Flexible Modeling and Execution of Semantic Manufacturing Processes for Robot Systems

Ingmar Kessler, Alexander Perzylo
fortiss, Research Institute of the Free State of Bavaria associated with Technical University of Munich
Guerickestraße 25, 80805 München, Germany
{ikessler, perzylo}@fortiss.org

***Abstract*—The potential benefits of digital transformation for manufacturing companies include reduced costs, increased interconnectedness, and improved adaptability. Semantic Web technologies such as IRIs, RDF graphs, OWL ontologies, and SPARQL requests are a well-known and actively researched approach for supporting these transformation efforts. One challenge with this concept of knowledge augmentation is identifying *where* and *how* to integrate such semantic technologies into a manufacturing system, as it could require frequent translations into other non-semantic representations, which may entail a loss of expressivity and other disadvantages. Therefore, this work aims to use semantic technologies in a knowledge-augmented robotic manufacturing platform as directly and natively as possible. This approach includes the semantic modeling of manufacturing processes (similarly to flow charts) and context knowledge such as generalized mechanisms of how to apply them. All of this semantic knowledge is instantiated and persistently stored in a Robot Knowledge Base application, which implements mechanisms to automatically derive the next robot skill invocations and their parameter values during process execution. These semantic description models and the Robot Knowledge Base were tested in simulation as well as integrated into a physical mobile robot system with an articulated arm tackling an industrial use case.**

***Index Terms*—Knowledge-Based Robot Systems, Semantic Process Models, Automated Mapping & Execution, OWL Ontologies, RDF, SPARQL, Semantic Web, PPR, Tasks, Capabilities, Skills**

I. Introduction

Digital transformation is a key challenge of manufacturing companies to stay competitive in current and future markets [1]. One aspect of this is increasing automation on both the hardware and software level to lower costs but also to increase speed and flexibility. For robotic manufacturing systems this means that they would no longer be isolated and static work stations, but increasingly autonomous and adaptive systems with a wider set of automated operations. This requires increased interconnectedness to base decisions on additional knowledge and understanding as well as to share generated manufacturing knowledge with other actors in a product's lifecycle management.

This leads back to a general need in digital transformation to move away from insular information systems and towards platforms as software-defined infrastructures that connect different people and systems within and across the boundaries of departments and organizations [2]. To enable both humans and machines like this, proprietary interfaces that exchange raw data are not enough. Instead, semantic knowledge would be formally represented using OWL ontologies and other Semantic Web technologies such as SPARQL, so that it can be more easily shared and combined.

Therefore, this work presents an approach for using semantic knowledge to firstly model products, processes, and resources (PPR) and to secondly apply it in an automated robotic manufacturing system. Section II gives a short overview of other high-level control and semantic approaches for robot systems. Section III describes how semantic manufacturing processes and context knowledge can be flexibly modeled on a task and object level. Section IV shows how they can be applied during the execution of a manufacturing process to automatically derive suitable skill invocations and parameter values. Section V illustrates how the semantic models and the software component were used as part of a larger robot system in an industrial use case and Section VI concludes this work.

II. Related Work

There are several approaches to high-level robot control such as task planners, state machines, behavior trees [3], or flow charts [4]. Data structures such as trees of world states, data flows, local/global variables, blackboards, or databases are varyingly suitable and common to such approaches – each on its own or in combination with others. Furthermore, various efforts have been made to introduce semantic representations and technologies into these approaches and therefore the used data structures.

Semantic technologies such as triplestores, i.e., RDF graph databases, are designed to natively support and leverage the advantages of semantic representations. This includes functionality for (and based on) OWL ontologies such as import, export, and persistent storage; built-in automatic inference and consistency checks; and the semantic query and update language SPARQL. For example, an OWL ontology uses preexisting or new properties based on the formally defined OWL vocabulary and it is stored in a triplestore as a set of asserted triple statements. Thus, the built-in reasoner of a triplestore can automatically compute an additional set of inferred triple statements that are also part of the set of total triple statements that is available to any SPARQL query. Therefore, using semantic representations without semantic technologies would mean losing some of their practical benefits.

The combination of and translation between different representations and technologies to achieve high-level robot control is an established approach in research (which may also entail

The final published version of this work is available online at: https://doi.org/10.1109/ETFA61755.2024.10710791

additional effort and other issues). For example, task planners find (in the simplest terms) a sequence of actions from an initial state to a goal state, but whether an action is possible in the current state may be infeasible to express and evaluate on a purely symbolic level. Consequently, advanced hierarchical planners [5] have found success in real-world scenarios [6] by combining symbolic task planning and subsymbolic geometric motion planning. Another example are state machines where the robot system dynamically reacts to environmental changes or signals to transition from one state (or action) to another. There, information may be represented in a volatile and local manner at runtime, as data flows between the states. RAFCON [7] is based on advanced, hierarchical state machines and has also been used in robot systems in combination with a separate database that acts as a central world model [8].

Such combinations and translations have also been applied to semantic representations and technologies. For example, a PDDL task planner has been combined with dynamic and static OWL ontologies that have been loaded into the OWL API by using a fluent and query interface that maps entities between the planner perspective and the OWL perspective [9]. However, while this has shown some benefits (and current limitations), it still requires additional effort to define the interface each time and some PDDL atoms still remain outside the mapping by the interface. Furthermore, the knowledge itself is still modeled separately in two different representations, PDDL queries have a lower expressivity than OWL 2 [10], and in PDDL no new objects can be created (at runtime) [10]. As this example illustrates, the challenge of *where* and *how* to integrate semantic technologies into robot systems to best leverage their advantages remains.

