Modular Robot Software Framework for the Intelligent and Flexible Composition of Its Skills

Lisa Heuss[1(✉)], Andreas Blank[2], Sebastian Dengler[1], Georg Lukas Zikeli[2], Gunther Reinhart[1], and Jörg Franke[2]

[1] Institute for Machine Tools and Industrial Management (*iwb*), Technical University Munich (TUM), Garching near Munich, Germany
lisa.heuss@iwb.mw.tum.de

[2] Institute for Factory Automation and Production Systems (FAPS), Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Erlangen, Germany

Abstract. Current trends such as mass customization necessitate an agile and transformable production. In this context, robotic technologies are seen as a key enabler. But, to date, industrial robots lack the flexibility to easily adapt to changing needs. Therefore, a modular skill-based software framework aiming for free configurability is presented here. A generic task control allows varying incoming tasks to be processed, based on the actual skills of the robot. In this way, the flexible composition of a robot's skills can be achieved, according to the actual situation.

Keywords: Autonomous robot · Configurability · Skills

1 Motivation

Customized products combined with the shortening and dynamization of product life cycles are challenging trends for the manufacturing industry [3]. To stay competitive, robotic technologies are seen as a key driver towards achieving an agile and transformable production [16]. Future robotic systems must be able to complete a wide variety of tasks at short intervals. Autonomous mobile robots in particular therefore demonstrate great potential [1]. However, this is contrary to the current situation. Today's robots lack the flexibility and reconfigurability to respond to changing needs. Great system complexity requires well trained experts to develop and operate those systems [8,16]. Promising to overcome these shortcomings is the skill paradigm [4,5,10,12,15,17–19]. A skill refers to a functionality or service a system offers. By first dividing the robots' functionalities into these skills, they can be freely orchestrated regarding the given task.

Within the scope of this paper, mobile robots should be enabled to autonomously take over varying production tasks, as coordinated and assigned by an superordinate planning system [6]. The tasks considered include commissioning diverse machine parts, pre-assembling these during transport or supporting workers during assembly. The focus of this paper lies on the skill-based

Published by Springer Nature Switzerland AG 2019
F. Ameri et al. (Eds.): APMS 2019, IFIP AICT 566, pp. 248–256, 2019.
https://doi.org/10.1007/978-3-030-30000-5_32

software framework of the robots and addresses two main goals. First, the robot's skills should be freely configurable and easily adjustable to the changing requirements. Second, the robot should be flexible regarding the task handed over by intelligently composing its skills based on generic task control. The following section presents the related work. Then, the skill-based software framework will be introduced in detail and evaluated within two applications. Finally, a conclusion will be given.

2 Related Work

In reviewing related work [4,15,17–19] about the modeling of skills, hierarchical decomposition can be identified as a common key principle. Thereby, the primitives at the lowest level are the reoccurring sub-functions. Skills are composed of primitives and considered as reoccurring robot behaviors. Tasks refer to an ordered sequence of skills aiming to achieve a specific goal. Input and output parameters are used to permit the parametrization of skills [4,17]. The definition of preconditions and effects for individual skills allows for planning a skill sequence to achieve a given task [10,15]. A common state machine can be used to provide generic status information during the execution of skills [4].

Multiple approaches have been introduced for the application of the skill paradigm. Taking into account a whole production system, [12] introduces a holistic framework for the flexible orchestration of devices, based on their offered skills. In a similar context, [4] defines skills as solution-neutral and describes them in this way, semantically correlating to the tasks of a product assembly. Automated matchmaking thus becomes possible. These ideas correlate with the job the superordinate planning system should assume within this work.

With regard to industrial robots itself, skill-based approaches play a significant role in facilitating robot programming, e.g. in developing programming languages [18], skill-based programming systems [5] or combining these with programming by demonstration techniques [17]. [19] introduce a development platform to configure the robot skills by choosing from reusable software modules which are then deployed as apps. These approaches provide a good insight into how to make the robot software freely configurable or how to teach the robot new skills. Some generic task control is however needed in order to coordinate the function call of its skills according to varying tasks sent by a planning system.

