Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track - Short paper

# Flexible Robotic Assembly Based on Ontological Representation of Tasks, Skills, and Resources

**Philipp Matthias Schäfer[1], Franz Steinmetz[2], Stefan Schneyer[2], Timo Bachmann[2], Thomas Eiband[2], Florian Samuel Lay[2], Abhishek Padalkar[2], Christoph Sürig[2], Freek Stulp[2], Korbinian Nottensteiner [2]**
[1]Institute of Data Science, German Aerospace Center (DLR)
[2]Institute of Robotics and Mechatronics, German Aerospace Center (DLR)
p.schaefer@dlr.de

**Abstract**

Technology has sufficiently matured to enable, in principle, flexible and autonomous robotic assembly systems. However, in practice, it requires making all the relevant (implicit) knowledge that system engineers and workers have – about products to be assembled, tasks to be performed, as well as robots and their skills – available to the system explicitly. Only then can the planning and execution components of a robotic assembly pipeline communicate with each other in the same language and solve tasks autonomously without human intervention. This is why we have developed the Factory of the Future (FoF) ontology. At its core, this ontology models the tasks that are necessary to assemble a product and the robotic skills that can be employed to complete said tasks. The FoF ontology is based on existing standards. We started with theoretical considerations and iteratively adapted it based on practical experience gained from incorporating more and more components required for automated planning and assembly. Furthermore, we propose tools to extend the ontology for specific scenarios with knowledge about parts, robots, tools, and skills from various sources. The resulting scenario ontology serves us as world model for the robotic systems and other components of the assembly process. A central runtime interface to this world model provides fast and easy access to the knowledge during execution. In this work, we also show the integration of a graphical user front-end, an assembly planner, a workspace reconfigurator, and more components of the assembly pipeline that all communicate with the help of the FoF ontology. Overall, our integration of the FoF ontology with the other components of a robotic assembly pipeline shows that using an ontology is a practical method to establish a common language and understanding between the involved components.

## 1 Introduction

Flexible and autonomous robotic assembly systems promise to make the production of small lot sizes more affordable. To automate robotic assembly efficiently, all components of that robotic assembly pipeline need to communicate in a common formal language. With robots and their interfaces slowly becoming standardized as well as digital twins and knowledge about processes becoming further advanced, communication between the components is in principle feasible. However, a lot of knowledge – about products to be assembled, tasks to be performed, and robots with their different configurations and skills – only exists implicitly, for example in the form of manually written code or as knowledge of system engineers. Only if this knowledge is made formal and explicit, a computer can reason about it and flexible robotic assembly can be done autonomously. A formal model enables all components within the robotic assembly pipeline – from workcell configuration over planning to execution – to communicate with each other in the same language and to have a common understanding of the world.

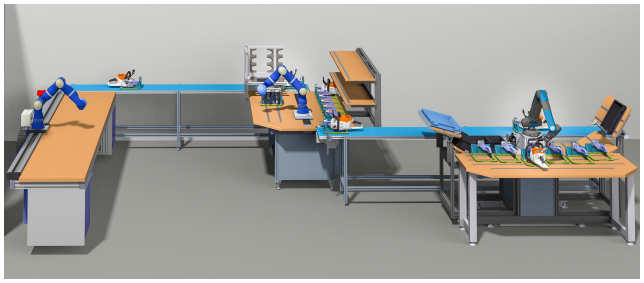

Figure 1: A rendering from the CAD model of our Factory of the Future lab, used to automate the assembly of a chain saw.

