Does your Robot have Skills?

Bøgh, Simon; Nielsen, Oluf Skov; Pedersen, Mikkel Rath; Krüger, Volker; Madsen, Ole

Published in:
Proceedings of the 43rd International Symposium on Robotics

Publication date:
2012

Document Version
Accepted author manuscript, peer reviewed version

Does your Robot have Skills?

Simon Bøgh, Oluf Skov Nielsen, Mikkel Rath Pedersen, Volker Krüger and Ole Madsen

Department of Mechanical and Manufacturing Engineering,
Aalborg University
Aalborg, Denmark
e-mail: {sb, olnie, mrp, vok, om}@m-tech.aau.dk

***Abstract*— This article presents a unifying terminology for task-level programming of highly flexible mobile manipulators in industrial environments. With a terminology of tasks and object-centered skills, industrial applications are analyzed in the logistic and assistive domains. The analysis shows that many tasks can actually be solved with a small library of predefined building blocks, called skills. In logistics domains, it can be exploited that the sequences of skills needed to solve specific tasks follow certain patterns, whereas in assistive tasks, the sequence vary a lot, and tasks must be taught to the robot by a user, based on Standard Operating Procedures or process knowledge. An overview is presented on how to implement a skill-based architecture, enabling reuse of skills for different industrial applications, programmed by shop-floor workers. The terminology of *tasks, skills*, and *motion primitives* is introduced and designed to separate responsibilities of robot system developers, robot system integrators and shop floor users concerning programming of the robot system. The concept of testable pre- and postconditions assigned to the skills are proposed to help both in developing skills and asserting when a skill is applicable.**

Keywords: industrial robots, mobile manipulation, flexible automation, robot skills, task-level programming.

I. INTRODUCTION

The paradigm-shift from mass production to mass customization and the request to dynamically scale the throughput on production lines up and down have changed the demands for automation systems, as solutions are needed that are more flexible than today's production lines with conveyor belts and stationary industrial robots. A solution to this is to develop robots that are more flexible in the way that they can easily be moved between different workstations and solve a larger variety of tasks. In addition, many tasks in production of small or medium volumes involve transportation of parts between workstations and machines. This call for new types of robots that

- are able to move around autonomously in industrial environments,
- can be programmed on the fly, and
- can cope with uncertainties due to not completely known environments.

A concept for targeting the mobility issue is autonomous industrial mobile manipulation (AIMM), where a manipulator is mounted on a mobile platform.[13] For this type of flexible robots, classical robot-centric programming paradigms do not suffice, as robot programs are typically developed from scratch dedicated for a specific task and environment. In order to speed up programming of flexible robots, and even let shop floor workers do it, another paradigm than traditional robot-centric programming needs to be adopted - a task-level programming paradigm.

In task-level programming, a sequence of actions, performed on objects, specify a production-related goal. The sequence of actions leading to the fulfillment of this goal can either be planned or specified by an operator.

A. State of the art

Many approaches to task-level robot programming exist already in different fields of robotic research. Task-level programming specifies what the robot should do in terms of actions on the objects involved in the task and not necessarily to the full extent how it should be achieved (for instance in terms of manipulator trajectories). This is similar to the symbolic representation of tasks (goals) of one of the earliest developments in autonomous robot, Shakey, which uses STRIPS rules and the STRIPS planner to reason about which actions may lead to accomplishing a goal [10]. The symbolic representation of STRIPS is however not very tightly connected with the physical world. This topic is dealt with in the formulation of Object-Action Complexes (OACs) [12], a framework for cognitive robots to identify possible actions based on the objects it perceives and infer the objects based on what actions can be performed on them. The tight connection between actions and objects and the formulation of OACs is a basis for how robust robot skills can be defined. Although, in industrial scenarios as addressed here, objects and possible actions are expected to be provided to the robot by a user and not learned by the robot itself.

A different approach for robots to reuse knowledge on how to solve tasks is to share recipes (programs) over networks like The Semantic Web or for industrial applications in the Knowledge Integration Framework (KIF) [5]. In KIF for example an abstract representation of how an emergency-stop button is assembled using guarded motions is stored, such that it can be used on robots even of different type if the same basic capabilities are available. There are however some features that this approach lacks; a formal description of which conditions the program will work under, the preconditions, and a description of the effects of executing the program, the postconditions.

