electronics

MDPI

Article

ORPP—An Ontology for Skill-Based Robotic Process Planning in Agile Manufacturing

Congyu Zhang Sprenger [1,*], Juan Antonio Corrales Ramón [2,*] and Norman Urs Baier [1]

[1] Institute of I3S, School of Engineering and Computer Science, Bern University of Applied Sciences, 3400 Burgdorf, Switzerland; norman.baier@bfh.ch
[2] Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain
* Correspondence: congyu.zhangsprenger@bfh.ch (C.Z.S.); juanantonio.corrales@usc.es (J.A.C.R.)

Abstract: Ontology plays a significant role in AI (Artificial Intelligence) and robotics by providing structured data, reasoning, action understanding, context awareness, knowledge transfer, and semantic learning. The structured framework created by the ontology for knowledge representation is crucial for enabling intelligent behavior in robots. This paper provides a state-of-the-art analysis on the existing ontology approaches and at the same time consolidates the terms in the robotic task planning domain. The major gap identified in the literature is the need to bridge higher-level robotic process management and lower-level robotic control. This gap makes it difficult for operators/non-robotic experts to integrate robots into their production processes as well as evaluate key performance indicators (KPI) of the processes. To fill the gap, the authors propose an ontology for skill-based robotics process planning (ORPP). ORPP not only provides a standardization in the robotic process planning in the agile manufacturing domain but also enables non-robotic experts to design and plan their production processes using an intuitive Process-Task-Skill-Primitive structure to control low-level robotic actions. On the performance level, this structure provides traceability of the KPIs down to the robot control level.

Keywords: ontology; robot process; robot task; robot skill; KPI

Citation: Zhang Sprenger, C.; Corrales Ramón, J.A.; Baier, N.U. ORPP—An Ontology for Skill-Based Robotic Process Planning in Agile Manufacturing. *Electronics* **2024**, *13*, 3666. https://doi.org/10.3390/electronics13183666

Academic Editor: Xin Geng

Received: 15 August 2024
Revised: 10 September 2024
Accepted: 11 September 2024
Published: 14 September 2024

1. Introduction

Robots and robotic manipulators play a role of fundamental importance in the manufacturing industry, particularly for complex but repetitive tasks. However, the success of industrial robotics to date has depended strictly on controlled, static environments [1]. Therefore, the industry requires advancements in robotic systems to reduce programming efforts and to adapt to changing environments due to the new requirements of agile production. The field of cognitive robotics deals with the development of robot systems equipped with awareness of their capacities and limitations in changing environments and agile production [2].

Ontology-based knowledge representation has been studied for its functionality in achieving robust autonomous robotic systems. When such a system is equipped with knowledge representation, the cognitive skills provided to the robot allow it to perform tasks, reason the knowledge base, and interact with various environments autonomously [3]. Ontology methods provide formalized concepts, which may then be used to construct a knowledge base using classes, axioms, and relationships. Thus, a shared domain vocabulary may be defined, making both collaboration and reusability simpler tasks [4]. Characteristics of ontology include formalization and standardization. Ontology provides clear structure and accurate data through which knowledge sharing and reusing among heterogeneous systems is made more convenient [5].

This research is supported by the European Horizon 2020 ACROBA project [6]. This project aims to develop an AI-driven cognitive robot platform that can be easily and flexibly

adapted to different agile manufacturing scenarios. There are five industrial pilot lines in this project. The research must support the realization of the five industrial pilot lines. Therefore, these industrial manufacturing processes provide not only requirements and inputs for the research work but also targets for the validation process.

IEEE has developed the standard ontologies for robotics and automation [7]. This standard consists of a core ontology for robotics and automation (CORA) and other ontologies to support CORA. CORA covers the fundamental concepts, their relations, and axioms in this domain. CORA is aligned with the Suggested Upper Merged Ontology (SUMO), which is an upper-level ontology that provides definitions of basic ontological concepts for all domains to use as a foundation [8]. To enable ontology reuse, it is recommended to build upon existing ontologies and standards. However, the detailed concepts regarding robotic processes are not defined in this ontology.

Through the literature review, it was found that although the use of ontology for task-planning in robotics applications has been researched, its implementation in industry has yet to see major advancements. In the literature, the use of terms and concepts in robot process area are not strictly regulated, and there is a need to provide a standardized use of terms to avoid misuse, misunderstanding, and miscommunicating of concepts in this area. KPIs are used to evaluate requirements' fulfillment. They are usually defined at the process level. However, in the robotics area, "task" is often the central focus instead of "process". If measurements and controls are designed on the robotic task level while KPIs for requirements are set at the process level, the traceability of the requirements cannot be easily tracked to the robot control level. Therefore, the link between process to task needs to be defined.

For the aforementioned reasons, the authors have developed a more specific domain ontology based on CORA to the robotic process domain. This ontology provides additional knowledge for the "process" and "task planning" concepts. It defines the terms and their relationships in this domain as well as provides the link from KPIs (high-level process management layer) to robot actions (low-level robot control layer). With this link, the KPIs defined for the process can be calculated automatically by the robotic system based on robot actions. Pedersen et al. have presented a robot skill model that is well accepted in the robotics industry [9]. It contains three abstract layers: task, skill, and primitive. This model has helped in determining the terms for domain ontology.

In the following sections, a problem description is presented first with a systematic literature review of the state of the art, and gaps in the current research are identified. In the research method section, the ontology development process is described, and a new ontology is proposed. Then, there is prototyping and the applications of the ontologies in the ACROBA project [6] are described. Finally, conclusions are drawn, and an outlook for future research is provided.

2. Literature Review and Problem Description

2.1. Literature Review

Planning complex robotic processes in a real-world environment requires sophisticated domain knowledge, which can be acquired from ontology-based knowledge representation and reasoning techniques. Using knowledge representation provides cognitive skills, which enable robotic platforms to perform tasks, make decisions, and interact with various environments [3]. The benefits of implementing ontologies for robotic task planning are investigated, and to identify any gaps in this field, two research questions are raised:

- Is there an existing ontology for robotic process planning that covers the whole industrial manufacturing domain—light-out manufacturing and collaborative manufacturing?
- Is there any traceability of high-level process KPIs to low-level robotic control?

Research related to ontology-based robotics task planning covers different robotic application areas such as human-robot collaboration, autonomous robotics, and industrial manufacturing. There are two papers [10,11] that thoroughly reviewed the ontologies in

the robotics domain. They provided us with good foundations and contributed a lot to our literature review. The authors will touch on each area in the following paragraphs.

2.1.1. Human-Robot Collaboration

A large amount of research has been conducted on the use of ontologies for Human–Robot Collaboration [12]. OCRA is an Ontology for Collaborative Robotics and Adaptation. This ontology focuses on collaboration and plan adaptation. When executing industrial robotic tasks, the robot collaborates with humans and is able to decide what to do based on the adaptation plans with the support of the ontology [13]. However, the validation of OCRA is limited to one scenario of a human and a robot collaboratively filling a tray [13]. Sharework Ontology for Human–Robot Collaboration (SOHO) describes an ontology for Human–Robot Collaboration that aims to flexibly represent the collaborative production process [14]. The main drawback is that SOHO is not developed for realistic collaborative industrial scenarios. Umbrico et al. [15] propose an extension of SOHO. An experimental evaluation was carried out on the developed representation and reasoning technology, justifying its efficacy. However, this work focused solely on Human–Robot Collaboration scenarios and has yet to be developed further. A knowledge storage and reasoning framework, Ontologenius [16], allows to modify the knowledge base during execution, and it maintains these changes even when powered off. This system has been tested in human-related scenarios.

2.1.2. Autonomous Robotics

Extensive research work for ontologies can also be found in autonomous robotics. ORO provides a framework for knowledge storage and reasoning. In this framework, robots are enabled with cognitive skills to perform different tasks such as task planning and replanning and collaboration with robots and humans in a complex environment [10]. The autonomous robot architecture ontology (ROA) [17] aims to provide a conceptual framework allowing people and robots to share and exchange information about robot architectures. The conceptualization is similar to the meta-models of concept representation languages such as the Unified Modelling Language (UML) [18]. Yuguchi et al. [19] explore the usage of ontologies for home service robots to equip them with the ability to perform tasks using home appliances. However, the framework could only execute sequential tasks to control home appliances. Robot Task Planning Ontology (RTPO) is an indoor service-oriented robot task planning ontology [5]. Based on this ontology, the authors developed a task planning algorithm that is used for robots to autonomously assign high-level tasks [5]. Beetz et al. introduced KnowRob2 [20], based on its predecessor, KnowRob [21]—which is an advanced knowledge processing system that accomplished the performance of a multitude of complicated manipulation tasks such as making a pizza, conducting chemical experiments, and setting tables. The aim of KnowRob2 was to improve the capabilities of robotic agents to acquire open-ended manipulation skills and reasoning when faced with realistic manipulation actions. KnowRob2 was proposed as a query-answering system, where queries are connected to tasks and refer to images captured by the agent and its motions. Although KnowRob2 is advanced, it has not yet been linked to any industrial scenarios.

2.1.3. Industrial Manufacturing

Manufacturing does not have many ontology-based task planning-related research. Weser et al. presented an ontology-based metamodel for manufacturing capabilities for production resources. Their paper describes a hierarchical ontology architecture that formally defines capabilities as abilities of manufacturing resources for different domains and use cases. However, this paper did not touch on the abstraction of robot skills [22]. In the German Basys 4.0 Initiative, Perzylo et al. also pointed out that capabilities are important and formally modelling them enables the interoperability of hardware and software. In this paper, the concept of using manufacturing skills is detailed [23]. The

Factory of the Future ontology (FoF) [24] attempted to model assembly tasks needed for product assembly as well as the robotic skills needed to perform these tasks. Although the results of this project indicated that the use of ontology is practical in establishing a common understanding between all involved components, the FoF ontology is limited in its ability to model the current state [24]. The authors pointed out that in the future, they aim to extend FoF for HRC and more manufacturing methods. ROSETTA [22] attempts to characterize capabilities of manufacturing resources using a formal definition of capabilities. It implemented a set of ontologies of robot skills aiming to create intelligent support for reconfiguration and adaptation of manufacturing robotic cells. This paper indicated that the formal representation of ontologies is suitable for encoding knowledge, and ontology models are easily maintained and reused. Saxena et al. [25] present a knowledge engine RoboBrain, which may learn and share knowledge representations such that robots may carry out various tasks. However, RoboBrain is still in the experimental stage.

