Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-13 (2019) 1590–1596

Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective

Alexander Perzylo [*] Julian Grothoff [**] Levi Lucio [*]
Michael Weser [***] Somayeh Malakuti [****] Pierre Venet [***]
Vincent Aravantinos [*] Torben Deppe [**]

[*] *fortiss - An-Institut Technische Universität München, Munich, Germany (e-mail: perzylo@fortiss.org)*
[**] *RWTH Aachen University, Chair of Process Control Engineering, Aachen, Germany (e-mail: {j.grothoff,t.deppe}@plt.rwth-aachen.de)*
[***] *KUKA Deutschland GmbH, Augsburg, Germany (e-mail: {michael.weser, pierre.venet}@kuka.com)*
[****] *ABB Corporate Research Center, Ladenburg, Germany (e-mail: somayeh.malakuti@de.abb.com)*

Abstract:
In distributed manufacturing systems, the level of interoperability of hardware and software components depends on the quality and flexibility of their information models. Syntactic descriptions of input and output parameters, e.g., using interface description languages (IDL), are not sufficient when it comes to evaluating whether a manufacturing resource provides the capabilities that are required for performing a particular process step on a product. The semantics of capabilities needs to be explicitly modelled and must be provided together with manufacturing resources. In this paper, we introduce concepts developed by the German BaSys 4.0 initiative dealing with semantically describing manufacturing skills, orchestrating higher-level skills from basic skills, and using them in a cognitive manufacturing framework.

Keywords: Intelligent manufacturing systems, modeling of assembly units, device integration technologies, knowledge modelling and knowledge based systems, ontology-based models interoperability

1. INTRODUCTION

Many manufacturing industries currently experience a shift in production paradigms. Instead of producing high quantities of the same or rather similar products over a long period of time, they have to satisfy the market demand for customized or even individualized products. As a result, their lines of products may have a multitude of different variants, which may only be produced in small lot sizes.

Traditional manufacturing automation solutions cannot cope well with the changed requirements. According to Kinkel and Weißfloch (2009), the total cost of ownership for an industrial robot is mainly driven by operational costs, which are around two thirds of the overall balance compared to a little less than a quarter for the initial investment of buying the robot. They are dominated by labor costs for setting up, reconfiguring, and programming the robot. As a consequence, financial viability cannot be achieved for small batch production.

An emerging approach to tackle this issue in agile production environments is based on modular production resources that can be efficiently (re-)used when designing and adapting manufacturing systems. In this paradigm, production resources offer so-called skills, i.e., parametrizable, executable function blocks that provide predefined functionalities described by associated capability models.

They can be matched with the requirements of a manufacturing process, in order to assign process steps to compatible resources. Such distributed systems need a formal specification of the capabilities of their resources, so that the matchmaking process can be carried out automatically. This additional layer of abstraction allows to seamlessly exchange components in a workcell from different manufacturers, if they provide the same type of skill.

In this paper, we introduce the approaches taken by the German BaSys 4.0 initiative[1] to formally describe and interpret manufacturing skills and their capabilities. First, we discuss how the information that needs to be represented in a skill description can be encoded. Secondly, we introduce a concept that uses semantic annotations of existing information models (e.g., based on the OPC UA for Devices standard) to enrich device models with additional information on their provided skills. Thirdly, we introduce an approach for the automatic orchestration of higher-level skills from a given set of basic skills using our own domain specific language.

[1] https://www.basys40.de/

10.1016/j.ifacol.2019.11.427

Fourthly, an ontology-based approach to integrate the capability paradigm in a cognitive manufacturing system is explained. In this approach, semantic descriptions of manufacturing processes, interaction objects, and manufacturing workcells provide a semantically rich layer of information that can be interpreted in order to map process steps to available resources in a given workcell.

2. RELATED WORK

Epple et al. (2017) show that properties may be used to facilitate a semantic base for the description of skills. The expression semantics of the corresponding property-value statements can be used for capability matching based on offered and required skills. An implementation of property-based capability models can be found in the work of the openAAS project[2], which aims to embed properties in the Industrie 4.0 context (Plattform Industrie 4.0, 2018). Further standardization efforts led to the current DIN SPEC on *Data Exchange on the Base of Property Value Statements* (DIN SPEC 92000).

