Cognitive Robotics
AAAI Technical Report WS-12-06

A Taxonomic Framework for Task Modeling and Knowledge Transfer in Manufacturing Robotics

Jacob Huckaby and Henrik I. Christensen
Cognitive Robotics Lab
Center for Robotics & Intelligent Machines
Georgia Institute of Technology
801 Atlantic Drive NW
Atlanta, GA 30332, USA

Abstract

Robust methods for representing, generalizing, and sharing knowledge across various robotics systems and configurations are important in many domains of robotics research and application. In this paper we present a method for modeling tasks and robot skills to simplify the programming and reuse of knowledge between robots in manufacturing environments. Specifically, we propose an assembly taxonomy designed to represent the decomposition of high-level, complex assembly tasks into simple skills and skill primitives that the robot must use in a specified sequence. By using programming by demonstration to populate the taxonomy, we propose a method to easily interact with and reuse knowledge in various manufacturing robotics systems, making it possible to reduce programming time and overhead. We present both a detailed discussion of this taxonomy, as well as an example of how the taxonomy can be applied to an assembly task.

Introduction

The popularity of robotics today is growing across all industries. Even in manufacturing, where robots have traditionally been commonplace, new research has aimed at investigating how to insert robotics even more into the manufacturing and assembly process. One problem that continues to plague the adoption of robotics at all levels of industry is one simple fact: programming robots is hard. It requires deep technical knowledge, it is time consuming, and programming robots is done by specification with respect to the system requirements. Thus when a manufacturing environment changes, or a robot is replaced, there is a large amount of overhead that goes into making the new system work.

One method proposed to address this problem is programming by demonstration (Friedrich, Kaiser, and Dillmann 1997), (Kaiser and Dillmann 1995), (Dillman et al. 1999), (Rogalla and Ehrenmann 1998). This has been a popular research topic in manufacturing environments almost since the advent of industrial automation (Nilsson and Johansson 1999). For decades researchers in both industry and academia have looked for ways to reduce the overhead that automation entails. To reduce this load, programming by demonstration aims to teach a robot tasks by demonstrating how those tasks are performed. Programming in this way requires less technical knowledge, is more intuitive, and would thus cut the necessary time required to program the robot.

To make this feasible, we need a method to model the tasks necessary to enable the robot to achieve its programming objective. We begin by specifying a taxonomy that defines skill primitives, or robot-specific skills that a robot can perform such as pick-up or detect, and constraints related to those skill primitives. Having this taxonomy be well-defined allows us to unambiguously specify how these task components can be used to build higher-level tasks. Programming by demonstration could then be used to learn generalized skills to populate the taxonomy.

We are interested in finding a generalized representation for objectives or tasks that can be applied outside of the context in which they were learned, such as on different robots or in different applications. Skills learned in one setting or context could then be transferred across multiple systems and configurations for assembly tasks. In this work, we provide a well-defined taxonomy for describing skill primitives and constraints for manufacturing tasks.

It is important to note here that the proposed framework is not in any way limited to the manufacturing domain. The taxonomic decomposition of tasks and objectives for knowledge transfer and reuse is applicable in many areas of robotics, including lab automation, home and service robotics, and medical robotics. One could imagine a taxonomy for ***Prepare Meal*** and ***Mix Chemicals*** alongside the ***Assembly Action*** taxonomy. The manufacturing domain, and subsequently the assembly taxonomy is our demonstrating example for a deeper idea that we are trying to illustrate: a way of representing tasks in different domains, along with all of the domain-specific knowledge and requirements need to perform those tasks, in such a way that knowledge can be reused and shared effectively.

This paper proceeds as follows. We first address previous work done in the area of knowledge reuse in general robotic systems, including work done in programming by demonstration, industrial manufacturing, and knowledge representation in domestic and service robotics, in the Related Works section. The System section then discusses the proposed assembly taxonomy, along with the different components that compose the taxonomy and the justification for their inclu-

sion. A motivating demonstration involving a simulated assembly task is presented and discussed in Experiments, and a brief discussion of the conclusions and future work follows in the Conclusion.