2.2. Problem Definition

The findings are summarized as follows:

- There is no existing ontology for robotic process planning that covers the entire manufacturing process. Most of the ontologies are focused on the service robotics domain aiming to achieve robot autonomy.
- There is an increasing amount of effort that has been put into Human–Robot Collaboration (HRC). However, in the industrial robotics area, the amount of work is less extensive.
- No key performance indicators (KPI) were included for robotic process planning and control because the research focus was mainly on the robot control level, which is the closest to the robot. Process on the other hand is at the top. It goes down to task, and then the task can be performed by robot.
- The research focus is either tool-centric or product-centric.

The literature review indicates clearly that while several attempts at developing ontology-based knowledge frameworks for robotic systems have been made, none of these have been developed to a state advanced enough to be adopted for and used to fully operate industrial processes. The use of ontology has been explored extensively for situations involving human–robot interactions. This is mainly because a level of ambiguity exists between human instructions and what a machine understands. While some papers did focus on ontology-based task planning for robotics in industrial systems, these projects were experimental and require further work. Therefore, it can be concluded that ontology-based task planning for robotic industrial systems is a field that requires further work to develop a robust system, which could be adopted in industry.

What caught the authors' attention is that there is hardly any work that describes the robot process level. Process level refers to the level where objectives are set and KPIs are measured for performance evaluation. It is different from the task level where the focus is on robot actions. This fact creates a big gap between performance management and robot control. At the process level, KPIs like cycle time, accident rate, product quality, etc. need to be calculated and measured for management reporting. When process management is separated from robot control, it is hard to evaluate the performance of the processes, and it is especially difficult to know how to reach a better KPI level by optimizing the lower-level robot performance.

The research in this field has two focuses: tool-centric (a manufacturing task is described based on available hardware and software components) and product-centric (describing the product and associated production steps independent from specific production resources). To profit from both approaches, not only do robot skills need to be implemented but also the capability of the production resources needs to be modelled [22].

3. Methods—Ontology Development

In the past years, researchers have proposed various kinds of ontology development approaches. This paper follows a combined approach that considers both the SABiO ontology development approach [26] and the hybrid framework used by [27]. The overview is shown in Figure 1. It is created based on [26,27].

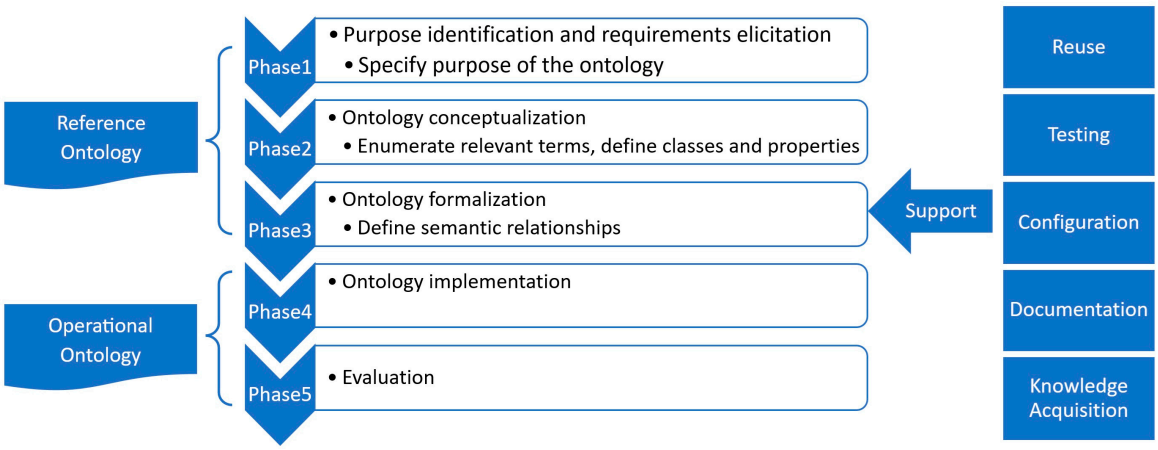


Figure 1. Combined ontology development methodology.

In this merged approach, there are five main phases that represent the main development process. The first three phases are used to develop a reference ontology (described in Section 3), and the last two phases are used to develop an operational ontology (described in Section 4). Reuse, testing, configuration, documentation, and knowledge acquisition describe the supporting process in ontology development. The SABiO method differentiates the reference ontology and operational ontology and divides the ontology development processes into main and supporting processes [26]. It provides guidance for these processes. The hybrid framework [27] has very similar phases to the SABiO method, but it provides more detailed structures for the different phases. For instance, in Phase 2, it recommends enumerating relevant terms, defining classes and their properties. The reason to combine these two is to profit from the advantages of both methods.

The reference ontology and operational ontology are described in SABiO as follows:

"A reference ontology is a special kind of conceptual model. An operational ontology, in turn, is a machine-readable implementation version of the ontology, designed with the focus on guaranteeing some desirable computational properties" [26].

Therefore, the first three phases are used to develop a reference ontology, and the last two phases are used together with the first three to develop an operational ontology. The details of the reference ontology development are described in the following sections. Section 3.1 focuses on identifying requirements of the ontology, ontology conceptualization, and ontology formalization.

3.1. Ontology Purpose Identification and Requirements Elicitation (Phase 1)

The goal of this step is to identify the purpose of the ontology and its intended uses. This step is crucial as it generates the requirements that are the prerequisite for constructing an ontology. This step involves the ontology engineer, domain experts, and potential ontology users. Figure 2 shows the overview of this phase, and it is taken from [26].

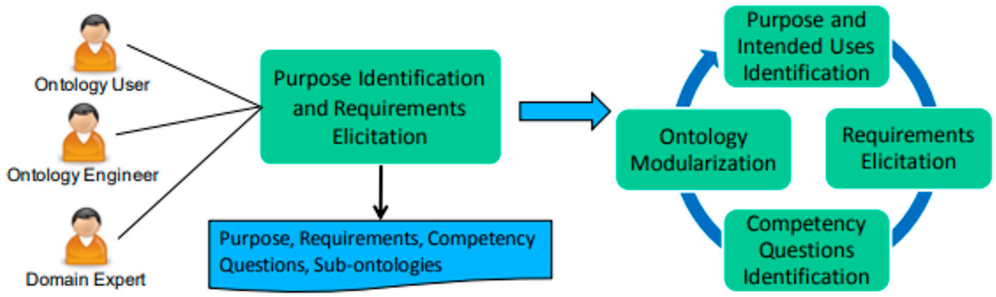


Figure 2. Ontology purpose identification and requirements elicitation (Source: [26]).

3.1.1. Competency Questions

To determine the scope of the ontology, we need to find out what is relevant and what is not by using competency questions (CQs). These CQs capture the functional requirements for the ontology. We use the top-down strategy for forming CQs. The complex questions are asked first and then they are decomposed into simpler ones. In Table 1, a set of competency questions are listed.

Table 1. Competency questions and answers.

CQ Nr	Competency Question	Answer
CQ1	What is the purpose of the ontology?	The ontology aims to support and facilitate the robotic process automation and human robot collaboration.
CQ2	What is the scope?	The ontology will include information on robotic processes. This includes pick and place, inspection, assembly, robot control, task planning, etc.
CQ3	Who are the intended end users?	There are two groups of intended users: 1. system engineers, programmers, and software engineers for support in AI functionalities; 2. Operators and engineers for error diagnostics and system queries.
CQ4	What is the intended use?	• To support robotic process modelling • To support PDDL robotic task planning • To create the sematic mapping from process goal to robot actions • To support Human–Robot Collaboration • To create a common understanding of the terms in the domain between hardware and software engineers, programmers, operators, and process designers so that knowledge is communicated consistently.
CQ5	What types of robotic processes for manufacturing exist?	There are two general types. One is applying robotics in light-out manufacturing processes. In these processes, the main goal of robots is replacing human workers and achieving automation. The other type is to apply robotics in collaborative manufacturing processes. The main goal for these processes is for robots and human to work together safely while increasing efficiency and/or improving human work conditions.
CQ6	How do robotic processes work?	These are processes that involve robots in the manufacturing steps. The process must enable communication with robots and control of robots.
CQ7	What are the robotic processes composed of?	A robotic process consists of different tasks. It can be human tasks, robot tasks, or combined tasks that are performed by humans and robots together.
CQ8	What are the links between these components?	These tasks can be independent or dependent. Sequential tasks are performed one after another (often the output of the former task is the input of the later task). Parallel tasks can be performed at the same time without influencing one another.
CQ9	What impacts robotic processes?	The speed of the robot, the performance of the robot, the safety of the robot, human factors (such as trust, fatigue, stress), and error diagnostics
CQ10	How are robotic processes evaluated?	There are KPIs that measure the process performance, such as cycle time, productivity, accident rate, etc.

Table 1. *Cont.*

CQ Nr	Competency Question	Answer
CQ11	What are the types of robotic tasks?	Human tasks, robot tasks, and human robot collaboration tasks that are performed by human and robot together.
CQ12	What do robotic tasks consist of?	Tasks can contain subtasks. Robot tasks often consist of many robot actions that are called by different names in the research and in practice—for instance, skills, capabilities, functions, etc.
CQ13	What are the links between these components?	It is similar to tasks; these components can also work dependently or independently.
CQ14	What are types of Human–Robot Collaboration tasks?	They are different based on the task types (coexistence, sequential, parallel, cooperate, teaching, collaborate) and collaboration mode (Fence, safety-rated monitored stop, hand guide, speed and separation monitoring, power and force limiting).
CQ15	What impacts Human–Robot Collaboration?	Human factors (how humans feel and behave around the robot), safety of the robot, safety measures, safety control
CQ16	How do humans and robots work together?	Humans and robots can work together in different ways: coexistence (working on their own, do not have anything to do with each other), sequential (they do the work sequentially one after another, they are dependent), parallel (they work at the same time but not on the same part), cooperate(they work on the same part but not at the same time), teaching (humans guide the robot to learn something, the human is in control, the robot is in teaching mode), collaborate (the human and robot work together on the same part and at the same time).
CQ17	How is safety insured when humans and robots work together?	There are different safety modes based on ISO standards, and these safety modes must be ensured to guarantee safety.

CQ1–CQ4 are top level questions that cover the ontological rationale. CQ5–CQ17 are more detailed questions used to elicit domain requirements.

3.1.2. Modularization

As the domain knowledge is quite complex, the target ontology is broken down into sub-ontologies: core robot process ontology and application specific ontologies. The core robot process ontology contains the generic robot process ontology, and it is called ORPP. The ACROBA project-related concepts are planned in the application ontology named ACROBA. This ACROBA ontology will gather project-specific and use case-specific knowledge. The concepts related to the graphical user interface (a tool to model robotic processes) are planned in an application ontology called RTMN. RTMN is the name of the modelling language [28]. RTMN has different notations such as Process, Task, Skill, Primitive, Sequence Flow, Condition, Decision, Error, Start/End, etc. The details are defined in the RTMN paper [28] and its successor RTMN2.0 [29]. With the support of this core robot process ontology, the graphical user interface (GUI) generates its components (ontology helps populate the GUI). The concepts related to task planning are in the PDDL application ontology. PDDL stands for Planning Domain Definition Language [30]. It is a planning language widely used in the robotic industry to define planning problems. This paper focuses on the core ORPP ontology.