Malakuti et al. (2018a) describe the main challenges of capability modeling and compare multiple description methods. The authors present a *classification scheme* based on various dimensions. This emphasizes the complexity of clustering different skills and highlights the need for skill matching across different hierarchies. Based on Pfrommer et al. (2014), an *engineering model* is proposed consisting of *product*, *process*, and *resource* properties. In this context, a capability can be fully classified, if it is known which process is carried out on which product and by which resources. Thus, the capability match in the engineering context is based on the description of a requested capability, derived from the process, and offered capabilities that are provided by manufacturing resources. Moreover, the executable skills of an automation system offer their capabilities through a service-oriented architecture. The contribution also compares different description methods based on property models and ontologies.

The semi-automatic tool SFIT is supposed to support production planning engineers in managing product variability. Bayha et al. (2016) have developed a meta-model for describing processes and manufacturing resources with a particular focus on product variants. By continuously evaluating resource capabilities and production constraints, the tool can inform its user about occurring incompatibilities between a particular product variant and the current configuration of the production line.

Perzylo et al. (2016) developed semantic description languages based on the Web Ontology Lanugage (OWL) for creating models of manufacturing processes, objects, and production environments. The semantic process models consist of sequences of tasks that are applied to referenced interaction objects. The environment models define the available manufacturing resources and their physical arrangements, in which the tasks are supposed to be carried out. Geometric information of the product and its parts is also semantically encoded through a boundary representation (BREP) ontology, as described by Perzylo et al. (2015).

[2] https://github.com/acplt/openAAS

A concept that combines OWL ontologies for the semantic description of web services (OWL-S) and OPC UA methods was introduced by Katti et al. (2018). The authors demonstrate the generation of orchestration plans based on loosely coupled production systems that can be interpreted by cloud-located manufacturing execution systems. An overview of various approaches to web service matchmaking, including OWL-S, is given by Klusch (2012).

3. PROPERTY-BASED CAPABILITY DESCRIPTION

In general, a skill provides the (executable) capability of something to cause an effect on something. The effect may be described by the change of property values of the object, i.e., the product. The subject that may cause the effect, also known as resource, may be described by related properties as well. In order to describe the skill itself, more than just an aggregation of these properties is needed. Moreover, the effects have to be classified and possible derived effects should be expressible as well. As a consequence, a dedicated semantic capability description for skills is needed. Semantic capability models have to rely on global standards to provide a common understanding of their concepts and to support interoperability between different devices. Following this approach, three assumptions can be established in order to utilize properties in the context of skill-based engineering.

First, a skill may be described by a set of properties. Thus, a skill can be considered a property carrier. The semantics of each property is only valid in the context of its specific skill. For example, the property *recess depth* has to be defined in the context of each carrier, e.g., drill, laser, hone. If a skill classification hierarchy can be provided, properties may be automatically derived from parent concepts. Otherwise they are unique for a skill.

Secondly, a skill can be characterized by its input, output, and transient conditions. Therefore, the skill's input, output, and transition elements are carriers of properties. In a technical environment, the relevant characteristics of a skill may be described by a corresponding process in the sense of the definition in control technology: "Complete set of interacting operations in a system by which matter, energy or information is transformed, transported or stored" (IEV 60050-351). Consequently, the input, output, and transient elements may be classified also with respect to matter, energy, and information. Changes in these properties may be described by statements to the values of their properties.

Thirdly, a skill may be seen as a property itself, so that it may represent the presence or absence of a skill for a carrier. As a result, other relevant entities in the context of engineering, like processes, products, and resources (Pfrommer et al., 2014), may be described by listing their required or offered skills.

One important goal during the engineering process is the selection of resources or resource types that are suitable to perform a specific process or process step. Depending on the scenario, this selection task can be extended to the challenge of finding a process that transforms one (intermediate) product state into another. Often this challenge is complicated by the lack of machine understandable knowledge and still has to be done manually. The matching of needed and offered skills can be supported by property-based skill descriptions. Following assumption three, products, roles, or resources are matched based on their skills. Two comparison objects have to carry the same skill type for being recognized to be potentially compatible. Not only the skill identifiers themselves have to be compared, but their properties and the properties of their process elements. A majority of comparisons may be completed via statements dealing with scalar property values. They can be automated, if the properties provide machine readable and standardized semantics, which is the goal of the DIN SPEC 92000. As a consequence, full or partial capability checks are possible.

In summary, properties can be utilized in skill descriptions, if they adhere to agreed-upon semantics. For performing a capability check, the carriers as well as their properties have to be classified. The correct and deterministic evaluation of single property comparisons is ensured by linking to standardized property specifications via global identifiers.

4. SEMANTIC ANNOTATIONS OF DEVICE DESCRIPTIONS

A way of providing the capability descriptions introduced in the previous section is to use and extend already existing device descriptions.

