Parametrization of “Therbligs” for Specifying Manufacturing Tasks for Collaborative Robots

Joseph Kim,[1]* Margaret Pearce,[2]* Majid Aksari,[2] Brittney Johnson,[1]
Alex Padron,[1] Julie A. Shah,[1] and Bilge Mutlu[2]

***Abstract*— Changing needs of the manufacturing industry, including the need for higher flexibility, shorter production cycle, and improved health and wellbeing of human workers, have resulted in the emergence of collaborative robots. This integration requires the translation of currently manual processes into human-robot collaborative task descriptions. Such translation, however, is challenging due to the dissimilarities between human and robot skills and the variability of processes across industries. Our goal is to develop a generalized model that can represent task descriptions in a way that both human and robot workers can interpret and execute. In this paper, we present our modeling approach based on hierarchal task analysis (HTA) and a parametrization of task primitives called *therbligs* in order to represent manufacturing tasks for human-robot teams. We demonstrate the applicability of our approach for modeling real-world processes, its generalizability across five manufacturing tasks, and its use across multiple collaborative robot platforms.**

I. INTRODUCTION

Collaborative robots promise significant benefits in manufacturing, increasing productivity of human labor, allow greater flexibility in production, and improve ergonomics of manual tasks. They are envisioned to assist human workers with fetch-and-delivery, inspection, kitting, grasping, and other common tasks in industrial settings [**?**]. For an effective collaboration, these robotic systems not only need to exhibit safety capabilities in close-proximity interactions [**?**], but they also need to seamlessly integrate into existing manufacturing processes that are currently manual. Thus, there is a growing need for modeling existing and new manufacturing tasks in a manner that is interpretable to and executable by both humans and robots. To address this need, we propose a new modeling approach that offers a unifying language in describing manufacturing tasks for collaborative robots.

Our approach builds on a language developed to describe tasks for humans, called *therbligs* [**?**], which includes a set of fundamental motions required to perform manual operations in a workspace. To make this language equally expressive for robots, we augment the original definition of therbligs with parameters required for robot execution. We then utilize these “robot therbligs” as task primitives in a formal task-analysis framework to fully represent a work process (Section **??**).

[1]Joseph Kim and Julie A. Shah are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {`joseph_kim`, `julie_a_shah`}`@csail.mit.edu`. Brittney Johnson and Alex Padron are with the Department of Electrical Engineering and Computer Science {`bjohns`, `alex_p`}`@mit.edu`

[2]Margaret Pearce, Majid Aksari, and Bilge Mutlu are with the Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706, USA {`mlpearce`, `majid`, `bilge`}`@cs.wisc.edu`

*J. Kim and M. Pearce contributed equally to this work.

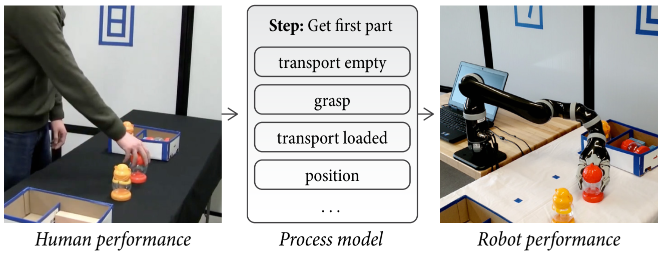


Fig. 1. An illustration of the proposed framework for modeling currently manual processes for robot performance. Left: the human performs a simulated “kitting” task. Middle: the second step of the task model is expressed in the proposed parameterized *therblig* primitives. Right: A robot is performing the task based on the defined therbligs.

We illustrate our modeling approach on five manufacturing tasks (Section **??**) and demonstrate its intuitive use with an interactive authoring environment (Section **??**). We envision our modeling approach playing a central role in integrating collaborative robots into currently manuals tasks by converging on an unifying grammar to express tasks for human-robot collaboration. Our representation has the potential to be incorporated on computational models for optimal planning, task scheduling, and task allocation.