A solution to this could be to associate the program with a skill description template, as in SKORP [2] - an approach to skill-based robot programming using a

graphical representation of skills, where the skill template serves as a communication tool between the skill developer and the application programmer. However, these pre- and postconditions need to be connected to the real robot, i.e. to be testable before they actually justify their existence.

B. Contribution

In this study different application scenarios are analyzed to provide a terminology that can assist in all levels of task-level robot programming, from the developer of the basic capabilities of the robot to the shop floor worker who will program new tasks. Using this terminology,

- a concept of robot skills is utilized to analyze production processes, and identify what skills are needed for certain task domains, and
- the skills and tasks abstraction is proposed to encapsulate, convey, and reuse expert knowledge to create robust robot programs.

II. Definition of Tasks and Skills

In this section we will introduce our definitions of the key terms of this paper. Similar to papers that model human action using an abstraction hierarchy of action primitives, actions and activities [6,15], or that model language out of phonemes, words and sentences [17], we will denote the same type of hierarchy by using the terms *motion primitives, skills* and *tasks*. Motion primitives perform basic motion commands of the robot, and skills are basic functionalities the robot has.

A. Tasks

Tasks are in this context, quite intuitively, characterized by attaining a certain production-related goal in the factory hall, e.g. *fill feeder A or load part B into machine C*. We say that a task can be decomposed into a sequence of skills from the set *S* of skills if there exists a robot that is able to complete a task by using the skills from *S*. Tasks are defined based on measurable state variables, and the robot uses its skills to change these state variables. State variables can be either measured with vanishing uncertainty by dedicated sensors, e.g. by those that are built into the manufacturing systems, or by sensors on the robot, such as vision, torque or tactile sensors.

B. Skills

Skills are the foundation of task-level programming, and provide the building blocks for the robot to complete the task. Which skills are available to a robot depends on its hardware and its sensors. In Sec. IV-A we discuss a method for defining skills that are somewhat independent of the hardware they run on, without loss of generality.

How to automatically select the *right* set of skills to accomplish a task, however, is an open question.[16] In the present paper we suggest finding the skills by analyzing real-world implementations and Standard Operating Procedures (SOP) from an industrial partner. This way, the identification of the set of skills is consistent with human intuition.

One core property and main justification for using skills is their object-centeredness. Classic robot programs are usually based on 3D coordinates, e.g. a `pick up` function requires the object to be at an a-priori defined 3D location. Skills, on the other hand, are not applied on 3D locations but on objects, i.e. `pick up <object>`. In order to instantiate e.g. the `pick up` skill on `object`, the robot will use a sensing device such as a video camera or a range scanner to first detect and then localize the object. Once the 3D location is available, the robot is in principle able to execute the classic function for picking up the object.

A second core property of a skill is that each one needs pre- and postconditions to ensure and verify a correct functioning: Before the robot can execute a skill, all preconditions need to be fulfilled, e.g. reachability of the object is a precondition of the `pick up <object>` skill. If the object is not reachable, the skill cannot be executed. A check of the postconditions will verify if the expected outcome of the skill was satisfactorily met, i.e. that the executed skill was successful. Thus, the pre- and postconditions are effectively a query on the world state, that evaluates to true or false.

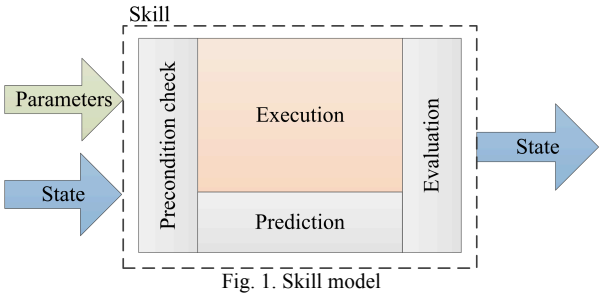

Fig. 1. Skill model

Robot skills have two very distinct features; *execution* and *inspection*, each requiring a different form of object interaction. Thus, a robot skill is expected to modify the state of the real world and concurrently update the systems state variables. A model of a robot skill is shown in Fig. 1. Queries on the state variables and input parameters (which are provided at task-level programming time) serves as a means of testing if the preconditions of the skill execution are met, either by prior knowledge or ad hoc inspection. If the preconditions are met, the skill is executed, based on the parameters and the state variables. Parameters are thus stored in the task description and are for instance objects or locations, e.g. `<red box>` for the `locate` or `pick up` skill or `<warehouse>` for the `move to` skill.

