented as written natural-language lists of mid-level descriptions of task actions. Although this representation is human-interpretable, implementing tasks into robots based on these descriptions is challenging [35]. Furthermore, users without the necessary experience in developing collaborative applications may generate implementations which are not generalizable across robot platforms and are ill-suited for task-level analysis of plan efficiency or safety. Therefore, we need a representation that enables the user to capture task elements from natural-language descriptions or from qualitative observations of the task and to specify task elements for humans and robots to perform.

2. Matching task elements with worker skills

Answering the question of which aspects of the task that robots and humans should perform is critical to realizing the promise of human-robot teaming for improved productivity and worker safety. This requires effectively matching human and robot skills to elements of the task, considering the cost of the human or the robot performing the elements. Furthermore, while a simple matching can determine whether a specific task element can be performed by a human or a robot, it does not help the user determine whether it should be performed by a human or a robot given specific task expectations and requirements, such as speed (a robot that can perform a task element may be too slow) and ergonomic safety (a task element that a human worker can perform much more efficiently may be ergonomically unsafe for the human worker). Hence, there is a need to match task elements to the skills and capabilities of human workers while considering outcomes such as efficiency and safety at a task level.

3. Supporting exploration across robot platforms

When engineers in industry are considering converting a manual process into one involving a collaborative robot, either as automation performed by the robot or collaboration between the robot and a human operator, they are faced with the decision of using manufacturer-provided software environments (*e.g.,* Universal Robots Polyscope[2]), utilizing third-party tools (*e.g.,* Artiminds[3]), or developing a custom software solution built on top of low-level APIs. Compounding the problem of making an informed choice is a potential lack of experience in developing collaborative human-robot teaming applications [29]. Therefore, we need to provide a tool that enables users to quickly and easily evaluate their tasks for multiple robot platforms before purchasing a particular robot. For example, an engineer interested in understanding whether a Universal Robots UR5 robot or a Franka Emika Panda robot would better fit into a given task would likely have to implement the same task for both robots using different programming tools or setups, as highlighted by the creators of the CoSTAR robot programming environment [16]. Furthermore, if the engineer is interested in seeing alternative task plans in action to further refine them, the user must program each plan individually. Users should be provided visual or demonstration-based robot programming tools in order to easily program robots and integrated planning tools to easily handle skill-based task allocation. Thus we need to enable the user to quickly and easily develop, deploy, view, and modify task plans for end-to-end exploration across multiple collaborative robot platforms.

4. Developing an intuitive and effective authoring pipeline

A final technical challenge is to enable users to rapidly and iteratively capture task models for manual work, explore human-robot task plans, and deploy the created plans on robot platforms for assessment, refinement, and training. Although users might have prior experience with robot programming tools, such as the demonstration or visual-programming tools that are commonly used to program collaborative robots, we must create intuitive software tools that users can quickly learn and use in order to effectively facilitate the complex process of human-robot teaming.

Technical Solutions

Below we detail the technical approach to our authoring environment, *Authr*, addressing each technical challenge in order.

1. Creating a shared representation for human-robot work

The goal of our representation is to facilitate the translation of natural-language task descriptions to a formal representation that remains interpretable to human collaborators, yet robots can understand and perform without having to update their underlying implementation. The tasks being translated are generally in the form of written natural-language lists composed of task specific actions or tasks that can be observed by an engineer as they are being performed by an operator. Although examples of translating task-specific actions into robot action primitives exist in various domains (*e.g.,* cooking [4] and route-navigation [6]), these solutions tend to be

[2] https://www.universal-robots.com/

[3] https://www.artiminds.com/

highly contextual or robot-specific. One promising solution is *Therbligs*, as proposed by Gilbreth and Gilbreth [13], which address the issue of defining operational action primitives for human work. Researchers have since applied *Therbligs* to modeling or specifying robot behavior in various contexts [25, 24, 23, 36, 1].

Our representation builds on *Therbligs* and is further inspired by Pearce *et al.*'s work [36] where they modeled human-robot tasks using Hierarchical Task Analysis (HTA) with lowest-level sub-tasks allocated between humans and robots. In work analysis literature, HTA decomposes tasks into nested sub-tasks until sufficient detail is achieved to perform work actions [41]. Our representation adopts this approach with two important changes. First, our approach only considers three levels in HTA, operationalized as the *Plan*, *Task*, and *Therblig*, where *Therbligs* function as sub-tasks. Second, *Therbligs* have both high- and low-level parameters. High-level parameters include *Agents*, *Things*, *Destinations*, while low-level parameters include numerical values, such as gripper effort, time, and cost.

Agents, Things, and Destinations—In our representation, *Agents*. *Things*, and *Destinations* are used to fully specify the high-level behaviors of *Therbligs*. Consider the action of a robot placing an item, such as a mug, in a container for shipping. In this case, we can think of the transport action as the *Therblig*. High-level parameters, such as *Agents*, *Things*, and *Destinations* serve to characterize these therbligs and more clearly define their behavior. Thus, in the mug packing example, the *Therblig* is specified by the *Agent* performing it (the robot), the *Thing* being moved (the mug), and the *Destination* it is moved to (the shipping container). Thus, the combination of high-level parameters and the *Therbligs* serve to symbolically define the action to the engineer. In *Authr*, we consider an *Agent* to be any physical actor in the work environment that performs a relevant action within the context of the *Plan* and has a type (human or robot). *Things* are regarded as objects within the environment that are manipulated by *Agents* in the *Plan*. *Destinations* are operationalized to combine semantic labels, such as the described location (*e.g.,* the "shipping container" in the example above) with a concrete position and orientation that a robot could act on. Due to this representation, *Authr* enforces a strict set of spatial expectations on the workspace, meaning real-time dynamics and variablity are not considered in the current version of the system. As such, this solution works well for clearly-defined workspaces (*e.g.,* kitting), but not for ones with variable *Thing* counts or positions (*e.g.,* bin-picking).

Plan, Tasks, and Therbligs—At the highest level of our HTA approach is the *Plan*, which in *Authr* reflects the entirety of the human-robot collaborative work being designed. The *Plan* is composed of *Tasks*, which represent high-level descriptions of behaviors used to achieve specific processes in the *Plan*. *Tasks*, in turn, are composed of *Therbligs*. The full set of 18 *Therbligs* includes physical actions, cognitive processes, and behaviors that are both physical and cognitive [13]. For the current implementation of *Authr*, we focus on physical actions, resulting in the following list of *Therbligs*: (1) *Transport Empty*, (2) *Transport Loaded*, (3) *Grasp*, (4) *Release Load*, (5) *Rest*, and (6) *Hold*. These *Therbligs* are also listed in Figure 2 along with their descriptions. By focusing exclusively on physical *Therbligs*, our task space is generally constrained to pick-and-place-type tasks (*e.g.,* kitting, assembly, palletization). Some limited tool use can be created in an *ad hoc* manner (*e.g.,* grasping a screwdriver and defining screwing rotation through multiple *Transport Loaded Therbligs*), but tasks requiring cognitive evaluation (*e.g.,* force-sensed peg-in-hole or component inspection) are currently not addressed in our representation. Further work is needed to operationalize cognitive and mixed cognitive-physical *Therbligs*.

Setting an *Agent* for the high-level parameter of a *Therblig* has the effect of allocating it to that *Agent*, and leaving it empty prompts automated allocation. Other high-level parameters, such as *Things* and *Destinations* are required. These high-level parameters are used to generate pre- and post-conditions of each therblig, which serve to describe when the *Therblig* can be performed and what the effect on the workspace will be. In the mug packaging example from above, given that the action was configured with the robot, the mug, and the shipping container, we can say that for this action to be performed, the robot must be both holding the mug, and the space for the mug in the container must be empty. At the end of the action, both the mug and the robot will be positioned at the container. The full breakdown of parameters, pre-conditions, and post-conditions for our *Therbligs* are shown in Figure 2.

