Intuitive Instruction of Robot Systems: Semantic Integration of Standardized Skill Interfaces

Junsheng Ding, Ingmar Kessler, Markus Knauer, Andreas Dömel, Sebastian Riedel, Stefan Profanter, Alexander Perzylo, Christoph Willibald, Sebastian Brunner, Arsenii Dunaev, Le Li, Manuel Brucker

fortiss GmbH
Munich, Germany
ding@fortiss.org

Institute of Robotics and Mechatronics
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
markus.knauer@dlr.de

Agile Robots SE
Munich, Germany
sebastian.riedel@agile-robots.com

***Abstract*—This work aims at facilitating the integration of industrial robots and other devices such as their gripper tools at small and medium-sized enterprises (SMEs). For this purpose, an intuitive user interface for the skill-based instruction of robot systems is combined with standardized OPC UA-based skill interfaces that support various hardware and software resources from different manufacturers. Special emphasis is laid on supporting different user groups with varying levels of expertise. Production system engineers are provided with a detailed graphical user interface (GUI) for hierarchically defining new skills by combining preexisting ones. System operators receive a simplified view with limited complexity for process instruction and changing high-level task parameterizations. The skills and relevant semantic context knowledge about products, processes, and resources (PPR) are formally represented in OWL ontologies to enable hardware-agnostic process descriptions that can be deployed to different production environments, while automatically deriving parameterizations for skill invocations. The proposed concept has been qualitatively evaluated in two real-world robot workcells based on a smartphone accessory packaging use case.**

***Index Terms*—Intuitive Robot Programming, Semantic Process Models, Standardized Skill Interfaces**

I. Introduction

Small and medium-sized enterprises (SMEs) face frequent product changes due to their customers' needs and have to reprogram their robot manufacturing systems more frequently. However, the intuitive adaption of robot programs to new product configurations is still challenging due to the large variety of hardware types from different manufacturers with no unified programming interfaces [1], [2]. With this background, our work aims to simplify the integration of industrial robots into SMEs by reducing the time and expertise required to reconfigure and adapt systems to new processes by following a Plug-and-Play approach [3].

For the setup and operation of production workcells consisting of different robots, tools, and sensors, it is currently necessary to implement proprietary interfaces for each component from the corresponding manufacturer. This effort can be significantly reduced by firstly using a common communication protocol such as the open, standardized OPC UA middleware, and by secondly defining and standardizing the programming and information model of involved devices accessed over this protocol. For OPC UA, the standardization of information models is achieved through the definition of so called OPC UA Companion Specifications which have been introduced to facilitate the flexible exchange of devices such as image processing or robotics from different manufacturers within the framework of production system engineering [4]. Since the current focus of Companion Specifications related to industrial robots and grippers primarily revolves around data reading for monitoring purposes, we base our integration approach on the extended device information model proposed in [5] that includes and defines active control of device motions via a standardized skill interface. Hence, we employ OPC UA and newly developed implementations of this extended device information model to facilitate the hardware-agnostic, skill-based programming approach presented in this work.

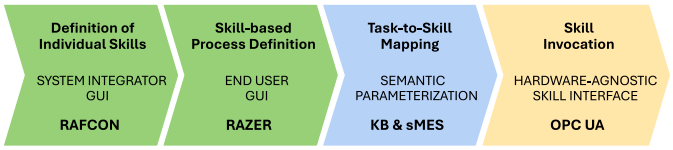


Fig. 1: Workflow for the intuitive instruction of robot systems: Defining additional skills in a GUI (optional), instructing a robot system via skill sequences in a second GUI, mapping of generic task representations to skill implementations and their parameters, and invoking identified skills via OPC UA.

To strengthen human-robot collaboration in SMEs, it is important to enable the intuitive programming of robots [6]. Robot systems that attempt to meet this requirement for simple programming by end users are already being sold. However, these are usually only adapted to specific robot types and manufacturers and cannot simply be extended with new skills.

In our approach (see Fig. 1), a standardized skill library for devices in robot systems is created that represents the capabilities of commonly used components via standardized interfaces (e.g., robot, parallel gripper, vacuum gripper). The skills are

The authors acknowledge the financial support by the Bavarian Ministry of Economic Affairs, Regional Development and Energy in the project IRISS (grant number DIK-2008-0010//DIK0212).

semantically modeled in a knowledge base to augment their formal descriptions with additional context knowledge about products, processes, and resources (PPR). Building on this, a graphical user interface (GUI) for system integrators is adapted and integrated for programming new high-level composite skills (for specific applications) via the aggregation of low-level basic skills of robots and their tools. A second GUI is implemented for end users to leverage their domain expertise in production via the intuitive instruction and parameterization of manufacturing processes with the existing skills.

II. Related Work

A. Skill-Based Programming

A skill is defined as the basic functionality that a system component makes available to other components [7], [8], [9], [10], which has various properties and parameters that need to be set for execution. Programming skills and skill sequences can be facilitated by GUIs, e.g., ROS Commander [11] or FlexBE [12]. These approaches use visual representations of hierarchical state machines or state transition diagrams [13] to visualize the execution flow of skill executions as intuitively as possible. In our previous works on RAZER [14], RAFCON [15], or intuitive robot instructions [16], the users are supported by intuitive robot programming interfaces.