The postconditions are two-part in relation to the skill; prediction and evaluation. The prediction specifies formally what the expected effect of executing the skill is, and can thus be used to select an appropriate skill for achieving a desired goal state. The evaluation checks that the state variables after execution is within an expected range and updates the state variables to reflect the actual state after the skill execution.

Since skills are goal-oriented, the prediction of a skill must devise a change in the state variables. This change

can either occur by letting the robot interact with objects or by letting the robot inspect the scene to gain further information on the current state.

According to the vision, we want to set up industrial robot tasks on the task level. To do this we need to identify what skills the robot needs in the possible application domains of flexible mobile manipulators. By using the definition of skills presented in this section, we now look for these patterns in real-world industrial scenarios.

III. Skill-based Analysis of Industrial Applications

Two approaches are pursued in order to identify skills for industrial applications. Firstly, the skills are found through analysis of industrial tasks, industrial implementations and laboratory experiments - Fig. 2 top - and secondly, skills are identified through analysis of *Standard Operating Procedures* (SOP), which are descriptions of how tasks are manually performed by the operators, hence human-action to robot skill mapping - Fig. 2 bottom.

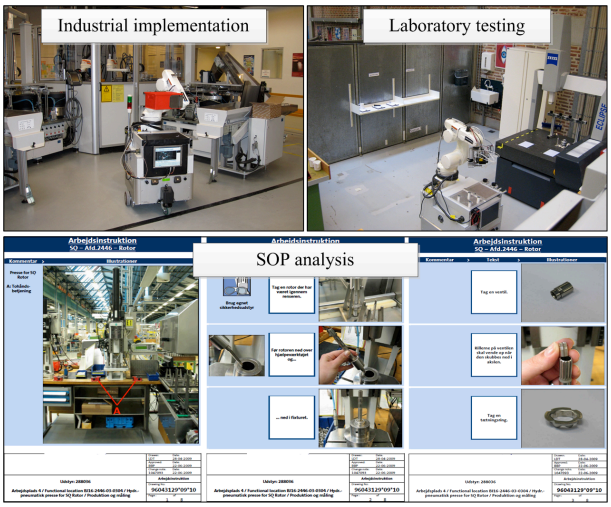

Fig. 2. Background for analysis - industrial implementation, laboratory testing and analysis of Standard Operating Procedures (SOP)

One important aspect of the analysis is that it is implicitly based on the natural language and communication between the people who work in the production, so the identified skills are identified based on what one finds intuitive for the given task held against the definition of skill.

A. Analysis

In Bøgh et al. [3] more than 566 industrial tasks are analyzed to identify the application categories potentially suitable for mobile manipulators, shown in Fig. 3. In the present paper, three classes of logistic tasks (transportation, multiple part feeding and single part feeding) and two classes of assistive tasks (machine tending and assembly) are investigated to identify which skills are needed in these particular categories.

In logistics tasks, the robot needs to cover larger distances so the mobility of the mobile manipulator is essential. For assistive tasks there can also be a need for mobility, but typically in a more limited production area. Assistive tasks are generally more value-adding tasks compared to logistic tasks.

Logistic	Assistive	Service
Transportation	Machine Tending	Maintenance, Repair and Overhaul
Multiple Part Feeding	Assembly	Cleaning
Single Part Feeding	Inspection	
	Process Execution	

Fig. 3. General application categories for AIMMs. Green - focus for first approach: analysis of industrial implementation and laboratory test. Blue - focus for second approach: analysis of Standard Operating Procedures

1) Analysis of Logistic Tasks: In Fig. 4 the three types of logistic tasks are illustrated. The basic skill sequences for transportation, multiple part feeding and single part feeding are presented in the following.