As such a common formal model, our first contribution is the *Factory of the Future (*FoF) ontology (published in OWL format with documentation (Schäfer et al. 2021)) that brings all aspects of the assembly process together based on the existing standard *IEEE Standard Ontologies for Robotics and Automation (*IEEE1872). At its core, this ontology models the tasks that are necessary to assemble a product (e. g. connections) and – as counterpart – the robotic skills with their preconditions and effects that can be employed for their completion. We use the FoF ontology as a basis for scenario specific ontologies, that model concrete workcells, skills, tasks, and artifacts. As our second contribution, we demonstrate how a scenario ontology is filled from various sources, with its contents being generated predominantly automated. This scenario ontology then serves as the initial content of a knowledge base that supplies all components of a robotic assembly pipeline with the required information. Our third contribution is the *World Interface*, a component offering fast and abstract access to the knowledge base at runtime through multiple protocols. It is connected with all components of our robotic assembly pipeline, such as the front-end for graphical task definition, the planner for skill sequence generation, and a reconfigurator for workcell layout optimization.

As a development goal, the knowledge base formed from the FoF and scenario ontologies shall provide a common source for reasoning in the overall pipeline. In this paper, we present our recent developments and activities towards that vision. At first, we present related ontologies in Section 2, followed by a description of our ontology in Section 3 and its application in the robotic assembly pipeline in Section 4. The approach is discussed in Section 5. We finish with conclusions and a brief outlook on future work in 6.

## 2 Related Work

In recent years, multiple initiatives were started to formalize knowledge in the robotic domain. The standard IEEE1872 provides a very basic ontology for robotics and automation (IEE 2015). While being very general, it stays abstract in many aspects and the level of detail is insufficient to apply it in a realistic scenario. At least two study groups are currently working on further standards based on IEEE1872, one for tasks (Balakirsky et al. 2017) and one for autonomous robotics (Olszewska et al. 2017).

(Beßler et al. 2018) presents assembly planning through reasoning over an ontology. The approach proofs to be very efficient in solving complex assembly tasks. Nevertheless, the overall production environment with multiple workcells is not represented.

(Jacobsson 2015; Jacobsson et al. 2016; Järvenpää et al. 2018; Perzylo et al. 2019) cover different aspects of modeling skills and capabilities of robotic devices and served as inspiration for FoF ontology's skill component. In particular, we chose a comparable approach in the representation of skills and expanded up on it.

(Wildgrube et al. 2019) presents a semantic model for connections between objects through so called semantic mates, similar to FoF ontology's connectable elements in the connection component.

(Perzylo et al. 2015) presents, similar to this paper, the use of ontologies for semantic representation of much of the robotic process and its environment (*ubiquitous semantics*). While the focus lies on object detection and pose estimation, task execution, and human-friendly interfaces, the paper does not go into detail how the system components communicate with each other.

## 3 Ontology

The FoF ontology is a domain ontology (Section 3.1). For specific assembly scenarios we create scenario ontologies (Section 3.2) based on the FoF ontology. These serve as a basis for the application ontology, axioms created during runtime of the robotic assembly pipeline (Section 4).

We base FoF on the ontologies of IEEE1872 (IEE 2015), which themselves are based on the *Suggested Upper Merged Ontology (*SUMO) (Niles and Pease 2001). SUMO is an upper ontology. IEEE1872 extends SUMO with the ontology CORAX and defines its core ontology, *Core Ontology for Robotics & Automation (*CORA), on top of them. It also defines POS for spatial information and RPARTS for parts of robots. To specify units of parameter values, we use the

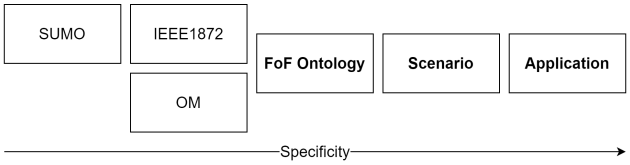


Figure 2: Ontology Hierarchy. Each box represents an ontology, building on all ontologies to the left of it. Ontologies written in bold are described in this paper.

*Ontology of units of Measure (*OM)[1]. Fig. 2 shows the hierarchy of the involved ontologies.

### 3.1 FoF Ontology

As a basis for the FoF ontology, we define robotic assembly to be the process of assembling artifacts by robotic workcells completing tasks through their skills. In our terminology, a task defines a goal that can be fulfilled by executing a sequence of skills. We split the ontology into multiple components, which are briefly introduced in the following paragraphs.