The contribution of our work in this regard is a focus on investigating the direct and native usage of semantic representations and technologies to minimize the disadvantages entailed by such combinations and translations. This is done via our work on a flexible robotic manufacturing system where abstract processes modeled in OWL ontologies (similarly to flow charts) are applied to and executed in specific workcells by using generalized mechanisms in a *Robot Knowledge Base*. In line with the above, we also mainly use standard OWL reasoning and SPARQL requests instead of, e.g., custom rule languages, to preserve the advantages of using standardized Semantic Web technologies in the first place. These were deliberate choices to investigate this approach, while knowing that they would make it (at first) less suitable, e.g., to task planners due to (initially) lower performance among other reasons, as this is not (yet) a focus of this approach and a known limitation.

For the semantic representations, our approach is based on the PPR paradigm [11], where products, processes, and resources as well as how they relate to each other are formally represented in OWL ontologies so that semantic manufacturing knowledge can be abstracted and reused, with skills tying these models together [12]. Furthermore, a semantic capability model [13] to match production requirements with resource capabilities can enable adaptivity and automatic reconfiguration of a manufacturing system as well as complement the PPR models. In the context of manufacturing systems, a skill can be defined as "a specific realization of a functionality that is provided by a hardware or software component" [14] and acts as an interface, e.g., in the OPC UA middleware, abstracting the underlying implementation.

Other works utilize semantic representations and technologies in the context of automated manufacturing systems as well, but with different focuses. For example, [15] uses a capabilities and skills ontology to bind abstract "capability processes" to available production resources to automatically transform them into executable "skill processes". That approach is based on standard-compliant extensions to an existing BPMN (Business Process Model and Notation) graphical modeling tool and execution engine, which facilitates the integration of production processes with typical IT functionalities, such as user interactions and notifications. However, the "capability processes" are stored as BPMN-compliant XML files instead of being modeled in ontologies and together with likewise missing product ontologies this means that there is no comprehensive semantic world state or relevant context knowledge, e.g., for expressive SPARQL queries to represent conditions in the processes. Other work [16] presents intricate capability models for automatic matchmaking in a semantic context of products, processes, and resources, but the results are only shown as SPARQL queries in the graphical ontology editor Protégé and not integrated into an automated process execution. Likewise, [17] focuses on how PPR ontologies can benefit decision workflows across team boundaries (e.g., product designers and mechanical engineers) by similarly using capabilities and skills. However, the results are again only shown as SPARQL queries in the graphical ontology editor Protégé and as a linear task sequence that does not indicate any branching or dynamic reactions at runtime.

In comparison, as stated previously, our approach focuses on modeling abstract process models in OWL ontologies similarly to flow charts and their automated application. During the process execution of a robotic manufacturing system, they are automatically mapped to the current resource configuration in a specific workcell by generalized mechanisms in a *Robot Knowledge Base*. This is one of the aspects that are intended to enable our overall *Platform Engineering* approach using knowledge augmentation and capability exploration to realize Everything-as-a-Service (XaaS) in the manufacturing domain. This work is an extension of the approach in our previous work [18] with a closer focus on the semantic manufacturing process models and their application (as part of real-world robot systems), which were only briefly described before.

III. Semantic Description Models

The semantic description models are a set of OWL ontologies that formally represent and relate the product to be built, the production process that creates the product, and the manufacturing resources that perform the process, via various classes, properties, and instances. From the manufacturing perspective, they provide all relevant information to enable

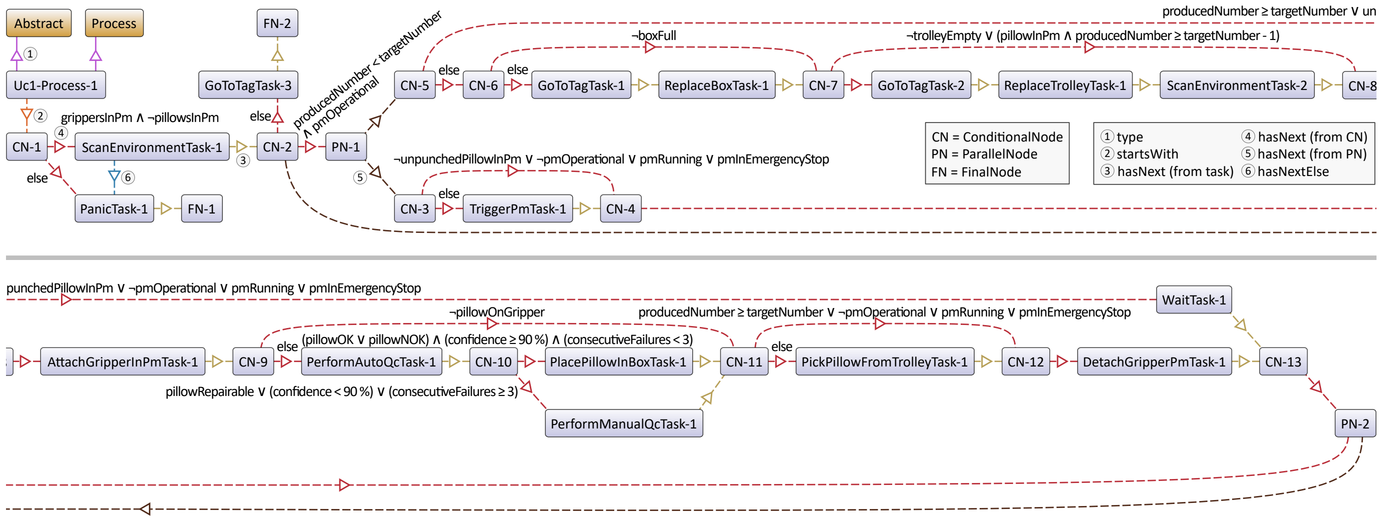

Fig. 1: OWL ontology of an abstract process with tasks, control nodes, and SPARQL-based semantic conditions to manufacture pillows of two types. It is similar to flow charts and begins with *Uc1-Process-1* and ends once *CN-2*'s condition is no longer true. Task parameters and semantic effects are not shown. Only one *hasNextElse* edge is shown, representatively.