II. BACKGROUND

A. Task Primitives

In robotics, task primitives are commonly used to break down high-level tasks into a set of sub-tasks that can be directly executed. Task primitives allow programming of robots at a *task-level* specification, rather than programming in the motor command space of the robot [**?**]. Much of the research involving task primitives have involved work in symbolic planning [**?**], imitation learning [**?**], and classification of human demonstrations [**?**]. The need for developing a “standardized” set of task primitives, however, has been largely overlooked.

It is important to highlight the differences of task primitives to motor primitives [**?**], [**?**]. Task primitives focus on the identification of atomic work elements in order to accomplish a goal. Meanwhile, motor primitives focus on finding low-dimensional subspaces of high degree-of-freedom robots for the purpose of making imitation and reinforcement learning algorithms tractable.

Traditionally, task primitives have been defined in an *ad hoc* manner, utilizing pre-existing commands for a targeted application or for a specific robot platform [?]. For example, a primitive set for a cooking robot [?] may consist of {`pour, mix, bake`}, while a route-navigation robot may use a set consisting of {`go until, take road, park`} [?]. Although prior research includes several examples of building primitives automatically [?], [?], the learned primitives are often represented in the format of spatio-temporal trajectories, which may not always be easily interpretable to humans. In this paper, we propose the use of primitives inspired from work in human factors and operation management. This approach involves leveraging existing language used to describe work elements for humans and then augmenting the language with parameters required for robot execution.

B. *Task Modeling*

While choosing a set of task primitives is an important part of the modeling problem, these primitives must be contextualized in a higher-level representation of the task structure and flow in order to represent the task more completely. Killich et al. [?] provides a comprehensive overview of various task models and classifies them using three categories: *state-oriented models*, *event-oriented models*, and *Petri net-based models*. State-oriented models, such as Statecharts [?], focus on capturing the *structural* information of tasks by using a complex set of system states. Event-oriented models, such as Activity Diagrams [?], focus more on the *behavioral* aspects of tasks and describe the overall flow of control using conditionals and logic. Lastly, Petri nets [?] focus on modeling of concurrency and synchronization of tasks and processes, especially for distributed and multi-agent systems. Many of the modeling techniques are now part of the larger Unified Modeling Language (UML) [?], which provides a rich methodology for general-purpose task modeling.

Our work adopts a popular modeling approach called Hierarchical Task Analysis (HTA) [?], which is closely related to Activity Diagrams in UML. HTAs involve an event-oriented hierarchical decomposition of operations into tasks and sub-tasks, specifying goals to be achieved at every layer. Due to its transparency and self-descriptiveness, HTAs are an effective tool for communication between task analysts and workers involved in the task [?]. The use of human-interpretable task primitives and HTA promises to make robot programming intuitive not only for system designers but also for non-experts on factory floors.

III. OUR MODELING FRAMEWORK

A. *Hierarchical Task Analysis*

HTA is a widely used scientific methodology to model human tasks in ergonomics and human factors [?]. One advantage of HTA is its ability to describe activities with 'abstract descriptions.' This level of abstraction becomes useful in human-robot collaboration modeling, especially when the actual optimal sequence of activities is yet to be defined. HTA commences by stating the overall goal that needs to be achieved, and then is redescribed into a set of sub-operations and the plan specifying when they are carried out.

Identification of opportunities for collaborative robots first requires analysis of work at the appropriate level. Tasks must be described at a level with sufficient granularity such that they can be translated to a robot interpretable language. We follow the main principles of HTA analysis as described by Stanton [?]: (1) At the highest level, a task consists of an operation that is defined in terms of its goal. (2) Each task is then decomposed to sub-operations in a hierarchy.

As Stanton notes, the sub-goal hierarchy could be decomposed indefinitely and there is no definitive termination criteria. In fact, determining when to end the analysis is one of the known difficulties of HTA. For collaborative robots, we terminate the decomposition when each sub-operation is defined by a core vocabulary executable by humans and robots.

B. *Therbligs*

For the shared vocabulary, we use *therbligs* [?], which are a set of elemental motions required for tasks in industrial applications. Therbligs have been widely used in the field of motion economy, human factors design, and ergonomics, and have been used by system engineers to optimize work processes [?]. The list of therbligs and their original descriptions are shown in Table **??**. The use of therbligs solves the question of when to terminate the process of decomposing HTAs. The lowest level in the hierarchy for each HTA operation should consist of a single therblig.