Currently, graphical skill-programming interfaces are also provided by cobot manufacturers, such as Franka Robotics or Universal Robots, but they only support rigid procedures with limited interfaces to their own robot models. In addition, some software companies offer programming environments for multiple robot models, e.g., drag&bot[1], ArtiMinds[2], and Mujin[3]. However, the possibilities of generic robot programming here are limited to the functionalities provided by the robot manufacturer in its controller. Furthermore, although programs are typically created through a unified interface, they still only work for specific robot types and are not easily transferable to other robot systems.

B. Control of Robot Systems

The control of industrial robot arms currently mainly occurs via proprietary programming languages of their respective manufacturers. Program code written for robots of one manufacturer is not transferable to robots of others, resulting in a high effort to integrate different robots, tools, and sensors. One approach to achieve vendor-agnostic control of robot systems is to control robots over a real-time joint position, velocity, or torque streaming interface. This is typically done in research and academia, e.g., *ros_control* [17] and the *Robotics Library*[4] [18], but more recently Intrinsic[5] is also known to follow this path. This approach enables (and necessitates) to implement part of the trajectory planning and control in a vendor-agnostic way outside a vendor's control box. While the benefits are vendor-agnostic, unified motion parameterization (e.g., treatment of singularities and multiple IK solutions), advanced robot trajectory control (e.g., jerk optimization), and potentially more advanced collision detection and motion planning, the main drawback is that only a few robot vendors provide an industrial-grade and public real-time interface which allows this kind of control (including, e.g., access to kinematic calibration data of the robot). Furthermore, the higher computational load and the increased deployment complexity for achieving a deterministic real-time control environment on an industrial computer (next to the robot control box) needs to be considered.

(a) Type 1 box with CN charger. (b) Type 2 box with DE charger.

Fig. 2: Product variants of smartphone packages with different shapes and accessory types in our real-world use case.

Another approach would be to generate vendor-specific program code from a vendor-agnostic task specification. This approach is followed, e.g., by ArtiMinds. It avoids deployment complexity with the mentioned drawback of relying on the robot's inherent control modes and capabilities.

With [5] and in the spirit of OPC UA Companion Specifications, we follow a similar approach, but replace vendor-specific program generation with vendor-specific OPC UA adapters and a simple but sufficient vendor-agnostic programming model on the level of OPC UA. Aside from the benefits of OPC UA's information model, which can be queried by any other part of the system, it is also a more semantically meaningful representation of a robot as a *resource* (together with other resources following the PPR paradigm) in an overall robot system. In addition to previous work like [19], where the OPC UA middleware serves as the basis for connecting hardware components, we also consider the programming of complex skills from basic ones in a GUI for expert users.

C. Semantic Description Models

The semantic skill paradigm extends the definition of functional interfaces of skills to enable the reusability of already implemented functionalities. Semantic information is modeled to reduce the integration effort of hardware and software components from different manufacturers into a complex overall system. This includes fundamental information such as kinematic and geometric models of components [19] as well as their capabilities [20]. In the context of industry, standards of the Semantic Web have been employed for such information models [21].

[1]https://www.dragandbot.com
[2]https://www.artiminds.com
[3]https://mujin-corp.com
[4]https://www.roboticslibrary.org
[5]https://www.intrinsic.ai/capabilities