Related Works

Given that programming robots is a difficult task requiring considerable technical knowledge, a great deal of prior work has focused on addressing how to make robot programming more intuitive and more accessible to operators with less technical experience. Much of this work has focused on finding efficient and effective methods for representing system knowledge to accomplish this task.

The long-studied problem of programming by demonstration has attempted to enable robots to generalize knowledge about certain tasks so that this knowledge can be used in solving problems in a different context than that in which they were originally learned. This is accomplished by modeling information about the motion and task, and different methods have been proposed for representing this knowledge. One philosophy has worked on encoding task knowledge as a function of motion. Examples of this type of representation include dynamical systems (Ijspeert et al. 2001) and object-action complexes (Krüger et al. 2011).

Another philosophy has espoused the view that one can effectively represent important system knowledge symbolically, such as using skill trees (Konidaris et al. 2011) and topological task graphs (Abbas and MacDonald 2011). This symbolic approach to knowledge representation assumes that the system has more inherent knowledge (it knows how the relationships between the symbols and physical instantiations behave), while it allows for the modeling of more high-level concepts than motion-based representations. Others that have used this symbolic approach to address the issue of knowledge reuse or knowledge transfer in areas other than manufacturing include (Ekvall, Aarno, and Kragic 2006), (Ekvall and Kragic 2006), (Nicolescu and Mataric 2003)

Similar to the programming by demonstration paradigm, other work has used different knowledge representations to simplify the robot programming problem. The work of Lyons, et. al. (Lyons and Arbib 1989) defined a model for robot computation using port automata. Kosecka, et. al. (Kosecka and Bajcsy 1993) used a discrete event systems framework to model tasks and behaviors for robotics. More recent work includes that of Kress-Gazit, et. al., (Kress-Gazit, Wongpiromsarn, and Topcu 2011), (Finucane, Jing, and Kress-Gazit 2010) using linear temporal logic to model task specifications, and Dantam, which takes a grammar-based approach to represent sensorimotor information (Dantam and Stilman 2011).

Given a specific knowledge representation, a number of other projects have addressed the problem of representing, storing, and transferring knowledge between various robotic systems.

In the SIARAS (Skill-based Inspection and Assembly of Reconfigurable Automation Systems) project (2011), a system is developed to assist in the automatic reconfiguration of automation systems. This is done due to the need for lightweight (low overhead) processes to address current manufacturing demands. System components are designed both to represent skills and parameters, as well as the process flow. This is done using a specific ontology. A skill server is designed to aid a human operator to match process requirements with the representations in the database.

Another project working on robot deployment in manufacturing environments in the ROSETTA (RObot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning) project (2012). This project tries to design industrial robotic systems that are suitable for working around and collaborating with humans in the manufacturing process. One important aspect of their approach is a skill repository.

RoboEarth (Waibel et al. 2011), (Zweigle et al. 2009) is a project aimed at creating a global repository for all knowledge relevant to a robotic systems, including information on environments, object models, action recipes, and semantic information. The architecture is organized in three layers, with the top layer being the global database, which acts as an information server, a second layer containing hardware independent functions such as action recognition and semantic mapping, and a bottom layer consisting of robot-specific implementations. Knowledge representation and processing is handled by the KnowRob system. (Tenorth and Beetz 2009)

System

In this section we will look in detail at a taxonomy for assembly actions. To understand the organization of the proposed taxonomy, we must look at and analyze how a robot interacts with its environment.

Robots interact with the world by working toward some objective, such as constructing an airplane wing. These objectives can be decomposed into a sequence of tasks that must be accomplished to successfully achieve the objective, such as attaching two sheets of metal together using a bolt. Tasks can further be broken into skills, like threading a bolt, and skill primitives, which are robot-specific actions like closing a gripper around an object.