Semantic annotations or tags are means to augment an existing piece of information with extra information. This approach can be used, for instance, to describe the semantics of proprietary information based on specific standards; a typical example is to augment a proprietary parameter description with the corresponding parameters defined in eCl@ss or the IEC Common Data Dictionary (CDD).

In our approach, each device has a list of properties, and offers a list of services. The capabilities of the device to perform specific actions are called device skills. Each skill is defined using a set of properties and corresponding services that are a subset of the entire device's properties and services. In order to describe skills in a machine-readable form, we need to define a skill name as well as a set of relevant parameters and services. Given a standardized definition of a skill and its parameters/services, the existing device parameters must be augmented accordingly to specify their correspondence with the standard.

Fig. 1 depicts a generic example of a device description that has been augmented with skill descriptions through semantic annotations. Here, a device description defines a set of device parameters and services. Each skill definition groups a subset of these services and parameters that are relevant to the skill. It is defined by a name and the references to the corresponding device parameters and services, as indicated by the *annotates* relation. If a skill implementation adheres to a particular skill standard, the corresponding skill specification can be linked through the *refers* relation using its uniform resource identifier (URI).

Most of the current device descriptions lack skill descriptions. To flexibly extend the existing device descriptions with skill descriptions, we may consider various options. For instance, a) including the skill descriptions in the asset administration shell (Plattform Industrie 4.0, 2018) of de-

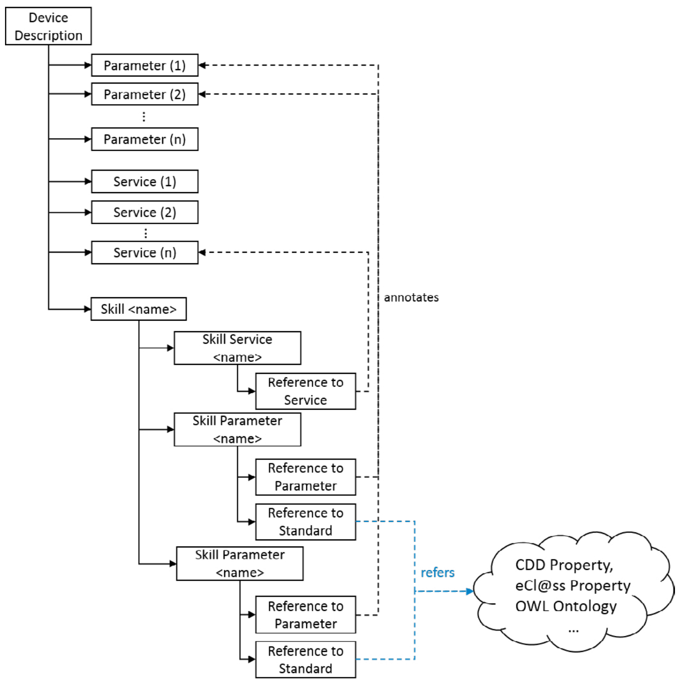


Fig. 1. Skill definition based on semantic annotations of a device description

vices as submodels; b) defining skills using semantic annotations offered by the OPC UA for Devices standard [3] ; and c) adopting XML mechanisms to extend existing XML-based device descriptions using semantic information, as described in Malakuti et al. (2018b).

5. SKILL COMPOSITION USING A DOMAIN SPECIFIC LANGUAGE

Based on a set of device-level capabilities that are provided in a particular workcell, composite capabilities can be orchestrated from basic capabilities in order to match a requested behaviour of the system. For this purpose, we designed an appropriate *Domain-Specific Language* (DSL) to declare those skills on the basis of how they interact with their environment, while abstracting away from the operational implementation details. Synthesis algorithms can then be used for generating the logic that will search for and orchestrate existing skills, in order to achieve the desired operational behaviour.

Domain specific languages promote rapid prototyping of systems by concentrating on defining the syntax and semantics of computer languages, whose terms embody the domain for which a system is being built.