the execution of a manufacturing process by modeling it and relevant context knowledge on a task and object level. In this work, such a process model consists of a directed graph of tasks and control nodes such as conditional, parallel, and final nodes (see Figure 1).

This approach, which resembles flow charts, was deliberately chosen, because it appeared to be the simplest one that is feasible when using semantic technologies as natively as possible, while others could be investigated in the future. For example, a task planner approach may require extensive investigation into how to solve performance and other scalability issues with the management of many world states during the search of the state space, since the entire database repository could be viewed as the world state. An advantage of the chosen approach is that the conditions of conditional nodes can be represented semantically as SPARQL queries that have access to the entire semantic world state.

Furthermore, flow charts are well-known to and easily understandable by application domain experts, who may not be robotics or ontology experts, thus potentially facilitating knowledge exchange by centering it around initially informal flow charts. Due to their similar structure, they could then be translated and formalized relatively easily into a semantic

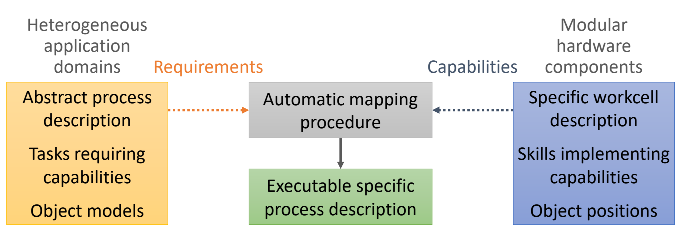

Fig. 2: High-level overview of the inputs and outputs of the automatic mapping procedure in the *Robot Knowledge Base*.

process model. More generally, expert knowledge is often only stored implicitly in the heads of experienced employees or encoded implicitly in source code or a robot's low-level programming. Hence, semantic approaches like the one in this work aim to flexibly model such expert knowledge explicitly in ontologies to make it more easily reusable and maintainable.

In this work, the focus is on the semantic descriptions themselves, which is why they were modeled (semi)manually, for now. Previous works [18], [19] show how GUIs and converter tools can be used to create similar models either manually in an intuitive way or automatically. Additionally, the idea is that, as much as possible, semantic descriptions are only modeled once and then shared and reused.

Semantic manufacturing processes like in Figure 1 are called abstract processes to distinguish them from so-called specific processes that represent one execution or production run in a specific workcell (see also Figure 2). In an abstract process, each task and its parameters (e.g., a robot, tool, or object) can be referenced by multiple tasks and other entities. Additionally, they are also abstract entities, which means that they are essentially placeholders or unbound variables in a template. Thus, an abstract process and each of the abstract tasks and other instances in it can be mapped multiple times (and potentially differently) to the current resource configuration and physical object positions of a specific workcell. This is also how entities within a process model can be deep-linked to and make use of entities within product and resource models.

This enables an abstract process to be flexibly deployed in multiple workcells without needing to adjust it, storing multiple specific processes for logging purposes, and reusing object-level task parameters within a single process, e.g., during multiple iterations of a loop. Multiple abstract instances of the same object type are possible in one abstract process and, due to a form of unique name assumption (UNA) that is

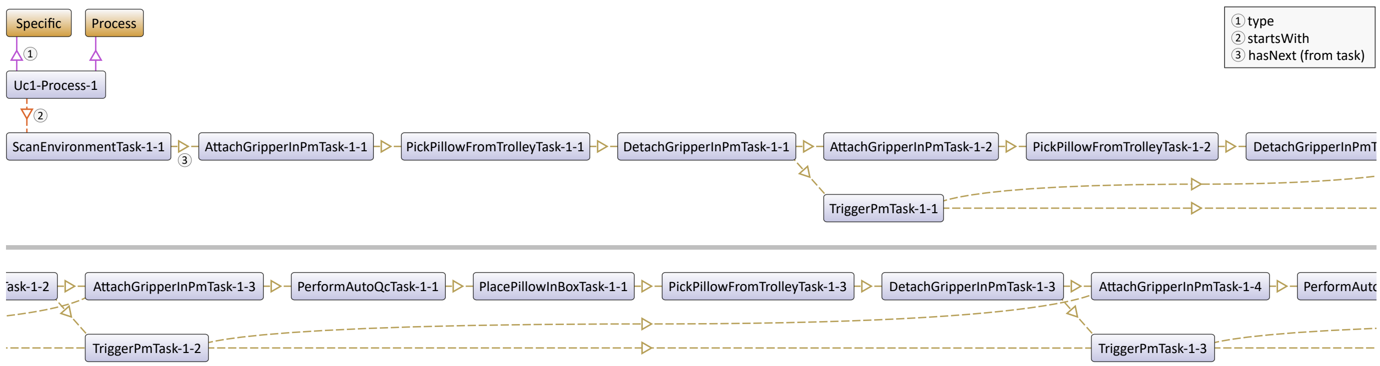

Fig. 3: Excerpt of the OWL ontology of a specific process that was mapped from its abstract process to the specific workcell in Section V and executed on its physical robot system. The local names of the abstract and specific process are the same, but their namespaces differ. The *hasNext* property constitutes a partial ordering where some tasks can be executed in parallel.

made here, they are mapped to different specific instances. If an abstract task parameter instance has already been mapped to a specific robot, tool, or object in the current specific workcell, this mapping is reused by default for succeeding tasks that share this parameter. If no such mapping exists yet, a still unmapped specific instance is found based on the abstract instance's types and properties, which may include, e.g., its parent object in the semantic scene graph or literal values.