The ***Assembly Action*** taxonomy (figure 1) is composed of different skills and skill primitives than enable a robot to achieve some assembly objective. It is an example of the hierarchical organization that exists in this type of structured task decomposition. The taxonomy is broad enough to be used in a wide range of settings, but expressive enough to describe the requirements for performing these tasks.

One clear advantage of the modeling taxonomy described here is the ability to leverage tools associated with formal modeling languages like SysML (sys 2010), specifically code generation and simulation. We can use the model taxonomy to do automatic code generation, significantly simplifying the task of generating and maintaining the software implementation of the desired task. We can also use the model to do simulation of the task, which greatly reduces the overhead associated with testing these potentially large, complex systems.

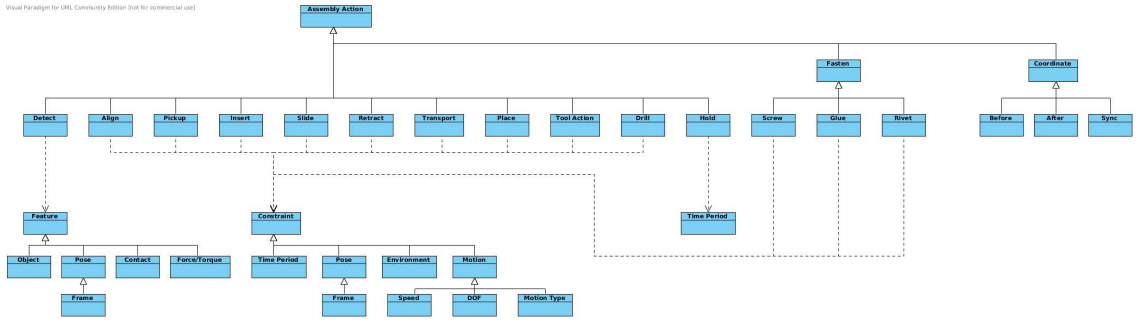

Figure 1: Taxonomy for manufacturing assembly task.

Another benefit of representing skills in this way is that it allows our representation to be independent of both hardware and implementation. The simplest building blocks in our taxonomy are the skill primitives, which are atomic actions that a robot is able to perform. Similarly, the ***Transport*** skill simply allows the task model to specify that a movement action must be performed, without worrying about the details of which algorithm must be used to perform that action. In fact any algorithm or combination of algorithms or approaches could be used to perform that movement action through any space, while still satisfying the requirements of ***Transport***.

In the next sections we will discuss various components of the taxonomy.

Skills

Here we discuss the skill primitives (SP) that have been included in the taxonomy, as well as their function and justification for inclusion. We also continue using the example of programming by demonstration to populate the taxonomy using generalized skills obtained through programming by demonstration techniques. As it is instructive to consider these SP's in light of the various needs required to create them in the taxonomy, we examine a number of these requirements in the acquisition and eventual execution of the generalized skills. Specifically, acquisition requires that the system can recognize occurrences in the environment, such as actions or objects, and a particular parameterization. Execution requires certain perceptual capabilities as well as an appropriate control law. Analyzing the skills in this way allows us to have a better grasp of the diversity in sensory requirements for recognition and parameterization, and the needed control strategies.

Many of the skill primitives in the taxonomy are straight forward and self-descriptive in their function. For example, the ***Align*** SP is a necessary component that allows a robot to be able to align itself with some feature in the environment, such as an object's pose, relative to some constraints. The ***Insert*** SP is needed to perform any type of insertion task, such as peg-in-hole-type tasks, again relative to certain constraints.

To acquire a generalized representation for ***Align***, the robot needs to recognize objects in the environment with which it is to align, while the parameterization could be

Task	Acquisition		Execution	
	Recog.	Params	Percept	Control
Alignment	Dual Obj	Rel. Pose	Rec/Servo	3D Servo/Contact
Pickup	Obj/Action	Obj Desc	Rec/Servo	Servo Approach
Insert	Obj/Action	Pos / Const	Vision/Force	Seq / Impedance
Slide	Action	Start/End	1-2D Force	Impedance Law
Retract	Action	Space	NA	Path Plan
F. Screw	Action	Torque	Force	Insert/Torque
F. Snap	Action	Force/Event	Force	Force Disc

Figure 2: Acquisition and execution requirements for skill primitives.

some relative pose. To execute this skill, the robot would need perception for recognizing the object to align with as well as perception for servoing, and there would need to be a controller capable of visual servoing to the required pose (relative to the object.)