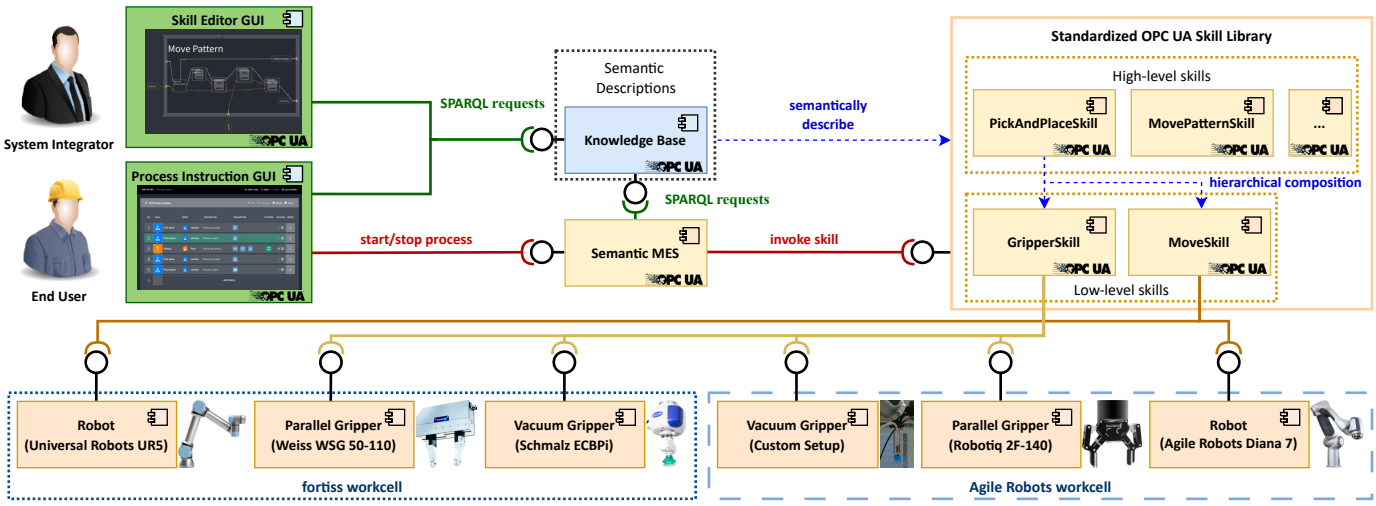


Fig. 3: Overall system architecture of the proposed approach. Hardware from various manufacturers is abstracted via standardized OPC UA skill interfaces. The skills are semantically described and can be hierarchically composed into higher-level skills. Intuitive GUIs enable the programming of skills and instruction of processes for different user groups.

III. Use Case

For the demonstration of the proposed approach for an intuitive-to-instruct and hardware-agnostic robot system, the packaging of smartphone accessories has been chosen as a use case. Currently, this is typically carried out manually due to a high variation in the product configurations and a comparative easiness in instructing a human compared to programming an automation system, especially when taking into account variability with respect to packaging for different target markets as well product variants. Our scenario consists of the following process steps: 1) taking the bottom of a box from an infeed tray and placing it in a working fixture, 2) taking a smartphone charger from an infeed tray and placing it into the box, 3) taking the lid of a box from an infeed tray and placing it on the box to close it, and 4) placing the final assembled box in an outfeed tray. Fig. 2 shows two example smartphone accessory package variants whose assembly can be semantically described in process models that link to the different involved product models of the subparts. We focus on the packaging task for a charger, while packaging tasks for other subparts would be handled analogously.

In the following Sections IV, V, VI, and VII, we will introduce the main components of our concept, while continuing with this guiding use case as an example. We defined four desirable criteria that an intuitive-to-instruct and hardware-agnostic robot system should fulfill and that will receive a qualitative evaluation in Section VIII:

C1: Reuse of the same standardized OPC UA skill interface by multiple organizations to develop interoperable implementations that can communicate with each other in the same robot system.

C2: Reuse of the same standardized OPC UA skill interface in different OPC UA adapters for different hardware component types, such as robots from different manufacturers in two workcells being controlled via the same *CartesianLinearMoveSkill* interface.

C3: Reuse of a semantic process model in different workcells through hardware abstraction, such as the same semantic assembly process being performed in two workcells with different hardware configurations.

C4: Reuse of generic software components, such as the knowledge base and the semantic manufacturing execution system in two different workcells.

IV. System Overview

The system architecture is designed for the proposed approach for an intuitive-to-instruct and hardware-agnostic robot system. Its main components and standardized interfaces are shown in Fig. 3. Within our system, standardized OPC UA interfaces are used to control the hardware via basic device skills (see Section V). To integrate them into the intuitive programming and instruction framework, the standardized skills and their interfaces are formally represented in semantic resource models (see Section VI).

The knowledge base (KB) [19] is responsible for the persistent storage and interpretation of all relevant semantic information about the production system. This includes the manufacturing resources, the manufacturing process and its subtasks, as well as the product to be manufactured (see Section VI). The KB is implemented with a GraphDB[6] triplestore, which provides an HTTP REST API for data interchange.Within our system, the KB implements a wrapper around the triplestore and provides an OPC UA-based interface. It enables other components in the overall system to interact with the semantic information via SPARQL queries and updates or more specialized services. The knowledge representation is modeled using semantic description languages that were defined as ontologies