To represent parts and products as well as devices (e. g. robots) of the assembly process, we use subclasses of `Design` and `Artifact`, defined by the CORAX, respectively the SUMO ontology. We differentiate between representations of object types and representations of individual objects. Furthermore, individuals can be defined to be abstract (name prepended with an underscore) to refer to any individual of that type, i. e. to serve as a placeholder.

We developed the *connection* component to represent connections between artifacts. As with artifacts and designs, there are two levels of classes and properties: one for the abstract types of connections and one for concrete connections between artifacts.

We annotate artifacts in general and storage devices in particular with classes and properties from the *grasp set* component and *place set* component, respectively. The former models knowledge about how grippers can grasp artifacts, while the latter models knowledge about where and how we can place artifacts in a storage device.

The *workcell* component models robotic workcells. A robotic workcell is an environment, in which robotic systems assemble artifacts with the help of other devices. Storage devices, located in robotic workcells, hold the resources of an assembly process, i. e. artifacts that may colloquially be called parts, products, and tools.

With the classes and properties of the *state* component, we represent snapshots of the state of one or more robotic workcells. A state consists of arbitrary individuals that are part of or exist within a robotic workcell. Because of this, there might be multiple individuals within the knowledge base representing the same physical object in different states. With state constraints, we added classes that resemble statements of predicate logic about the individuals comprising a state. They are used to express preconditions, goals, and effects of tasks and skill.

Assembly tasks for robotic workcells are represented by classes and properties of the *task* component. Tasks are de-

[1] https://github.com/HajoRijgersberg/OM/

fined by preconditions and goals through state constraints. We group tasks into task containers, which are intended to be assigned to a robotic workcell. There are two versions of tasks and task containers. The plain versions may reference abstract artifacts in their preconditions and goals and serve as templates (e. g. `Task`). From these so called runnable versions can be created (e. g. `RunnableTask`). Those have to reference concrete artifacts and are assigned to robotic workcells for execution.

We represent skills of robotic systems through classes and properties of the *skill* component. Skills are defined by preconditions and effects through state constraints. There are three versions of skills. Plain skills represent abstract skills of robotic systems, parameterized skills are skills parameterized for solving a specific task, and runnable parameterized skills are skills that are fully specified for a given runnable task and associated robotic workcell. We group parameterized skills into skill sequences, each skill sequence being able to solve a task.

The *parameter* component provides classes and properties to represent skill parameters of various types. A parameter type specifies, among other things, its value type. The value of a parameter is only assigned for the parameterized and runnable versions of a skill.

In addition to the robotic assembly related components, we extended the POS ontology of IEEE1872 with data properties for concrete coordinate values through the *posx* component and added properties for sequences (`first*`/`next*`) in the *sequence* component.

### 3.2 Scenario Ontology

The FoF ontology defines classes and properties required for representing any robotic assembly scenario. Based on this domain ontology, for each scenario a specific scenario ontology can be described by subclasses, properties, individuals, and property axioms. In particular, artifacts, workcells, and their skills must be represented.

We fill our scenario ontologies from various sources of existing knowledge, which is converted into an *Web Ontology Language (*OWL) representation. Representations of parts, products, and devices (e. g. robots) are converted from an existing object database (Leidner et al. 2012). Further information about these artifacts, including connections, place sets, and how they form workcells, are extracted using a custom script from the FoF *computer-aided design (*CAD) model of the lab shown in Fig. 1. Information about skills is converted from data from RAFCON (Brunner et al. 2016), a *graphical user interface (*GUI) application for the development of skills using hierarchical state machines. Also, a certain degree of manual editing is still involved.

Such a scenario ontology, together with the ontologies it is based on, forms the initial knowledge base at runtime of a robotic assembly pipeline.

## 4 Robotic Assembly Pipeline

In the following, we describe the robotic assembly pipeline components' use of the FoF ontology. It serves three purposes: it (i) models the knowledge about the assembly environment and process, (ii) serves as the basis for the knowledge base that centrally holds the information about a concrete scenario, and (iii) defines how to share information between pipeline components.