To acquire ***Insert***, the robot would need to recognize both objects and actions related to the ***Insert*** operation, while parameterizing pose and constraints. To replicate the action, the robot has perceptual needs both in vision, and in force at the end effector. The control would require an impedance control law.

Transport, a skill to move the robot from one location to another, requires action recognition, as well as start and goal locations for a parameterization. The execution perceptually could require object recognition (for obstacle avoidance, depending on environment), as well as a path planning and following control law.

Similar to ***Transport***, ***Slide*** also requires action recognition to identify the slide action, and a start and goal location to parameterize the action. Execution of ***Slide*** requires the ability to have perception of forces at the end effector, and an impedance controller for the control.

The ***Screw*** skill primitive can be learned solely through action recognition, and is parameterized by the torque on the end effector. The generalized ***Screw*** action is performed by have force perception at the tool tip, and using a torque-based controller to insert.

Constraints

Many of the SP's in our Assembly taxonomy are parameterized relative to certain physical constraints. These constraints include ***Time Period***, ***Pose***, ***Environment***, and ***Mo-***

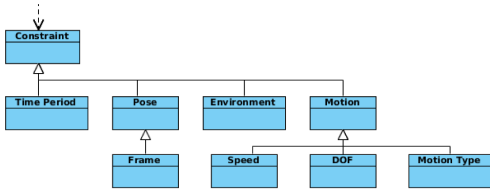

Figure 3: Constraints on skill primitives.

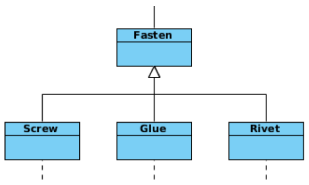

Figure 4: Fasten skill.

tion.

Time Period can be used by ***Hold*** to describe how long something should be held in the specified pose, and can also indicate that some action must be completed in a certain amount of time. Actions such as ***Align*** and ***Transport*** will often be done with respect to some goal or destination pose. Examples of ***Environment*** constraints include workspace limitations and restrictions, the need to follow a particular surface, and constraints in which motion planning through the environment is done in the presence of obstacles. ***Motion*** constraints relate to the actual motion of the robot, and include ***Speed***, ***DOF***, and ***Motion Type***. ***Speed*** and ***Motion Type*** could be used when a robot needs to follow a trajectory with a specific velocity, such as in a welding task.

Flexibility

The claim has been made that the proposed taxonomy is flexible enough to accommodate a wide variety of variations in the task model. This is evident in ***Fasten***. The ***Fasten*** skill is divided into several different skill primitives; ***Screw***, ***Glue***, and ***Rivet***.

This was done to be able to more accurately model tasks in manufacturing environments. While a single ***Fasten*** SP could be sufficient to represent task decompositions in some contexts, it is not sufficient for a wider range of tasks in the broad context of manufacturing. In a similar manner, while the ***Drill*** SP could in some contexts be a complete description of robot capabilities, in other situations it may not be expressive enough. In these situations the taxonomy is able to exercise dynamic abstraction, creating separate Drill SP's that capture the nuances required under the now more general Drill skill. This ability to be able to adjust the level of abstraction for any skill to suit needed modeling requirements in different situations is one strength of this representation.

Coordination

One essential capability to be able to be flexible and robust enough to represent all types of manufacturing tasks is the ability to encapsulate the coordination of multiple systems operating in concert to achieve the manufacturing objective. Ideally the representation would be able to handle robots of differing configurations and specifications (e.g. both robots with identical specifications tasked with different roles, such as identical robots on opposite sides of a conveyor line, as well as robots with different specifications entirely, such as serial manipulators working with parallel platforms or mobile platforms.)