Alongside high-level parameters (*Agents*, *Things*, and *Destinations*, we need a way to standardize and compare the quality of *Therbligs* to sufficiently allow for shared representation of these collaborative tasks. Low-level parameters support this reasoning by providing low-level information required to execute an action by the robot, but may not be required by the human (*e.g.,* gripping effort). In this way, if a robot can theoretically perform the task, and if it is allocated the *Therblig*, it has the necessary information to perform the task. Low-level parameters also provide comparative power to *Therbligs* allocated to separate *Agents*. Time to complete an action can be simulated by the robot, but knowing the time for the human to perform the task would be necessary for determining which *Agent* is fastest at performing it. Likewise, if a task is hard for a human but easy for the robot (or vice versa), being able to weigh these values is necessary for thoughtful allocation of tasks. One common metric of difficulty is ergonomic strain, and being able to flexibly define this cost for a given *Therblig* and *Agent* can empower the engineer to construct programs that provide robotic assistance where it is needed most.

2. Enabling effective task allocation in human-robot teams

A critical challenge in authoring human-robot collaborative tasks is the gap between engineers' ability to construct single-agent programs and the know-how of designing interactive tasks. Even if some tasks or sub-tasks are only executable by one *Agent*, the rest of the interaction needs to be planned in a way that incorporates those restrictions on agent allocation. Since many engineers from the manufacturing domain have access to task specifications (albeit non-interactive, manual ones), we sought a representation that translates this type of non-agent focused representation into an interactive plan. Our

operationalization of *Therbligs*, along with the parameters we specify, allows for a direct translation from their task specifications to the shared *Therblig* representation, from which the interactive *Plan* is constructed. This construction process requires any non-specified *Agents* to be allocated to a given task, all while accommodating specified (*i.e.,* parameterized) *Agents* and considering cost and time estimates.

Our proposed allocation process is performed in a series of steps. First, the *Plan* is checked using the SMT solver Z3 [8], in which the pre- and post-conditions of each therblig are translated into first-order logic and verified. This same algorithm is used continuously during plan construction to provide feedback about program correctness to the user. Next, the *Plan* is further checked that all needed parameters are defined, since not all parameters need to be set for verification to succeed. Following this parameter check, the *Plan* proceeds to allocation. For this purpose, a breadth-first search through the interaction is utilized, resulting in a set of possible interaction traces. In the worst case, the state size upon applying each *Therblig* t_n of $t_1, t_2, ..., t_n$ has an upper bound of 2^n for two agents. However, we observe that users typically chain together consecutive *Therbligs* for an *Agent* into individual *Tasks* (*e.g.,* pick-and-place: *Transport Empty* $\rightarrow$ *Grasp* $\rightarrow$ *Transport Loaded* $\rightarrow$ *Release Load*). Due to the pre- and post-constraints, *Things* act as tokens constraining the growth of the state space, and the state space grows instead with 2^m where m is the number of *Tasks*. Since these traces are modeled as a single sequence of consecutive actions (for computational efficiency), the *Plan* is then parallelized such that allocated *Therbligs* are performed as soon as possible while maintaining first-order logic. The resulting traces are compared for overall time and cost, using the provided time and cost weights, and the optimal interactive *Plan* is returned.

The method described above was chosen because it most closely matched the formulation of the *Plans* as provided by the users, namely an initial state and a set of non-allocated *Therbligs* to perform. With some additional work, and some caveats, the *Plans* can be converted into standard planning-based problems. The first caveat is that due to the nature of our approach, the ordering of *Therbligs* is constrained for a given *Agent*. Combined with the token-like nature of *Things*, this means that users can specify sequences of *Transport Loaded* *Therbligs*, thereby creating waypoints, with certainty of the ordering that the *Agent* will visit them. Additionally, if some intermediate placement of an *Agent* or *Thing* is required, but not captured in the final state, a coarse planning will not result in this configuration, unless segmented into multiple planning problems or supplying additional explicit goals.

Therblig	Parameters	Pre-/Post-conditions
Transport Empty	agent, dest.	**Pre:** ¬ *gripping*
Reaching for thing with empty hand/gripper		**Post:** *location(agent)=dest.*
Grasp	agent, thing, effort	**Pre:** ¬ *gripping* ^ *location(agent)=location(thing)*
Grasping object with hand/gripper		**Post:** gripping
Transport Loaded	agent, thing, dest.	**Pre:** *gripping* ^ *location(agent)=location(thing)*
Moving thing in hand/gripper to destination		**Post:** *location(agent)=dest.* ^ *location(thing)=dest.*
Release Load	agent, thing	**Pre:** *gripping*
Releasing thing from hand/gripper		**Post:** ¬ *gripping*
Hold	agent, thing, dur.	**Pre:** *gripping* ^ *location(agent)=location(thing)*
Pausing while holding thing in hand/gripper		**Post:** *none*
Rest	agent, dest., dur.	**Pre:** ¬ *gripping* ^ *location(agent)=dest.*
Pausing for specified duration without grasp		**Post:** *none*

Figure 2. A description of the *Therbligs* implemented in *Authr*, including parameters, pre-conditions, and post-conditions.

While the allocation and parallelization algorithms specified were sufficient for the complexity and size of *Plans* considered in this study, it is important to consider how such methods compare to more conventional planning approaches. To this effect, we ran benchmarks with our process and a *Multi-Objective Divide-and-Evolve* (*MO-DAE*) algorithm, which is an evolutionary algorithm which supports multi-objective planning [9, 20]. Since interactive design is a key component in the user's workflow, we needed any algorithm to be sufficiently fast. Thus, we capped the maximum compute time at 90 seconds for quick user feedback. As input, we modeled three *Plans* (shown in Figure 3) in *Authr* based on real-world manufacturing tasks. Each *Plan* was evaluated five times with each algorithm. Our algorithm was deterministic, so there was no variation other than slight differences in compute time.

The first *Plan* models a kitting task (assembling objects into containers or kits) in which there is a grouping of four toys and four batteries on the left side of the workspace. The goal of this *Plan* is to move one toy and one battery into four separate boxes, located to the right. The second *Plan* models a circuit board assembly task. The initial state of this *Plan* consists of a group of four nuts located to the right of a PCB, and two cables positioned above the board. To complete this assembly process, one nut must be screwed onto each corner of the circuit board, and each of the two cables connected to the circuit board. The third *Plan* models a repair task in which two faulty components are replaced by new components on a circuit board. The two components are functionally different, with one requiring considerable cost for the human to place but not remove. This repair task starts with the two distinct faulty components attached to the circuit board and the two distinct new components off to the side. The goal of this *Plan* is to remove both faulty components, placing them to the right, and replace them with the new components of the same type, located to the left.

In order to estimate the *Therblig* times for the human *Agent*, we set up a physical representation of each task and recorded the amount of time it took for a human to perform each *Therblig*.

For each of the three *Plans* we evaluated, we targeted different time and cost metrics. In the first *Plan* (kitting), the objective was focused on minimizing time, so the optimization weights for time and cost were set to 0.6 and 0.4, respectively. In the second *Plan* (assembly), the objective was to minimize cost, so time and cost weights were configured at 0.05 and 0.95, respectively. Of concern in this *Plan* was the ergonomic strain for humans in screwing in the nuts. Thus, the cost of this action was configured to be higher for human than robots (0.9 vs 0.2). In the third *Plan* (repair), we set the cost weight to 0.6 and the time weight to 0.4. In this *Plan*, we were interested in the effect of a high cost related to a single action and *Agent*, as opposed to a class of actions.