To model and manage complex behaviors, such as the switch between and parallel production of the pillow types in Figure 1 (see also Section V) or more generally multiple iterations of a loop, several mapping properties between abstract and specific instances are available. *wasMappedFrom* is the default mapping property and causes a mapping to be reused, *wasMappedFromArchived* causes a mapping not be reused (e.g., so that not the same but the next object on an infeed tray is picked in the next loop iteration), and *wasMappedFromFree* causes a mapping to be done from scratch and only represents that the previous mapping existed (e.g., for logging purposes). Such mapping property assertions can be modified, e.g., by

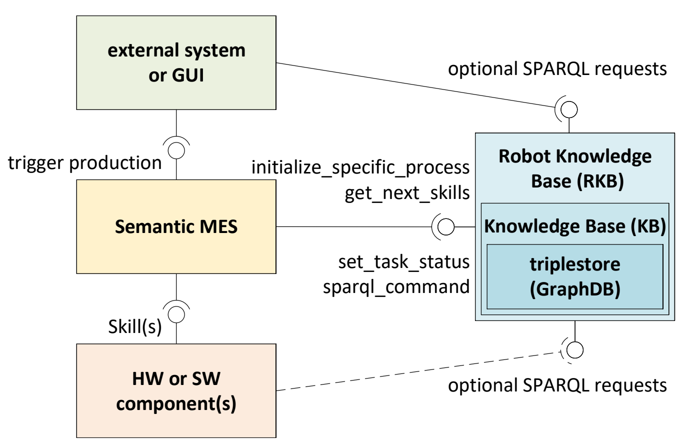

Fig. 4: Overview of a generalized system architecture focusing on the *RKB*. The dotted line indicates that a skill may query the *RKB*, but by default its parameters should suffice.

semantic effects, which are modeled as SPARQL updates, of tasks or control nodes in an abstract process.

The hardware and software abstraction of abstract processes is further enabled by modeling the skills that a resource provides. Taxonomies of skill types can provide standardized interfaces for parameterizing and calling commands of robots and tools from different manufacturers [14]. When mapping an abstract process to a specific workcell, each task is matched to a skill based on the required capabilities of the task and the implemented capabilities of each available skill that is provided by the resources in the workcell. A capability could range from simply its type to, in principle, e.g., a compatible grasp, payload, or gripper span [13], [20]. The actual skill parameter values for invoking it can then be automatically derived from the task parameters and context knowledge. This is modeled in the OWL ontologies as a SPARQL CONSTRUCT query for each skill type that is associated with it, and therefore its subtypes, so that the parameters can be processed and adapted.

Similarly to the skill parameters, the mapping of tasks and task parameters is defined for each task type in the class taxonomy using one or multiple SPARQL updates. They are modeled as so-called SPARQL commands within the OWL ontologies by defining, among other things, an IRI, the request string, and a certain number of parameters [18]. This allows them to be called from inside and outside the *Knowledge Base* in a common manner, e.g., by associating these so-called mapping updates with the task types using the same *hasMappingUpdate* property and giving them the same parameters. This is one of the occasions where OWL reasoners are useful, as they allow to automatically infer whether a SPARQL command applies to a task instance based on whether it was associated with any of its explicit or implicit types.

IV. Automated Process Execution

It is not enough to define semantic models, because they also need to be filled with instances and processed in an application. The *Knowledge Base (KB)* software component was created to persistently store all of this semantic knowledge and provide, e.g., domain-independent general services such as *sparql_command* or *related_sparql_command* [18] to parameterize and execute the SPARQL commands described in Section III. While their mechanism is general, they can be used to call specially-defined SPARQL commands during process execution to, e.g., add new specific object instances detected by a perception component or object statuses by an automatic quality check component.

Figure 4 shows a generalized system architecture diagram with a focus on the *KB*. It uses the RDF4J API to transparently connect to either an embedded or a remote triplestore, in this case *Ontotext GraphDB*. The *KB* is independent of any particular domain such as robotics or manufacturing. The *Robot Knowledge Base (RKB)* is a subclass of the *KB* and provides services useful to the robotics domain like *initialize_specific_process*, *get_next_skills*, and *set_task_status*, but it is still independent of any particular project or use case, i.e., those are modeled semantically in the OWL ontologies without changing its source code. Both the semantic description models, including the skill models for the resources, and the *RKB* are independent of any particular middleware, with OPC UA and ROS1 wrappers having been implemented for use in a larger robot system.