Consider the example of kitting, which is a common manufacturing task in which individually separate items are packaged and supplied together as one unit. A sub-operation of "retrieve a part" in kitting would be decomposed further using following the sequence of therbligs.

1) **Transport empty** to (part)
2) **Grasp** (part)
3) **Transport loaded** (part) to kit
4) **Position** (part) at specified orientation
5) **Release load** onto kit

Table **??** shows the full HTA of an example kitting scenario where two parts need to be retrieved. The task is decomposed to sub-goals of retrieving a smaller part, retrieving a larger part, and preparing for the next empty kit. In this scenario, the smaller part needs to be "selected" (plan 1.1) from a box of identical objects (e.g., grabbing a screw from box of screws). The larger part is stationed by itself but needs to be "positioned" (plan 1.4) at a specific orientation within the kit. Each sub-goal is decomposed by a sequence of therbligs. The motion level breakdown of sub-goals may appear verbose, but it provides two important benefits. First, it ensures consistency in the level of granularity used in the lowest level of the HTA. Second, therbligs are low-level enough such that they can be naturally extended as instructions for a robotic worker.

We classify therbligs according to whether they involve physical manipulation, cognitive decision-making, or both. The classification can facilitate task allocation in human-robot teams, using information on each worker's strengths

TABLE I

THERBLIGS WITH THEIR DEFINITIONS AND PARAMETRIZATIONS

Therblig	Parameters
Physical	
Transport Empty	pos. (x,y,z), orient. (φ,θ,ψ), ang. (φ',θ',ψ'), arm (*int*)
Reaching for an object with an empty hand (precursor for Grasp)	
Grasp	grasping position, grasp effort, arm
Grasping an object with the active hand	
Transport Loaded	goal position, orientation, angle, arm
Moving an object (grasped) using a hand motion	
Release Load	arm
Releasing control of an object (letting go)	
Hold	duration, grasp effort
Hold a grasp (while other hand performs a use or an assemble function)	
Position	position/orientation, arm
Positioning and/or orienting an object in location (may be part of Transport Loaded)	
Preposition	position/orientation, arm
Positioning and/or orienting an object for the next operation	
Rest	position/orientation, duration
Resting to overcome fatigue, consisting of a pause in the motions	
Cognitive	
Search	object, region boundaries (x,y,z), search routine*
Attempting to find an object using the eyes and hands	
Find	(No parameters)
A momentary mental reaction at the end of the Search cycle	
Select	object, select criteria*, decision constraints*
Choosing among several objects in a group (identifying)	
Inspect	object, quality conditions*
Determining the quality of an object using the eyes and/or other senses	
Plan	initial/goal states, planning constraints*
Determining the course of next action	
Cognitive & Physical	
Assemble	objects, goal criteria*, assembly constraints*
Joining multiple components	
Disassemble	objects, goal criteria*, disassembly constraints*
Separating multiple components	
Use	tool, tool methods*
Manipulating a tool in the intended way during the course working	

TABLE II

HTA TABLE OF KITTING TASK

Task components	Robot parameters
Kitting Task	
Plan 0: Do 1 then 2 then 3	
1. Retrieve small part (sp) to kit	
2. Retrieve large part (lp) to kit	
3. Prepare for next empty kit	
Retrieve small part to kit	
Plan 1: Do once 1-5 then exit	
1. **Select** (small part)	object (small part), select criteria*, decision constraints*
2. **Transport empty** (sp)	object pos. (x,y,z), orient. (φ,θ,ψ), angle (φ',θ',ψ'), arm (*int*)
3. **Grasp** (small part)	grasping position (x',y',z'), grasp effort, arm (*int*)
4. **Transport loaded** (sp) to kit	goal position (x,y,z), object orient. (φ,θ,ψ), angle, arm (*int*)
5. **Release load** onto kit	arm (*int*)
Retrieve larger part to kit	
Plan 2: Do once 1-5 then exit	
1. **Transport empty** (lp)	object pos. (x,y,z), orient. (φ,θ,ψ), angle (φ',θ',ψ'), arm (*int*)
2. **Grasp** (large part)	grasping position (x',y',z'), grasp effort, arm (int)
3. **Transport loaded** (lp) to kit	goal pos. (x,y,z), obj. orient. (φ,θ,ψ), ang. (φ',θ',ψ'), arm (*int*)
4. **Position** (large part)	object position (x,y,z), orientation (φ,θ,ψ), arm (*int*)
5. **Release load** onto kit	arm (*int*)
Prepare for next empty kit	
Plan 3: Do once 1-2 then exit	
1. **Preposition**	position (x,y,z), orientation (φ,θ,ψ), arm (*int*)
2. **Rest**	duration