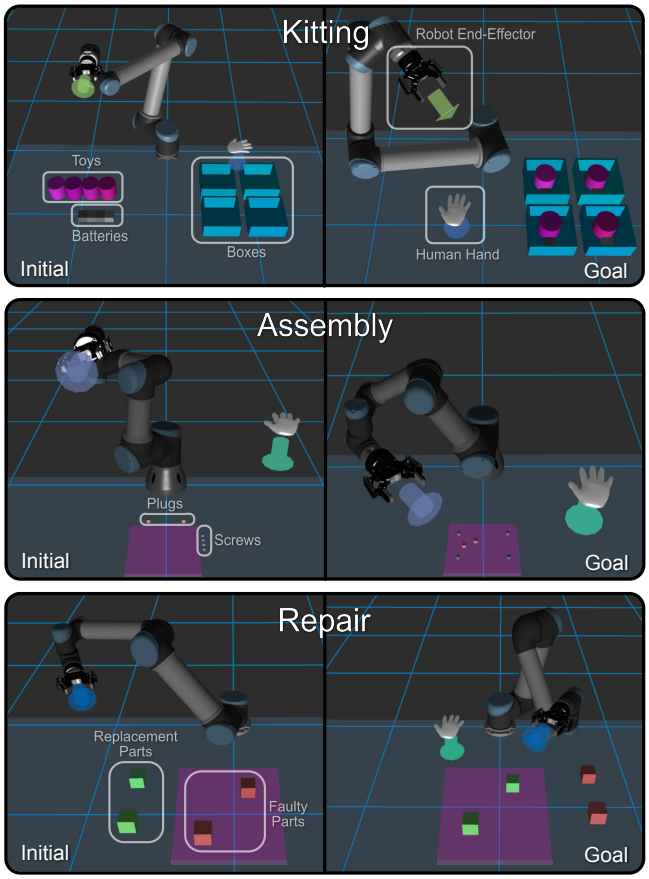

Figure 3. For the technical evaluation, we constructed three manufacturing tasks: Kitting, Assembly, and Repair. For Kitting, *top*, a toy (cylinder) and a battery pack (cube) were moved to each container. In Assembly, *middle*, screws (grey cylinders) are placed in each of the four corners of a PCB and rotated, while two cables (pink cubes) are placed in the center. Finally Repair, *bottom*, features two faulty components (red cubes) being removed and replaced with new parts (green cubes).

Results of both methods (*Authr* Allocation versus *MO-DAE*) for the three different *Plan* types are shown in Figure 4. This evaluation showed that while the two methods were roughly comparable for optimizing in *Plan* time, cost, and overall score, the compute time for these similar metrics was less for our implementation. Since a focus of *Authr* is to enable the exploration of *Plans*, especially through iterative refinement, we chose to utilize the simpler implementation outlined above for further testing. However, we note that while this method was sufficient for these purposes, alternative methods may be superior with *Plans* of different size or complexity.

3. Implementing task plans into a collaborative robot.
Authr connects to a server developed for Robot Operating System (ROS) [37], running on Ubuntu. We chose to develop our system in ROS to enable future integration with physical robots. For robot trajectory planning, estimating *Therblig* action time, and simulation, our implementation uses MoveIt [7], specifically using Open Motion Planning Library (OMPL) [44]. Because MoveIt is freely available and configurations are easily made, this choice makes adding additional robots to *Authr* straightforward. While the current implementation allows the user to choose from the Franka Emika Panda or Universal Robots' UR3, UR5, and UR10, any robot with a MoveIt configuration could be added. Utilizing a standard inverse-kinematics and motion-planning tool enables us to achieve our goal of a shared representation by converting our spatial-semantic representations to robot-specific control. Thus, *Therblig* behavior implemented on top the motion planner achieves *Agent*-agnostic functionality.

4. Facilitating the exploration of human-robot task plans.
Authr integrates the above representations and technologies into a visual programming environment. This environment is built using the Angular web framework [15] as a browser-based application, which connects to a ROS-based server using Robot Web Tools [45]. *Authr* has three modes, setup, plan, and simulate , which a user works through in five main steps, (1) setting up the workspace, (2) creating tasks, (3) adding therbligs, (4) parameterizing, and (5) simulating the result (Figure 5). The first step is for a user to set up a workspace, as the *Agents*, *Things*, and *Destinations* created in this step will be needed for the following steps. Next, the user can create a *Task* and a set of *Therbligs* that will be performed in this *Task* by dragging them from a library and dropping them in the *Task* container. The user can then parameterize the *Therbligs* and review any errors identified by *Authr*. Based on these errors and the remainder of the *Plan*, the user can either decide to create another *Task* or continue adding *Therbligs* to an existing *Task*. At this point, the user can also navigate to the simulation view to see the plan played out. After reviewing the simulation, the user can create another *Task* to add to their *Plan* if desired. Below, we detail how users would perform each step.

Workspace Setup—This phase lets users set plan-level parameters and configure *Agents*, *Things*, and *Destinations*. One or two *Agents* (one human and/or one robot) must be defined. Users also specify *Things*, which include cubes, spheres, cylinders, and containers and can be customized with size and color. When *Agents* or *Things* are created, *Destinations* that specify

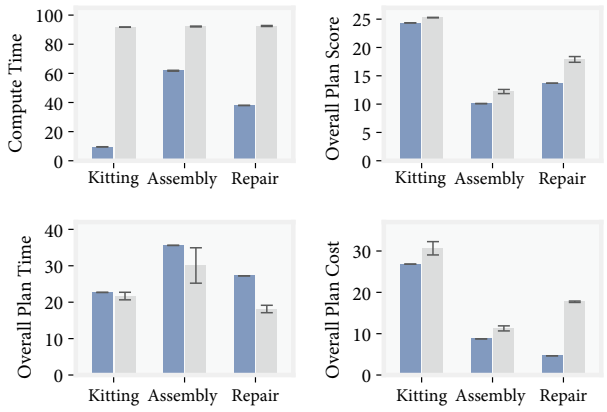

Figure 4. We evaluated the *Authr* Allocation and Parallelization algorithms (blue) versus a *MO-DAE* planner (grey) with three different *Plans* (*Kitting*, *Assembly*, and *Repair*) on 4 different metrics (*Compute Time*, *Overall Plan Score*, *Overall Plan Time*, and *Overall Plan Cost*). Lower scores for all metrics are desirable.

Figure 5. The three modes in *Authr*. In *setup*, users first configure the workspace; *Destinations* are able to be added, deleted, and modified, and each *Agent* and *Thing* gets assigned an initial location in the scene. Moving into planning in the *Plan* Tab, *Tasks* are represented as containers for *Therbligs* and are ordered from left to right. Within each *Task*, *Therbligs* are ordered from top to bottom. *Therbligs* and *Tasks* are also configured. In *simulate*, after designing an interaction, users can simulate the actions of human and robot *Agents*.

their initial locations are automatically created. Additionally, users can specify new, unpaired *Destinations* as waypoints or goals. While configuring *Destinations*, users can inspect the robot action times for all possible *Destination* pairs in a table generated by the motion planner. If a *Destination* is unreachable, the robot time entry is marked invalid, prompting adjustment from the user. Users are able to adjust the placement of *Agents*, *Things*, and *Destinations* within a 3D simulation view or, alternatively, through manual entry.

Creating Tasks—Users develop their collaborative plans through a drag-and-drop mechanism. Users can create any number of *Tasks*, which will be executed from left to right. As users are developing their *Tasks*, they may find that they are repeatedly creating *Tasks* containing the same sequence of *Therbligs*. Users create macros by exporting a *Tasks* as a template of parameterized *Therbligs*. When a user then drops a macro into a *Task* container, the macro expands back into a sequence of those parameterized *Therbligs*.

Adding Therbligs—Users can drag *Therbligs* from the source drawer and drop them into *Task* containers. *Therbligs* can be rearranged within and across *Tasks*.