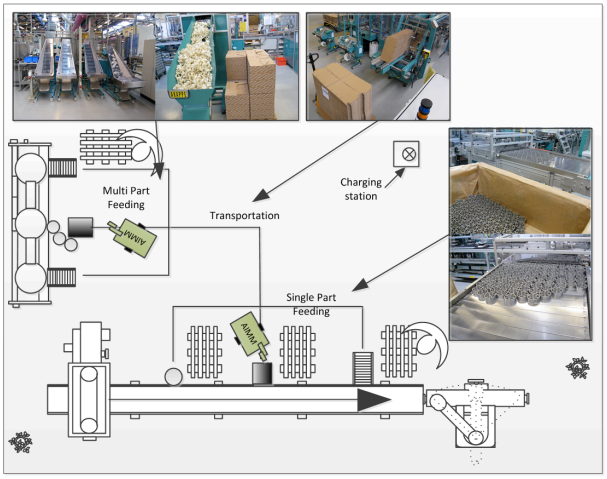

Fig. 4. Logistic tasks: Transportation, multiple part feeding, and single part feeding[4]

Transportation is the process of moving parts and work pieces between workstations and storages. Transportation tasks involve physical separation of locations larger than the workspace of the robot manipulator. In basic transportation tasks, there is not necessarily any direct contact or communication with production machines nor is there any advanced manipulation involved, i.e. only a few degrees of freedom is required (e.g. a forklift, as opposed to a robotic arm).

Multiple part feeding is the process of loading several components at a time into part feeders or machines. The two basic multiple part feeding setups utilize either kanban boxes that are unloaded into feeders, or shoveling of parts, where parts can be in bulk on pallets placed at a workstation, and the mobile manipulator will shovel the parts into a feeder right next to the pallet. In Fig. 5 the general skill sequence for a multiple part feeding task is illustrated. Much like in transportation tasks, this involves moving to a storage to pick up kanban boxes or small load carriers (SLCs) typically placed on a shelf system, then moving to a designated workstation, and unloading the content of the box into a part feeding machine. The difference is that here the task is initiated by a signal from a machine, and parts are emptied into the machine.

Single part feeding is the process of loading components, one at a time, into feeders and machines. The difference compared to multiple part feeding, is that the robot is in direct contact/interaction with objects. Thus, the mobile manipulator will pick up one part at a time, e.g. from a pallet, and place the part into a machine or onto a conveyor belt. The robot might have to move between the picking and placing locations.

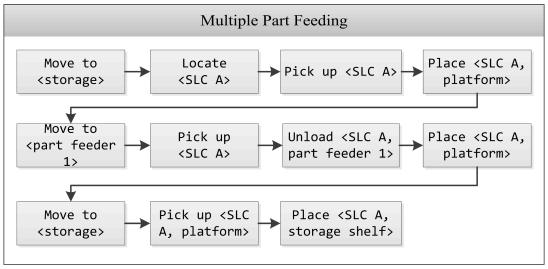

Fig. 5. Skill sequence for multiple part feeding tasks

It is easy to see that the principal sequences of skills are almost similar for this domain, even though the order and length of the sequence can vary. Although the sequences for logistic tasks follow a specific pattern, there can be significant gains in planning the tasks, as the robot is typically able to carry several parts or containers at a time. This kind of planning is essentially solving travelling salesman problems for which order to visit what location.

2) Analysis of Assistive Tasks: For machine tending and assembly tasks, SOPs (shown in Fig. 6) are analyzed to identify skills. Each step of the instructions is mapped to its respective robot skills. The specific details of the SOPs will not be shown due to confidentiality reasons from the industrial partner.

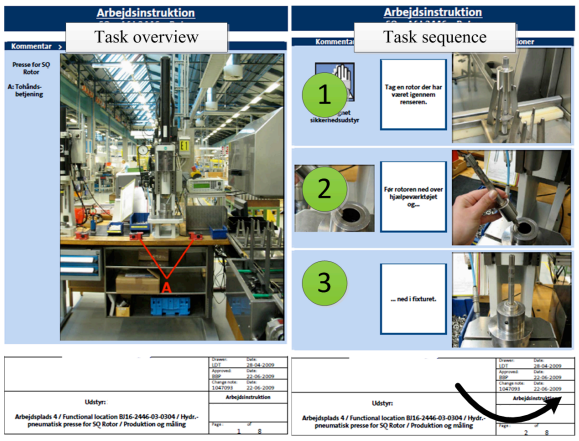

Fig. 6. Example Standard Operating Procedure for operators

Machine Tending is the process of loading/unloading materials into machinery for processing but also includes aspects like opening/closing doors, pressing buttons and turning knobs. Assembly is the process of fitting components together, e.g. into larger or completed parts. Pre-assembly tasks are typically carried out before assemblies as semi-finished goods.