[6]https://www.ontotext.com/products/graphdb

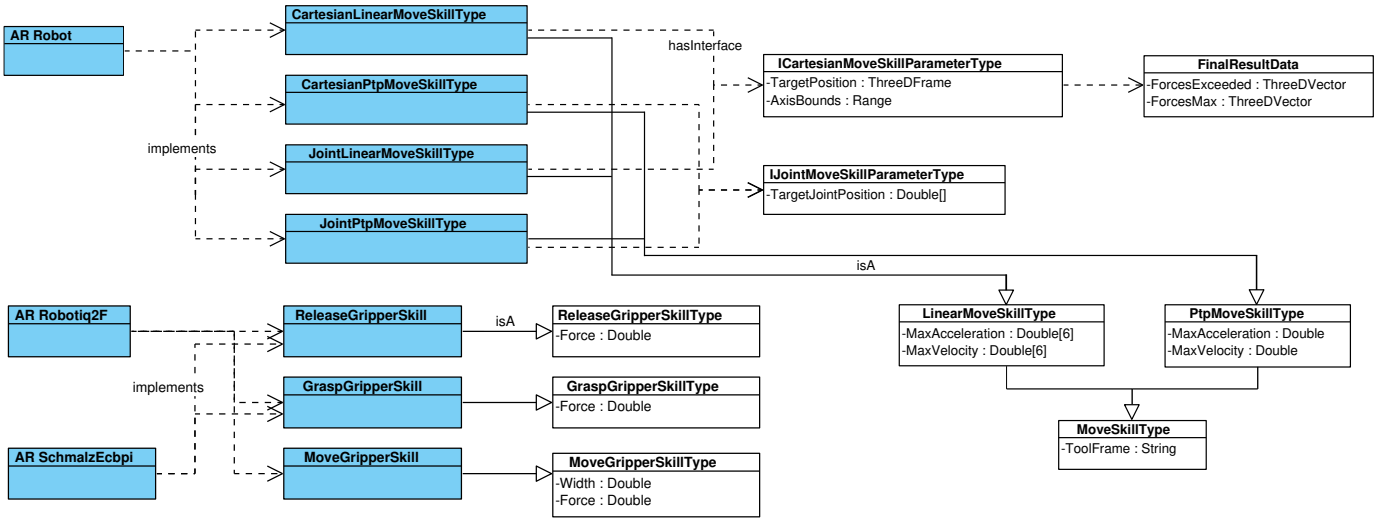


Fig. 4: The device class *AR Robot* for different robot models from multiple manufacturers as well as the device classes *AR Robotiq2F* and *AR SchmalzEcbpi* for parallel and vacuum grippers. Each device class implements executable skills that are derived from standard skill types and exhibit respective parameterization interfaces.

based on OWL 2 (Web Ontology Language). The KB also provides reasoning functionality for the semantic information via OWL 2 and GraphDB to detect logical inconsistencies and automatically derive implicit facts from the explicitly represented ones.

The semantic MES (sMES) [19] is the central system component responsible for managing and monitoring the execution of manufacturing processes in production environments and utilizes the semantic information in the KB to increase system autonomy. The hardware and software components announce themselves via OPC UA Local Discovery to the sMES, which detects the skills that they provide. During process execution, the sMES calls the KB, which returns the next skill to be invoked along with appropriate parameter values based on the semantic description models (see Section VI). With these results, the sMES finds, parameterizes, and invokes the next skill, and sends its results to the KB afterwards. With this approach, the system enables the operator to define on an abstract level a manufacturing process that can be executed autonomously on different hardware configurations with the standardized skill interfaces.

Two types of GUIs have been integrated into the system for different user groups. RAFCON [22] serves as a skill editor interface for system integrators (see Section VII-A) and RAZER [14] serves as as process instruction interface for system operators (see Section VII-B). Both GUIs do not directly interact with skill components, but primarily with the KB to enable operators to create, monitor, modify, or control semantic manufacturing processes. The execution of a manufacturing process can be started by selecting it from a list of created processes and sending its identifier to the sMES.

V. Standard Skill Interfaces

In order to enable hardware-agnostic programming of workcells, hardware capabilities have to be represented and interfaced with in a standardized way. To achieve the same outcome of a hardware-agnostic task on different hardware configurations, the effects of controlling hardware need to be sufficiently modeled as well. Our standardized skill concept (skill state machine, parameterization, capabilities) is based on work by fortiss [5] and leverages OPC UA's information modeling capabilities to provide the necessary semantic integration into the system architecture. Industrial robots and grippers are modeled through several standardized skill types such as *LinearMoveSkill* or *MoveGripperSkill* that are implemented and referenced in OPC UA adapters around hardware-specific control interfaces to the hardware. This extra software layer is necessary as neither the current OPC UA Companion Specifications nor other industrial standards are available which specify a standardized control and programming model of hardware devices like industrial robots from different manufacturers (contrary to, e.g., the GenICam[7] standard for industrial cameras).

At the moment, our hardware adapters implement three device classes: *AR Robot*, *AR Robotiq2F*, and *AR SchmalzEcbpi*. Each device class further implements various skills according to their respective OPC UA device class definition. We reused the skill and parameterization information models as described in [5]. The input parameterization of a skill is derived through specialization (*isA*) and interface usage (*hasInterface*), and the output parameterization is derived through the provision of a *FinalResultData* attribute. Some of the modeled device skills and their input/output parameterization are sketched out

[7]https://www.emva.org/standards-technology/genicam

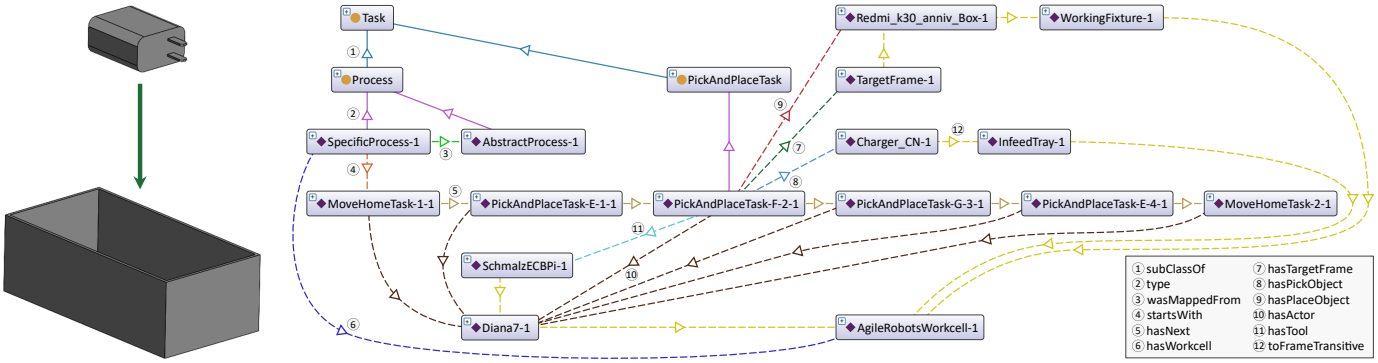


Fig. 5: Simplified excerpt of a specific process ontology during its execution (right) that illustrates one loop iteration to assemble one box, charger, and lid (see Fig. 2a) where *PickAndPlaceTask-F-2-1* places *Charger_CN-1* into *Redmi_k30_anniv_Box-1* (left).