This ontology aims to contribute to the robotics manufacturing domain and opt for a standardization that simplifies robotic task programming. The characteristics of the ontology are described as follows:

- achieves modularity, reusability, extensibility, knowledge sharing, and knowledge reasoning
- contains process, task, and robot action knowledge
- creates a link between processes management and robot action
- based on upper-level ontology, extends domain-level ontology, and adds application-level ontology

- combines the tool-centric approach (a manufacturing task is described based on available hardware and software components) and the product-centric approach (describing the product and associated production steps independent from specific production resources) [22]
- supports the populating of a graphical user interface (GUI) to design, create, simulate, and run the robotic processes
- supports the PDDL task planning

3.2. Ontology Conceptualization and Formalization (Phase 2&3)

3.2.1. Conceptualization

The ontology conceptualization phase requires knowledge acquisition. Based on [27], the process of extracting knowledge is divided into three steps. Firstly, find the relevant terms for the ontology. Secondly, define classes based on the terms. Thirdly, specify properties for the classes. Figure 3 gives an overview of two phases, and it is taken from [26].

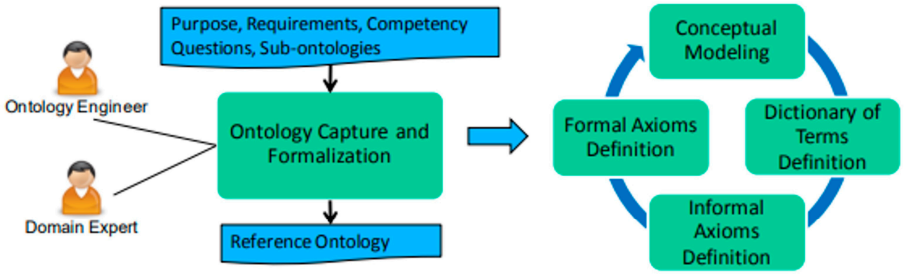


Figure 3. Ontology conceptualization and formalization (Source: [26]).

- Find the relevant terms for the ontology.

For ontology-based systems to be effective, they must involve standardized sets of terms that define a common understanding of the domain and may be shared amongst different systems [23,31]. To do that, the most-used terms in this area must be identified.

○ From Literature

A literature review was conducted to capture the knowledge of robotic process ontology. In Figure 4, the commonly used terms are listed. For each reference, the year of the publication, the name of the ontologies, their application fields, ontology language used to model the ontology as well as the relevant terms they used in their ontologies are presented in Figure 4. The "y" is an abbreviation for "yes", which indicates that this term is mentioned in the ontology. The total number of the terms noted in the different literature is calculated. Among the most used terms in the different papers, "Task", "Skill", and "Primitive" are identified as the most common terms. Sometimes nomenclature is not uniform (e.g., functions, capabilities), but the meanings fall into these main categories—task, skill, and primitive. Hardly any ontology addresses the process level. KPIs are also missing from the existing ontologies. Therefore, there is a need to standardize the terms used and shared in the robotic process domain. This will provide a common understanding of the domain and allow the ontology to be shared amongst different systems. These reusable knowledge pieces can be used to ease robot programming. To agree on the meaning of terms, the definitions in literature are checked first.

There are different definitions of the term "Task" in the literature. Many researchers define task as "a sequence of skills with specified parameters and conditions that aims to solve a specific goal [9,11,14,22,24,32]. PMK defines task as a subclass of Action, while ORO's task definition is also an action but in the specific context of robotics. KnowRob 2.0 uses DUL action to classify tasks, and it views a task as an even type that classifies an action to be executed [10]. Olszewska et al. consider a task as the interpretation of a robot of what the operator wants to achieve on the robot behavior level [17].

Year	Name	Application Field	Ontology Language	Terms												
				Task	Skill	Primitive	Process	Action	Behavior	Plan	Goal	Capability	Funtion	Device	Collaboration	Agent
2009	KNOWROB	service robot	Web OWL	y				y	y			y	y	y		y
2010	ORO	collaborative robot	RDF triples													
2014	RoboBrain	service robot	Graph structure													
2017	ROA	service robot	OWL	y	y	y	y		y		y		y			
2018	KnowRob2.0	service robot	Web OWL	y				y								
2018	ROSETTA	inductial robots	OWL	y	y	y					y			y		
2019	RTPO	service robot	OWL	y				y								
2019	BaSys 4.0	inductial robots	OWL	y	y	y						y		y		
2020	SOHO	collaborative robot	FOL	y	y	y						y	y			y
2020	C4I	industrial robot&collaborative robot	OWL	y	y	y						y				
2021	FoF	service robot	OWL	y	y											
2022	SOHO extension	collaborative robot	OWL	y		y						y	y		y	y
2022	OCRA	collaborative robot	FOL,OWL DL							y	y				y	y
2022	OTP	service robot		y							y					
2022	Rosetta extension	inductial robots	OWL	y	y	y										
			Total	12	7	7	1	3	2	1	4	5	4	2	2	4

Figure 4. Analysis of ontology terms related to robotic task planning [10,11,14,17,22,24,32,33].

The term "Skill" is also used in different ways in literature. Pederson et al. see robot skills as "intuitive object-centered robot abilities that allow robots to complete a task, which can easily be parameterized by a non-expert" [9]. Some authors describe skills as a parametrizable and executable functionality to achieve a particular task or goal [22,24]. Others use "abstract capability of a device" [10,11] or "low-level operations/primitives" [14] or "observed/desired actions of the robot" [17] or "robot programs" [33].

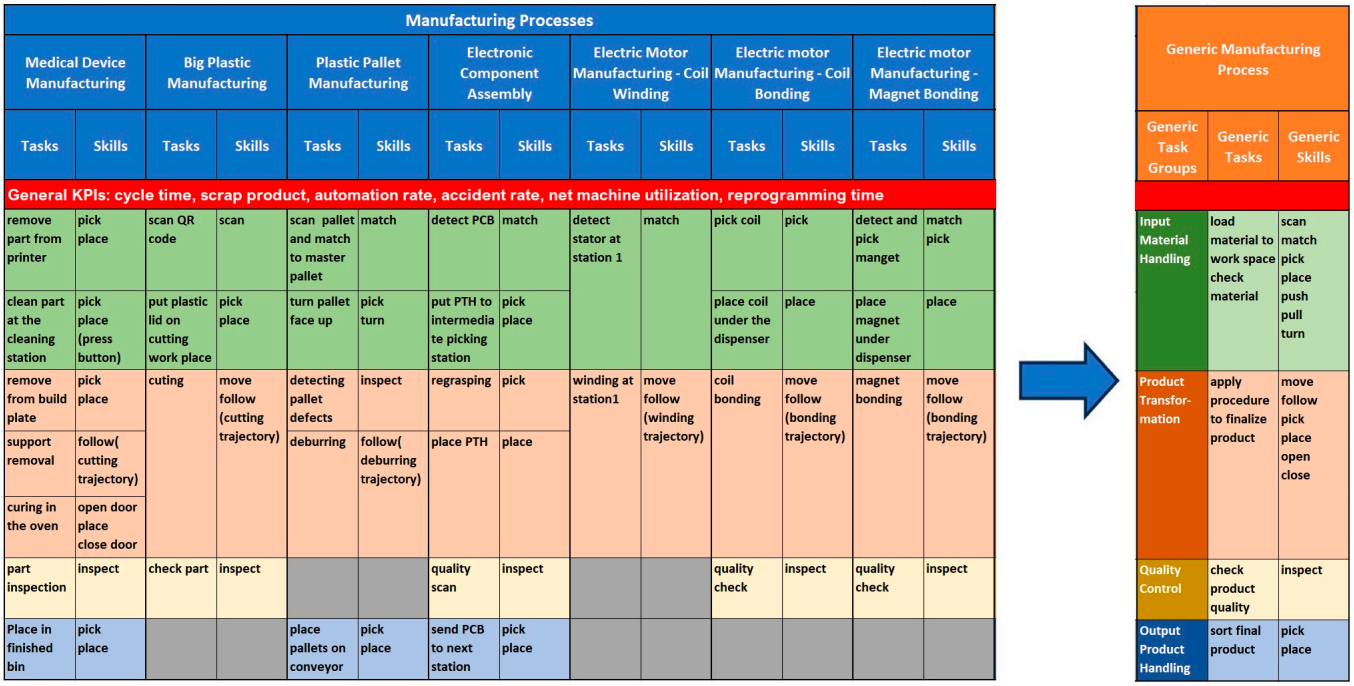
The description/definitions of the term "Primitive" vary. Pedersen et al. pointed out that primitives are single operations related to the robot system such as motion planning and gripper opening [9]. Schou et al. defined "device primitives as a set of device functionalities that abstract specific implementation details" [32]. Umbrico et al. focused on capability and defined interaction primitives as ActingCapability and SensingCapability [14]. Olszewska et al. [17] used "function" to describe primitive, which is similar to Pedersen et al.'s definition.

○ From Use Cases in the ACROBA Project

In addition to the literature review, we also analyzed the five use cases in the ACROBA project. Five industrial pilot lines from the ACROBA project are analyzed. There are seven processes: medical device 3D printing, big plastic manufacturing, plastic pallet bur handling, electronic components assembly, electric motor manufacturing coil winding, coil bonding, and magnet bonding.

The authors started their analysis by breaking the processes down into different tasks groups. Then, the different task groups were split into tasks. After this step, the tasks were further divided into smaller steps that are called skills. Figure 5 shows the results of the analysis. The columns with a blue header cover the analysis of the seven processes, and the columns with the orange header cover the generic definitions of a manufacturing process after consolidating the seven processes. The generic task groups, generic tasks, and generic skills are derived from the consolidation of the seven processes. The authors identified four generic task groups: input material handling (material preparation for production), product transformation (the core step for manufacturing a product), quality control (product quality check), and output product handling (sorting the products to different places). In each of the task groups, the generic tasks are identified. Afterwards, generic robot skills are derived from these generic tasks. These skills are match (to recognize and map the scanned material to the master data), pick (to pick up something from somewhere), place (to put something somewhere), move (move from one location to another), push (to move forward), pull (to move backward), turn (to move with an angle), follow (to generate and execute a trajectory),

open (to open something), close (to close something), and inspect (to scan and check the quality). In the future, the authors plan to analyze more production processes to derive more generic skills.