An example of instructions in a SOP for assembly could be; *Pick up the rotor from the fixture* or *Place the valve into the rotor shaft*. This will respectively result in a *pick up* skill and a *place* skill. In a machine tending task, an example could be: *Open machine door* or *Push Start-button*, which would result in an *open* skill and *push* skill.

B. Results - Skill Portfolio

An overview of the identified skills is presented in Table I, where a formal description is provided for each skill. It can be seen that this short set of skills is sufficient for solving a large set of industrial tasks, especially for logistic applications, where the *check, align, open, close, press, release* and *turn* skills are not necessary. A condition for this to be possible, is however the ability to make the skills generic such that a large set of objects can be handled with the same skill.

TABLE I
SKILL PORTFOLIO

Skill	LT[a]	AT[b]	Description
Move to	✔		To go from one location (station) to another in the factory
Locate	✔	✔	To determine or specify the position of an object by searching or examining
Pick up	✔	✔	To take hold of and lift up
Place	✔	✔	To arrange something in a certain spot or position
Unload	✔		Unload a container: to remove, discharge or empty the contents from a container
Shovel	✔		To take up and move objects with a shovel
Check		✔	Quality control: the act or process of testing, verifying or examining
Align		✔	To make an object come into precise adjustment or correct relative position to another object
Open		✔	To move (as a door) from a closed position and make available for entry, passage or accessible
Close		✔	To move (as a door) from an open position
Press		✔	To press against with force in order to drive or impel
Release		✔	To let go or set free from restraint e.g. release a button
Turn		✔	To turn a knob or handle

[a] Logistic Tasks
[b] Assistive Tasks

IV. IMPLEMENTATION RECOMMENDATIONS

This section presents the considerations relevant for implementing task-level programming on industrial robots. A key consideration for these recommendations is how to integrate the different types of actors involved in creating robotic systems in an implementation architecture for task level robot programming. In this work, that is approached by separating the implementation into layers.

A. Implementation layers

Fig. 7 shows how a three-layered architecture for flexible robots can look like. The architecture is similar to what is the presented by Gat [11] and Björkelund et al. [5]. Imposing constraints on the implementation, as using layered architectures and setting up a skill model

with a defined structure, is a step in a different direction than component based architectures like ROS and OROCOS [8], which focus on providing great freedom to the individual component designers. That fits well with the demands of researchers, creating their own components, but for other actors to be able to reuse the components the lack of structure and well defined pre- and postconditions makes it difficult to determine when a certain component is applicable.[7] Component-based middleware may however be a feasible way to implement a layered architecture due to its flexibility, scalability and focus on reuse of software. Dividing the control into components and layers may also be a way to enable integration of the architecture on different robots if the same set of capabilities can be provided.

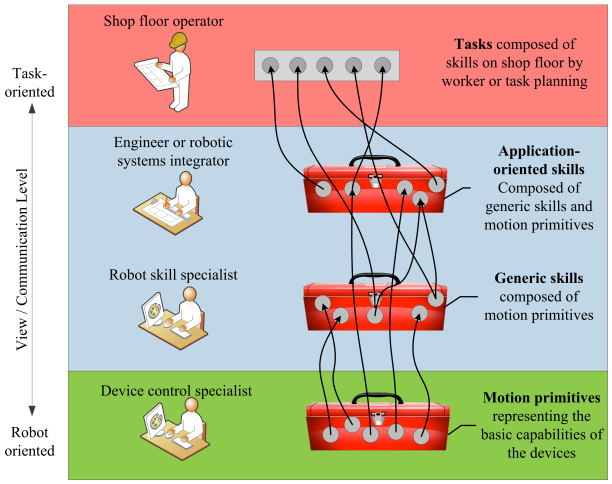

Fig. 7. The implementation layers of the skill terminology. Green: *motion primitive* layer. Blue: *skill* layer. Red: *task* layer

1) Motion primitives: The three-layered architectures have a motion primitive layer at the bottom level responsible for implementing the real-time control loops of the robot. This layer is similar to the level of programming that typically is carried out on industrial robots, but with the addition that movements can be interrupted and altered in real-time, based on feedback from sensors. For assistive task domains, hybrid force/position control modes or the Impedance Controller [1] are also considered as part of this layer. The capabilities are typically developed by device specialists at the robot manufacturer, but if the real-time layer is exposed to other developers, additional sensor-based robot control modes can be added, like for instance visual servoing.[9]