and weaknesses. One potential strategy would be to assign repetitive and physically-intensive actions such as "transport empty" and "transport loaded" to the robot, while cognitive tasks such as "inspect" and "plan" are performed by the human worker.

C. *Parametrization of Therbligs*

Therbligs help define primitives in HTA, but they lack necessary information for robot execution. For example, the therblig "transport loaded (part) to kit" (plan 1.4 in Table **??**) does not indicate where precisely on the kit the part should be placed. Humans are naturally proficient at inferring correct locations and robust to changes in work configurations. In communication with a robot, however, each primitive needs to be augmented with parameters for execution. These parameters are listed for each therblig in Table **??**.

It is important to highlight that the parametrization is *task-relevant* and not *robot-centered*. Parameters are defined at a layer of abstraction above the space of robot-specific controllers (i.e, they are independent of robot's motion planner, forward and inverse kinematic solvers, perception capabilities, etc). The goal of parametrization of therbligs is on *task-driven* control of robots. Below we highlight parametrization for each class of therbligs.

- Physical therbligs such as "transport empty" and "transport loaded" require information about the desired initial and goal states of the system (i.e., the *position* and the *orientation* of the object and its corresponding goal configuration). These parameters are generally specified in the global coordinate frame of the working environment. *Angle* represents the orientation (using the local coordinate frame) of the robot's end effector. *Arm* is an indicator variable for choosing a specific end effector for multi-arm robots. "Grasp" requires specification of the object's appropriate grasping position and a parameter for how much force is applied (*grasp effort*).
- Cognitive therbligs involve parameters specifying the criteria and the constraints (noted with *) of decision making algorithms. For example, "search" requires the operator to specify a finite search region and a scanning routine (e.g., spiral). "Select", "inspect", and "plan" require enumeration of constraints for perception and planning algorithms. Operators' natural descriptions of problem constraints can be translated into computational language for implementation (e.g., perception [**?**], planning [**?**]).
- Cognitive & Physical therbligs involve a complex sequence of physical manipulation and high-level decision making consist of object assembly, disassembly, and tool use. Underlying planners would require specification of goal conditions, assembly constraints (e.g., need to assemble legs first before tabletop), and tool methods using representations like affordances [**?**].

Overall, the goal of parametrization is to be specific enough such that a task can be accurately completed without the need to resolve ambiguity while being flexible such that it can be applied to any collaborative robot.

Overall...

IV. IMPLEMENTATION

It is important that collaborative robots are adaptive to new tasks and easy to instruct. Thus an important module is the human-machine interface. Users need to be able to describe a series of tasks and their component steps in a way that is intuitive to them and also communicates the necessary information to the robot. We designed the interface shown in Figure 1, which allows a user to author high-level tasks and therbligs with robot parameters. The design focus was on graphical and easy-to-use representation such that users with minimal training can quickly model tasks.

A. Authoring Environment

The authoring environment enables users to rapidly prototype task plans and execute them on robots. The authoring environment is composed of three sections: the *timeline*, the *therbligs library*, and the *edit panel*. In the *timeline*, users can create high-level blocks corresponding to sub-goals of their HTAs. Users can then add therbligs by dragging and dropping one of the therblig templates into one of the high-level blocks. The created therbligs can then be rearranged within a high-level block, moved to a different block, or deleted. The *edit panel* is used to set/modify parameters for individual therbligs. Once the user is satisfied with the timeline, they can execute the sequence on the robot.