Parameterizing—While the user is developing the task structure they may open a contextual menu by clicking on an element. If the element is a *Therblig* in the source drawer, then the contextual menu provides an informational description of the *Therblig*. Selecting a macro from the source drawer displays the sequence of parameterized *Therbligs* saved within. Clicking on a *Task* brings up the ability to export it as a macro. Finally, selecting a *Therblig* contained within a *Task* provides access to its parameters.

The *Therblig* contextual menu affords configuration of both high- and low-level parameters. High-level parameters (*Agents*, *Things*, and *Destinations*) are presented as icons with a drop-down list for configuration. All *Therbligs* request an *Agent* parameter and may also request *Thing* and/or *Destination* parameters. Unique to *Agent* parameterization is an option to defer to the allocation algorithm, presented as an *optimize* option in the *Agents* drop-down list. Low-level *Therblig*-specific parameters (*e.g.,* time, cost, effort) are presented when applicable. For time and cost, when a user provides a human as the parameterized *Agent*, the contextual menu simply requests the time parameter. However, when the *Agent* is deferred to allocation, both time and cost for the human *Agent* and cost for the robot *Agent* need to be specified.

The parameter view also provides the user with feedback on any errors associated with that *Therblig*. In addition to identifying missing parameters, the same Z3-based verification algorithm used in the allocation process is executed upon plan updates, and provides helpful error messages, *e.g.,* "*Agent* must not be gripping." As a visual shorthand, the interface also indicates the presence of errors for a *Therblig* with a red notification icon within its tile.

Simulating—Users enter the simulation phase to evaluate their resulting *Plan*. On entry of this phase, *Authr* runs the *Agent* allocation algorithm on the designed *Plan*. With successful allocation, the user may start, pause, stop, and reset the simula-

tion in real-time. Robot simulation is handled through MoveIt, and human simulation is simply linear interpolation between *Destinations*. Also shown in the simulation view is a timeline for each *Agent*'s allocated *Therbligs*. The timeline representation, *a la Interaction Blocks* [39], was chosen due to the inherent temporality that it affords. Clicking on a *Therblig* within the timeline will expand a context menu displaying its duration and cost. While simulating the *Plan*, the *Therblig* being performed in the 3D simulation by an *Agent* is also highlighted within the timeline.

If the user enters simulation with an invalid plan, the view is replaced with a list of errors detected. While the user is simulating the *Plan*, they may find that their *Therblig* sequence is not performing as they intended (*e.g.,* they forgot to indicate a *Transport Loaded* to a way-point *Destination*). The user may then switch to either the setup or planning phase to fix the error or refine their plan.

USER EVALUATIONS

To gauge the ability of our technical solutions to support the creation of task plans for human-robot teams, we carried out two evaluation studies. The first study focused on our solution to the first technical challenge, creating a shared representation, and assessed the extent to which *Authr* provided users with an appropriate vocabulary to model tasks. The second evaluation focused on our solution to the second technical challenge, translating task models into human-robot task plans, and evaluated *Authr*'s ability to effectively allocate task steps to human and robot *Agents*. Both evaluations also measured the general usability of and user experience with *Authr*.

Evaluation 1: Shared Task Representation

The first evaluation aimed to assess the ability of our HTA- and *Therblig*-based framework to support the modeling of manual tasks as well as the general usability of the software. To achieve this goal, we asked engineers and engineering students to implement a simple kitting task using *Authr*. This evaluation used a version of *Authr* without the Simulate Mode and was constrained to manual allocation of *Therbligs*. This version provided a simulation view in setup where users could move the robotic arm for virtual kinesthetic teaching.

Participants

A total of eight participants were recruited from a university campus. All participants (5 males, 3 females) were native English speakers with an average age of 27.63 ($SD = 21.61$). They either held or were pursuing a degree in either industrial engineering or mechanical engineering.

Procedure

After providing informed consent, participants interacted with an early version of *Authr*, which lacked the simulation and optimization components considered in the later evaluation. Participants were shown a short 9-minute video explaining how to use the software and the different types of *Therbligs* they could use. This video walked users through designing a simple pick-and-place task with a single *Thing*. Next, participants watched a video of a human actor, see Figure 6, performing a kitting process with three different types of *Things*, and were then asked to implement that process as a *Plan* in *Authr* that was performed by a robot. While full simulation was not present in this version, participants were able to utilize a simulated robot (Universal Robots' UR5) for defining the locations of *Agents*, *Things*, and *Destinations*. After completing the task, participants received compensation at rate of $12/hour.

Measures

Participants were given as much time as they needed to design their *Plans* and were asked to verbalize their thoughts in a think-aloud procedure [11, 46]. After the task, users completed the System Usability Scale (SUS) [5, 3], USE [26], and a short demographic survey.

Results

Video data was transcribed and coded for emergent themes. These themes are discussed below.

Theme 1: Planning and Strategy—All eight participants created generally similar *Plans*, with a few differences. Only one chose to group all their *Therbligs* into a single *Task*. The remainder chose to group their *Therbligs* into separate *Tasks*, based on the item being moved.

One participant switched from a single *Task* design to a three-*Task* design after setting up the first group of *Therbligs* in a *Task*. At the time, the singular *Task* contained (*Transport Empty*, *Grasp*, *Transport Loaded*, and *Release Load*), as well as an additional *Transport Empty* which returned the robot back to its initial position to prepare for the next sequence:

> So I suppose I could do 3 *Tasks*—that'd probably be pretty easy. *Grasp*, *Transport Loaded*, *Release Load*, go back to neutral position. I suppose that kind of makes "Only Task" [Name of the one *Task*] not make much sense. Let's grab *Transport Empty*, *Grasp*, *Transport Loaded*... So that kinda makes this *Transport Empty* not make any sense to do, because then all I am going to do is say *Transport Empty* again right at the start of this [The next *Task*]. (P05)

In this excerpt, the participant made two adjustments. The first was the aforementioned switch to three *Tasks*, instead of one. In so doing, the participant also realized that the *Transport Empty* they were performing, which returned the robot back to its initial location, was actually unnecessary, as it was immediately followed by the first *Transport Empty* of the next *Task*.

When structuring their *Tasks* in this way, participants also tended to notice parallels between the *Tasks* they created,

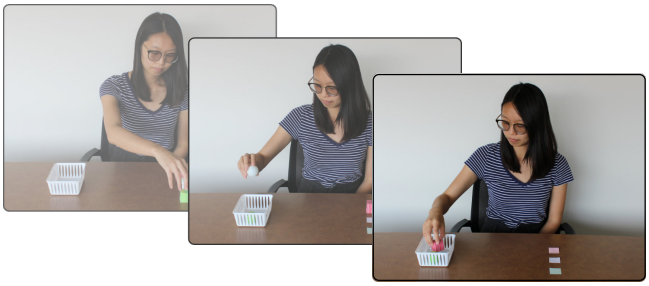

Figure 6. Participants viewed a video of an actor performing a simple kitting task and used *Authr* to translate it to a human-robot task.

prompting many to comment or request some way to either loop through or copy *Tasks*:

> It seems like I am doing the same actions over and over, so it would be nice if I could use the same *Task*. (P02)

Exporting *Tasks* as copy-ready macros had been planned but not implemented by the time of these evaluation sessions. These comments provided justification for adding this in the next evaluation.