2) Skills: The layer deserving the most attention is the skill layer, as it is the layer that is exposed to the end user when the operator is programming new tasks on the robot. It is also important in the sense that it is responsible for providing reuse of robotic sub-programs at a higher level than the intrinsic generic motion primitive-layer. Currently much robot programming is carried out by engineers or robotic system integrators, and even though many applications for stationary industrial robots have similarities, tasks are usually programmed from scratch. For flexible robots, it is necessary to minimize that effort, and adding functionality in the building blocks is a way to do that. However, subprograms, as skills can be considered, can only be reused if they are both generic and robust for the possible set of applications.

Generic skills provide basic functionality on objects, and they follow the basic skill model presented in Sec. II. Being the low-level implementation of skills, these are sequences of motion primitives, where the exact choice of motion primitives is determined online. Generic skills will be implemented by a robot skill specialist, who utilizes the motion primitives for creating more advanced, yet versatile, program building blocks, and establishes the prediction and the pre- and postcondition checks. Strict preconditions ensure that skills with a sufficient robustness can be developed. An evaluation of the skill execution is also essential to ensure the robustness of a program, and statistics of skill executions can be stored to diagnose and improve robot programs and the skill itself. Skills can also be nested within other skills to provide useful higher-level functionality.

Application-oriented skills are skills developed by the customer in-house in order to solve application specific needs for that company only. This allows the customer not to be restricted to the generic portfolio of skills, and instead increase the number of tasks solvable for the flexible robot in specific industrial applications, when a suitable generic skill is not available. Application-oriented skills can be expensive to develop in terms of man-hours as they are programmed from motion primitives and generic skills by engineers or robotic system integrators with little possibility of reuse.

3) Tasks: The task layer contains an abstract description of what the robot is doing in terms of the parameters and state variables. This layer must interface with end users, programming and initiating tasks, and systems controlling the production line, e.g. manufacturing execution systems (MES).

B. Challenges in implementation

Although industrial environments are much more organized than for instance households, it is inevitable that disturbances will occur, e.g. positions of parts or pallets are not completely known. Therefore the robot must be able to update its state variable continuously. Exactly when the robot must update specific state variables is a matter of future research, but at least checking the variables involved in the next skill to be executed may be useful.

A benefit of the skill model shown in Fig. 1, incorporating predictions and checks of pre- and postconditions, is the possibility of implementing error handling in a planning layer. E.g. the robot can be able to backtrack in the case of skill execution errors (the evaluation is unsatisfactory) or when a skill cannot be executed (the pre-conditions are not met). The robot can then find a sequence of skills that will lead to the correct states. Small-scale testing has revealed that it is feasible to integrate a STRIPS planner, at least for logistic tasks.

V. Preliminary Experiments

Several preliminary experiments have been conducted following the approach outlined in this paper, testing some of the aspects of skills shown in Fig. 1. The approach looks very promising, and experiments incorporating all the aspects of skills is among future work.

In one experiment, a multiple part feeding scenario is programmed by visual demonstration by a user. The user points at a box the robot shall pick up, and points at a feeder in which the contents shall be emptied. In this experiment we have implemented

- a very simple set of state variables (gripper and box both have the states full/empty),
- user-supplied parameters (the user specifies which box/feeder should be the focus of the skill), and
- object-centered execution (the system knows, based on the state variables, which objects should be located, and detects the coordinates of the box/feeder at runtime).

The evaluation of the expected outcome and verification of preconditions is not implemented in this particular experiment, since it is merely a proof-of-concept.

VI. Conclusion and Future Work

In this paper a unifying terminology for flexible mobile manipulators in industrial environments is presented, inspired from both research in robot systems architectures and cognitive robots. We see it as a vital part of developing architectures for robot programming that the end users are kept in focus. Our contribution provides a clearer understanding of how different actors - from robot control specialists through integrators to shop floor workers - contribute to programming industrial robots on a task-level.

Logistic and assistive scenarios have been analyzed to reveal that, for many tasks, only a limited set of skills is necessary. The logistic application domain in particular seems promising for skill-based robots, as only a few skills are needed, and many tasks can be solved without complex interactions with machines and other objects. The introduction of skills provides a systematic approach to reuse robotic programs, and being able to specify skills formally in terms of pre- and postconditions can be a way to create a clear view of what the purpose of a skill is.