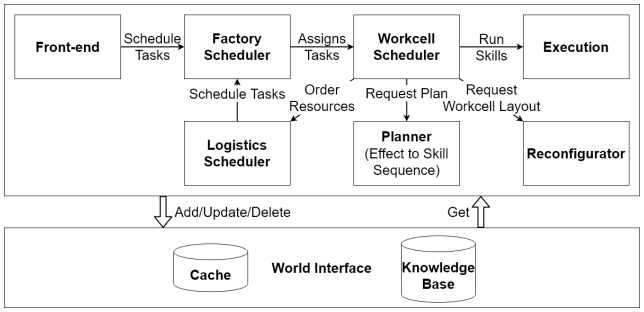


Figure 3: Architecture of the Robotic Assembly Pipeline

In Fig. 3, the communication architecture of the robotic assembly pipeline is schematized. All components read from, add to, or update the knowledge base. World Interface described in Section 4.1 serves as the central interface. When components communicate with each other about individuals (e. g. to schedule a task), they reference them using their *Internationalized Resource Identifier (*IRI).

After the description of World Interface, the front-end (Section 4.2), planner (Section 4.3), and reconfigurator (Section 4.4) components are show-cased. The *factory scheduler* forwards tasks to the different *workcell schedulers*, which, for example, order artifacts required for a task from the *logistics scheduler* and run a generated plan using the Execution.

### 4.1 World Interface

Typical triplestores, used for holding ontological data, usually offer the query language *SPARQL Protocol and RDF Query Language (*SPARQL) as an interface. The result of those queries is a plain table, which makes the access to hierarchical data difficult. As an example for hierarchical structures in the FoF ontology, consider task container sequences that contain a list of task containers which themselves each contain a list of tasks.

To mitigate this and further issues, we developed World Interface using a patched version of the *Object-RDF Mapping (*ORM) library SuRF[2]. World Interface serves as the central runtime interface to the knowledge base. It offers an *application programming interface (*API) for the basic *create, read, update, and delete (*CRUD) operations, which can be accessed using different protocols. Currently, WebSocket and *OPC Unified Architecture (*OPC UA) are supported, an HTTP-based *Representational state transfer (*REST) interface is planned.

World Interface parses the ontology and dynamically creates Python classes with attributes corresponding to the ontology's classes and their properties. For example, the property `hasGoal` with domain `Task` becomes an attribute `hasGoal` of the `Task` class. As a special case, sequences, which are represented using first/next properties in the ontology, are converted into lists.

[2]https://github.com/franzlst/surfrdf

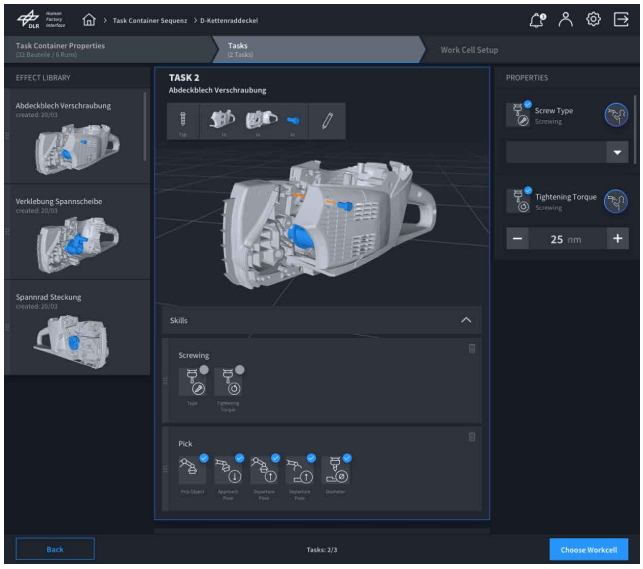


Figure 4: Illustrative image of of the HFI front-end. It shows a task defined by an effect (a connection) and a skill sequence that has been calculated to fulfill the task.