Manufacturing Processes															
Medical Device Manufacturing		Big Plastic Manufacturing		Plastic Pallet Manufacturing		Electronic Component Assembly		Electric Motor Manufacturing - Coil Winding		Electric motor Manufacturing - Coil Bonding		Electric motor Manufacturing - Magnet Bonding	
Tasks	Skills	Tasks	Skills	Tasks	Skills	Tasks	Skills	Tasks	Skills	Tasks	Skills	Tasks	Skills
General KPIs: cycle time, scrap product, automation rate, accident rate, net machine utilization, reprogramming time													
remove part from printer	pick place	scan QR code	scan	scan pallet and match to master pallet	match	detect PCB	match	detect stator at station 1		pick coil	pick	detect and pick manget	match pick
clean part at the cleaning station	pick place (press button)	put plastic lid on cutting work place	pick place	turn pallet face up	pick turn	put PTH to intermediate picking station	pick place			place coil under the dispenser	place	place magnet under dispenser	place
remove from build plate	pick place	cuting	move follow (cutting trajectory)	detecting pallet defects	inspect	regrasping	pick	winding at station1	move follow (winding trajectory)	coil bonding	move follow (bonding trajectory)	magnet bonding	move follow (bonding trajectory)
support removal	follow(cutting trajectory)			deburring	follow(deburring trajectory)	place PTH	place						
curing in the oven	open door place close door												
part inspection	inspect	check part	inspect			quality scan	inspect			quality check	inspect	quality check	inspect
Place in finished bin	pick place			place pallets on conveyor	pick place	send PCB to next station	pick place						

Generic Manufacturing Process		
Generic Task Groups	Generic Tasks	Generic Skills
Input Material Handling	load material to work space check material	scan match pick place push pull turn
Product Transfor-mation	apply procedure to finalize product	move follow pick place open close
Quality Control	check product quality	inspect
Output Product Handling	sort final product	pick place

Figure 5. Skills analysis on five industrial pilot lines.

- Define classes and their properties.

In addition to studying the definition of terms in the literature and analyzing the usage of the terms in the ACROBA use cases, the model of Pedersen et al. [9] also contributed to the decision of the classes. Pedersen et al. presented their model with three abstraction levels: task, skill, and primitive [9]. This model is commonly used in the robotics industry. The main classes are based on [9] with modifications and extensions. The following classes are decided for the ontology, and they are described here in natural language.

Robotic Process: this is a process that involves robots in production. Robot Process is a process. There is at least one Robot as Agent participating in the process. A Robot Process consists of at least one Robot Task. It can have human tasks, robot tasks, and Human–Robot Collaboration (HRC) tasks.

Task: It is part of a Robot Process. It has three types: Robot Task, Human Task, and Human–Robot Collaboration Task.

Robot Task is defined as a compound activity that is undertaken by robots to perform complex robot actions. It consists of robot skills.

Human Task is an activity that is performed by a human as an agent. It has at least one output to the system that indicates the Human Task is finished.

Human–Robot Collaboration Task (HRC Task) is a compound activity that is performed by both a human and a robot to achieve a common goal. There are five types of HRC tasks: Coexistence Fence (CF), Sequential Cooperation SMS (SS), Teaching HG (TH), Parallel Cooperation SSM (PS), and Collaboration PFL (CP) [29].

Robot Skill (in short, Skill) is classified as generic skills and application skills. Generic skills are skills that can be reused by every application. Application skills are created just for usage in one specific application. Robot Skills are defined as intuitive robot abilities for performing an autonomous mission. Skills should be easily understood by non-experts for parameterization. A skill is made up of a combination of primitives and/or other skills. Skills are used to complete higher level tasks [9].

Robot Primitive is defined as an atomic robot action for performing a single operation. Primitives can be combined to form more complex robot actions—Robot Skills. They are the lowest level model elements and provide a direct link to hardware elements. Primitives are not hardware dependent. Normally, the same type of device provides the same primitives, and therefore the devices within the same type should be interchangeable without affecting the control layer. For example, all grippers can "Grasp" and "Release". Primitives are mapped to executable codes on the robot system to manage the low-level robot controls.

KPI: Key performance indicators (KPIs) are a set of metrics that evaluate the performance of processes or tasks. KPIs specify requirements. KPIs represent quantitative measures of critical success factors (CSFs) in enterprises. KPIs play an essential role in seeking improvements and reaching better performance.

Requirement is defined as the target goal of a process/subprocess/task. Each requirement can have one or more KPIs evaluating its achievements. A requirement can have multiple KPIs.

The properties, conditions, and cardinality of the terms are defined in Table 2.

Table 2. Class and Properties.

Class	Property	Condition	Cardinality
Process	name goal sequence input output feedback precondition postcondition		has at least one task
Task	name sequence input output feedback precondition postcondition		has at least one human task or robot skill
Human Task	name human agent output	no robot involvement	has at least one human agent
Robot Task	name sequence input output feedback precondition postcondition	no human involvement	no human agent
HRC Task	name sequence inputs outputs feedback precondition postcondition HRC task type HRC mode safety level	human and robot are both involved in completing the activity	has at least one human task and one robot task

Table 2. *Cont.*

Class	Property	Condition	Cardinality
Robot Skill	name sequence input output feedback precondition postcondition	no human involvement	has at least one primitive
Robot Primitive	name input output feedback precondition postcondition	has robot as an agent	atomic
KPI	name formular		
Requirement	name goal		

3.2.2. Formalization

In the process of formalization, the informal terms and axioms that are written in natural language will be described in a formal language. It is recommended to reuse or extend existing ontologies whenever possible, especially for upper ontologies, because it can reach a higher level of maturity and compatibility as well as increase acceptance [14]. Therefore, the intension is to connect the ORPP ontology to the standard ontologies for robotics and automation: CORA (IEEE standard ontology for robotics and automation) [7] and its theoretical foundation SUMO (Suggested Upper Merged Ontology) [8].

CORA is an IEEE standard ontology for robotics and automation. CORA aims to create a common vocabulary/language for the robotics and automation domain. It formally defines robots, robot parts, robot groups, robot positions and configurations, robot autonomy levels, and robotic systems [14,17].

SUMO is the Suggested Upper Merged Ontology, which is an upper-level ontology that provides definitions of basic ontological concepts for all domains to use as a foundation [8]. To ensure reusability and alignment with existing standards, the ORPP and its supporting ontologies are developed based on CORA [7] and SUMO [8].

Developing ORPP based on CORA and SUMO will ensure reusability and alignment with existing standards. The ontology architecture is shown in Figure 6. It shows that ORPP consists of two levels of ontologies (in blue): domain ontology and application ontology. The domain ontology "ORPP Core" covers the main robotic process concepts. The application ontologies cover the task planning (PDDL), use case ontology (ACROBA), and the Robot Task Modeling Notation (RTMN) concepts. The PDDL, RTMN, and ACROBA ontology do not lie in the scope of this paper. They are marked in grey.

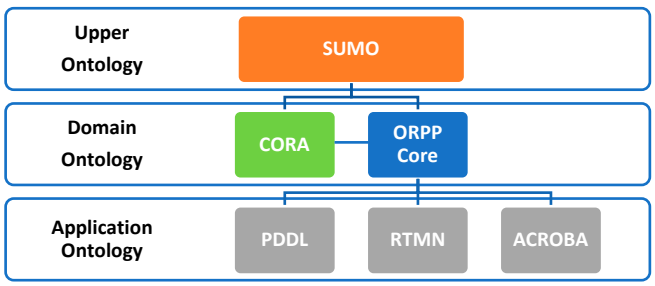


Figure 6. Ontology architecture.

Figure 7 presents the overall concepts based on the ORPP architecture. It is modeled as a UML diagram. The list of terms is presented in Appendix A.

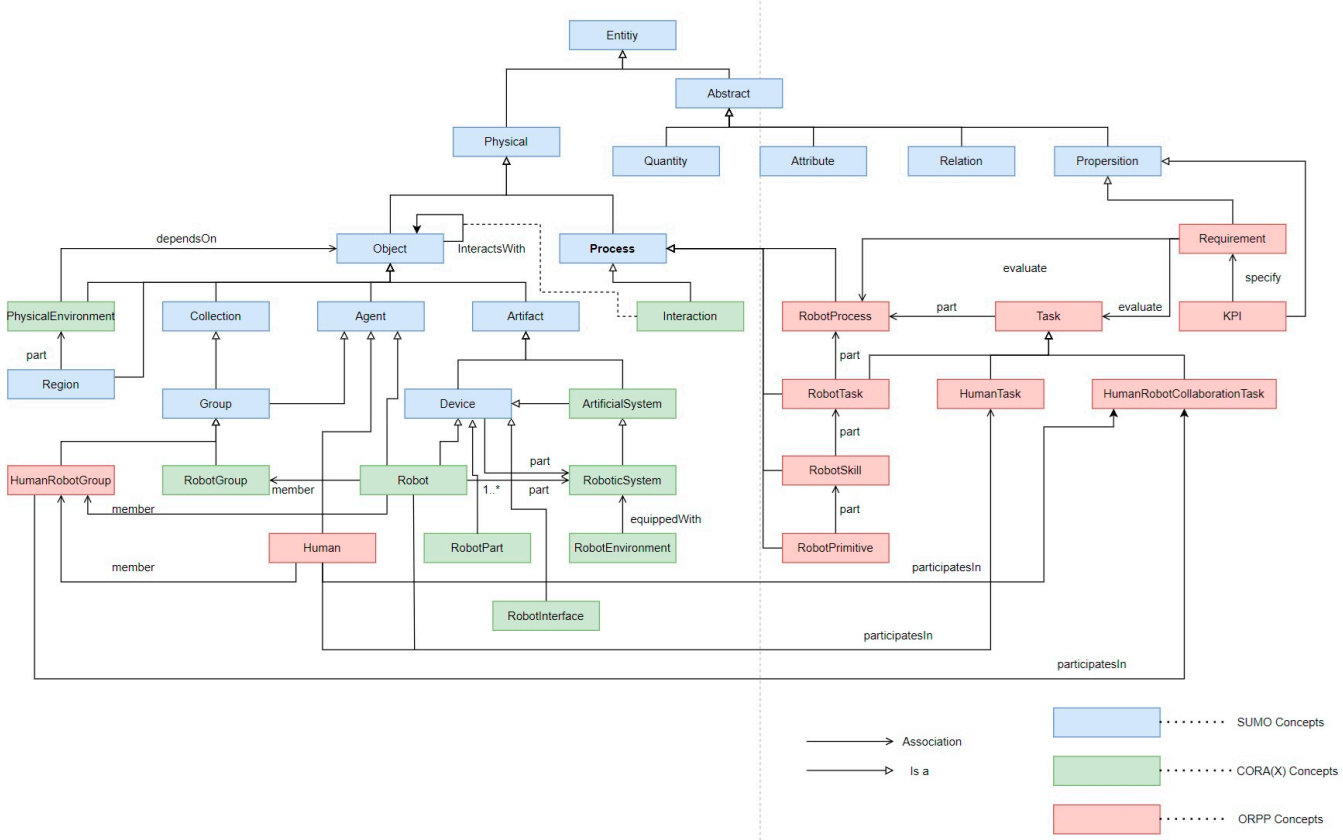


Figure 7. Taxonomy of the main concepts of ORPP and their relation to the other ontologies. 1..* in UML means at least one. In this figure it means device has at least one robot part.