Usually, the *RKB* primarily communicates with a *Semantic MES (sMES)* component that calls the *RKB*'s services. Since the *sMES* works on the skill level, whereas the *RKB* works on the task level, these services translate between the levels and provide the information in an appropriate format. The *sMES* primarily cares about the parameterization, invocation, and results of skills, which is why it does not receive tasks and their parameters, but the skills and their parameters to be invoked, with only the matching task as a reference. For example, when the *RKB* performs the matching between tasks and skills, in some cases there may not be a single matching skill available to directly execute a supertask, but this supertask could still be achieved by matching its subtasks to skills and

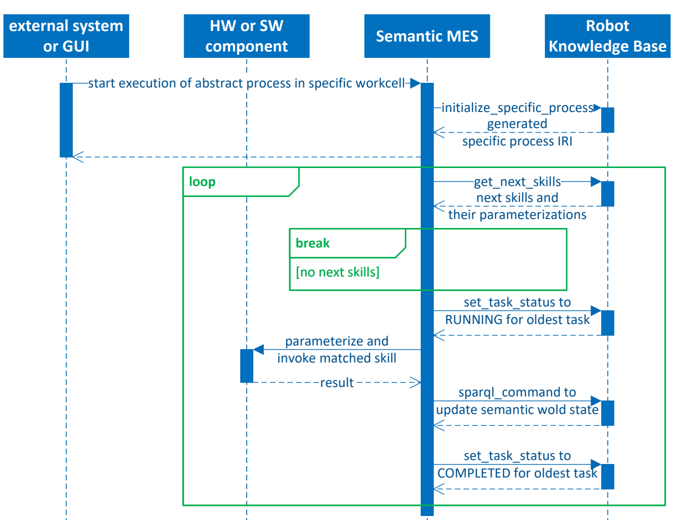

Fig. 5: Simplified sequence diagram of a process execution illustrating interactions between system components over time.

invoking those. Then, the *sMES* never receives the supertask and does not call *set_task_status* to set its status to *running*, as this is handled implicitly by the *RKB* when the *sMES* sets the first subtask's status to *running*. Hence, skill invocations and their parameters are not sent directly from the *RKB* to low-level HW and SW components, which are resources such as robots, tools, and sensors that provide those skills. Instead, the skills and their parameters are received by the *sMES* using a generic interface (that uses the N-Triples syntax), such that new skill types can be added to the OWL ontologies without changing any source code in the *RKB* or *sMES*.

As explained in Section III, an abstract process is mapped to a specific workcell to automatically generate an executable specific process, from which suitable skill invocations and their parameter values can be derived for the given production environment. The *RKB* performs this in the so-called automatic mapping procedure (see Figure 2). It is not performed all at once at the beginning of the process execution, but in the *get_next_skills* service after each task ends to dynamically react with both the mapping updates and the SPARQL-based semantic conditions to events or sensor information at runtime. For example, this can be automatic quality check results that have been sent using the *sparql_command* service (see also Figure 5). Using this service also keeps knowledge in the ontologies, instead of, e.g., putting database request strings into other components' source code.

Specific task instances are generated by the mapping updates associated with the task types as described in Section III. After mapping a task or its parameters, the *RKB* checks whether the mapping was successful, i.e., whether the specific task instance exists and whether it has all parameters implied by its class. Afterwards, the task-to-skill matching is performed. It is considered distinct from the mapping, because, e.g., a *PickAndPlaceTask* could have a robot, gripper, and object as parameters, but the robot may not provide a *PickAndPlaceSkill*. In this case, a composite skill software component could provide a *PickAndPlaceSkill* and the robot, tool, and object identifiers would be its parameters [14]. After a task or control node ends, its SPARQL-based semantic effects are applied (see Section III). They can, e.g., update the symbolic (or geometric) position of an object in the semantic world state or more specifically in its semantic scene graph.

The *RKB* detects whether the mapping and matching have been successful as well as whether a final node has been encountered when no next task has been be found. If not, error handling is performed by following the *hasNextElse* property in the abstract process instead of the default *hasNext* property. This enables the error handling to be modeled semantically on the task level in the ontologies. The *RKB* can also be set to retry the task parameter mapping until it succeeds, e.g., until a perception component detects that a missing object has been placed in the workcell or an empty infeed storage has been refilled. If no skill could be matched to a task, the *RKB* first checks if any subtasks were defined for the task in the abstract process or a so-called task template has been associated with its type, which defines on a semantic level how appropriate