in Fig. 4 with device classes and implemented skills being highlighted in blue.

The modeling of robot motion skills is based on describing and modeling the type of motion (*LinearMoveSkillType* or *PtpMoveSkillType*) and the type of target parameterization (*ICartesianMoveSkillParameterType* or *IJointMoveSkillParameterType*). The corresponding parameters (e.g., *MoveSkillType::ToolFrame*) of the different types accumulate in the final skill type definitions (e.g., *CartesianLinearMoveSkillType*) and then form the final input parameter set of the corresponding skill. Currently, only skills with the interface *ICartesianMoveSkillParameterType* have readable output parameters. These are modeled in OPC UA via a *FinalResultData* attribute, which in this case informs about the forces generated during the movement. The actual modeling in OPC UA is more detailed (including optional parameters, default values, skill state machine, derivation of types from OPC UA standard types, etc.) and the reader is referred to [5]. The information models defined there were expressive enough for the chosen use case and therefore no additional information modeling on the skill level was necessary.

The gripper interfaces are structurally simpler than the robot interfaces in terms of parameterization and mainly differ in that the *MoveGripperSkill* (for approaching a finger position with a specified gripper span) is only offered by parallel grippers, such as the Robotiq 2F gripper. OPC UA adapters implementing the aforementioned information models have been developed for Agile Robots' as well as Universal Robots' robots, Robotiq's as well as Weiss's parallel grippers, and Schmalz's integrated ECPBi vacuum gripper.

VI. Semantic Description Models

In the semantic description models, a *product model* (or other entity) can be referenced not only by a model number or a file name. Instead, Semantic Web technologies such as globally unique identifiers (URIs) and OWL ontologies are used to formally represent both them and their internal entities as well. Their consistent references and a semantic modeling of their properties enable deep links to them and external applications to extract and use them in subsequent procedures.

For example, the exact geometry of the 3D models of the components in the simulation of the workcells (see Fig. 9a and Fig. 9c) including their internal entities can be formally represented. These entities can be directly referenced due to their modeling that uses the OntoBREP ontology (Ontology-based Boundary Representation) [23]. These OntoBREP models include entities for each surface, curve, and point with their own URIs for linking to/from other (potentially external) entities. With a self-developed transformation tool, these semantic descriptions can be automatically generated from industry-standard STEP or IGES files. Additional information can also be automatically determined during the transformation and annotated in their semantic models, such as the sizes of surfaces and volumes. In addition to geometry, other product properties can also be semantically described, such as dimensions, material, gripping points, or subparts. This enables process descriptions to reference individual parts of a product, e.g., for task parameterization, such as the relative geometric positions of multiple slots in infeed/outfeed trays based on their classes in the product taxonomy.

An abstract *process model* semantically describes in a declarative fashion the manufacturing of a product via a sequence of tasks and their reusable object-level parameters. This description is intended to be relatively hardware-agnostic, so that the same process can be automatically adapted when deploying it to a changed hardware configuration of a workcell or to a different workcell altogether.

This deployment is done by the automatic mapping procedure in the KB during the execution of a specific process that is generated by mapping an abstract process to a specific workcell [19]. Essentially, an abstract process can be thought of as a template where abstract task parameters are free variables that will be bound to specific instances in a workcell based on their types and other properties. This generates a new specific instantiation of this abstract process for the current hardware configuration and the real object positions. Fig. 5 illustrates such a specific process in the OWL ontologies including its tasks that correspond to the steps in Section III and its links to entities from the product and resource models. A specific process description also serves as a semantic execution log of

a production run that is linked to the manufactured product instances and can be used to calculate certain KPIs. More complex workflows beyond simple linear sequences are also possible, such as conditional branching. For this use case, e.g., the abstract process assembles smartphone packages in a loop until all infeed trays are empty or all outfeed trays are full.

How certain types of tasks and parameters are mapped is modeled in the OWL ontologies as well and is generically applied by the KB [19]. This also includes matching each task in a process to a skill provided by the resources in a workcell based on a comparison of the required and offered capabilities, respectively. This also enables the parameterizations of the skills being automatically derived based on the task parameters and other semantic context knowledge. Each task is either directly matched to a skill, such as a high-level composite *PickAndPlaceSkill* or, if none is available in this workcell, a previously modeled semantic task template can generate for some simple scenarios a sequence of subtasks that are matched to basic skills to achieve the same effect. In either case, the process is executed by sequentially invoking the matched skills.