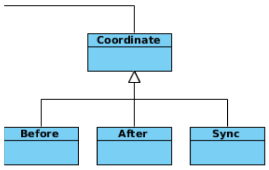

Figure 5: Coordinate skill

This is one of the strengths of the proposed taxonomy, in that it is general enough to be able to cope with these different platforms and systems as they perform various tasks to achieve the overall objective. The ***Coordinate*** skill allows for this collaboration to be specified. It specifically allows for discrete tasks performed by individual robots to be put in the proper sequence for successful completion of the task, using the ***Before***, ***After***, and ***Sync*** SP's. Actions that can be performed in parallel do not need to be explicitly coded, and can be synchronized in terms of the tasks and need to be completed in sequence.

Tool Use

Another important aspect to consider with robots working in manufacturing environments is making sure that the representation can accommodate the ability to use tools in the accomplishment of the desired task. Tools are required to complete most complex manufacturing tasks. This important requirement is handled in our taxonomic framework in two ways. First, the constraint parameterization allows for specifications to be given in different frames of reference, indicated by the ***Frame*** constraint. The reference frame associated with a particular tool could then be given whenever the tool is in use by the robot.

An essential function needed by the robot in order to use a tool is the ability to actuate (or otherwise activate and use) the tool. For this reason the ***Tool Action*** SP exists in the taxonomy. This enables the system to functionally operate the tool, such as activating a drill, or actuating the mechanism for dispensing glue on a gluing tool.

Mobility

For the proposed framework to capture the capabilities of various robotic systems, the concept of mobility must also be addressed. In the context of the assembly task, there are several different types of robots that may use this skill, from simple mobile platforms to full mobile manipulation systems.

The taxonomy that has been discussed in the previous sections can be used to describe mobility, by utilizing the

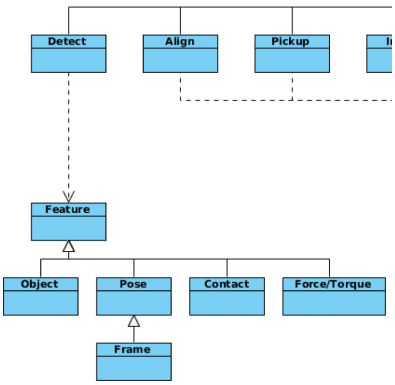

Figure 6: Detect skill.

Transport and ***Hold*** SP's. By using a different parameterization, ***Transport*** can be used to move a mobile platform from one configuration to another, just as it can move the end effector (or some object in the end effector) of a manipulator to a desired location. Similarly ***Hold*** can be used in the same context as a manipulator, instructing the robot (in this case a mobile platform) to hold a specific configuration for a certain amount of time.

Using the ability of this system to capture mobile systems along with the Coordinate skills, one could imagine a task description that includes not only manipulators but mobile platforms as well, to perform various tasks and improve efficiency in the system.

Perception

While not explicitly a part of the proposed Assembly taxonomy, an indispensable part of the effective use of the taxonomy is the interaction with the perception subsystem. Many different types of perception are required to execute an assembly task, and the taxonomy is able to facilitate this interaction.

The ***Detect*** SP is essentially the interface module into the perception actions. Constraints or parameters into this skill primitive are defined as features to be detected, and include ***Object***, ***Pose***, ***Contact***, and ***Force/Torque***.

These feature types handle the various types of perception that is required in assembly. This would cover sensors and algorithms that have to do with pose estimation and object detection, such as cameras and 3D scanners, and other methods of finding and localizing desired objects for manipulation. Other sensors of interest that are useful in this context include contact sensors and force/torque sensors for detecting contact within the environment and performing complex, accurate manipulation tasks.

It is important to note that the taxonomy is designed to be independent of both hardware and algorithmic implementation. While the various parameters of ***Detect*** provide a window into or method of interacting with the perception system of the robot, this does not depend on the type of sensor used or the algorithm or method used to extract meaning from the sensor.