In order to illustrate how the DSL paradigm can be applied to the problem of describing and operationalizing skills, let us introduce a simple example. Assume we wish to describe a robotic arm that is able to extend to its full limit under a certain time threshold and report when the movement has been completed. Assume additionally that we also have a description of the skills of a linear actuator and a sensor to provide the high-level skill of extending the arm until its limit. In this example, the linear actuator may not by itself report whether it has reached its extension limit,

[3] https://github.com/OPCFoundation/UA-Nodeset

```
Skill Extend Arm

Physical State Variables
timeToExtend : time
maxLength : distance
position : distance

Actuators
extend() : (no precondition) -> position = maxLength

Sensors
done_extend() : Reached(position = maxLength) &
    !(Elapsed(InstantOfCall(extend) + timeToExtend))
fail_extend() : !(Reached(position = maxLength)) &
    Elapsed(InstantOfCall(extend) + timeToExtend)
```

Fig. 2. Declaration of the *Extend Arm* skill

```
Skill Sensor

Physical State Variables
objectPresent : boolean

Actuators

Sensors
object_detected() : Reached(objectPresent = true)
object_left() : Reached(objectPresent = false)
```

Fig. 3. Declaration of the *Sensor* skill

but composing it with a sensor that has the skill to detect objects would potentially provide the desired behaviour.

In Fig. 2, the *extend arm* skill is described using a purposefully built DSL. The DSL includes three main blocks:

- *Physical State Variables*, describing the physical dimensions the skill operates on.
- *Actuators*, describing the physical effects on the world as a function of the *physical state variables.*
- *Sensors*, describing the constraints that the system is supposed to respect when operating in the physical world.

Note that the constraints provided in the *Sensors* block of the specification define in a domain-specific and human-readable manner, how the arm should operate. For example, they state that in order for the arm extension to be successful, the *maxLength* (maximum length of the extension) should be achieved (and reported) and the *timeToExtend* (time threshold for the arm extension) should not be exceeded.

While the *extend arm* skill is described in Fig. 2 in a declarative manner, we have designed an algorithm that allows building an orchestrator for it, which can assume the role of an operational controller. The inputs for our algorithm are:

- A set of other skills (which hereon we call *slave* skills), described using the same skill DSL as presented above, that will potentially provide services that can be orchestrated by the *extend arm* skill (which hereon we call the *master* skill). In Fig. 3, we depict the declaration of the *sensor* skill.

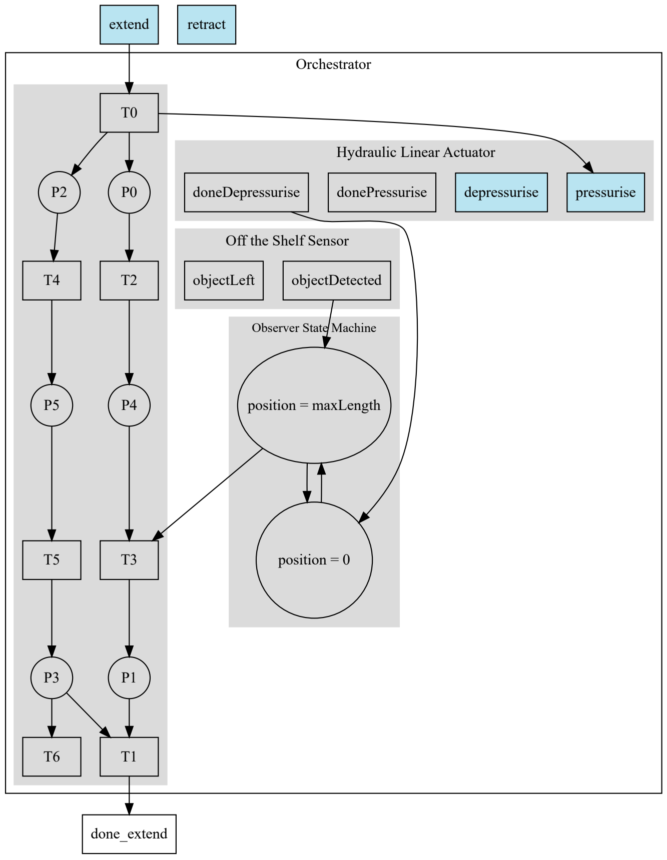


Fig. 4. An automatically synthesized orchestrator

- Information about the semantic mapping between the meaning of certain events in the *master* and in the *slaves*. An exemplary input of such information would be a human answering the question:
  "**Does the sensor detecting an object imply that the arm has been fully extended?**"
  This question has to be answered by a human as there are no means to compute the answer automatically without having a much deeper understanding of the composition of skills, which is exactly the thing we are attempting to build. Nonetheless, the formulation of the questions itself can and has been automated in our tool.

Taking the information above as input, our algorithm performs the matching between the master and slave skills and generates an orchestrator containing the logic that synchronizes the slaves to achieve the skill of the master. Fig. 4 shows a graphical visualization of such an orchestrator. The orchestrator contains the following types of elements:

- The outside border of the main square represents the orchestrator itself and contains *actuator* and *sensor* ports.
- Inside of the main orchestrator, the slave skills are represented as black boxes (where the operational behaviour is abstracted from) again with actuator and sensor ports.
- A state machine that provides an observer to the environment, in which the skill operates (given in this

case by the observation of the sensors of the slave skills).