Theme 2: Destination Configuration—The most commonly cited difficulty participants mentioned centered not around *Therbligs*, but rather the specification of *Destinations* in the 3D workspace. The challenge was that to move a *Destination* or *Thing* in the workspace, users had to click and drag various toggles around the entities. This challenge could be due to lacking a metaphor that they were familiar with [47]. The controls were not immediately intuitive to users:

> I am going to move... how do I... Oh that's not it. Um oh I see, OK. I didn't know how to move it at first, and now I see that you have to move it like mutually orthogonal in either of the 3 Cartesian directions. (P05)

In order to constrain the space of *Destinations* to the smaller set of valid *Destinations* for a given robot arm, we used a procedure in which users moved a marker around the scene, and the arm attempted to match that pose. Setting the position and orientation would copy the robot pose to the *Destination*, as a direct parallel to kinesthetic guidance [32] which could be used in a physical workspace to specify locations to the robot. However, this approach did not seem intuitive to users in this context, as suggested by the following excerpt:

> But it is hard to know what moves what. I got it eventually. And then the robot it isn't super clear like where the base is and the where the head starts, and then you have to move the robot to set the *Destination*, which... And then like checking how far it can go—that didn't really make sense to me. (P03)

To address this confusion, for the final evaluated version of our tool, each *Destination* was manipulated directly, and an indicator showed when it was reachable by the robot *Agent*. However, for future versions of *Authr*, this capability may be added back in, for when users have a physical robot on the scene and wish to use the physical robot to configure the destinations and object locations more easily, much like interfaces such as Polyscope and RAZER [42].

Theme 3: Simulation—A common comment by participants was that upon completion, many wished to confirm the accuracy of their *Plans* by seeing it in action through simulation:

> OK, it looks like it is all good, but I do not know how to test it. (P06)

Indeed, one such participant made an error in their *Plan* that likely would have been discovered through simulation. In specifying the goal *Destinations* for the *Transport Loaded Therbligs*, they incorrectly used the initial locations of the objects as *Destinations*, instead of the goal location specified. This does not create an error, since a *Transport Loaded* to the same *Destination* is valid, albeit non-useful. The resulting *Plan* would have shown no movement of items in the scene

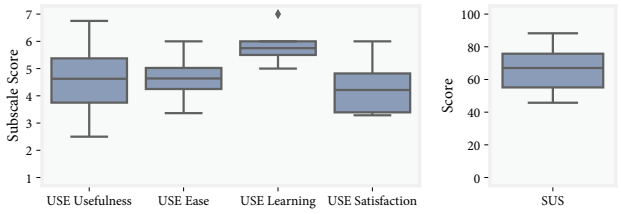

Figure 7. USE and SUS scores from Evaluation 1.

to the goal location. This provided further justification for supporting iterative refinement through the design of *Authr*.

The quantitative data from the measures of usability and user experience can be seen in Figure 7. The subscales of USE had the following scores: Usefulness ($M = 4.56$, $SD = 1.35$), Ease ($M = 4.63$, $SD = 0.856$), Learning ($M = 5.81$, $SD = 0.579$), and Satisfaction ($M = 4.29$, $SD = 0.990$). The average SUS score was 67.3, ($SD = 14.1$).

Evaluation 2: Agent Allocation

For the second evaluation, we turned our focus toward automatic *Agent* allocation. Specifically, we studied the ease to which participants author manual *Agent* allocation *Plans* in comparison to authoring automatic *Agent* allocation *Plans*. To understand how engineers may use *Authr* to perform these tasks, we started participants with a more complex sorting *Plan* where *Agents*, *Things* and *Destinations* are already defined. We then asked them to implement the *Tasks* necessary to complete the *Plan*. Once completed, the experimenter would load in a different sorting *Plan* and repeat the experiment with a different allocation type. The four conditions in the experiment were counter-balanced.

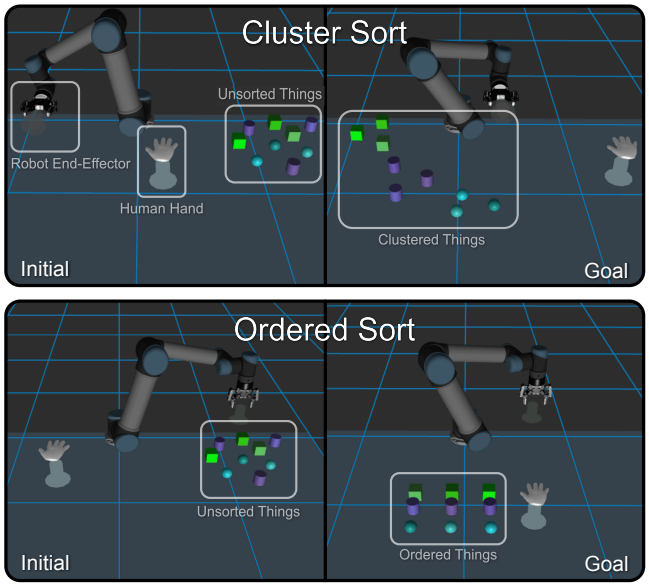

Figure 8. For Evaluation 2, we constructed 2 comparable tasks, Cluster Sort and Ordered Sort. For Cluster Sort, top, participants organized blocks into clusters by type, and in Ordered Sort, bottom, participants organized blocks into a grid.

Participants

Another eight participants (6 males, 2 females), aged 21 on average ($SD = 0.93$), were recruited for Evaluation 2. All participants were native English speakers and either held or were pursuing degrees in industrial or mechanical engineering.

Procedure

After providing informed consent, participants interacted with the next version of *Authr*, which added full simulation, verification, automatic *Agent* allocation, and the adjustments to the interface based on feedback from Evaluation 1. The version of *Authr* described in the *Technical Approach* section was used in this evaluation. A Universal Robots' UR5 was used for simulation and design. As in Evaluation 1, participants were first shown a short eight-minute video explaining how to use the software, and the different types of *Therbligs* they could use. This video walked users through designing a different basic pick-and-place task from Evaluation 1. Next, participants were instructed on how to view the robot action time table and use macros by the experimenter. The experimenter worked through one example, and the macro was deleted after demonstration. Two different sorting tasks, see Figure 8, were provided each with nine *Things* (three cubes, three cylinders, and three spheres) that needed to be moved to their goal state according to the condition. The *Things* had initial positions off to the right side of the simulated workspace. In both cases the initial positions of *Things* were identical. In the Cluster Sort, participants were asked to move the *Things* from the unsorted cluster into three sets of clusters, based on type. Similarly, in the Ordered Sort, participants were asked to move the *Things* into a grid formation.

To start the task, participants were provided a reference document containing the task description with the sorting objective, defined *Plan* workspace, a time estimate table for human actions, and cost tables for both *Agents*. Time and cost tables were the same between *Plans* except for robot timing due to differences in *Destination* positions and orientations. When constructing a *Plan* in the manual allocation condition, the participant was asked to explicitly define the *Agents* without the allocation algorithm. The experimenter also provided the participant with scratch paper and a calculator, and instructed the participant to solve the allocation to the best of their ability. When constructing a *Plan* in the automatic allocation condition, the participants were instructed to only use the automatic allocation. For both allocation conditions, participants were given as much time as they needed to work through the *Plan*. Participants were also allowed to use simulation throughout their design process. Once they felt that their *Plan* was finished they were given a series of questionnaires assessing their experience with the tool, after which they would move onto the second case. Except for the sorting objective and method of *Agent* allocation, the procedure for the second *Plan* was the same as the first. After the participant completed the second *Plan* and the associated questionnaire, they received compensation at rate of $12/hour.

Measures and Analysis

Participants were given as much time as they needed to design their *Plans*. As in Evaluation 1, participants were asked to verbalize their thoughts in a think-aloud procedure. After completing the task, users completed the SUS [5, 3], USE [26], and NASA Task Load Index (TLX) [17], as well as a short demographic survey.