For communication between the pipeline's components and World Interface, the generated class instances are JSON-serialized. When a read operation is performed (e. g. a request for certain individual), instances of the Python classes get recursively populated with the information from the ontology. If, for example, a certain `TaskContainer` is requested, an on-the-fly created instance of the Python class `TaskContainer` is returned, with the `tasks` attribute filled with corresponding `Task` instances, each again having all attributes set. Likewise, when adding or updating entities, World Interface takes instances of the Python classes as input and converts them into triples.

As this process is computational heavy, entities are cached. Changes are only written back to the ontology on command.

### 4.2 Front-end

The front-end is a GUI (see Fig. 4 for a screenshot) to the assembly process in form of a web application. Users can intuitively define tasks in terms of their goals. They can then assign these tasks to a robotic workcell, which triggers a message to the execution component with the task's IRI. The front-end also provides information about the progress of each task, periodically querying World Interface for new information. In addition, it allows to control the execution.

### 4.3 Planner

The purpose of our planner is to create a skill sequence that solves a given task (Sürig 2021). It translates all necessary entities from the ontology into a *Planning Domain Definition Language (*PDDL) domain and problem representation. A skill in the ontology, for example, is translated into an action in PDDL. Given the PDDL representation, a PDDL planner generates a sequence of parameterized actions (a plan) that solves the problem. Subsequently, the plan is converted back to an ontological representation in the form of a skill sequence consisting of parameterized skills.

### 4.4 Reconfigurator

The reconfigurator allows for automated planning of an optimal workcell layout. It optimizes the location of fixtures for skill sequences with fast execution times, while meeting reachability and collision constraints. The approach is based on minimizing path lengths (Bachmann et al. 2021) and obtains all required information from the ontology.

## 5 Discussion

We showed that IEEE1872 is a useful basis for a robot ontology in practice. We expanded upon it with knowledge of the robotic assembly domain.

The resulting FoF ontology is not only based on theoretical considerations and related literature, but has been iteratively improved by practical needs during the integration of the robotic assembly pipeline. A common theme of our adaptations was the need to differentiate between knowledge available at different points of a pipeline's runtime.

We needed to differentiate between abstract types of artifacts, the properties of which where representable before runtime, and concrete instances of types, only some of which are representable before runtime (e. g. robots but not concrete products). The CORAX ontology provides us with `Design` as a class for artifact types, for which we implemented analogues in the FoF ontology's connection component. Another example are skills of robots that are known before runtime. They need to be parameterized for particular robots and parts that should act and be acted upon, respectively, which can only be done at runtime. Furthermore, there is a need to distinguish between abstract robot skills, partially parameterized skills after planning, and fully parameterized skills for execution. In the FoF ontology, we represent this with individuals of three different classes: `Skill`, `ParamterizedSkill`, and `RunnableParameterizedSkill`.

Despite being sufficient for our current applications, the FoF ontology is limited regarding modeling the current state. For example regarding connections, no kinematic constraints or kinematic chains can be modeled. Furthermore, the current concepts of connections are limited in their semantic expressiveness, e. g. regarding geometrical alignment.

## 6 Conclusion

The FoF ontology serves as a common language for the components of robotic assembly systems and grounds the implementation of their interfaces. It allows representing complex process chains and their resources consistently in multiple phases of robotic assembly processes, from planning to execution. The FoF ontology models the preconditions and goals of tasks as well as preconditions and effects of skills. This can be used to verify skill sequences generated automatically by an assembly planner. We plan to investigate further applications and reasoning methods using the ontology in future work. The FoF ontology, as currently published, is restricted to robotic agents. We plan to extend it with entities for modeling human-robot collaboration and to include a broader set of manufacturing methods.

## Acknowledgments

The authors would like to thank the overall HFI team for the discussions, in particular, Ismael Rodriguez, Christoph Willibald and Markus Knauer for their contributions towards improving the ontology.

The work described in this paper was partially funded by the DLR-internal project "Factory of the Future", by the Bavarian Research Institute for Digital Transformation (bidt) and the European Commission through the project H2020 AnDy (GA no. 731540).