- A set of Petri nets that orchestrate the operation of the master skill. Each Petri net corresponds to the operationalization of a behaviour of the master skill (e.g., *done* or *fail*). It is responsible for coordinating the messages coming from and to the environment of the master skill, from and to the slave skills and synchronizing with the observer state machine.

The complete description of the syntax and semantics of the skill DSL is beyond the scope of this paper. Also, while several publicly available and competent DSL workbenches exist, we have experimented with our skill expression and interpretation ideas using the popular EMF framework, which is distributed with the Eclipse software development environment.

6. ONTOLOGY-BASED INTEGRATION LAYER FOR COGNITIVE MANUFACTURING SYSTEMS

In order to make use of semantic capability models, they must be integrated in a semantic manufacturing framework that can process and interpret models of manufacturing processes and their requirements, as well as workcells and their offered capabilities.

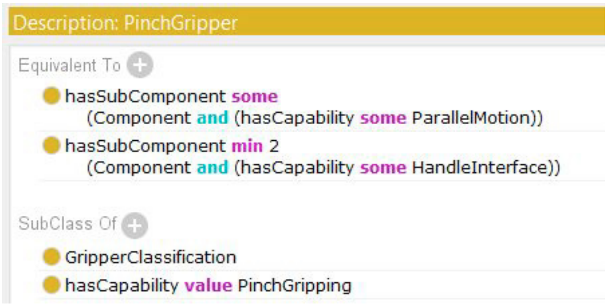
We use the Web Ontology Language[4] (OWL) as a formal language to specify our own semantic description languages for encoding the required models. The effort of explicitly representing such knowledge helps to separate knowledge from code, and as a result the created models can be easily understood, maintained, shared, and reused.

Semantic description languages specify a common vocabulary for specific domains, defining concepts and relations between instances of these concepts. Hence, they match the requirements to model the concept of capabilities and associated relationships. The advantage of a capability ontology is its generic symbolic representation. It is based on a logical formalism that allows to integrate and combine knowledge, and to perform automatic reasoning through logical inference. This enables the system to combine capabilities with other concepts, e.g., processes and their individual steps, or interaction objects. Implicit dependencies to capabilities can be automatically inferred for a given situation by the underlying logic of a capability ontology.

6.1 Inferring capabilities from component taxonomies

Related to section 5, the orchestration of capabilities can be part of a cognitive manufacturing system. The following example was developed for the domain of industrial grippers. A capability ontology was combined with component and geometry models in order to infer higher-order capabilities based on component hierarchies.

A pinch gripper can grasp something, if one of its subcomponents can execute a *ParallelMotion* and two of its subcomponents offer the capability *HandleInterface*. This assumption was modeled in an OWL ontology. Fig. 5 shows the corresponding axioms for a generic *PinchGripper* class in the Protégé ontology editor. A class defining a particular gripper comprised of a gripper body and two gripper fingers is modeled in Fig. 6. The gripper body and fingers further have the mentioned capabilities asserted through the *hasCapability* object properties.


Description: PinchGripper

Equivalent To

● hasSubComponent some
(Component and (hasCapability some ParallelMotion))

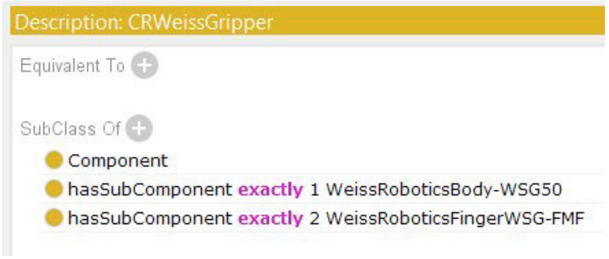
● hasSubComponent min 2
(Component and (hasCapability some HandleInterface))

SubClass Of

● GripperClassification

● hasCapability value PinchGripping

Fig. 5. View of the Protégé editor showing axioms that specify an abstract *PinchGripper* based on the presence of the capabilities *ParallelMotion* and *HandleInterface* in its subcomponents


Description: CRWeissGripper

Equivalent To

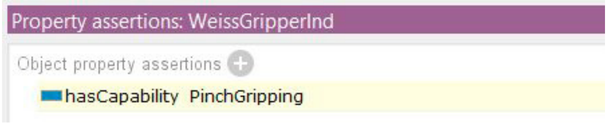
SubClass Of

● Component

● hasSubComponent exactly 1 WeissRoboticsBody-WSG50

● hasSubComponent exactly 2 WeissRoboticsFingerWSG-FMF

Fig. 6. View of the Protégé editor showing the sub component axioms of a particular pinch gripper class called *CRWeissGripper*


Property assertions: WeissGripperInd

Object property assertions

▬ hasCapability PinchGripping

Fig. 7. View of the Protégé editor showing the inferred object property assertions for a particular instance of the *CRWeissGripper* class called *WeissGripperInd*. The capability *PinchGripping* was automatically inferred.