Results

Outcome measures were computed for each produced *Plan*: Plan Time, Plan Cost, and Plan Score. Plan Score reflects the weighted average of Plan Time and Plan Cost that the participants attempted to minimize. Additionally, scores for the SUS, USE subscales, and the NASA TLX subscales were computed. Each outcome measure was analyzed with a repeated measures one-way Analysis of Covariance (ANCOVA), modeling allocation method while controlling for *Plan* type (*Sorting* or *Ordering*). Plan Time, $F(1, 6) = 6.8274$, $p = .0400$; and Plan Score, $F(1, 6) = 6.9791$, $p = .0384$, were found to be significant, where Automatic procedures outperformed Manual in these cases. No other measures were significant. We note that these results are based on a small sample and require further validation and contextualization within qualitative findings.

Video data were transcribed and coded for emergent themes for each *Plan* implemented by the participants. Given the complexity of the *Plans* compared to Evaluation 1, engineers constructed *Plans* that showed greater variation. This included both boundaries for where the *Plans* were divided up into *Tasks*, and when manually allocating *Agents*, how they chose to handle prioritizing duration and cost. Frequently, participants would express distaste for the workload when performing manual allocation. These themes are discussed below.

Theme 1: Workload—When performing allocation *manually*, many participants, after fully understanding the problem space, articulated negative attitude or frustration with the process. One such participant stated that they did not want to perform all the necessary calculations:

> Okay. I don't really want to do all the calculations for figuring out exactly which ones. ... So many different possibilities. (P15)

This apprehension towards the problem resulted in a variety of approximation strategies, detailed below. Others expressed distaste for the work required to compare charts of times. In the following excerpt, a participant who first performed the automated allocation procedure remarks about how it differed from the manual process:

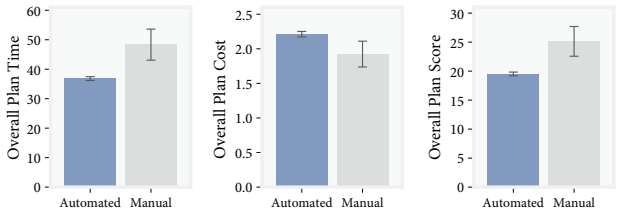

Figure 9. Resulting Overall *Plan* Cost, Time, and Scores for Automatic and Manual procedures. Cost refers to the objective corresponding to effort or wear (depending on *Agent*), and Score refers to the overall score, based on the weighted Time and Cost. Lower scores for all measures are more desirable.

> It's definitely less frustrating calculating the cost than it is trying to figure out the quickest path going through the the chart the first way... Cause like quick mental math, I personally liked that better than trying to stare at the chart... (P10)

When using automated allocation, participants simply had to provide accurate times for the human, and costs for the robot and human. While this certainly amounted to some tedious calculation of effort and data entry, as seen here, participants tended to prefer this to the complex considerations of the manual approach. We will discuss options for reducing this potentially tedious activity in the future directions.

That is not to say that the automatic method was without its difficulties. A small number of participants did initially have some trouble orienting to the more abstract representation of the task. This problem of representing these more abstracted, agent-agnostic actions is therefore in need of further research.

Theme 2: Strategies—Due to the workload required in the manual allocation procedure, most participants developed various heuristics or algorithms to determine the best allocation.

Some participants chose to approximate by looking at trends. In the following excerpt, one participant averaged rows in the robot times, to get a rough expected duration. This was compared to the human times:

> ...and with the sphere, all of the robot numbers look like they're kind of a little bit higher than three for where they're moving it to. Um, if they like start at the sphere and like move to it. So I was gonna make the human do that one. (P10)

Others took a more conditional approach, looking for cases where patterns held or did not hold:

> What's the cost of moving the robot? The cost of moving the robot is always cheaper. Wait... If it's always cheaper... Yeah. Then if time is cheaper than cost should be cheaper for the robot. We should only consider it when... Since we're only working with these four and the robot's cost is lower than these. So anyway... yeah. Only if the human time is shorter then we take cost into consideration. (P16)

Heuristics weren't the only way to approach the problem, however. Other participants opted to iterate on a *Plan*, moving *Agents* from *Task* to *Task*. In some cases, participants would leverage the simulation to assist with this process.

Theme 3: Mistakes and Errors—As seen above, most participants, when manually allocating *Agents*, developed various heuristics to determine the best allocation. Unfortunately, not all these heuristics were valid, or consistently applied.

The excerpt from the discussion of workload also illustrates this line of thinking:

> So I'm just going to put these as the robot because it seemed cheaper... Yeah. I'm just going to give the robot two of them. Sure. We'll say that's good. (P15)

This a common error seen in these heuristics. When participants used less concrete heuristics, they tended to place a higher-than-necessary emphasis on cost than described in the instructions. While cost was almost always lower for the robot (the exception was *Rest*), the cost was usually small compared to duration, while weights were equal.

Yet, even larger errors were sometimes made. In the following case, the participant incorrectly opted to assign all *Therbligs* to the robot, in order to minimize the overall cost, but forgot to consider benefits of concurrency:

> I feel like I'm not going to use the human at all. I feel like the time that it's using—like the 0.66 seconds—will offset the cost of 0.2 versus 0.05 for the transport. That's my... I know... I'm going to go forward with moving only the robot. Cost seems like a better benefit. (P10)

This error resulted in a nearly two-fold increase in total *Plan* score (38.6) when compared to their automatically-allocated *Plan* (20.2). This illustrates the large potential for errors when dealing with such complex allocation tasks in a completely manual manner.

Theme 4: Modified Workflow—In addition to the differences in the tactics of problem solving, participants appeared to display a more streamlined reference $\rightarrow$ calculate $\rightarrow$ configure process in the automated condition and a more fragmented approach in the manual one. Consider the following process for *P16* as they configured a series of therbligs:

> The participant begins by creating a series of empty *Therbligs*. *P16* then considers the scene from the *Setup* tab, decides to work on the second cylinder first based on its location, and then tentatively proceeds by assigning the robot, choosing to alter the plan later if desired. *P16* switches to consult the reference document (on a separate browser tab), but then needs to check where the robot arm would be. Next, the participant needs to check the table of robot times to see the time estimate based on that information and then switch back to the reference to calculate the expected time and cost parameters: *"What are we working with... the cylinder. Four seconds."* Finally getting the information, *P16* switches back to the *Plan* view and enters the information. Throughout the process, the participant performs multiple context switches to reference different information from different sources (some internal to the *Authr* tool and some external), which requires the participant to frequently verbally re-situate after each context switch.

Compare this pattern with the same participant configuring a plan using the automated process as follows:

> The participant creates a series of *Therbligs* in a *Task*, configuring the high-level parameters for *Things* and *Destinations*. *P16* then switches to the reference document in the separate tab. The participant obtains the estimated time and cost parameters and returns to the *Plan* tab to enter the information.

Because the automated process utilizes its own information (*i.e.,* robot time estimates) in the allocation, participants did not need to reference, handle, or utilize this information, reducing the need to reference internal data. While the participant had to perform some additional calculations, as seen in the themes above, this process was preferred over the manual approach. This preference was likely in part because of the more streamlined reference $\rightarrow$ calculate $\rightarrow$ configure process of the automated method.

DISCUSSION

Our evaluation studies provided us with a better understanding of the extent to which the technical solution we have developed enabled users to create human-robot task plans. Our findings from Evaluation 1 indicate that the shared task representation that incorporated ideas from HTA and work modeling served as an appropriate framework to model tasks that users observed from video. The representation not only provided an effective set of building blocks to construct plans, but it also helped users identify *Therbligs* that were unnecessary in achieving the task goal. The evaluation also identified a number of usability issues, specifically in positioning and assigning *Destinations* for *Therbligs*, which informed the improvements we made in *Authr* prior to Evaluation 2. Finally, participants repeatedly expressed a desire to simulate the actions they were modeling, providing evidence that simulation is a critical component of any task-planning approach or environment.