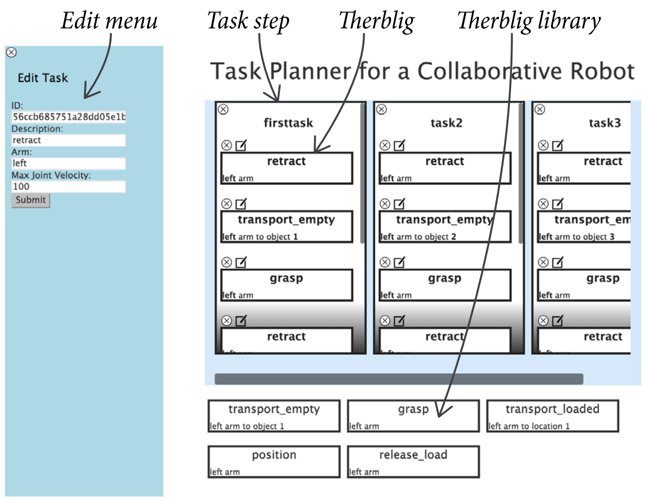
During execution, hierarchy of the HTA is "flattened-out" to a linear sequence of therbligs. Limitations to our timeline-based representation include the lack of conditional branches and loops. Such functionalities will be incorporated in the subsequent versions of the interface. We would like to incorporate a separate panel for simulation of the robot execution, such that users can visually inspect the process before commanding the actual robot. We would also like the interface to provide estimated performance metrics, such as execution time or ergonomics score, that users can use for iterative task refinement.

B. Implementation on Robot Platforms

In order to maximize platform compatibility and dissemination to the robotics and manufacturing communities, we

Two robots have been implemented, which John and Alex know more about.

The different therbligs of a manufacturing task is communicated to the robot using the Graphical User Interface (GUI). As an example, here we explain how we can translate the "Position" therblig to a language that robot can understand. The "Position" therbelig is defined as positioning and/or orienting an object in the defined location. Each therblig has a unique "messageTypeCode" that the GUI sends along with the parameters to the robot interface. In the GUI we have four parameters for the position therblig: *orientation*, *angle*, *arm*, and *max_joint_vel*. Orientation and angle are integers



ID:
56ccb685751a28dd05e1b
Description:
retract
Arm:
left
Max Joint Velocity:
100

Fig. 2. A screenshot of our authoring environment.

that represent different hand positions. Orientation can be 0 or 1 which correspond to the gripper being flat to the ground, or vertical (rotated 90 degrees either way). Angle can be 0,1,2 where 0 represents gripper pointed at the ground, 1 represents flat, and 2 means pointed up. The arm parameter specifies whether we use the left(0) or right(1) hand. max_joint_vel is an integer specifying velocity and is used if the robot has the option of changing the velocity. The robot interface expects the same parameters that the GUI sends along with the messageTypeCode. It will raise an exception if the passed parameters are invalid or it can not successfully complete the action.

V. CASE STUDIES

To gauge feasibility of our modeling framework, we implemented five processes based on common work activities. We selected processes that are generalizable to a wide variety of industries. We have already presented one of the processes (kitting) along with its full HTA table. We briefly describe the other four in the interest of space. Detailed descriptions of all the processes are available online [**?**].

Each case study is a simplification based on real-world processes that we observed on factory site visits or discussed during interviews with system engineers and factory workers. Processes are intentionally simplified for the purposes of this paper such that the robot can complete all of the steps without needing a human partner. We have also reduced the processes to their main functions to try to show as many therbligs as possible without being repetitive. For example, a kitting process may involve retrieving dozens parts and placing them in one area. We limit the number of parts involved such that sufficient variety of therbligs are displayed without redundancies.

A. Processes

Assembly process involves two parts: a larger one and a smaller one. After the larger part is positioned into place, a worker retrieves the smaller part and attaches it to the larger part.

Stocking process mimics stocking shelves in a supply area. We pick two different parts from inside of a box and arrange them on a shelf.

Quality control processes verify that a part or an object meets predefined criteria. In this process, we place a part on a scale and determine if it is within an acceptable weight range. Based on whether or not the check passes or fails, the part is placed in an acceptable or defective bin.