[15] follows the trend of cognitive robotics and introduces SkiROS as a skill-based software architecture for robots, which allows them to autonomously plan and execute tasks based on their skills. [10] present a similar concept, while their focus is more on the information models used. Both approaches show the potentials of superordinate task management within the robot architecture. However, to date, these approaches are only applied to kitting tasks and there is a need to extend these to other production domains.

In summary, the initially stated requirements can not be fulfilled by the current state of the art. In order to be flexible regarding the tasks handed over,

there is a need to encapsulate the software realizing a specific skill and provide a generic interface to use it. In addition, generic task management is needed, which works independently from the underlying skills. This is the prerequisite for allowing the easy composition of the robot's skills across the different domains of production.

3 Skill-Based Robot Software Framework

In order to enable robots to autonomously complete varying tasks, a skill-based software framework is presented (see also [6]). An overview is given in Fig. 1.

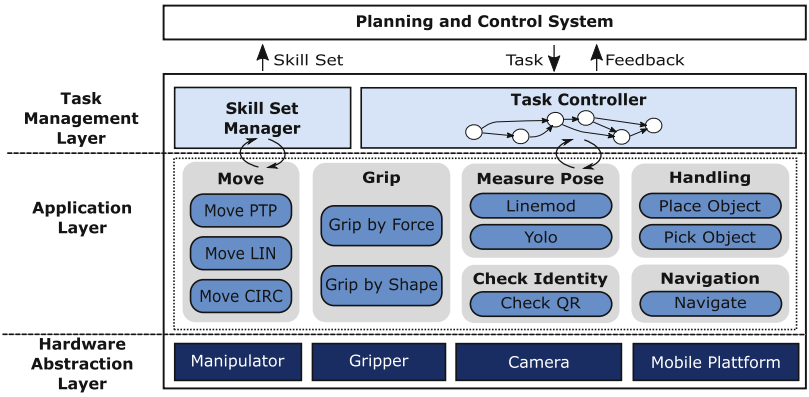

Fig. 1. Overview of the skill-based robot software framework

The modular robot software framework consists of three consecutive layers. At the lowest level, the hardware components of the overall system are integrated, with the aim of hardware abstraction. The application layer provides the skills that enable the robot to achieve tasks. It is expected that robots in the future will have to achieve a diverse range of tasks within short time cycles. Furthermore, there are almost unlimited possibilities for combining the skills in different ways. As a result, it is not possible to store all related knowledge permanently in the robot. Therefore, thematically related skills are combined in an app. Similarly to in the consumer sector, a robot's apps can be freely configured in terms of the current task, or can even be downloaded on a situation-specific basis. Tasks are handed over to and processed by the task management layer. The skill-set manager is responsible for drawing up a list of all the skills the robot offers. It passes this list to the planning and control system. Based on a semantically correlating description of skills and tasks, automated matchmaking as suggested by [4] or [12] can be performed. In this way, tasks to be completed get assigned to suitable and available robots. These are then handed over to and processed by the task controller of the individual robot, which subsequently coordinates the execution of its skills. The application layer is further explained in Sects. 3.1 and 3.2 and Sect. 3.3 describes the task management layer.

3.1 Modeling of Skills

Derived from the related work, *primitive skills* refer to the capabilities of a robot at the lowest level, which can not be further divided. *Composite skills* arise through the combination of primitive or lower-level composite skills and thus allow the modeling of complex robot behaviors on different hierarchy levels. As an example, by composing primitive skills like "move", "grip" and "measure pose", composite skills for "handling" can be achieved.

As well as modeling skills on different hierarchy levels, skills can be described on different levels of abstraction. Within this context, *abstract skills* describe the robot capabilities independently of the executing hardware and the underlying operating principles, whereas *specific* skills inherit from abstract skills and take these into account. [6] Thus, specific skills provide an opportunity to implement different building blocks for an abstract skill, which can be flexibly used depending on task-specific boundary conditions. For example, a gripper offers the primitive skill "grip"; by taking the operating principle into account, it can be separated between the two specific primitive skills "grip by force" and "grip by shape". Within this work an extended version of the skill taxonomy of [4] is used as a basis for the abstract primitive and composite skills.