The second evaluation, which focused on assessing the effectiveness of our technical solutions in enabling engineers to allocate task steps to a human-robot team, revealed that automated methods for such allocation is critical. Participants in our study were overwhelmed by the combinatorial complexity when considering the conjunction of task step ordering, cost, time, and *Agent* allocation. To handle this complexity, many participants developed heuristics to simplify the allocation problem, but these heuristics were ineffective or detrimental when they were not applied consistently or correctly. Finally, we found that designers using the automated approach showed a more streamlined workflow for designing and configuring their plans. Overall, our findings indicate that automated methods that handle the complex computational process of task assignment to multiple *Agents* are essential for any task-planning environment.

Limitations and Future Directions

Authr offers a novel approach to the design of collaborative robotic programs, but it does have certain limitations that motivate a number of future research directions.

First, our solution to the shared task representation problem only focused on physical manipulation tasks, although real-world tasks will require additional capabilities such as perception and tool use. The original set of *Therbligs* that informed the development of our solution proposed a set of *cognitive Therbligs* such as *Search* and *Inspect* as well as a set of *cognitive and physical Therbligs* such as *Use* [13]. In our future work, we plan to extend the current set of *Therbligs* with the ability to perform cognitive actions and tool use. The *Use Therblig* in particular is worth exploring as it can potentially handle a large number of tools and be applicable outside of manipulation-type tasks. The challenge with operationalizing *Use* in an agent-agnostic manner is that a semantic description of tool use is generally enough for humans to act but is insufficient for robot instrumentation. That is, the tool's behavior would need to be algorithmically defined. We suggest two potential ways to approach the operationalization of *Use*. In the first approach, *Authr's* high-level parameters could be extended with *Tools*. With well-defined behaviors, the *Use Therblig* may be sufficiently descriptive as it could utilize the capabilities each tool defines. An alternative approach is to decompose *Use* into a larger number of *Therblig-like* actions (*e.g., Use-Screwdriver*), each addressing a different behavior.

We designed *Authr* on the extendable infrastructure called ROS. While our current solution does not directly connect to physical robot devices, a minor configuration change and the inclusion of an additional (usually standard) driver would allow for this capability, since the control is based on inverse kinematics and simply publishes joint instructions. Our evaluations have thus far not tested this implementation, but in future work we plan to test the inclusion of a physical device as a component in the *Authr* task creation and evaluation processes. This plan includes the potential to use the physical robot as an input device for object and goal specification via PbD, as well as a more grounded output for simulation. PbD could also be investigated as a way of simultaneously accumulating data on human activity time and effort, potentially reducing the amount of parameterization required.

Our solution also does not address the complexities of variations in object attributes, positions, and counts that would be encountered in complex, real-world tasks. Some of these problems could be modeled by extending our simulation process to account for variance, resulting in more robust time estimates. Furthermore, future work should incorporate computer vision and sensing capabilities to respond to this variation and the movements by the human collaborator in real-time.

Low-level parameterization of *Therbligs* is also currently limited to manual entry of values except robot action time. Several options to provide sensible default values instead of relying on user entry could be explored. First, *Things* can provide relevant information such as grip-effort based on their intrinsic properties. Second, a human simulation could be used to generate time estimates similar to the current approach for robot *Agents*. Lastly, *Authr* could allow users to specify cost functions algorithmically, taking into consideration time, weight, grip effort, and so on, providing engineers with flexibility to define their own metric without the burden of manual entry.

A shop-floor human-collaborator mode for *Authr* should also be explored. Currently, *Authr* generates a static plan in simulation with the assumption that human *Agents* can follow it precisely. Future work should investigate methods of informing workers on their current goals, tasks, and future robot-collaborators actions. How human collaborator's task performance and cognitive load are affected while performing *Plans* developed in *Authr* alongside a physical robot should also be studied. Achieving these would require *Authr's* representation to handle both human synchronization and errors not addressed with the static programs currently generated. This behavior can potentially be exposed to engineers using either *cognitive Therbligs* or introducing more general programming control flow. *Authr* should also be extended to provide safety awareness such as minimum separation monitoring during execution and considered when allocating *Agents*.

Finally, our user evaluations modeled simple tasks (ordering and sorting) that may not represent the complexities of real-world tasks, *e.g.,* an assembly task. Our technical evaluation of

the *Agent* allocation algorithm used tasks derived from more realistic situations (kitting, assembly, and repair), demonstrating greater representational capability. In the future, we plan to carry out evaluations of *Authr* with engineers from the local industry in their own environments (instead of our laboratory) and ask them to create human-robot plans for the tasks that their organizations perform. We expect such evaluations to provide us with guidance on how to extend *Authr's* capabilities to further support complex real-world tasks.

CONCLUSION

In this paper, we sought to both better understand how to translate manual tasks for human-robot teaming, and create a tool, *Authr*, that facilitates this translation. This tool enables users, such as engineers, at the manufacturing facility to build models of existing tasks which are automatically converted into human-robot collaborative plans, providing end-to-end support for human-robot task teaming. To better understand this process, we have evaluated *Authr* with engineering students across two user studies. Our findings from these evaluations indicated that the shared task representation that we developed served as an appropriate framework for modeling existing tasks, and our automated agent-allocation approach facilitated the translation of these task models into plans that humans and robots can collaboratively perform. Our work serves as a first foray into the development of authoring tools for human-robot teaming within the manufacturing context and provides guidance for creating tools that can facilitate the real-world adoption of collaborative robot systems more generally.

ACKNOWLEDGMENTS

This work has been supported by National Science Foundation awards 1426824, 1651129, and 1925043. We would like to thank Julie Shah, Joseph Kim, Margaret Pearce, Majid Aksari, Brittney Johnson, Alex Padron, Abhay Venkatesh, John Balis, and Steelcase, Inc. for their contributions to the conceptual and software development of *Authr*.

REFERENCES

[1] H. Akrout, D. Anson, G. Bianchini, A. Neveur, C. Trinel, M. Farnsworth, and T. Tomiyama. 2013. Maintenance Task Classification: Towards Automated Robotic Maintenance for Industry. Procedia CIRP 11 (2013), 367 – 372.

[2] 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.

[3] Aaron Bangor, Philip Kortum, and James Miller. 2008. An Empirical Evaluation of the System Usability Scale. Intl. Journal of Human–Computer Interaction 24, 6 (2008), 574–594.

[4] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. 2013. Interpreting and Executing Recipes with a Cooking Robot. In Experimental Robotics. Springer, 481–495.

[5] John Brooke. 1996. SUS-A Quick and Dirty Usability Scale. Usability Evaluation in Industry 189, 194 (1996), 4–7.

[6] Guido Bugmann, Ewan Klein, Stanislao Lauria, and Theocharis Kyriacou. 2004. Corpus-Based Robotics: A Route Instruction Example. In Proceedings of Intelligent Autonomous Systems. 96–103.

[7] Sachin Chitta, Ioan Sucan, and Steve Cousins. 2012. Moveit![ros topics]. IEEE Robotics & Automation Magazine 19, 1 (2012), 18–19.

[8] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 337–340.

[9] Johann Dréo, Pierre Savéant, Marc Schoenauer, and Vincent Vidal. 2011. Divide-and-Evolve: The Marriage of Descartes and Darwin. Proceedings of the 7th International Planning Competition (IPC). Freiburg, Germany 91 (2011), 155.

[10] Shirine El Zaatari, Mohamed Marei, Weidong Li, and Zahid Usman. 2019. Cobot Programming for Collaborative Industrial Tasks: An Overview. Robotics and Autonomous Systems 116 (2019), 162–180.

[11] K. Anders Ericsson and Herbert Simon. 1998. How to Study Thinking in Everyday Life: Contrasting Think-Aloud Protocols with Descriptions and Explanations of Thinking. Mind, Culture, and Activity 5, 3 (1998), 178–186.