Delivery consists of bringing a set of parts to a secondary work station requiring these parts. Delivery can be done by just using the arms if the external location is within reach, or physically moving to a new location otherwise.

Kitting is introduced in Section III and described in detail in Table **??**.

B. Case Study Discussion

The resulting HTAs from the case studies illustrate the flexibility of therbligs and serve as examples of characterizing everyday tasks through our motion library. Through these examples, we observe that physical therbligs such as "grasp" or "position" are used more frequently than cognitive therbligs such as "inspect" or "plan". In general, cognitive therbligs are more broadly defined and are accompanied by physical therbligs, whereas physical therbligs are specific. Additionally, there are inherent trends and constraints on the ordering of therbligs. Intuitively, "release load" should not appear before "grasp": it doesn't make sense to let go of an object before grasping the object, so this is an implicit constraint. Other patterns in the HTAs show trends in therblig sequences. For instance, "grasp" almost always comes after "transport empty" because this sequence represents reaching for an object.

VI. EVALUATION

A. Performance Evaluation

Study Design: First, we ran a sample task on a robotic platform from the GUI multiple times. We measured the success or failure rate of the task as well as the average cycle time after 20 repeated runs of the kitting task. This gave us a sense of how predictable and reliable the GUI/ motion library combination is compared to using native ROS. Based on the type of failure experienced, we could infer whether the source was the GUI/ motion library or if it was a limitation of the robot itself.

Findings: Each of these runs required positioning 4 lanterns in a box. Out of $20 \times 4 = 80$ positionings by the robot, 1 was a failure, which made 1 out of 20 kitting runs unsuccessful. There were no errors caused by the GUI. The average time to specify the kitting task through the GUI was 73 ($SD = 19$) seconds, and the average time for the robot to finish the task after receiving the command through the GUI was 110 ($SD = 3$) seconds.

B. Usability Study

Study Design: We evaluated our modeling framework in design sessions with users who reported little to no experience in robotics. The main objective of study was to explore the ease of use and the usability of our modeling framework. In the study, users were asked to reproduce a manufacturing task on a robot after watching a video of human performing the task. The kitting task was chosen, one similar to Table **??**, but now consisting of four parts instead of two. Training consisted of instructional videos on the therbligs, the parameters, and the authoring environment. Training, on average, lasted 10 minutes. Participants were then given a textual description of the kitting task and a corresponding video of human demonstration. Then they proceeded with designing the task on the authoring environment. Participants were given as much as time necessary during the design step. Once the participants felt that their timelines were ready, they executed the sequence on the robot. Following the completion of the task, participants filled out the System Usability Scale (SUS) [**?**] and questionnaires on assessing task workload and intuitiveness of therbligs. In total, five participants took part in the study.

Findings: Four out of the five participants successfully replicated the kitting task on the PR2 robot. One participant made a mistake of inputing wrong orientation values for "Position" therbligs. However the overall sequencing of therbligs were accurate. Participants' timelines were also consistent with one another, i.e., they decomposed sub-goals per object, even when defining them via kits was a possibility. The SUS served as metric for overall usability of therbligs in combination with the authoring environment. The average score was 90 (SD = 13.3). Based on established guidelines for interpreting SUS scores [**?**], a score of 80 or higher places the system in the highest quartile amongst the survey of interfaces evaluated using the SUS.

Discussion: The use of therbligs was reviewed favorably by participants, many noting the ease of understanding and intuitive use (median Likert score of 4 out of 5). Participants correctly identified and modified appropriate parameters when using the therbligs. Participant did note, however, the burden of "over-specifying" therbligs with parameters such as *arm* and *grasp effort*, when they seldom needed to be changed from their default values. This was anticipated due to our proof-of-concept kitting task. More complex manufacturing tasks would require modification of various parameters. Regarding the authoring environment, participants positively commented on the simplicity of the user interface, especially the drag-and-drop feature and ability to edit individual therbligs.