3.2 Modularization of Algorithms into Apps and Strategic Applets

The skill-modeling gives a functional view of the robot capabilities. These given principles are further used to modularize the underlying software architecture of the robot by encapsulating it into apps or rather applets [6]. These consider the implementation view, build on the same structure and provide a generic interface containing a set of common control algorithms applied for all skills. An *applet* refers to the software implementation of a specific skill. An *app* is used as a generic construct to group a set of thematically-related applets into a software package, which together aim to perform a defined abstract skill.

Applying these concepts raises the central question of the robust and flexible programming of skill-building blocks at the outset when developing the robot's software architecture. By doing so, user-friendliness as well as constant extensibility of a multi-layer architecture are decisively influenced. In this respect, the conception of the architecture's lower levels already proves to be critical. For the realization of abstract skills as encapsulated apps, algorithms are first integrated into the system as independent functions. Of importance in this respect is the identification and definition of program sequences and interaction areas, which are representative for the algorithmic execution of functionalities offered by specific skills. Code fragments can be programmed or extracted from existing programs. Interfaces of separated functions are their prototypes and describe the data types of their return value. Depending on the semantic and functional nature of an abstract skill, several algorithms can be identified as its technical specification in the form of a specific skill. In order to guarantee a maximum degree of adaptability and the continuous modularization of the architecture,

it makes sense to continue collecting such functions in classes according to the guidelines of object-oriented programming. In addition to its functional character, a class is also coherent regarding semantics. Ideally, each skill should create a class with strategically flexible algorithms available to it, thus giving the entire model an intuitive character. Generally, a skill-building block can then be implemented by creating the class object representing an abstract skill in combination with accessing one of its strategically specified skills in the form of a class method. The result of this concept can be transferred to the framework of apps and applets (see Fig. 2) and is exemplified based on a bin-picking task frequently used within commissioning.

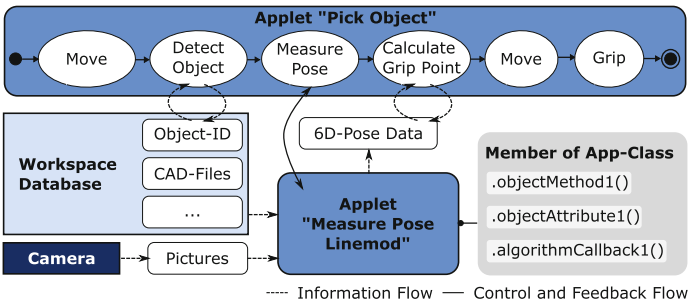

Fig. 2. Exemplary interaction of strategic applets within a skill-based system

In the presented work, ROS is used as implementation basis. Thus, the applet for the exemplary specific skill "measure pose with linemod algorithm" packages all software to run this process within ROS. In this way, the app provides the class-based collection of its strategic algorithms, which are then integrated within the applet. Strategy-based algorithms are similarly defined and used as can be seen in [7]. The app can further group a set of thematically-related applets, building on this same class into one software package. Applets are supposed to interact with all software and hardware resources in relevant system layers to adequately manipulate data. The call of the applet is done from the higher task controller or it can be further used to implement a higher-level composite skill. In this case it is integrated into an applet providing the scripted behavior of a picking skill. In both cases, the higher-lying calling entity handles the transfer of relevant messages via defined interfaces to the next applet.

3.3 Generic Task Control

In order to control the skills by the superordinate task controller, the idea is to include a standardized control interface to every applet. This is illustrated in the bottom part of Fig. 3 for the aforementioned applet to implement the composite skill "pick object". The control interface is defined by a state machine based

on the PackML Interface State Model [11]. Similar to the concept introduced by [4], the applet's state machine describes its current execution status. The state machine furthermore introduces a set of control functions (marked italic in Fig. 3) that can be used to start, stop, pause or reset the applet. Expected, as well as faulty behavior can be monitored via the applet's feedback messages in the form of the current state within the PackML-based state machine.