Figure 7: Airplane constructed using toy kit.

Control

Equally as important to the performance of the assembly task as the interaction with the perception subsystem is the ability of the taxonomy to interact with the control subsystem. With the exception of ***Detect*** and ***Coordinate***, all SP's interact with the control subsystem by calling on some controller to perform some action. ***Insert*** will call on a controller to change the position of the end effector, ***Slide*** will control on the contact interactions at the tool tip to slide the workpiece along some surface. There are many different types of controllers that are needed by the system to do assembly.

Here again it should be mentioned, however, that the taxonomy is independent of implementation. Just as ***Transport*** does not depend on the path planning algorithm used to plan the path to be taken, neither does it depend on the type of controller utilized to drive that path.

The control subsystem and the perception subsystem are tightly integrated. In most cases, the controllers called into action by the SP's will in turn call perception functions to provide feedback to close the loop. For example, ***Align*** will depend on the perception subsystem for the robot's pose to identify when it has achieved its goal. The perception system will often provide terminal conditions for the controller, which will in turn signal the system that the task is over and a new task can be attempted.

Demonstration

In this section we will discuss a demonstration prepared to show how the taxonomy can be used to model an assembly task. The example task we are using to illustrate this point is the simulated assembly of a toy airplane using a Baufix toy kit (see Fig. 7).

The toys parts used to assemble the plane are simple, brightly colored wooden construction parts such as screws, washers, blocks, nuts and boards. The model airplane was built using 31 individual parts.

Several UML-based sequence diagrams have been prepared to demonstrate a method of using the taxonomy to model the toy airplane assembly. The sequence diagram is intended to show the organization of an assembly objective into a sequence of actions that are to be performed, and utilizes the skills described in the taxonomy as discrete steps in the sequence.

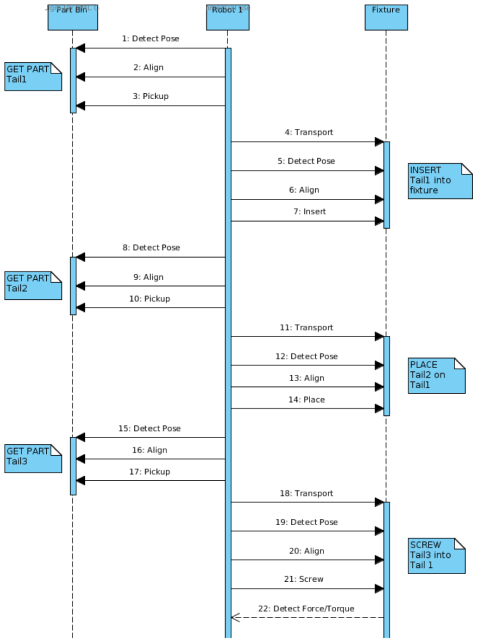

Figure 8: Sequence diagram for a single manipulator.

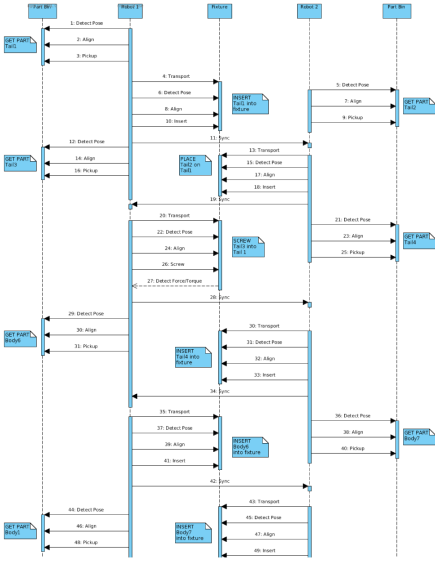

Figure 9: Sequence diagram for two manipulators.