[12] C. Ailie Fraser, Tovi Grossman, and George Fitzmaurice. 2017. WeBuild: Automatically Distributing Assembly Tasks Among Collocated Workers to Improve Coordination. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 1817–1830.

[13] Frank Gilbreth and Lilian Gilbreth. 1924. Classifying the Elements of Work. Management and Administration 8, 2 (1924), 151–154.

[14] Matthew Gombolay, Ronald Wilcox, and Julie Shah. 2018. Fast Scheduling of Robot Teams Performing Tasks With Temporospatial Constraints. IEEE Transactions on Robotics 34, 1 (2018), 220–239.

[15] Google. 2019. Angular. (Jan 2019). `https://angular.io/`

[16] Kelleher Guerin, Colin Lea, Chris Paxton, and Gregory Hager. 2015. A Framework for End-User Instruction of a Robot Assistant for Manufacturing. In Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 6167–6174.

[17] Sandra Hart and Lowell Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Advances in Psychology. Vol. 52. Elsevier, 139–183.

[18] Justin Huang. 2017. Enabling Rapid End-to-End Programming of Mobile Manipulators. In Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction. ACM, 343–344.

[19] Jun Kato, Daisuke Sakamoto, Takeo Igarashi, and Masataka Goto. 2014. Sharedo: To-do List Interface for Human-agent Task Sharing. In Proceedings of the Second International Conference on Human-agent Interaction (HAI '14). ACM, New York, NY, USA, 345–351.

[20] Mostepha Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. 2013. Pareto-Based Multiobjective AI Planning. In International Joint Conference on Artificial Intelligence. AAAI.

[21] Seung Han Kim and Jae Wook Jeon. 2007. Programming LEGO Mindstorms NXT with Visual Programming. In 2007 International Conference on Control, Automation and Systems. 2468–2472.

[22] Frank Klassner and Scott Anderson. 2003. LEGO MindStorms: Not Just for K-12 Anymore. IEEE Robotics Automation Magazine 10, 2 (June 2003), 12–18.

[23] Seung kook Jun, Madusudanan Narayanan, Pooja Agarwal, Abeer Eddib, Pragya Singhal, Satyanarayana Garimella, and Venkat Krovi. 2012. Robotic Minimally Invasive Surgical Skill Assessment Based on Automated Video-Analysis Motion Studies. 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob) (2012), 25–31.

[24] Hsien-I Lin and Chia-Hsien Cheng. 2014. Behavior-Based Manipulator Programming Based on Extensible Agent Behavior Specification Language. In 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014). 808–813.

[25] Hsien-I Lin and YP Chiang. 2015. Understanding Human Hand Gestures for Learning Robot Pick-and-Place Tasks. International Journal of Advanced Robotic Systems 12, 5 (2015), 49.

[26] Arnold Lund. 2001. Measuring Usability with the USE Questionnaire. Usability and User Experience Newsletter of the STC Usability SIG 8 (01 2001).

[27] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. ACM Transactions on Computing Education (TOCE) 10, 4 (2010), 16.

[28] Carlos Mateo, Alberto Brunete, Ernesto Gambao, and Miguel Hernando. 2014. Hammer: An Android Based Application for End-User Industrial Robot Programming. In 2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA). IEEE, 1–6.

[29] Joseph Michaelis, Amanda Siebert-Evenstone, David Shaffer, and Bilge Mutlu. 2020. Collaborative or Simply Uncaged? Understanding Human-Cobot Interactions in Automation. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). 1–12.

[30] George Michalos, Sotiris Makris, Jason Spiliotopoulos, Ioannis Misios, Panagiota Tsarouchi, and George Chryssolouris. 2014. ROBO-PARTNER: Seamless Human-Robot Cooperation for Intelligent, Flexible and Safe Operations in the Assembly Factories of the Future. Procedia CIRP 23 (2014), 71–76.

[31] George Michalos, Sotiris Makris, Panagiota Tsarouchi, Toni Guasch, Dimitris Kontovrakis, and George Chryssolouris. 2015. Design Considerations for Safe Human-Robot Collaborative Workplaces. Procedia CIRP 37 (2015), 248–253.

[32] Arne Muxfeldt, Jan-Henrik Kluth, and Daniel Kubus. 2014. Kinesthetic Teaching in Assembly Operations–A User Study. In International Conference on Simulation, Modeling, and Programming for Autonomous Robots. Springer, 533–544.

[33] Hai Nguyen, Matei Ciocarlie, Kaijen Hsiao, and Charles Kemp. 2013. ROS Commander (ROSCO): Behavior Creation for Home Robots. In 2013 IEEE International Conference on Robotics and Automation. IEEE, 467–474.

[34] Chris Paxton, Andrew Hundt, Felix Jonathan, Kelleher Guerin, and Gregory Hager. 2017. CoSTAR: Instructing Collaborative Robots with Behavior Trees and Vision. In Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 564–571.

[35] Chris Paxton, Felix Jonathan, Andrew Hundt, Bilge Mutlu, and Gregory D Hager. 2018. Evaluating Methods for End-User Creation of Robot Task Plans. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 6086–6092.

[36] Margaret Pearce, Bilge Mutlu, Julie Shah, and Robert Radwin. 2018. Optimizing Makespan and Ergonomics in Integrating Collaborative Robots into Manufacturing Processes. IEEE Transactions on Automation Science and Engineering 99 (2018), 1–13.

[37] 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.

[38] Dominik Riedelbauch and Dominik Henrich. 2018. Fast Graphical Task Modelling for Flexible Human-Robot Teaming. In ISR 2018; 50th International Symposium on Robotics. 1–6.

[39] Allison Sauppé and Bilge Mutlu. 2014. Design Patterns for Exploring and Prototyping Human-robot Interactions. In Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 1439–1448.

[40] Alexander Skoglund, Boyko Iliev, Bourhane Kadmiry, and Rainer Palm. 2007. Programming by Demonstration of Pick-and-Place Tasks for Industrial Manipulators using Task Primitives. In 2007 International Symposium on Computational Intelligence in Robotics and Automation. IEEE, 368–373.

[41] Neville Stanton. 2006. Hierarchical Task Analysis: Developments, Applications, and Extensions. Applied Ergonomics 37, 1 (2006), 55–79.

[42] Frank Steinmetz, Verena Nitsch, and Freek Stulp. 2019. Intuitive Task-Level Programming by Demonstration Through Semantic Skill Recognition. IEEE Robotics and Automation Letters 4, 4 (Oct 2019), 3742–3749.

[43] Frank Steinmetz, Annika Wollschläger, and Roman Weitschat. 2018. RAZER-A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization. IEEE Robotics and Automation Letters 3, 3 (July 2018), 1362–1369.

[44] Ioan Şucan, Mark Moll, and Lydia Kavraki. 2012. The Open Motion Planning Library. IEEE Robotics & Automation Magazine 19, 4 (December 2012), 72–82.

[45] Russell Toris, Julius Kammerl, David Lu, Jihoon Lee, Odest Chadwicke Jenkins, Sarah Osentoski, Mitchell Wills, and Sonia Chernova. 2015. Robot Web Tools: Efficient Messaging for Cloud Robotics. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 4530–4537.

[46] Maartin Van Someren, Yvonne Barnard, and Jacobijn Sandberg. 1994. The Think Aloud Method : A Practical Guide to Modelling Cognitive Processes. London: AcademicPress (1994).

[47] Chadwick Wingrave and Joseph LaViola. 2010. Reflecting on the Design and Implementation Issues of Virtual Environments. Presence 19, 2 (April 2010), 179–195.

[48] Chongjie Zhang and Julie Shah. 2016. Co-optimizating Multi-agent Placement with Task Assignment and Scheduling. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16). AAAI Press, 3308–3314.