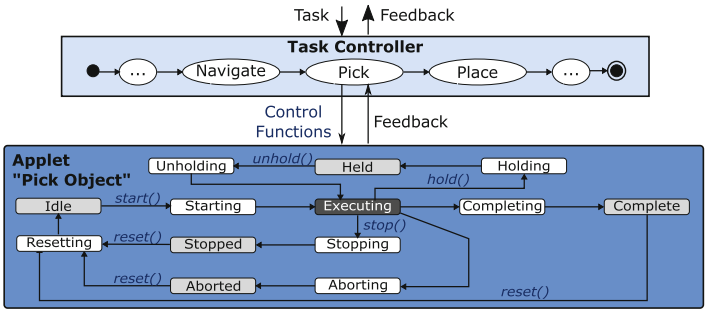

Fig. 3. Task controller interacting with an applet

The functionality that is provided by the application layer is used by the superordinate task management layer in order to perform given tasks. The task controller of the robot receives varying tasks from the superordinate planning and control system (see Fig. 3). Tasks have a complex, nested structure of sequential and parallel actions. Within the task controller, these are modeled as hierarchical finite state machines. The task controller dynamically builds these task state machines at runtime, by recursively iterating through the nested task structure and adding the actions as states to the state machine. Based on the generic control interface provided by each applet, the task controller is able to process the task independently of its structure and the actions it uses.

Within each state of the task state machine, the applet that offers the corresponding skill is addressed and controlled via the standardized generic control functions (see Fig. 3). Through the interface, the task controller is furthermore able to monitor the applet's execution. Faulty behavior of an applet can be detected and handled at task level. The task controller can additionally inform the planning and control system about the faulty behavior via status feedback.

4 Application Examples

Modular, Skill-Based Bin Picking: The methodology for the object-oriented modularization of skills is evaluated based on an existing bin-picking demonstrator. Figure 2 already showed a potential sequencing of specific primitive skills to achieve a composite picking skill. The system set-up is primarily concerned with

recognition, 6D pose determination and robust picking of texture-less metallic machine elements from an associated industrial partner. In this way, the six primitive skills are processed as applets by a SMACH state machine [2]. As shown in Fig. 1, encapsulated, strategy-specific functions are aggregated in semantically coherent classes for apps, as their methods and several applets based on class methods were created for each app. The actionlib [9] provided by ROS is used in this case as the syntactic framework for the latter, resulting in an encapsulated and intuitive way of working when sequencing them within the composite skill. A first approach to implementing primitive skills with a focus on the lower layers of an architecture was successfully evaluated by the demonstrator.

Flexible Task-Handling for Logistic Tasks: The generic task controller for the automatic building and processing of tasks as well as the standardized applet control interface are evaluated on a simulated TurtleBot [14] equipped with an OpenManipulator [13]. The mobile manipulator implements multiple primitive and composite skills for performing logistic tasks. By making use of these skills, the robot is able to process tasks as sets of actions. Through the use of the generic task controller, these tasks do not need to be manually programmed within the robot's control software but can be passed to the robot in form of a nested JSON object. The task controller is able to automatically build a SMACH [2] state machine and control the required applets through the standardized control interface. Additionally, it is possible to halt a running task by sending hold commands to the task controller. When errors are detected within an applet's execution, the task controller gets passed the information about them and stops its execution as well. Further behaviors for error recovery need to be considered within the skill implementation, whereas the applet already provides the organizational structure. Feedback about the preemption or abortion of tasks also gets passed to the planning and control system.

5 Conclusion

In order to enable autonomous mobile robots to easily adapt to changing needs, and to flexibly complete varying tasks coordinated by an superordinate planning system, a modular skill-based software framework is presented. First, to reduce complexity, skills are hierarchically structured into primitive and composite skills and are modeled on different levels of abstraction. These concepts are used to encapsulate the robot's software according to the guidelines of object-oriented programming into independent building blocks, referred as apps and applets. The free configurability of the robot's skills is thus achieved. The PackML state model is introduced as a common interface to control and monitor the execution of the robot's skills. In this way, a generic task control able to process tasks independently of their content and structure is achieved. Thus, the proposed architecture fulfills the prerequisites for use within various production domains.