The first diagram (figure 8) shows the assembly of the toy airplane using a single manipulator and a static fixture designed specifically to aid the robot in this airplane assembly task. This sequence diagram is composed of three main lines; the Robot, the Fixture, and a Part Bin. Each message from the robot to either the fixture or part bin represent an instantiation of a skill primitive (message 1 is a ***Detect***, message 2 is ***Align***, etc.) It is assumed in this simulation that all desired parts in the part bin are visible and accessible. Execution of the assembly sequence is performed from top to bottom (as indicated by the numbered messages.) Small groups of messages comprise stages in the assembly process. For example, messages 1-3 can be thought to represent the goal of getting part *Tail1* from the part bin, while messages 4-7 place *Tail1* into the fixture. While only a portion of the sequence diagram is shown (the first 17 actions), it took 258 actions, or instantiations of skill primitives, to complete the assembly of the toy airplane.

The second sequence diagram (figure 9) shows the same toy airplane assembly task, this time using two manipulators. The main differences between the single robot and dual robot diagrams are that in the dual arm example, a second robot main line as been added, along with another part bin main line. The second part bin main line serves two purposes: first, it makes the diagram easier to read by reducing the clutter across the main lines; second, it allows for the possibility of either having both robots use the same part bin or having two separate part bins. The ***Coordinate*** skill is used to synchronize collaboration between the two robots. The availability of a second manipulator allows for a greater variability in how the assembly task is performed due to the added dexterity, as well as a different design for the fixture.

The last diagram shown (figure 10) shows another modification on the single manipulator sequence diagram. This time the diagram has been expanded to include main lines for the perception and control subsystems, to show the different interactions between both of those systems, as well as those systems and the robot. The added interactions make the sequence diagram significantly larger. Note also the constant flow of feedback through the system, which more accurately shows how information and control flows through the system. This diagram shows the tight integration between control and perception, as the control depends on perception to provide feedback for the control loop. The types of perception and control required are passed as parameters to the relevant lines (that ***Detect*** should expect a ***Pose*** feature in message 1, for example.)

Conclusion

In this paper we have presented a taxonomy for assembly tasks in the domain of manufacturing and industrial robotics. We have proposed this taxonomy as a robust and flexible method for modeling assembly task descriptions that are generalizable across multiple hardware platforms in various configurations. This can be a relatively simple and efficient method for simplifying robot programming and reusing knowledge across these robotic systems, easing the transition to new systems and system configurations, as well as reducing time and financial overhead.

Current and future work with this taxonomy includes a hardware implementation of the demonstration discussed in the Experiments section, along with experiments demonstrating the effectiveness of how this taxonomy representation can be used to transfer learned knowledge from one robot to another. One assembly experiment will be run using a KUKA KR5-sixx R650, a standard 6-DOF manipulator. Further experiments will also be performed on other platforms to test the robustness of the generalized skills and skill primitives.

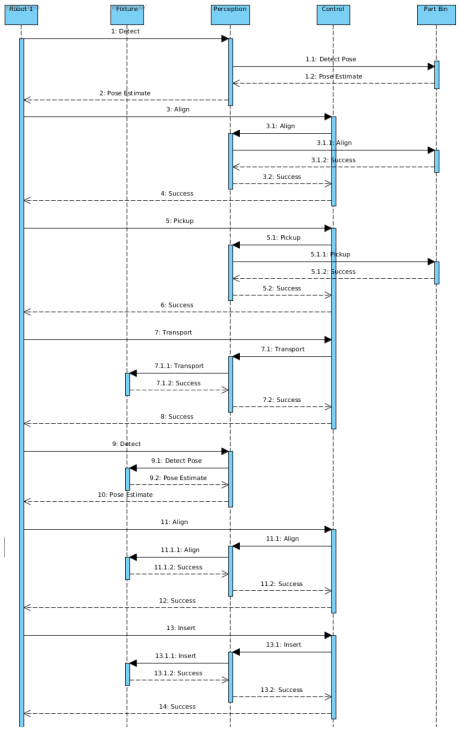

Figure 10: Expanded sequence diagram for single manipulator to include perception and control.

Acknowledgments

The authors gratefully acknowledge the support of Boeing for funding this research.