## References

Bachmann, T.; Nottensteiner, K.; and Roa, M. A. 2021. Automated planning of workcell layouts considering task sequences. In *Int. Conf. Robotics and Automation (ICRA)*. IEEE.

Balakirsky, S.; Schlenoff, C.; Fiorini, S. R.; Redfield, S.; Barreto, M.; Nakawala, H.; Carbonera, J. L.; Soldatova, L.; Bermejo-Alonso, J.; Maikore, F.; Goncalves, P. J. S.; Momi, E. D.; Kumar, V. R. S.; and Haidegger, T. 2017. Towards a robot task ontology standard. In *Volume 3: Manufacturing Equipment and Systems*. ASME.

Beßler, D.; Pomarlan, M.; and Beetz, M. 2018. Owl-enabled assembly planning for robotic agents. In *Proc. 17th Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*.

Brunner, S. G.; Steinmetz, F.; Belder, R.; and Dömel, A. 2016. RAFCON: A graphical tool for engineering complex, robotic tasks. In *Int. Conf. Intelligent Robots and Systems*, 3283–3290.

2015. IEEE standard ontologies for robotics and automation. *IEEE Std 1872-2015* 1–60.

Jacobsson, L.; Malec, J.; and Nilsson, K. 2016. Modularization of skill ontologies for industrial robots. In *Proc. 47st Int. Symp. Robotics (ISR)*, 1–6.

Jacobsson, L. 2015. A Module-Based Skill Ontology for Industrial Robots. Master's thesis, Lund University.

Järvenpää, E.; Lanz, M.; and Siltala, N. 2018. Formal resource and capability models supporting re-use of manufacturing resources. *Procedia Manufacturing* 19:87–94.

Leidner, D.; Borst, C.; and Hirzinger, G. 2012. Things are made for what they are: Solving manipulation tasks by using functional object classes. In *Int. Conf. Humanoid Robots (HUMANOIDS)*, 429–435. IEEE.

Niles, I., and Pease, A. 2001. Towards a standard upper ontology. In *Proc. Int. Conf. Formal Ontology in Information Systems (FOIS)*.

Olszewska, J. I.; Barreto, M.; Bermejo-Alonso, J.; Carbonera, J.; Chibani, A.; Fiorini, S.; Goncalves, P.; Habib, M.; Khamis, A.; Olivares, A.; de Freitas, E. P.; Prestes, E.; Ragavan, S. V.; Redfield, S.; Sanz, R.; Spencer, B.; and Li, H. 2017. Ontology for autonomous robotics. In *26th Int. Symp. Robot and Human Interactive Communication (RO-MAN)*. IEEE.

Perzylo, A.; Somani, N.; Profanter, S.; Gaschler, A.; Griffiths, S.; Rickert, M.; and Knoll, A. 2015. Ubiquitous semantics: Representing and exploiting knowledge,geometry, and language for cognitive robot systems. In *Proc. Workshop Towards Intelligent Social Robots - Current Advances in Cognitive Robotics, IEEE/RAS Int. Conf. Humanoid Robots (HUMANOIDS),*.

Perzylo, A.; Grothoff, J.; Lúcio, L.; Weser, M.; Malakuti, S.; Venet, P.; Aravantinos, V.; and Deppe, T. 2019. Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective. In *Proc. IFAC Conf. Manufacturing Modeling, Management, and Control (MIM)*.

Schäfer, P. M.; Schneyer, S.; Bachmann, T.; Knauer, M.; Steinmetz, F.; and Nottensteiner, K. 2021. Factory of the future ontology. https://doi.org/10.5281/zenodo.4650308.

Sürig, C. 2021. Achieving autonomous task execution by integrating a combination of ontologiesand semantic languages intoa human factory interface. Master's thesis, Technical University of Munich.

Wildgrube, F.; Perzylo, A.; Rickert, M.; and Knoll, A. 2019. Semantic mates: Intuitive geometric constraints for efficient assembly specifications. In *Int. Conf. Intelligent Robots and Systems (IROS)*. IEEE.