In the ongoing research, skills will be developed and tested in different application scenarios, with primary focus on assembly tasks and using skills for human-robot interaction for logistic tasks. A central issue here is how to choose the state variables that are to be tested in the precondition checks and the evaluation step in a consistent way. When the skills are in place, it must also be easy for the operators to use them for task-level programming. Therefore another focus of future research is human-robot interaction tailored to the domain of task-level programming. This could be, but is not limited to, the ability to specify sequences of skills in a GUI or through gesture recognition.

Acknowledgment(s)

This work has partly been supported by the European Commission under grant agreement number FP7-260026-TAPAS.

References

[1] Alin Albu-Schaffer, Christian Ott, and Gerd Hirzinger. “A Unified Passivity-based Control Framework for Position,Torque and Impedance Control of Flexible Joint Robots”.*The International Journal of Robotics Research*, *26*(1), pp 23–39, 2007.

[2] Colin C. Archibald and Emil Petriu. “Skills-Oriented robot programming”. *Intelligent Autonomous Systems III*, volume 3, pp 104–113, Pittsburgh, 1993.

[3] Simon Bøgh, Mads Hvilshøj, Morten Kristiansen, and Ole Madsen. “Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM)”. *The International Journal of Advanced Manufacturing Technology*, November 2011.

[4] Simon Bøgh, Mads Hvilshøj, Morten Kristiansen, and Ole Madsen. “Autonomous industrial mobile manipulation (AIMM): From research to industry”. *Proceedings of the 42nd International Symposium on Robotics*, 2011.

[5] A. Björkelund, L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx. “On the integration of skilled robot motions for productivity in manufacturing”. *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, pp 1–9, 2011.

[6] Aaron Bobick and Volker Krüger. “On Human Action”. *Visual Analysis of Humans*, pp 279–288. Springer London, 2011.

[7] Davide Brugali and Patrizia Scandurra. “Component-based robotic engineering (part i)[tutorial]”. *Robotics & Automation Magazine, IEEE*, 16(4):84–96, 2009.

[8] Herman Bruyninckx. “Open robot control software: the OROCOS project”. *Robotics and Automation, 2001. ICRA'01. IEEE International Conference on*, volume 3, pp 2523–2528. IEEE, 2001.

[9] Lars-Peter Ellekilde and Henrik I. Christensen. “Control of mobile manipulator using the dynamical systems approach”. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp 1370–1376, 2009.

[10] Richard E. Fikes and Nils J. Nilsson. STRIPS: “A new approach to the application of theorem proving to problem solving”. *Artificial Intelligence*, 2(3-4): pp189–208, 1971.

[11] Erann Gat. “On three-layer architectures”. *Artificial intelligence and mobile robots*, pp 195–210, 1998.

[12] Christopher Geib, Kira Mourão, Ron Petrick, Nico Pugeault, Mark Steedman, Norbert Krueger, et al. “Object action complexes as an interface for planning and robot control”. *IEEE RAS International Conference on Humanoid Robots*, 2006.

[13] Mads Hvilshøj, Simon Bøgh, Oluf Skov Nielsen, and Ole Madsen. ”Autonomous Industrial Mobile Manipulation (AIMM) - Past, present and future”. *Industrial Robot: An International Journal*, 39, pp. 120-135, 2011.

[14] Mads Hvilshøj, Simon Bøgh, Oluf Skov Nielsen, and Ole Madsen. ”Multiple Part Feeding - Real-world Application for Mobile Manipulators”. Assembly Automation, 32, pp. 62-71,2012.

[15] Volker Krüger, Danica Kragic, Ales Ude, and Christopher Geib. “The meaning of action: A review on action recognition and mapping”. *Int. Journal on Advanced Robotics, Special issue on Imitative Robotics*, 21(13):pp. 1473–1501, 2007.

[16] Volker Kruger, Dennis Herzog, Sanmohan Baby, Ales Ude, and Danica Kragic.” Learning Actions from Observations”. *IEEE Robotics & Automation Magazine*, 17(2), pp. 30–43, June 2010.

[17] Lawrence Rabiner and Biing-Hwang Juang. “Fundamentals of Speech Recognition”. *Prentice Hall*, Englewood Cliffs, 1993.