Acknowledgments. The results presented in this article were developed within the FORobotics (AZ-1225-16) research network. The authors would like to thank the Bavarian Research Foundation and all participating project partners for their funding and support of the project.

References

1. Bøgh, S., Hvilshøj, M., Kristiansen, M., Madsen, O.: Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM). Int. J. Adv. Manuf. Technol. **61**(5), 713–726 (2012)
2. Bohren, J., Cousins, S.: The SMACH high-level executive [ROS news]. IEEE Robot. Autom. Mag. **17**(4), 18–20 (2010)
3. ElMaraghy, H.A.: Flexible and reconfigurable manufacturing systems paradigms. Int. J. Flex. Manuf. Syst. **17**(4), 261–276 (2005)
4. Hammerstingl, V., Reinhart, G.: Skills in Assembly (2018). https://mediatum.ub.tum.de/1428286
5. Herrero, H., Moughlbay, A.A., Outón, J.L., Sallé, D., de Ipiña, K.L.: Skill based robot programming: assembly, vision and Workspace Monitoring skill interaction. Neurocomputing **255**, 61–70 (2017)
6. Heuss, L., Lux-Gruenberg, G., Hammerstingl, V., Schnös, F., Rinck, P., Reinhart, G., Zäh, M.: Autonome mobile roboter in der smart factory. wt Werkstattstechnik online **108**(9), 574–579 (2018)
7. Hoffmann, A.: Serviceorientierte Automatisierung von Roboterzellen: Modularität und Wiederverwendbarkeit von Software in der Robotik. Ph.D. thesis. University of Augsburg (2015)
8. Hvilshøj, M., Bøgh, S., Skov Nielsen, O., Madsen, O.: Autonomous industrial mobile manipulation (AIMM): past, present and future. Ind. Robot: Int. J. **39**(2), 120–135 (2012)
9. Janssen, R., van Meijl, E., Di Marco, D., van de Molengraft, R., Steinbuch, M.: Integrating planning and execution for ROS enabled service robots using hierarchical action representations. In: IEEE ICAR, pp. 1–7 (2013)
10. Kootbally, Z., Kramer, T.R., Schlenoff, C., Gupta, S.K.: Implementation of an ontology-based approach to enable agility in kit building applications. Int. J. Semant. Comput. **12**(01), 5–24 (2018)
11. Nøkleby, C.: OMAC PackML Unit Machine Implementation Guide (2016). http://omac.org/wp-content/uploads/2016/11/PackML_Unit_Machine_Implementation_Guide-V1-00.pdf
12. Pfrommer, J., Stogl, D., Aleksandrov, K., Schubert, V., Hein, B.: Modelling and orchestration of service-based manufacturing systems via skills. In: IEEE ETFA, pp. 1–4 (2014)
13. Robotis: OpenManipulator (2019). http://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/
14. Robotis: TurtleBot3 (2019). http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
15. Rovida, F., Crosby, M., Holz, D., Polydoros, A.S., Großmann, B., Petrick, R.P.A., Krüger, V.: SkiROS—a skill-based robot control platform on top of ROS. In: Koubaa, A. (ed.) Robot Operating System (ROS). SCI, vol. 707, pp. 121–160. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54927-9_4

16. SPARC: Robotics 2020 Mulit-Annual Roadmap (2016). https://www.eu-robotics.net/cms/upload/topic_groups/H2020_Robotics_Multi-Annual_Roadmap_ICT-2017B.pdf
17. Stenmark, M., Topp, E.A.: From demonstrations to skills for high-level programming of industrial robots. In: AAAI Fall Symposium Series 2016, pp. 75–78. AAAI Press (2016)
18. Thomas, U., Hirzinger, G., Rumpe, B., Schulze, C., Wortmann, A.: A new skill based robot programming language using UML/P Statecharts. In: IEEE ICAR, pp. 461–466 (2013)
19. Wenger, M., Eisenmenger, W., Neugschwandtner, G., Schneider, B., Zoitl, A.: A model based engineering tool for ROS component compositioning, configuration and generation of deployment information. In: IEEE ETFA, pp. 1–8 (2016)