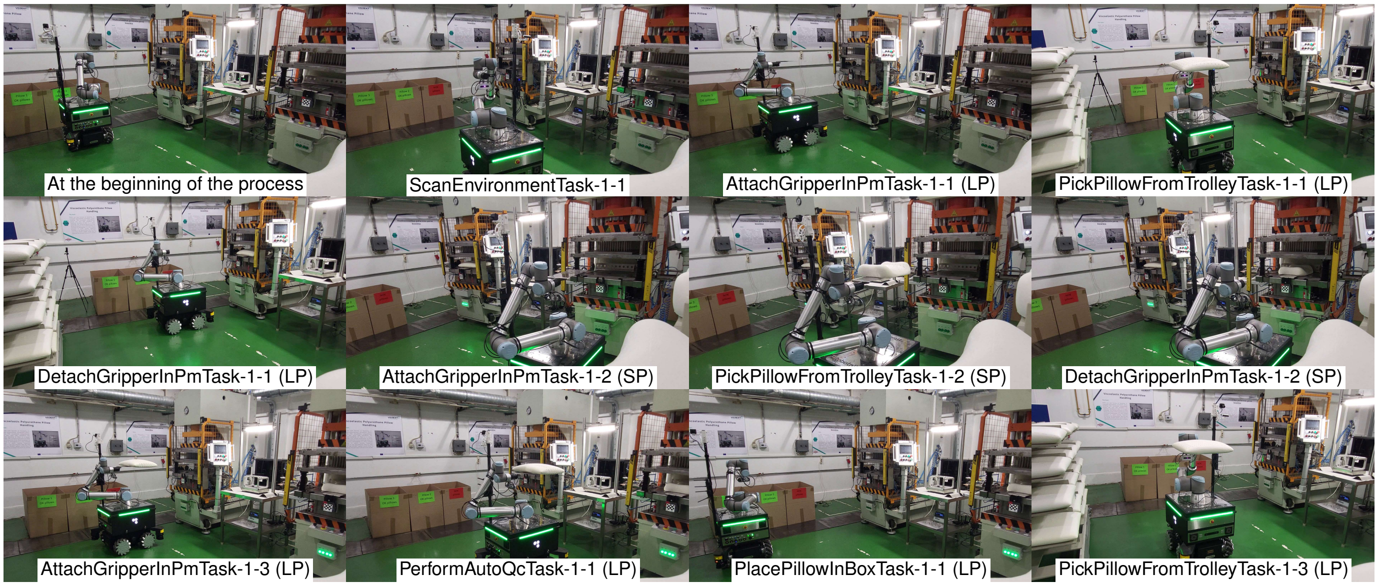

Fig. 6: Physical environment after the first few tasks in a real-world process execution. *(LP)* and *(SP)* refer to tasks with a large or small pillow. *TriggerPmTasks* have little visible change afterwards and are performed in parallel after *DetachGripperInPmTasks*.

subtasks could be automatically generated for it [18].

Figure 3 shows the first few tasks in a specific process ontology that was generated from the abstract process ontology in Figure 1 by mapping it to the specific workcell of the use case in Section V and executing it on the physical robot system. Unlike in an abstract process, the tasks in a specific process constitute a partial ordering in a directed graph without any loops or conditions and only one kind of *hasNext* property. Figure 3 does not show the matched skills as well as the mapped abstract and specific task parameters. Overall, a specific process ontology provides the semantic knowledge necessary at a current point in time during a process execution to derive suitable skill invocations and their parameter values. Afterwards, it is a history log on the semantic level for a specific production run, e.g., to calculate KPIs.

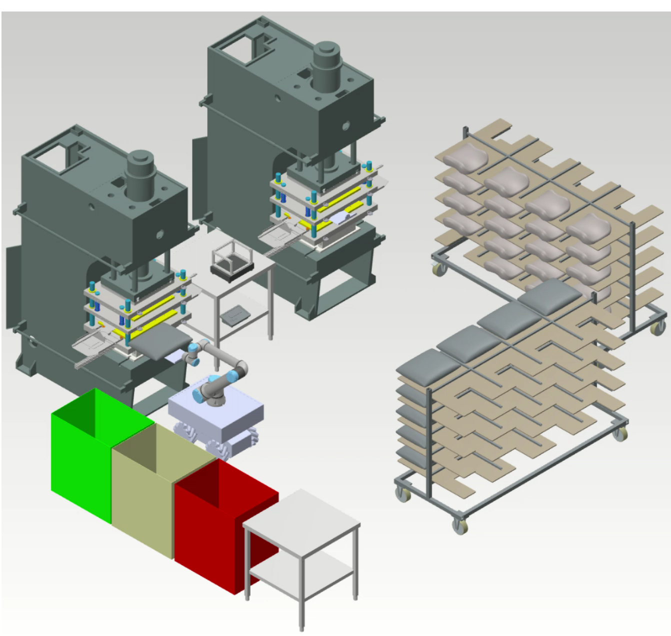

Fig. 7: SPARQL-based visualization of the current semantic world state in the *RKB* after *AttachGripperInPmTask-1-3*.

The simplified sequence diagram in Figure 5 illustrates how the system components in Figure 4 interact with each other during a process execution. First, an external system or a human using a GUI triggers the production run in the *sMES*, which in turn calls the *initialize_specific_process* service of the *RKB* to set up a specific process instance and context based on the given abstract process and specific workcell IRIs. Afterwards, the interaction between the *sMES* and the *RKB* loops until no more next skills can be returned. This consists of the *sMES* calling the *get_next_skills* service, which triggers the *RKB*'s automatic mapping procedure and returns a set of the next skills, including their parameters and matching tasks for reference. For each of these skills, the *sMES* sets the matching task's status to *running*, invokes the skill using the given parameter values, and then waits until the skill returns its result, e.g., *success/error* or an object status from an automatic quality check. Afterwards, *sMES* sets the task status to *completed* or *error* depending on the skill's result. Once the *RKB* returns no more skills to the *sMES*, e.g., because a *ProcessFinalNode* has been encountered, the *sMES* ends its process execution and the *RKB* sets the process's overall status to *completed* (or *error*).

V. Use Case

The semantic description models and the *RKB* have been used in several use cases as part of various robot systems across multiple research projects. In the VOJEXT project, technical partners and end users prepared and discussed relevant information about several use cases to iteratively define scenario specifications that include step-by-step descriptions of the process in plain text as well as the interfaces between components and the skills they provide.