- SUMO Concepts

SUMO classes are shown in Figure 6 in blue. Entity is the top SUMO category. It is the universal class of individuals. It has two disjoint sub classes Physical and Abstract. The definitions are quoted from [7].

Physical: "An entity that has a location in space-time".

Abstract: "Properties or qualities as distinguished from any particular embodiment of the properties/qualities in a physical medium. Instances of Abstract can be said to exist in the same sense as mathematical objects such as sets and relations, but they cannot exist at a particular place and time without some physical encoding or embodiment".

Process: "The class of things that happen and have temporal parts or stages. Examples include extended events like a football match or a race, actions like pursuing and reading, and biological processes. The formal definition is anything that occurs in time but is not an object".

Object: "Corresponds roughly to the class of ordinary objects. Examples include normal physical objects, geographical regions, locations of processes, and the complement of objects in the physical class".

Region: "A topographic location. Regions encompass surfaces of objects, imaginary places, and geographic areas. Note that a region is the only kind of object that can be located at itself".

Collection: "Collections have members like classes, but, unlike classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the collection".

Agent: "Something or someone that can act on its own and produce changes in the world".

Artifact: "An object that is the product of a making".

Group: "A collection of agents".

Device: "A device is an artifact whose purpose is to serve as an instrument in a specific subclass of a process".

Quantity: "Any specification of how many or how much of something there is. Accordingly, there are two subclasses of quantity: number (how many) and physical quantity (how much)".

Attribute: "Qualities that we cannot or choose not to reify into subclasses of object".

Set or Class: "The set or class of sets and classes, i.e., any instance of abstract that has elements or instances".

Relation: "The class of relations. There are three kinds of relation: predicate, function, and list. Predicates and functions both denote sets of ordered n-tuples. The difference between these two classes is that predicates cover formula-forming operators, while functions cover term-forming operators. A list, on the other hand, is a particular ordered n-tuple.".

Proposition: "Propositions are abstract entities that express a complete thought or a set of such thoughts".

- CORA(X) Concepts

CORA and its supporting ontologies classes are shown in Figure 6 in green. The definitions are quoted from [7].

Device: "Device is partitioned to Robot, Robot Part, Robot Interface, and Electric Device. Robots have other devices as parts".

Robot: "a robot is a Device (SUMO term) which participates as a tool in a process. A robot is also an Agent which is something that can act on its own and produce changes. Robots perform tasks by acting on the environment/themselves".

RobotPart: "RobotPart is further divided into robotActuatingPart, robotCommunicatingPart, robotProcessingPart and robotSensingPart".

RobotInterface: "Robot interacts with the world surrounding it through an interface. The RobotInterface is a device composed by other devices that play the roles of sensing device, actuating device and communicating device. Through the interface, the robot can sense and act on the environment as well as communicate with other agents. Therefore, the robot interface can be viewed as a way to refer to all the devices that allow the robot to interact with the world. Every robot interface must have a part that is either a robot sensing part, or a robot actuating part or a robot communicating part".

RobotGroup: "A robot is an agent, and agents can form social groups. According to SUMO, a group is 'a collection of agents', like a pack of animals, a society, or an organization. A RobotGroup is a group whose only members are robots".

ArtificialSystem: "ArtificialSystem is an Artifact formed by various devices (and other objects) that interact in order to execute a function. For any part of an artificial system, there is at least one other part it interacts with".

RoboticSystem: "Robots and other devices can form a RoboticSystem. A RoboticSystem is an artificial system formed by robots and devices intended to support robots to carry on their tasks. Robotic systems might have only one or more than one robot".

Interaction (CORAX): "An interaction is a Process in which two agents participate. It is composed of two subprocesses defining action and reaction. The action subprocess initiated by agent x on a patient agent y causes a reaction subprocess having y as agent and x as patient".

- ORPP Core Concepts and Axioms

The ORPP core is the focus of this paper. It is based on the model of Pedersen et al. [9] with modifications and extensions. Pedersen et al. presented their model with three abstraction levels: task, skill, and primitive [9]. The ORPP model has one more layer—the process layer. It is a very important part as it integrates the tasks and enables the requirement evaluation with KPIs. A process can be broken down into different levels—the primitive layer being the lowest abstraction level which consists of premade programs to

control robot actions. The skill layer is made up of generic and specific skills. The task layer consists of human tasks, robot tasks, and HRC tasks. These layers can provide traceability of the KPIs to robot primitives. They allow the KPIs to be calculated automatically in the robotic system. The contribution of the model has the following features:

- It adds a new process layer. It groups the robotic tasks together and provides an overview for the management with definable requirements/goals and KPIs. The process is the starting point of any requirement engineering analysis; it is highly important from the modelling perspective. Since one can track the process down to the primitive level, each robotic action is registered in the system and the KPIs can be calculated based on the information from the primitive layer. These KPIs then provide the evaluation of the requirements/goals.
- It extends the task layer to three task types (Robot Task, Human Task and HRC Task). Separating human and HRC tasks enables the application of different levels of safety controls.
- It enriches skill layers with skills themselves, meaning that skills can be created by not only primitives but also by skills. This enables the reuse of skills that are already created.

In this section, the First Order Logic is used to describe the informal terms and axioms in the ORPP ontology that are written in natural language in the former chapter.

RobotProcess: RobotProcess is a Process. It has at least one Robot as Agent participates in. A RobotProcess consists of at least one RobotTask. It can be composed of HumanTasks, RobotTasks, and HumanRobotCollaborationTasks.

```
(subclass  RobotProcess  Process)

(=>
  (and (RobotProcess ?x)
    (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?x)
          (Robot ?agent)))
    (exists (?task)
        (and (part ?task ?x)
          (RobotTask ?task))
    (exists (?task)
        (and (part ?task ?x)
          (or (HumanTask ?task)
            (RobotTask ?task)
            (HumanRobotCollaborationTask ?task)))))
  (instance ?x Process))
```

Task: Task is a Process. It is part of a RobotProcess. It has subclass RobtTask, HumanTask, and HumanRobotColloaborationTask.

```
(subclass  Task  Process)

(=>
  (and (Task ?x)
    (part ?x ?robotProcess)
    (or (RobotTask ?x)
      (HumanTask ?x)
      (HumanRobotCollaborationTask ?x)))
  (instance ?x Process))
```

RobotTask: RobotTask is a Task. It is part of a RobotProcess. It has at least one Robot as Agent. It cannot have Human as Agent. It consists of RobotSkills.

```
(subclass  RobotTask  Task)

(=>
  (and (RobotTask ?x)
    (partOf ?x ?robotProcess)
    (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?robotProcess)
          (Robot ?agent)))
    (not (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?robotProcess)
          (Human ?agent))))
    (exists (?skill)
        (and (part ?skill ?x)
          (RobotSkill ?skill))))
  (and (Task ?x)
    (instance ?x Process)))
```

HumanTask: HumanTask is a Task. It is part of a RobotProcess. It has at least one Human as Agent. It cannot have Robot as Agent.

```
(subclass  HumanTask  Task)

(=>
  (and (HumanTask ?x)
    (partOf ?x ?robotProcess)
    (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?robotProcess)
          (Human ?agent)))
    (not (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?robotProcess)
          (Robot ?agent)))))
  (and (Task ?x)
    (instance ?x Process)))
```

HumanRobotCollaborationTask (HRC Task): HRCTask is a Task. It is part of a RobotProcess. It has at least one Human and one Robot as Agent. There are five types of HRC tasks: Coexistence Fence (CF), Sequential Cooperation SMS (SS), Teaching HG (TH), Parallel Cooperation SSM (PS), and Collaboration PFL (CP).

```
(subclass  HumanRobotCollaborationTask  Task)

(=>
  (and (HumanRobotCollaborationTask ?x)
    (partOf ?x ?robotProcess)
    (exists (?human ?robot)
        (and (Agent ?human)
          (Agent ?robot)
          (participatesIn ?human ?robotProcess)
          (participatesIn ?robot ?robotProcess)
          (Human ?human)
          (Robot ?robot))))
  (and (Task ?x)
    (instance ?x Process)))
```

RobotSkill: RobotSkill is a Process. It is part of a RobotTask. It can also be part of a RobotSkill. It consists of Primitives and/or other RobotSkills. RobotSkill has Robot as its Agent. Human cannot be the Agent of RobotSkill.

```
(subclass RobotSkill Process)

(=>
  (and (RobotSkill ?x)
    (partOf ?x ?taskOrSkill)
    (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?taskOrSkill)
          (Robot ?agent)))
    (not (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?taskOrSkill)
          (Human ?agent))))
    (exists (?primitiveOrSkill)
        (and (part ?primitiveOrSkill ?x)
          (or (Primitive ?primitiveOrSkill)
            (RobotSkill ?primitiveOrSkill)))))
  (and (Process ?x)
    (instance ?x Process)))
```

RobotPrimitive: RobotPrimitive is a Process. It is part of a RobotSkill. RobotPrimitive is atomic. RobotPrimitives has Robot or RobotPart as Agent. Human cannot be the Agent of RobotPrimitive.

```
(subclass RobotPrimitive Process)

(=>
  (and (RobotPrimitive ?x)
    (partOf ?x ?skill)
    (exists (?agent)
        (and (Agent ?agent)
          (participatesIn ?agent ?skill)
          (or (Robot ?agent)
            (RobotPart ?agent)))
        (not (Human ?agent))))
  (and (Process ?x)
    (instance ?x Process)))
```

HumanRobotGroup: A Robot or a Human is an Agent, and agents can form social groups. According to SUMO, a group is "a collection of agents". A HumanRobotGroup is a group that has humans and robots as members. It participates in HumanRobotCollaborationTasks.

```
(subclass HumanRobotGroup Group)

(=>
  (and (HumanRobotGroup ?group)
    (exists (?member)
        (and (Agent ?member)
          (memberOf ?member ?group)
          (or (Human ?member)
            (Robot ?member)))))
  (instance ?group Group))
```