A *resource model* semantically describes the manufacturing resources in a workcell such as hardware and software components. This ranges from the skills they provide and the properties of the components themselves, to the topology of the workcell and the specific object instances it currently contains. For example, a smartphone box has a relative position in a slot of an infeed tray on a table, on which a robot's base is mounted. This robot can use a parallel gripper and a vacuum gripper that have their own tool center points (TCPs) based on their types and that are both mounted on a dual gripper adapter that is connected to the robot's end effector. By following this chain of relative positions, the target positions of the *CartesianLinearMoveSkills* are automatically adapted and derived based on the current workcell configuration during each *PickAndPlaceTask* in the process. Individual grasp parameters for certain object and gripper types are modeled in the ontologies or could be calculated by grasp planners, defined via teaching by demonstration, or handled within high-level composite skills.

VII. Intuitive Programming Interfaces

A. System Integrator Interface

The System Integrator Interface (skill editor GUI for experts) is used to integrate expert knowledge into the system in a reusable way. This is implemented via the creation of composites skills. The System Integrator Interface enables the easy combination of existing skills and available semantic information from the knowledge base into new higher-level skills. These composite skills enable the aggregation of more complex, combined, or application-specific operations; automatically incorporate derivable information; and elevate the skill parameters from technical parameters (e.g., 3D positions) to domain-specific parameters (e.g., storage locations, magazine slots, etc.).

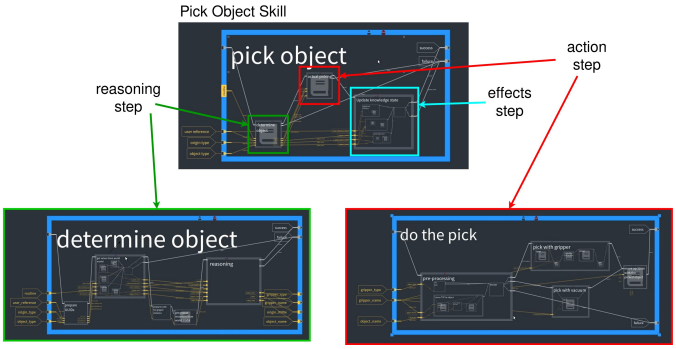


Fig. 6: RAFCON GUI for graphical programming with a *Pick Object* skill and the 3 phases *Reasoning*, *Action*, and *Effects*.

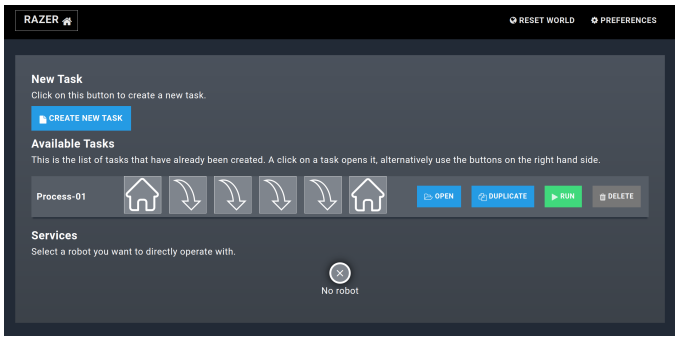


Fig. 7: RAZER GUI home screen showing an existing process.

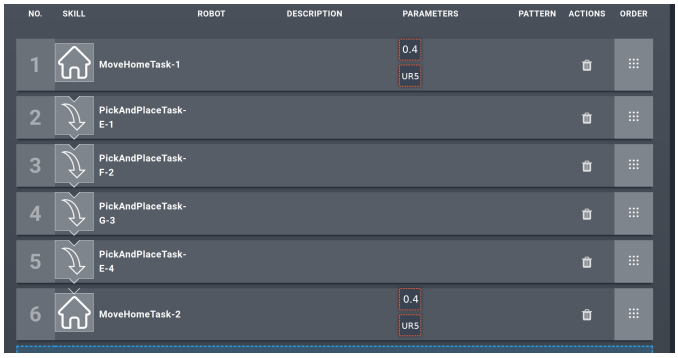
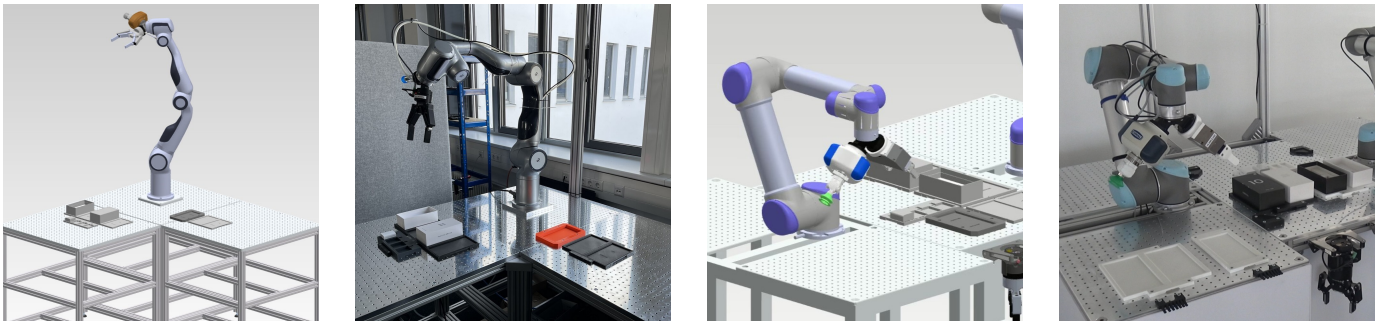


Fig. 8: RAFCON GUI task screen for a composed task sequence.