The use case that was chosen to exemplify the approach in this work is about automating the production of foam pillows. Currently, one human worker walks between multiple infeed storages (trolleys), workstations (punching machines), and outfeed storages (boxes). Additionally, they have to quality check each pillow as well as handle edge cases like any of the trolleys being empty or boxes being full. The project partners created together a collaborative mobile robot system with an articulate arm, where the semantic description models and the *RKB* were a higher-level part of this larger prototype demonstrator system. The mobile robot should handle most of the work on its own and only call a human worker when there is an error that it cannot handle on its own or an operation that was not yet feasible to automate like replacing full boxes, turning around half-empty trolleys, or replacing completely empty ones. An online video[1] gives a short overview of the VOJEXT project, a quick look at some of its other use cases, a small introduction to the industrial end user of the chosen use case, and a brief presentation by them about one of the real-world process executions during an integration workshop.

In this use case, the mobile robot needs to make decisions based on multiple data sources, sensors, and external systems such as human gestures, an automatic quality check station, and the punching machines' status (*operational*, *running*, or *inEmergencyStop*). The values of all of these inputs are represented in the semantic description models that are stored in the *RKB*, so that the semantic conditions in the abstract process in Figure 1 can take them into account. Additionally, in this use case, two pillows of different variants are produced at the same time in an interleaved fashion. This was modeled in the abstract process by reusing the same tasks while swapping the subtypes of the abstract task parameter pillow instances in each loop iteration. The selection of the right target box for the *PlacePillowInBoxTasks* was modeled similarly as a semantic effect following the *PerformAutoQcTasks*. It temporarily sets the subtype of the abstract box instance to correspond to good large pillows, good small pillows, or bad pillows of either type. This influences the selection of the specific box instance in the generic mapping update for the *PlaceObject* parameter that is associated with the *PlacePillowInBoxTask* type among others.

The semantic description models and the *RKB* can work on several abstraction levels. In other projects, such as [18], low-level skills like *CartesianLinearMoveSkill* and *GraspGripperSkill* along with geometric coordinates were part of the semantic description models. In the VOJEXT project, the partners requested that the abstraction levels of the system components are such that the OWL ontologies work mostly on a symbolic level with little to no geometric information. The abstract process and semantic conditions in Figure 1 are relatively complex for an introductory example, but were chosen to exemplify the approach in this work, because they cover the behavior of a robot system including several edge cases in a real-world industrial use case. A further example of such edge cases in addition to the previously mentioned ones was encountered during the real-world process executions. A punching machine may become inoperational and then, while the system has seamlessly continued working with only the other punching machine, may become operational again. When this happens, the system behavior differs depending on whether the punching machine is empty, or there is a gripper inside it, or there is still a pillow inside it that has either already been "punched" or not. The representation of this persists in the semantic description models stored in the *RKB* after a punching machine becomes inoperational and is taken into account by the semantic conditions once it becomes operational again.

Figure 6 shows the mobile robot in the physical environment at the facility of the industrial end user after each task during the beginning of a real-world process execution that corresponds to the specific process in Figure 3. *ScanEnvironmentTask-1-1* added new pillow instances to the semantic scene graph. Because the robot cannot physically unload a pillow from the tray-like grippers into a punching machine, *DetachGripperInPmTask-1-1* placed the entire gripper including the pillow in one of the punching machines, which is supported by the semantic scene graph. *TriggerPmTask-1-1* and *AttachGripperInPmTask-1-2* show that while a punching machine was working on the first large pillow, the mobile robot could in parallel already switch production to the other pillow type and pick a small pillow from the second trolley. Later collaborative tasks in the process would be replacing full boxes, replacing empty trolleys, and the manual quality check, which is performed when the confidence level of an automatic quality check result has been too low, a pillow has been determined to be repairable, or there have been too many bad pillows in a row.

During process execution, the ontologies in the *RKB* can be automatically queried using SPARQL not only to enable the execution but also to visualize the current semantic world state. Figure 7 shows a screenshot from a viewer component of a preexisting Web-based application that was adapted to periodically send SPARQL queries to the *RKB* and use their results to visualize the semantic scene graph of the workcell environment. Here, it shows the current semantic world state after *AttachGripperInPmTask-1-3* equivalently to Figure 6. There are still pillows in the trolleys, the boxes are not yet full, a gripper and the first small pillow have been placed in the second punching machine, and the mobile robot is now back at the first punching machine, where it has just attached a gripper and thereby picked the first large pillow back up.

VI. Conclusion

This work presented an approach for semantic PPR models and showed their application in a real-world industrial use case as part of a larger robot system. In particular, the focus was on

[1] https://youtu.be/tyyFJBAa44Q

how to flexibly model semantic manufacturing processes for different use cases in a generalized manner. This also included semantically modeled mechanisms of how to apply these models and a *Robot Knowledge Base* that implemented them, so that suitable skill invocations and their parameter values could be automatically derived during process executions.

A limitation of this work is that, while relevant PPR knowledge may also need to be defined for current industrial solutions or other approaches and intuitive GUIs or automatic data import were part of previous work, they were not within the scope of this work. Performance was sufficient for the encountered use cases, but additional features such as frequent updates to geometric positions and other approaches like task planners would require further investigation. While reusability, scalability, and maintainability were considered during the modeling of the semantic descriptions, these aspect would benefit from further investigation. Additionally, while the similarities between abstract processes and flow charts may make them more intuitive and SPARQL-based semantic conditions etc. appear to be expressive, they currently require ontology experts to write them.

Relevant future work includes improving usability related to the mentioned limitations and linking the semantic knowledge with larger outside systems, as this was one of the main reasons to use Semantic Web technologies in the first place. Domains other than robotics could also benefit from a knowledge-augmented and capability-exploring platform, and such semantic description models and an equivalent of the *Robot Knowledge Base* could be a part of that.