Requirement: Requirement is a subclass of Proposition. Requirements evaluate Task or Process. Requirement has KPI to specify it. A Task or Process can have multiple Requirements.

```
(subclass  Requirement  Proposition)

(=>
  (and (Requirement ?req)
    (or (evaluates ?req ?task)
      (evaluates ?req ?process)))
  (instance ?req Proposition))
```

KPI: KPI is a subclass of Proposition. KPIs specify Requirement. A Requirement can have multiple KPIs. KPI and Requirement are disjoint subclasses of Proposition.

```
(subclass  KPI  Proposition)

(=>
  (and (KPI ?kpi)
    (specifies ?kpi ?requirement))
  (and (subclass ?kpi Proposition)
    (subclass ?requirement Requirement)
    (disjointSubclass KPI Requirement)))
```

Human: Human is an Agent who can act on its own and produce changes. It participates in HumanTask or HumanRobotCollaborationTask. Human performs these tasks by acting on the environment/themselves.

```
(subclass  Human  Agent)

(=>
  (and (Human ?human)
    (participatesIn ?human ?task)
    (or (instance ?task HumanTask)
      (instance ?task HumanRobotCollaborationTask))
    (actsOn ?human ?environment))
  (and (Agent ?human)
    (canActOnOwn ?human)
    (producesChanges ?human)
    (instance ?human Human)))
```

- PDDL [30] Concepts

The PDDL definition covers domain description, problem description, predicates, actions, initial state, and goal state. The PDDL definitions can be described into an ontology. To use the ontology to support the PDDL, a mapping between the ontology elements and the PDDL constructs must be established. Such an ontology can support the PDDL by providing a structured and semantically rich representation of the domain concepts and relationships.

The mapping includes the following:

- Types in the PDDL are represented as classes in the PDDL ontology.
- Objects are instances of types in the PDDL, and they are mapped to ontology to individuals.
- The PDDL uses predicates to define relationships between objects, and these are described as object properties in ontology.
- Initial state specifies the state of the world before the start in the PDDL. Ontology uses object property assertions to represent these individual instances. Individual instances in the ontology are associated with their properties using object property assertions. It is the same for the goal state.

- Actions in the PDDL represent possible transitions between states. They can be created as classes and properties in ontology.
- Constraints are rules and limitations for actions in the PDDL. In oncology, they are described using additional axioms and properties.

The main concepts of the PDDL application ontology are Plan, Action, InitialState, GoalState, CurrentState, Domain, and Problem. The PDDL ontology is under development, and the basic concept is shown in Chapter 4 in the PDDL application scenario. The complete PDDL ontology will be presented after validation.

- ACROBA Concepts

The ACROBA use case ontology covers five industrial manufacturing use cases (medical device, large plastic, electric motor, and electronic components) in the ACROBA project [6]. As all use cases are different in terms of products and processes, the use case-specific knowledge in the application ontology is separated from the ORPP core because it allows the ORPP to cover only the generic concepts that lead to flexibility and reusability. The main concepts of the ACROBA ontology are as follows: Organization, User, Role, Product, ProductPart, ProductionPlan, and Device. The ACROBA ontology is under development and will be presented when it is finished.

- RTMN Concepts

RTMN [28] is a graphical modeling language that enables the users to intuitively model their production processes and execute them. As it is used for developing a graphical user interface (GUI), it is considered an application ontology. The authors developed RTMN in their previous research, and it is customized and aligned with their general approach for robot task planning. The concept used in RTMN will be modeled as classes and object properties in the ORPP ontology. This ontology enables the generation of the GUI. The RTMN concepts include Process, Task, Skill, Primitive, Sequence Flow, Condition, Decision, Start/End, and Notation. The RTMN is under development and will be presented when it is validated.

4. Results—Ontology Application

4.1. Ontology Implementation (Phase 4)

The ORPP core ontology was implemented using OWL/RDF in Protégé 5.5.0 [34]. The terms in ORPP are defined as classes in Protégé. Relationships are defined using object property. Data property defines data type properties. Individuals are the instantiations of the classes and properties.

Figure 8 is the relation graph of the implemented ontology. It shows the relationships between classes, object properties, and instances.

In Figure 8, the orange nodes are classes, and the blue arrows indicate they have subclasses. Relationships between classes are modeled as object properties in Protégé. Here, some of the most important classes and their related properties are described.

The Process Class has object properties: hasTask(process consists of tasks), isEvaluatedby (process is evaluated by requirements). Data properties include Input, Output, Feedback, Precondition, and Postcondition.

The Task Class has the following object properties: hasAgent (one can assign an agent to the task) and hasSkill. It has data properties: Input (e.g., 3D coordinates), Output (e.g., grasping pose), Preconditions (e.g., robot at A), Postconditions (e.g., robot at B), Feedback, and Sequence (a list of tasks to be performed in this task). Task has three subclasses: Robot Task, Human Task and Human–Robot Collaboration Task (HRC Task). The HumanTask has a human agent. The RobotTask has a robot agent. The HRC Task has human and robot agents. It also has DistanceHumanRobot (the distance between the human and the robot), HumanPosition (the human x, y, z location), RobotPosition (the robot x, y, z location), and RobotSpeed (the speed of the robot) as data properties.

The Skill Class has the following main object properties: hasAgent (Robot), hasPrimitive (Skill consists of primitives), and hasSkill (skills consists of other skills). It is a isSkillof

Task. isSkillof is the inverse property of hasSkill. Data properties include Sequence, Input, Output, Feedback (e.g., after running the skill, it returns an image from a camera), Precondition, and Postcondition.

The Primitive Class has the following main object properties: isPrimitiveof (which is the inverse property of hasPrimitive). Data properties include Input, Output, Feedback, Precondition, and Postcondition.

The Requirement Class has the following object properties: isSpecifiedby (requirements are specified by KPIs) and evaluates (requirement evaluates processes), and it is the inverse property of isEvaluatedby. The KPI Class has the following object properties: specify (KPI specifies requirements).

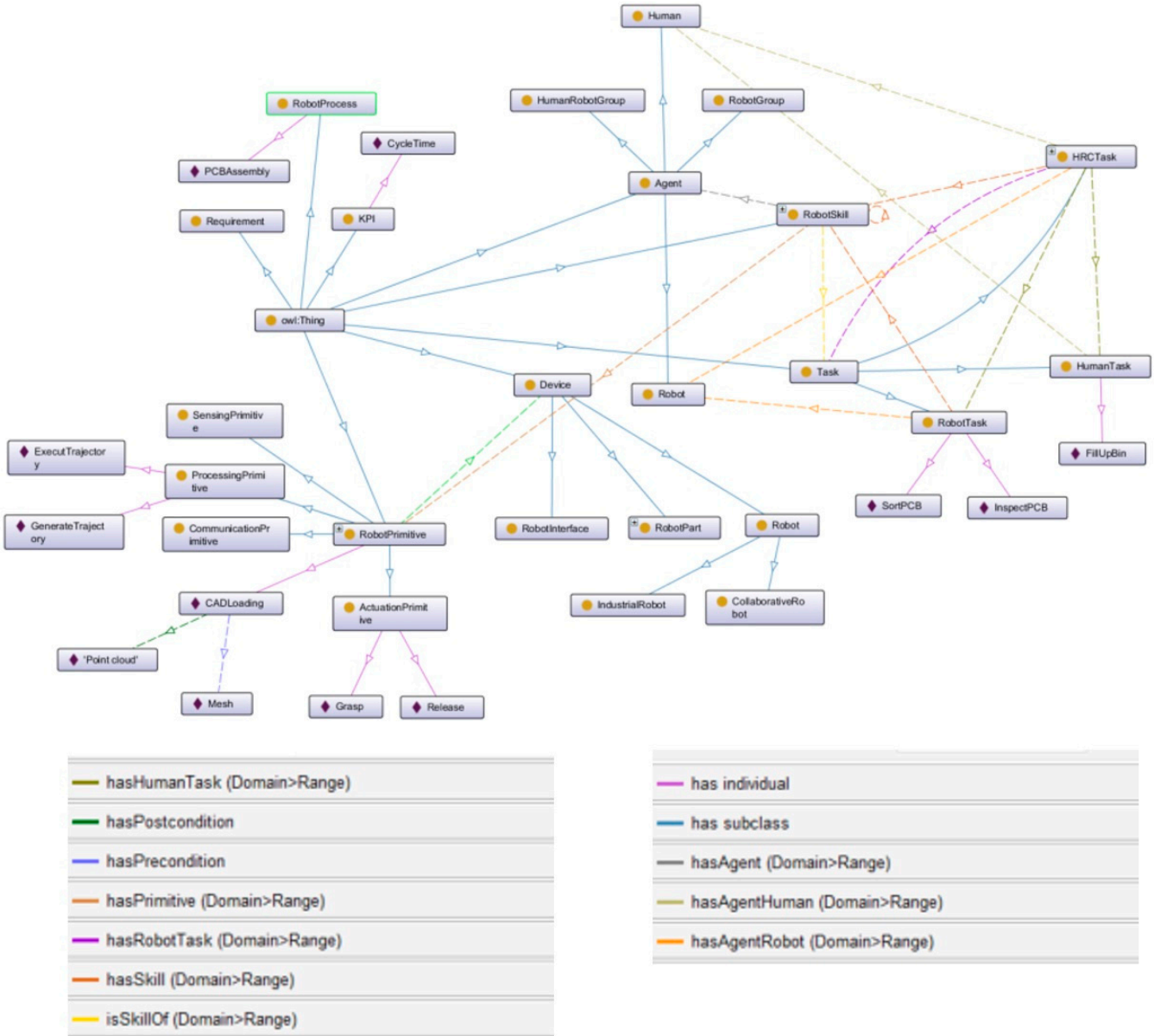


Figure 8. ORPP core relation graph.

4.2. Ontology Testing (Phase 5)—The Use of ORPP in the ACROBA Project

In the ACROBA project, the ORPP ontologies are used in three ways: 1. populating the GUI; 2. supporting task execution; 3. supporting task planning. The overview of the usage is shown in Figure 9.