VII. CONCLUSION

There currently is no widely accepted model for describing the 'vocabulary' for robot actions. Since language plays a central role in communication of domains between factory workers and roboticists, it is important that there be a standard by which a common understanding may be developed for the semantics of robot actions. For collaborative robots in manufacturing, we propose the use of *robot therbligs* as a set of action primitives. We show the generalizability of the model across common manufacturing tasks and highlight the interpretability of the model by human teammates. By having a formal set of action primitives along with a 'grammar' which ensures soundness of their sequencing, we envision robot therbligs embedded on hierarchical task analysis become a generalizable language to be used by collaborative robots.

VIII. APPENDIX

A. Acknowledgements

We thank John Gagliardi, Alex Padron, Edward Oakes, Christopher Bodden, and Robert Radwin for their contributions and to the National Science Foundation for supporting this work under award 1426824.

REFERENCES

[1] B. Matthias, S. Kock, H. Jerregard, M. Kallman, I. Lundberg, and R. Mellander, "Safety of collaborative industrial robots: Certification possibilities for a collaborative assembly robot concept," in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, 2011, pp. 1–6.

[2] N. Pedrocchi, F. Vicentini, M. Matteo, and L. M. Tosatti, "Safe human-robot cooperation in an industrial environment," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[3] F. B. Gilbreth and L. M. Gilbreth, "Classifying the elements of work," *Management Classics*, 1977.

[4] R. Alford, U. Kuter, and D. S. Nau, "Translating HTNs to PDDL: A small amount of domain knowledge can go a long way." in *IJCAI*, 2009, pp. 1629–1634.

[5] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[6] E. Drumwright and M. J. Mataric, "Generating and recognizing free-space movements in humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, vol. 2, 2003, pp. 1672–1678.

[7] M. J. Mataric, "Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics," in *Imitation in Animals and Artifacts*, 2000.

[8] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems*, 2009, pp. 849–856.

[9] O. Stasse, N. E. Sian, F. Lamiraux, T. Sakaguchi, A. Kheddar, and Y. Kazuhito, "Architectures and models for humanoid robots in collaborative working environments," in *ISR'08: International Symposium on Robotics*, 2008, pp. 354–359.

[10] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus, "Interpreting and executing recipes with a cooking robot," in *Experimental Robotics*, 2013, pp. 481–495.

[11] G. Bugmann, E. Klein, S. Lauria, and T. Kyriacou, "Corpus-based robotics: A route instruction example," in *Proceedings of Intelligent Autonomous Systems*, 2004, pp. 96–103.

[12] O. C. Jenkins and M. J. Matarić, "Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion," *International Journal of Humanoid Robotics*, vol. 1, no. 02, pp. 237–288, 2004.

[13] I.-W. Jeong, Y.-H. Seo, and H. S. Yang, "Effective humanoid motion generation based on programming-by-demonstration method for entertainment robotics," in *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on*. IEEE, 2010, pp. 289–292.

[14] S. Killich, H. Luczak, C. Schlick, M. Weissenbach, S. Wiedenmaier, and J. Ziegler, "Task modelling for cooperative work," *Behaviour & Information Technology*, vol. 18, no. 5, pp. 325–338, 1999.

[15] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987.

[16] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide*. Addison Wesley, 1998.

[17] W. M. Van der Aalst, "The application of petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

[18] N. A. Stanton, "Hierarchical task analysis: Developments, applications, and extensions," *Applied Ergonomics*, vol. 37, no. 1, pp. 55–79, 2006.

[19] J. G. Everett and A. H. Slocum, "Automation and Robotic Opportunities: Construction Versus Maufacturing," *Journal of Construction Engineering and Management*, vol. 120, no. 2, pp. 443–452, 1994.

[20] J. Wang, K. Markert, and M. Everingham, "Learning models for object recognition from natural language descriptions." in *BMVC*, vol. 1, 2009, p. 2.

[21] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Experimental Robotics*, 2013, pp. 403–415.

[22] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 3060–3065.

[23] "Therblig motion library repository," https://github.com/Wisc-HCI/therblig-motion-library, 2016.

[24] J. Brooke *et al.*, "SUS - a quick and dirty usability scale," *Usability Evaluation in Industry*, vol. 189, no. 194, pp. 4–7, 1996.

[25] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.