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme (H2020-DT-2019-2) under grant agreement 952197 (VOJEXT).

References

[1] P. Leão and M. M. da Silva, "Impacts of digital transformation on firms' competitive advantages: A systematic literature review," *Strategic Change*, vol. 30, no. 5, pp. 421–441, 2021. [Online]. Available: https://doi.org/10.1002/jsc.2459

[2] P. Brauner, M. Dalibor, M. Jarke, I. Kunze, I. Koren, G. Lakemeyer, M. Liebenberg, J. Michael, J. Pennekamp, C. Quix, B. Rumpe, W. van der Aalst, K. Wehrle, A. Wortmann, and M. Ziefle, "A Computer Science Perspective on Digital Transformation in Production," *ACM Trans. Internet Things*, vol. 3, no. 2, pp. 1–32, 2022. [Online]. Available: https://doi.org/10.1145/3502265

[3] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of Behavior Trees in robotics and AI," *Robotics and Autonomous Systems*, vol. 154, pp. 1–18, 2022. [Online]. Available: https://doi.org/10.1016/j.robot.2022.104096

[4] M. Weck and R. Dammertz, "OPERA – A New Approach to Robot Programming," *CIRP Annals*, vol. 44, no. 1, pp. 389–392, 1995. [Online]. Available: https://doi.org/10.1016/S0007-8506(07)62348-8

[5] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477. [Online]. Available: https://doi.org/10.1109/ICRA.2011.5980391

[6] B. Kast, S. Albrecht, W. Feiten, and J. Zhang, "Bridging the Gap Between Semantics and Control for Industry 4.0 and Autonomous Production," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 780–787. [Online]. Available: https://doi.org/10.1109/COASE.2019.8843174

[7] 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)*, 2016, pp. 3283–3290. [Online]. Available: https://doi.org/10.1109/IROS.2016.7759506

[8] P. Lehner, S. Brunner, A. Dömel, H. Gmeiner, S. Riedel, B. Vodermayer, and A. Wedler, "Mobile manipulation for planetary exploration," in *2018 IEEE Aerospace Conference*, 2018, pp. 1–11. [Online]. Available: https://doi.org/10.1109/AERO.2018.8396726

[9] T. John and P. Koopmann, "Towards Ontology-Mediated Planning with OWL DL Ontologies," in *Proceedings of the 36th International Workshop on Description Logics (DL 2023)*, 2023, pp. 1–14, CEUR-WS.org/Vol-3515. [Online]. Available: https://hdl.handle.net/1871.1/b728c56c-0f75-4e54-aaf4-7b54cd00f182

[10] S. Borgwardt, J. Hoffmann, A. Kovtunova, M. Krötzsch, B. Nebel, and M. Steinmetz, "Expressivity of Planning with Horn Description Logic Ontologies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, 2022, pp. 5503–5511. [Online]. Available: https://doi.org/10.1609/aaai.v36i5.20489

[11] A. Cutting-Decelle, R. Young, J. Michel, R. Grangel, J. L. Cardinal, and J. Bourey, "ISO 15531 MANDATE: A Product-process-resource based Approach for Managing Modularity in Production Management," *Concurrent Engineering*, vol. 15, no. 2, pp. 217–235, 2007. [Online]. Available: https://doi.org/10.1177/1063293X07079329

[12] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for Intelligent Industrial Robots," in *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*, 2012, pp. 1–6. [Online]. Available: https://lup.lub.lu.se/record/4679237

[13] M. Weser, J. Bock, S. Schmitt, A. Perzylo, and K. Evers, "An Ontology-based Metamodel for Capability Descriptions," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 1679–1686. [Online]. Available: https://doi.org/10.1109/ETFA46521.2020.9212104

[14] S. Profanter, A. Perzylo, M. Rickert, and A. Knoll, "A Generic Plug & Produce System Composed of Semantic OPC UA Skills," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 128–141, 2021. [Online]. Available: https://doi.org/10.1109/OJIES.2021.3055461

[15] A. Köcher, L. M. V. Da Silva, and A. Fay, "Modeling and Executing Production Processes with Capabilities and Skills using Ontologies and BPMN," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA52439.2022.9921564

[16] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, "The development of an ontology for describing the capabilities of manufacturing resources," *Journal of Intelligent Manufacturing*, vol. 30, no. 2, pp. 959–978, 2019. [Online]. Available: https://doi.org/10.1007/s10845-018-1427-6

[17] M. Ahmad, B. R. Ferrer, B. Ahmad, D. Vera, J. L. Martinez Lastra, and R. Harrison, "Knowledge-based PPR modelling for assembly automation," *CIRP Journal of Manufacturing Science and Technology*, vol. 21, pp. 33–46, 2018. [Online]. Available: https://doi.org/10.1016/j.cirpj.2018.01.001

[18] A. Perzylo, I. Kessler, S. Profanter, and M. Rickert, "Toward a Knowledge-Based Data Backbone for Seamless Digital Engineering in Smart Factories," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 164–171. [Online]. Available: https://doi.org/10.1109/ETFA46521.2020.9211943

[19] 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 *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2293–2300. [Online]. Available: https://doi.org/10.1109/IROS.2016.7759358

[20] A. Perzylo, J. Grothoff, L. Lucio, M. Weser, S. Malakuti, P. Venet, V. Aravantinos, and T. Deppe, "Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1590–1596, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019. [Online]. Available: https://doi.org/10.1016/j.ifacol.2019.11.427