In conjunction with the capability model, component hierarchy, and relations between these concepts, an OWL reasoner is able to infer the higher-order capability of *PinchGripping* for the given instance of the *CRWeissGripper* class, as indicated by the yellow background of the listed property assertion in Fig. 7.

6.2 Matching task requirements with workcell capabilities

The process of picking an object contains the sub task of grasping the object. First, let us assume that a human is supposed to grasp a screw. The intuitive answer to the question if the human can grasp a screw would be *yes*. However, to answer the same question for a particular robot is not that intuitive anymore. Humans have access to a variety of models about their environment gained from years of experience in interacting with it. They also excel in deriving new models from their accumulated knowledge.

Despite the multitude of available screws and different ways of performing a grasping action, humans share a

[4] http://www.w3.org/TR/owl2-primer/

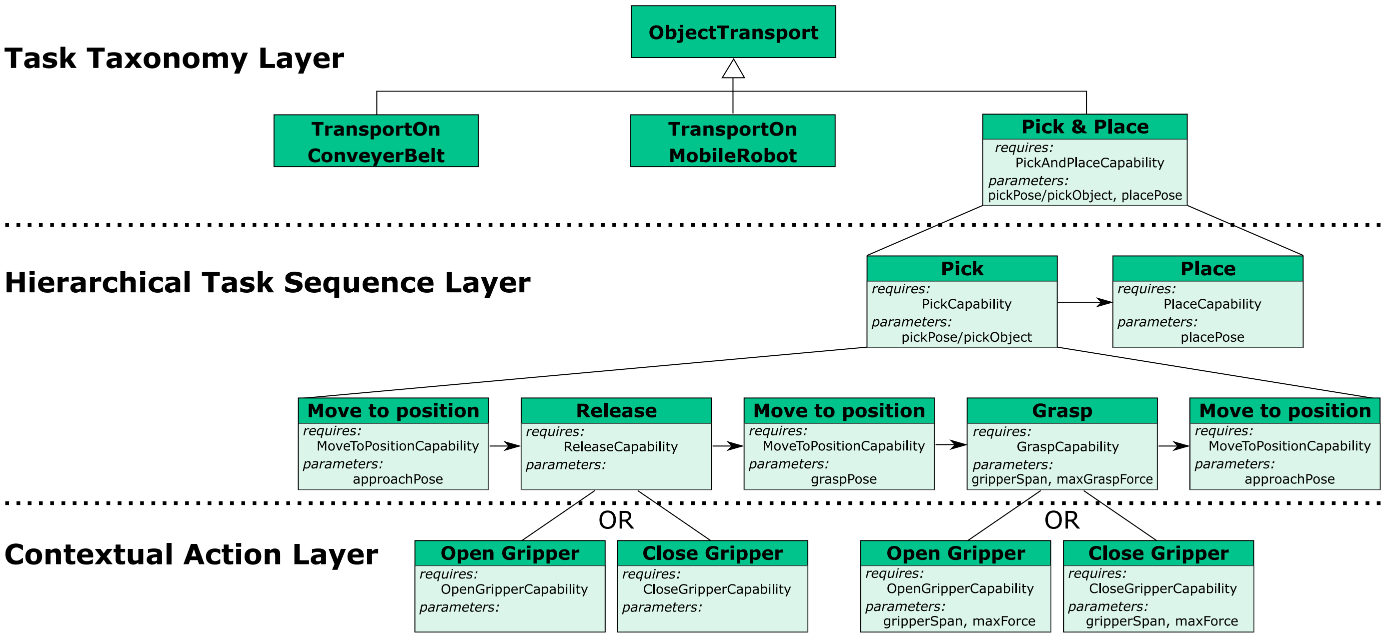

Task Taxonomy Layer
ObjectTransport
TransportOn ConveyerBelt
TransportOn MobileRobot
Pick & Place
requires: PickAndPlaceCapability
parameters: pickPose/pickObject, placePose
Hierarchical Task Sequence Layer
Pick
requires: PickCapability
parameters: pickPose/pickObject
Place
requires: PlaceCapability
parameters: placePose
Move to position
requires: MoveToPositionCapability
parameters: approachPose
Release
requires: ReleaseCapability
parameters:
Move to position
requires: MoveToPositionCapability
parameters: graspPose
Grasp
requires: GraspCapability
parameters: gripperSpan, maxGraspForce
Move to position
requires: MoveToPositionCapability
parameters: approachPose
OR
OR
Contextual Action Layer
Open Gripper
requires: OpenGripperCapability
parameters:
Close Gripper
requires: CloseGripperCapability
parameters:
Open Gripper
requires: OpenGripperCapability
parameters: gripperSpan, maxForce
Close Gripper
requires: CloseGripperCapability
parameters: gripperSpan, maxForce

Fig. 8. Visualization of an excerpt of a semantic description of a pick and place task with annotated required capabilities and parameters. All tasks are described in a taxonomy and are further specified through a sequence of subtasks. The black arrows denote a *precedes* relation for defining a pairwise partial ordering between the subtasks. The actions that are eventually performed might depend on the task's context, e.g., when a pipe can be grasped from the inside and the outside.