The implementation of the System Integrator Interface is based on the freely available software RAFCON [22]. The skills are represented in RAFCON via so-called libraries. These can be organized hierarchically and connected by logic as well as data flow. This makes it possible to develop highly complex behaviors with a large number of individual skills. Since each composition can be saved as a library, it is easy to reuse sub-problem solutions. To connect the System Integrator Interface with the knowledge base, code generators are used to create a RAFCON library from the semantic description of a skill. The composite skills created in the System Integrator Interface can also be exported fully automatically to the knowledge base.

The implementation as well as the complexity of a



(a) AR workcell in simulation (b) Physical AR workcell (c) fortiss workcell in simulation (d) Physical fortiss workcell

Fig. 9: Two workcell layouts at Agile Robots and fortiss used in the experiments, equipped with different hardware components providing similar functionalities. Both setups perform the same process of smartphone accessory packaging. Infeed trays (currently occupied) provide the parts, the assembly fixture (single slot) enables the execution of the assembly steps, and the outfeed tray (dual slot) is used to store completed products.

knowledge-augmented skill is illustrated in Fig. 6 using the example of a *Pick Object* skill. The implementation is based on a semantic model of products, processes, and resources as described in Section VI. Based on this, a higher-level skill can be graphically constructed in the System Integrator Interface, enabling the user to easily parameterize it using *Pick-Object-Type* and *Pick-Location*. Internally, this involves three phases: deriving the necessary geometric knowledge from the inputs and the knowledge base (*Reasoning*), parameterizing and invoking the corresponding robot and device skills (*Action*), and then storing the achieved effects back into the knowledge base (*Effects*). After being provided by the system integrator, the composite skills can be loaded and parameterized in the End User Interface.

B. End User Interface

The End User Interface (process instruction GUI for system operators) is used to parameterize a robot for new tasks or to optimize a current process. It is based on the software RAZER [14] and its aim is to enable users to instruct a robot intuitively, i.e., without expert knowledge. In order to achieve this, some simplifications have been made compared to the System Integrator Interface. The End User Interface is limited to a sequence of individual steps that can be parameterized independently of each other.

On the End User Interface home screen (see Fig. 7), users can create new tasks directly by selecting 'CREATE NEW TASK'. Existing tasks can be opened, edited, duplicated, deleted, or executed. The interface also provides information about available robot resources for task execution, which are displayed in the lower section. Fig. 8 shows the overview of a particular task, which is displayed when a task is opened. The End User Interface offers functionalities to manipulate this sequence, including the addition, modification, or removal of subtasks, as well as the ability to rearrange their order as necessary.

VIII. Experiments

In order to demonstrate our hardware-agnostic, skill-based programming approach, two workcells with different hardware configurations were set up (see Fig. 9), where the same semantic assembly process (see Section VI) for smartphone accessory packaging (see Section III) was performed. In the first workcell at Agile Robots, an Agile Robots Diana 7 robot with a dual gripper adapter was equipped with a Robotiq 2F-140 parallel gripper and a custom vacuum gripper setup (similar to a Schmalz ECBPi). In the second workcell at fortiss, a Universal Robots UR5 robot with a dual gripper adapter was equipped with a Weiss WSG 50-110 parallel gripper and a Schmalz ECBPi vacuum gripper. The devices in both workcells were integrated via OPC UA adapters that follow the standardized skill interfaces. Aluminum tabletops with hole patterns were designed for the flexible fixation of infeed trays (for smartphone boxes, USB chargers, and box lids), working fixtures, and outfeed trays, whose positions are also described in the semantic model of the workcell. Parallel to their physical construction, both workcells were also tested in simulations of production runs as shown in Fig. 9a and 9c.

During the experiments, our proposed system was able to automatically adjust the assembly process execution to the different hardware configurations and successfully perform the assembly process in both workcells. A qualitative evaluation was performed based on the criteria defined in Section III with the results being as follows:

C1: Use of different but compatible implementations of the standardized OPC UA skill interface in the sMES (by fortiss) and in the OPC UA adapters (by Agile Robots) in their workcell during successful process executions.

C2: Use of an UR5 robot and a Weiss WSG 50-110 gripper in the fortiss workcell, and a Diana 7 robot and a Robotiq 2F-140 gripper in the Agile Robots workcell during successful process executions, with their different OPC UA adapters sharing the same standardized robot and gripper skill interface, respectively.

C3: Use of the same hardware-agnostic semantic process model in both workcells during successful process executions, with parameterizations of skill invocations being automatically adapted based on the different hardware configurations.

C4: Use of the same generic KB and sMES software components in both workcells during successful process executions, without changing their source code and with them being fully parameterizable via the semantic description models and the skill components' OPC UA adapters.