References

Abbas, T., and MacDonald, B. 2011. Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 3816 –3821.

Dantam, N., and Stilman, M. 2011. The motion grammar: Linguistic perception, planning, and control. In *Robotics: Science and Systems*.

Dillman, R.; Rogalla, O.; Ehrenman, M.; Zoellner, R.; and Bordegoni, M. 1999. Learning robot behavior and skills based on human demonstration and advice: The machine learning paradigm. In *Proceedings of the 9th International Symposium of Robotics Research ISRR99*, 181–190.

Ekvall, S.; Aarno, D.; and Kragic, D. 2006. Task learning using graphical programming and human demonstrations. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, 398 –403.

Ekvall, S., and Kragic, D. 2006. Learning task models from multiple human demonstrations. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, 358 –363.

Finucane, C.; Jing, G.; and Kress-Gazit, H. 2010. Ltlmop: Experimenting with language, temporal logic and robot control. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 1988 –1993.

Friedrich, H.; Kaiser, M.; and Dillmann, R. 1997. What can robots learn from humans? In Gertler, J., ed., *Annual Reviews in Control*, volume 20. Elsevier. chapter Robotics, 167–172.

Ijspeert, A.; Nakanishi, J.; Shibata, T.; and Schaal, S. 2001. Nonlinear dynamical systems for imitation with humanoid robots. In *Proceedings of the IEEE International Conference on Humanoid Robots*. 219–226.

Kaiser, M., and Dillmann, R. 1995. Robot programming based on single demonstration and user intentions. In *3rd European Workshop on Learning Robots*.

Konidaris, G.; Kuindersma, S.; Grupen, R.; and Barto, A. 2011. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*.

Kosecka, J., and Bajcsy, R. 1993. Discrete event systems for autonomous mobile agents. *Robotics and Autonomous Systems* 12:187–198.

Kress-Gazit, H.; Wongpiromsarn, T.; and Topcu, U. 2011. Correct, reactive, high-level robot control. *Robotics Automation Magazine, IEEE* 18(3):65 –74.

Krüger, N.; Geib, C.; Piater, J.; Petrick, R.; Steedman, M.; Wörgötter, F.; Ude, A.; Asfour, T.; Kraft, D.; Omrčen, D.; Agostini, A.; and Dillmann, R. 2011. Object-Action Complexes: Grounded Abstractions of Sensory-motor Processes. *Robotics and Autonomous Systems* 59(10):740–757.

Lyons, D., and Arbib, M. 1989. A formal model of computation for sensory-based robotics. *Robotics and Automation, IEEE Transactions on* 5(3):280 –293.

Nicolescu, M. N., and Mataric, M. J. 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 241–248.

Nilsson, K., and Johansson, R. 1999. Integrated architecture for industrial robot programming and control. *Robotics and Autonomous Systems* 29:205–226.

Rogalla, O., and Ehrenmann, M. 1998. A sensor fusion approach for PbD. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

2012. The ROSETTA project page. http://fp7rosetta.org. Accessed 10 March 2012.

2011. Final Report - SIARAS (Skill-based Inspection and Assembly for Reconfigurable Automation Systems). http://cordis.europa.eu/search/index.cfm?fuseaction=lib.document&DOC_LANG_ID=EN&DOC_ID=121979181&q=.

2010. OMG Systems Modeling Language (OMG SysML) Version 1.2 Specification.

Tenorth, M., and Beetz, M. 2009. KnowRob — Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 4261–4266.

Waibel, M.; Beetz, M.; D'Andrea, R.; Janssen, R.; Tenorth, M.; Civera, J.; Elfring, J.; Gálvez-López, D.; Häussermann, K.; Montiel, J.; Perzylo, A.; Schiešle, B.; Zweigle, O.; and van de Molengraft, R. 2011. RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine* 18(2). Accepted for publication.

Zweigle, O.; van de Molengraft, R.; d'Andrea, R.; and Häussermann, K. 2009. Roboearth: connecting robots worldwide. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, 184–191. New York, NY, USA: ACM.