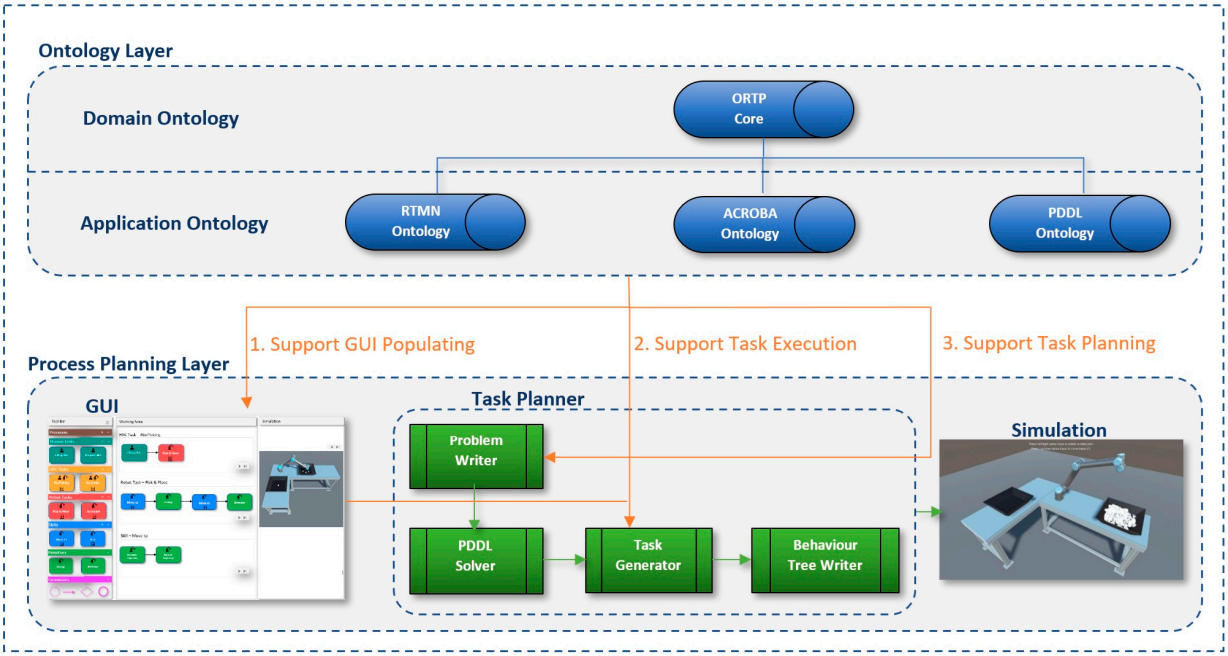


Figure 9. Uses of ORPP ontologies.

- Populating the GUI (Orange Arrow 1.)

The RTMN graphical user interface allows users to drag and drop tasks, skills, and primitives to compose a manufacturing process. To be able to load this information correctly in the GUI, ontology is used to generate the GUI sections with the correct processes, tasks, skills, and primitives. Users can log into the populated GUI and use the drag and drop function to model a process. To create a process model, one needs to first create the different tasks in the process, presuming skill and primitives are already available in the GUI. Then, one can use these created tasks to compose the process. The instances of the process model are saved back to the ontology as individuals. New robot skills can be created based on existing primitives and other skills. For example, the robot skill Pick is created by the robot skill Move and robot primitive Grasp.

- Supporting task execution (Orange Arrow 2.)

Process is executed by task sequences. Task sequence is given in the RTMN process model. ACROBA executes task sequences based on the core ORPP classes and relationships. Each task has a sequence of skills. For example, a "pick and place" task has the skill sequence Move, Match, Pick, Move, and Place assigned to it. Each skill has a primitive sequence. By launching a process, the sequences of tasks, skills and primitives are launched. These sequences are sent to the task planner (a module of ACROBA platform) and translated further into behavior threes.

- Supporting PDDL task planning (Orange Arrow 3.)

ACROBA ontology defines all the five use cases—their organization, users, their products, and processes. It supports the PDDL task planning by providing the domain and problem definitions in the ACROBA ontology. The PDDL planning aims to generate a skill sequence that solves a given task. The task planner translates all relevant information from the ontology into the PDDL [30] domain and problem representation. For example, a skill in the ontology is converted into an action in the PDDL [24].

4.2.1. PCB Assembly Application—Example of Orange Arrow 1&2

The testing of the ontology is based on a real-world scenario in the ACROBA project. The instantiation covers one simple assembly process. It is illustrated in Figure 10—a Printed Circuit Board (PCB) assembly process.

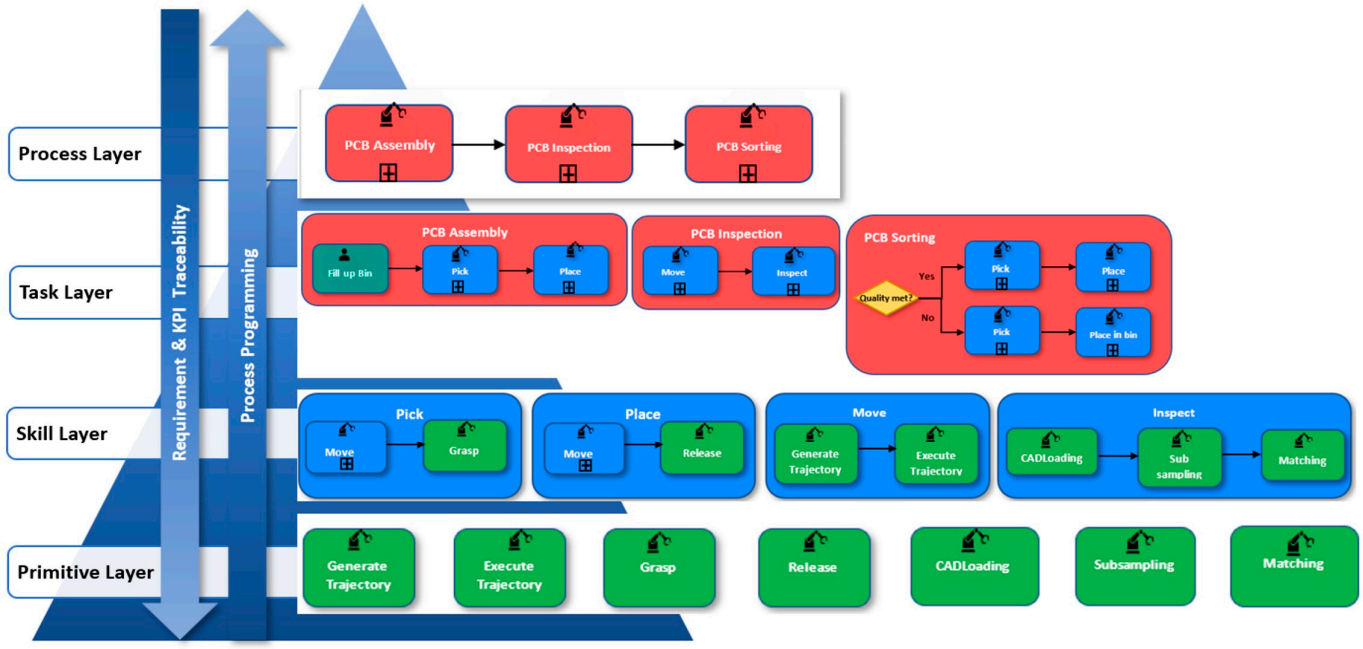


Figure 10. Projection of an ontology instantiation—a PCB assembly process.

The PCB is assembled by putting different kinds of pin through hole (PTHs) components on the board. The process is called PCBAssembly as an instantiation of the class process. This PCBAssembly process has three tasks: PCB assembly (AssemblePCB: an instantiation of a HRCTask), PCB inspection (InspectPCB: an instantiation of a RobotTask) and PCB sorting (SortPCB: an instantiation of a RobotTask). PCB assembly is conducted by a HumanTask—FillUpBin (bin is used as the container of the PTHs) and two RobotSkills—Pick and Place. The PCB inspection task is undertaken by RobotSkills—Move and Inspect. The PCB sorting uses two RobotSkills—Pick and Place—to sort the PCB at different places.

Figure 11 shows that HRCTask has an instance, AssemblePCB. It is part of the PCBAssembly process instantiation. This specific task has a Human Ron and a Robot UR5 as its Agents. It has HumanTask FillUpBin instantiation and two RobotSkills, Pick and Place. It has a sequence and some other uses in Figure 11.

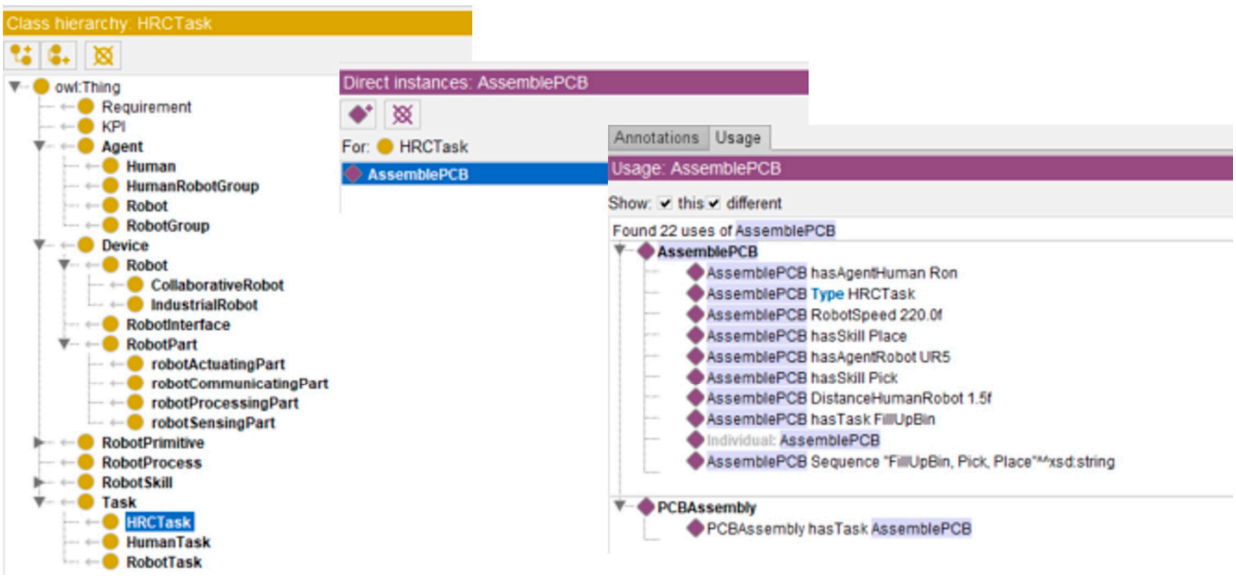


Figure 11. Individual HRCTask and its use in ORPP.

The ontology can be queried using different SPARQL queries to obtain the information to support the target uses. Below, Figure 12 is an example of a query to return the instantiation of the PCBAssembly process.

The instances of the Process, Task, Skill, and Primitives can be used to populate the GUI. The sequences are used to support task execution. Here, only the ORPP ontology is used. For this example, if the ACROBA, RTMN, and PDDL ontologies are ready, one can link the information and have a much more comprehensive ontology to be queried for complex planning scenarios. See Figure 13 for the information use.