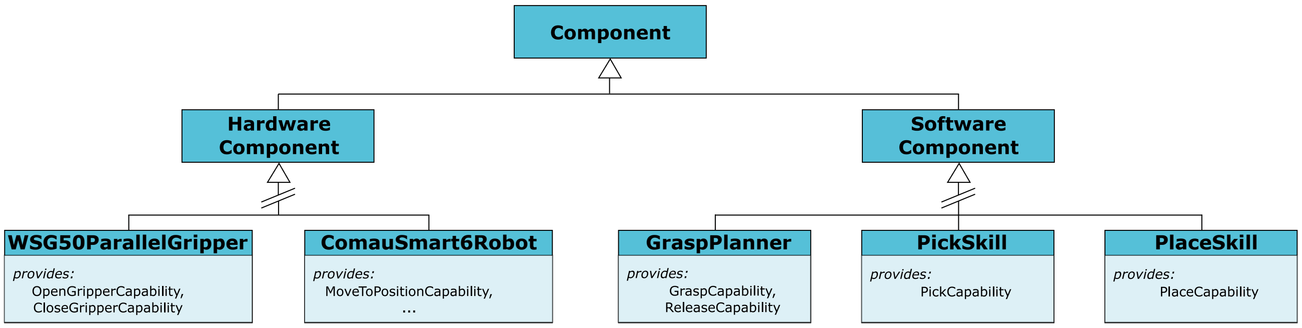

Component
Hardware Component
Software Component
WSG50ParallelGripper
provides: OpenGripperCapability, CloseGripperCapability
ComauSmart6Robot
provides: MoveToPositionCapability, ...
GraspPlanner
provides: GraspCapability, ReleaseCapability
PickSkill
provides: PickCapability
PlaceSkill
provides: PlaceCapability

Fig. 9. Visualization of an excerpt of a component taxonomy that can be used in semantic workcell models. It features a Comau Smart6 robot arm, a WSG50 parallel gripper, as well as related software skills with their provided capabilities.

common understanding of screws and the implications of grasping them. When humans evaluate whether or not they would be able to grasp an object they have never touched before, they try to relate all relevant aspects of the task to known concepts. The evaluation of their grasping capabilities will be based on individual assessments of, e.g., the material of the object, its dimensions, assumed weight, and surface properties, as well as their strength, reach, and hand span. In our approach, we endow manufacturing systems with means to evaluate their capabilities in a similar way.

The task of planning an action is getting more complicated if a process step requires a robot to pick up a particular M6 cap head socket screw with a length of 20 mm. Now, an instance of a screw and the environment in which the action shall be performed have to be considered. In order to perform such a task, additional models related to the environment, perception, planning, analytical algorithms, and optimizations are required.

Fig. 8 shows a visualization of a semantic process model for a hierarchically defined pick and place process. The process consists of two sub tasks (pick, place) that are partially ordered through a *precedes* relation. The pick task is further refined as a sequence of basic actions that are again annotated with a pairwise partial ordering. Each of the basic actions requires specific capabilities to be present on manufacturing resources that are to be considered for performing the actions.

An excerpt of a component taxonomy is visualized in Fig. 9. The modelled components are used as manufacturing resources in semantic workcell models. The contained resources are instances of concepts that have been defined in the taxonomy. They provide capabilities that have been modelled in a capability ontology and can be matched with the requirements of the pick and place process from Fig. 8.