The results of our experiments indicate the potential efficiency improvements in system configuration, adaptation, and operation with our proposed approach.

IX. Conclusion

This work introduces a knowledge-augmented and skill-based approach for the intuitive instruction and operation of robot systems. Based on formal representations of skills, which are provided by the manufacturing resources in a given production environment, the intuitive instruction of processes is enabled via an end user GUI, in which a sequence of high-level tasks can be defined. For the flexible extension of the system, a second GUI for system integrators can be used to hierarchically recombine basic skills into higher-level composite skills, which are made available again in the end user GUI. Due to the standardization of skill interfaces including their parameter sets for certain skill types, process descriptions can be hardware-agnostic making manual hardware-specific adjustments to them unnecessary. Required expertise in automation and robotics is reduced due to the intuitive GUIs and the system being capable of automatically parameterizing skill invocations based on the rich context knowledge and high-level instructions. Real-world experiments in two different workcells have shown that involved knowledge models and software components can be reused across different hardware configurations. As a result, a domain expert (who is not necessarily an automation expert) is enabled to operate complex robot systems.

References

[1] J. Pires, "Robotics for small and medium enterprises: control and programming challenges," *Industrial Robot*, 2006.

[2] G. Ajaykumar and C.-M. Huang, "User needs and design opportunities in end-user robot programming," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 93–95.

[3] A. Valera, J. Gomez-Moreno, A. Sánchez, C. Ricolfe-Viala, R. Zotovic, and M. Valés, "Industrial robot programming and upnp services orchestration for the automation of factories," *International Journal of Advanced Robotic Systems*, vol. 9, 2012.

[4] OPC Foundation, "OPC UA for Robotics Companion Specification – Part 1: Vertical integration," OPC Foundation, Tech. Rep., 2019.

[5] S. Profanter, A. Breitkreuz, M. Rickert, and A. Knoll, "A hardware-agnostic OPC UA skill model for robots, tools and other devices," in *Proceedings of the IEEE International Conference on Emerging Technologies And Factory Automation (ETFA)*, Zaragoza, Spain, Sept. 2019, pp. 1061–1068.

[6] R. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *VDI BERICHTE*, 2006.

[7] R. Froschauer, A. Köcher, K. Meixner, S. Schmitt, and F. Spitzer, "Capabilities and skills in manufacturing: a survey over the last decade of etfa," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2022, pp. 1–8.

[8] A. Bayha, J. Bock, B. Boss, C. Diedrich, and S. Malakuti, "Describing capabilities of industrie 4.0 components: Joint white paper between plattform industrie 4.0, vdi gma 7.20, basys 4.2," 2020.

[9] S. Malakuti, J. Bock, M. Weser, P. Venet, P. Zimmermann, M. Wiegand, J. Grothoff, C. Wagner, and A. Bayha, "Challenges in skill-based engineering of industrial automation systems," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 67–74.

[10] X.-L. Hoang, C. Hildebrandt, and A. Fay, "Product-oriented description of manufacturing resource skills," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 90–95, 2018.

[11] Hai Nguyen, M. Ciocarlie, Kaijen Hsiao, and C. Kemp, "ROS commander (ROSCo): Behavior creation for home robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 467–474.

[12] P. Schillinger, S. Kohlbrecher, and O. von Stryk, "Human-Robot Collaborative High-Level Control with an Application to Rescue Robotics," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 2796–2802.

[13] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 461–466.

[14] F. Steinmetz, A. Wollschläger, and R. Weitschat, "RAZER - A HRI for visual task-level programming and intuitive skill parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1362–1369, 2018.

[15] S. G. Brunner, F. Steinmetz, R. Belder, and A. Dömel, "RAFCON: A graphical tool for task programming and mission control," in *Robot World Cup*. Springer, 2016, pp. 347–355. [Online]. Available: https://elib.dlr.de/109241/

[16] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Republic of Korea, Oct. 2016, pp. 2293–2300.

[17] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf

[18] M. Rickert and A. Gaschler, "Robotics Library: An object-oriented approach to robot applications," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sept. 2017, pp. 733–740.

[19] A. Perzylo, I. Kessler, S. Profanter, and M. Rickert, "Toward a Knowledge-Based Data Backbone for Seamless Digital Engineering in Smart Factories," in *Proc. IEEE Int. Conf. Emerging Technologies And Factory Automation (ETFA)*, Vienna, Austria, 2020, pp. 164–171.

[20] M. Weser, J. Bock, S. Schmitt, A. Perzylo, and K. Evers, "An Ontology-based Metamodel for Capability Descriptions," in *Proc. IEEE Int. Conf. Emerging Technologies And Factory Automation (ETFA)*, Vienna, Austria, 2020, pp. 1679–1685.

[21] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Miguelanez, "An ieee standard ontology for robotics and automation," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 1337–1342.

[22] S. G. Brunner, F. Steinmetz, R. Belder, and A. Dömel, "Rafcon: A graphical tool for engineering complex, robotic tasks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-November. Institute of Electrical and Electronics Engineers Inc., pp. 3283–3290.

[23] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sept. 2015, pp. 4197–4203.