A logical examination of these capabilities can never guarantee that a specific task will be successfully performed by a manufacturing resource's skill. It merely is an indicator for guaranteed incompatibility in case of logical inconsistency. Due to the logical interpretation, reasoning components can identify the reasons for such an incompatibility, which can be used to generate human-understandable feedback to the operator of the manufacturing system. For instance, an incompatibility of a pick and place process and a robot workcell could be resolved by adding a more suitable gripper to the workcell. Whether a given grasp pose can be used to successfully grasp the object might not be decided on a symbolic level alone and require sub-symbolic evaluations, e.g., using a physics simulator. In this case, the sub-symbolic evaluation can be used to increase the manufacturing system's level of confidence that an action will succeed, while the symbolic pre-evaluation can be used to reduce the search space. In order to emphasize the separation of concepts, we use the term *capability* for symbolic/logical and *feasibility* for mathematical/analytical compatibility matching.

7. CONCLUSION

Cognitive manufacturing systems can leverage semantic capability descriptions of manufacturing resources to flexibly and efficiently design and (re-)configure production processes, thus saving time and money. The additional layer of abstraction enables process engineers to focus on developing and describing manufacturing processes independent from actual machines or software components. The mapping of abstract specifications of process steps (and their requirements) to executable code eventually happens when a process model is deployed to a workcell or production line and can be computed automatically.

The real-world applicability of this approach strongly depends on the depth of the capability models. Modeling all relevant aspects of complex process steps and related manufacturing skills is far from a trivial task. Even seemingly simple tasks such as robot motions require a substantial modeling effort, when velocities, accelerations, forces, or deviations from trajectories must be considered, e.g., when being used for robot-based grinding or polishing.

The benefits of capability-based systems engineering can be increased by the development of standardized sets of skills. The standards must define a common understanding of the skills of a particular domain that can be shared by resource providers and their customers' manufacturing execution systems. The German Mechanical Engineering Industry Association[5] (VDMA) works towards closing this gap through their OPC UA companion specifications, which have already been released in first versions for domains such as robotics or integrated assembly solutions.

ACKNOWLEDGEMENTS

The research leading to these results has been partially funded by the German Federal Ministry of Education and Research (BMBF) under grant agreement no. 01IS16022 through the project BaSys 4.0 (Basissystem Industrie 4.0).

[5] https://www.vdma.org/en/

REFERENCES

Bayha, A., Lúcio, L., Aravantinos, V., Miyamoto, K., and Igna, G. (2016). Factory product lines: Tackling the compatibility problem. In *International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, 57–64. Salvador, Brazil.

DIN SPEC 92000 (2018). Data exchange on the base of property value statements.

Epple, U., Mertens, M., Palm, F., and Azarmipour, M. (2017). Using properties as a semantic base for interoperability. *IEEE Transactions on Industrial Informatics*, 13(6), 3411–3419.

IEV 60050-351 (2009). International electrotechnical vocabulary - Part 351: Control technology.

Katti, B., Plociennik, C., and Schweitzer, M. (2018). SemOPC-UA: Introducing semantics to OPC-UA application specific methods. *IFAC-PapersOnLine*, 51, 1230–1236.

Kinkel, S. and Weißfloch, U. (2009). Estimation of the future user potential of innovative robot technologies in SMEs – Promising prospects. In *World Robotics 2009 Industrial Robots: Statistics, Market Analysis, Forecasts, Case Studies and Profitability of Robot Investment*, 376–381. VDMA.

Klusch, M. (2012). Overview of the S3 contest: Performance evaluation of semantic service matchmakers. In *Semantic Web Services, Advancement through Evaluation*, 17–34.

Malakuti, S., Bock, J., Weser, M., Venet, P., Zimmermann, P., Wiegand, M., Grothoff, J., Wagner, C., and Bayha, A. (2018a). Challenges in skill-based engineering of industrial automation systems. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 67–74. Torino, Italy.

Malakuti, S., Schmitt, J., and Gamer, T. (2018b). From Heterogeneity to Uniformity in Building Automation Systems via Semantic-based Engineering. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 115–122. Torino, Italy.

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

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

Pfrommer, J., Stogl, D., Aleksandrov, K., Schubert, V., and Hein, B. (2014). Modelling and orchestration of service-based manufacturing systems via skills. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–4. Barcelona, Spain.

Plattform Industrie 4.0 (2018). Details of the Asset Administration Shell, Part1 - The exchange of information between partners in the value chain of Industrie 4.0. Federal Ministry for Economic Affairs and Energy. Berlin, Germany. URL `https://www.plattform-i40.de`.