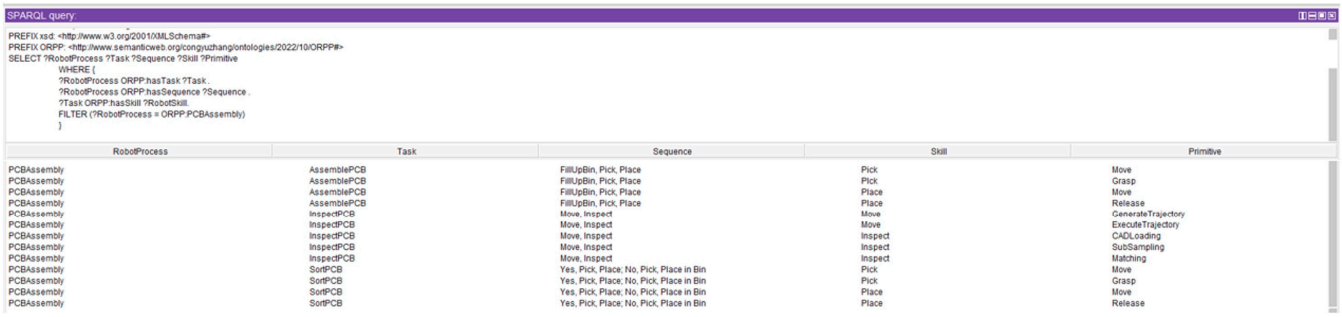


Figure 12. SPARQL query for PCBAssembly process instantiation.

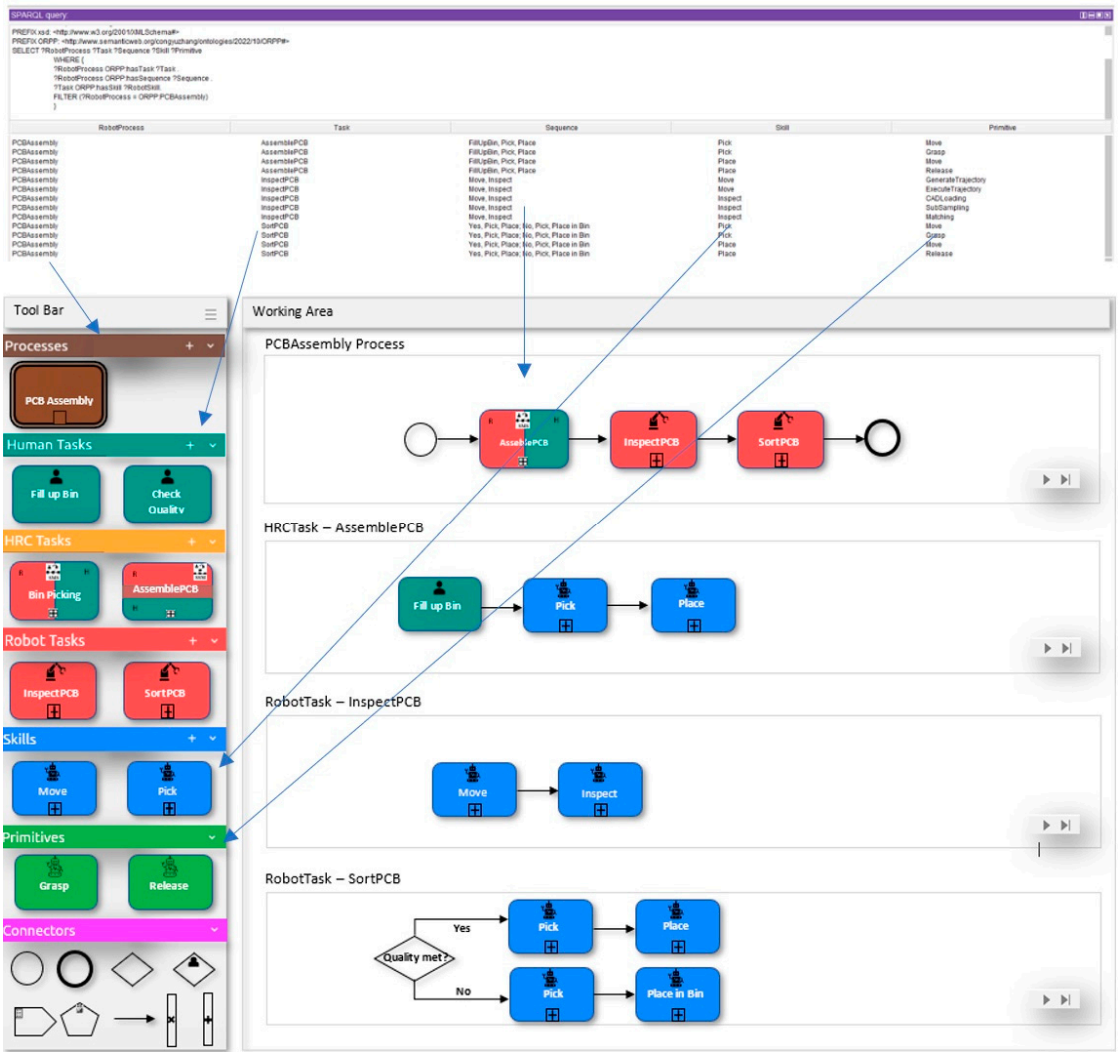


Figure 13. Ontology query support use of ontology.

The ontology queries can answer the following competency questions: CQ1, CQ4, CQ5, CQ7, CQ10, CQ11, CQ12, CQ14, and CQ16.

4.2.2. PDDL Task Planning Application- Example of Orange Arrow 3

- Cell Setup
  Figure 14 is a picture of the demonstrator. It shows the setup of the cell.
  This cell consists of the following components.
  - Robot: UR5e
  - Gripper: ROBOTIQ 2F-85
  - Camera: Zivid M+
  - There is a "grid" that has three by three holes, and their positions are (00,01,02, 10,11,12,20,21,22)
  - There are "cylinder" and "circular-container". A cylinder can be put into a circular-container. Both cylinders and circular-containers have three colors: red, yellow, and green.
  - There are three bins for the cylinder (they are bin red, bin yellow, and bin green), and each bin contains six cylinders. Each bin contains one color of the cylinders
  - There are three hangers for the circular-container (they are hanger red, hanger yellow, and hanger green), and each hanger contains one color of circular-containers.
- The Scenario

The task is to assemble the cylinders and circular-container in the grid. The assembly goal is to put cylinders into circular-containers on all holes of the grid with matching colors. In other words, the goal of this scenario is for robots to move cylinders and circular-containers from the initial state to the goal state.

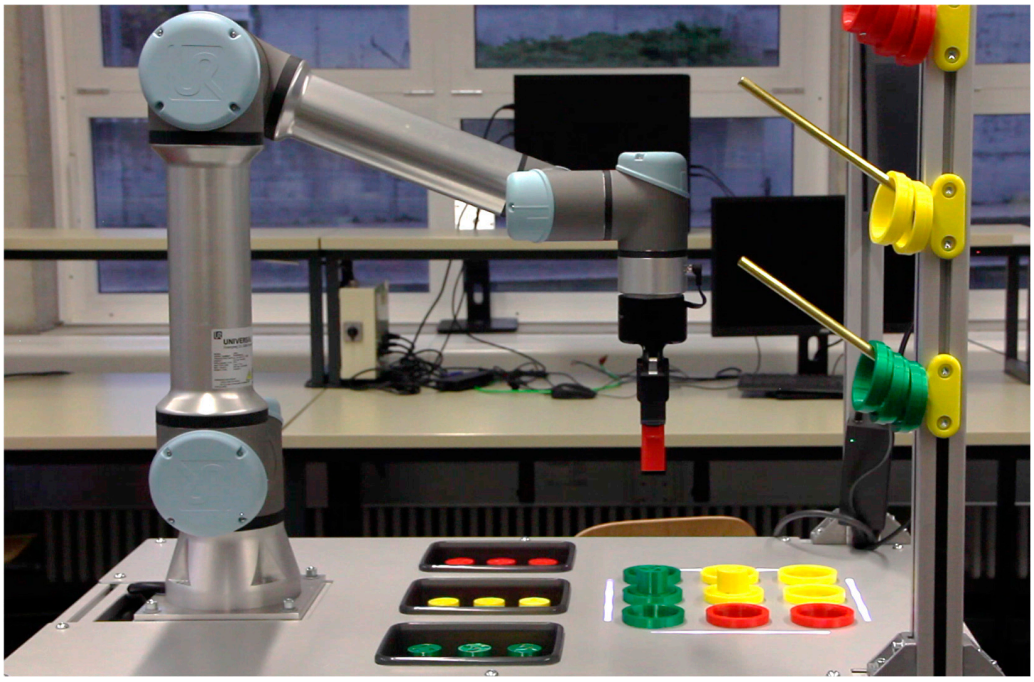


Figure 14. The demonstrator.

- The Initial State

The user decides and sets the initial state for all positions: (00, yellow circular-container, no cylinder), (01, yellow circular-container, no cylinder), (02, red circular-container, no cylinder), (10, yellow circular-container, no cylinder), (11, yellow circular-container, yellow cylinder), (12, red circular-container, no cylinder), (20, green circular-container, no cylinder),

(21, green circular-container, green cylinder), and (22, green circular-container, no cylinder). It is presented in Figure 15.

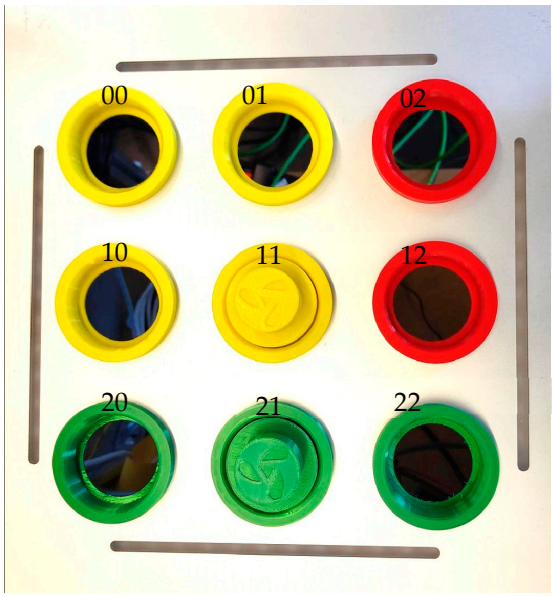


Figure 15. Initial state.

○ The Goal State

The user sets the goal state (see Figure 16), which is to put cylinders into circular-containers on all holes of the grid with matching colors.

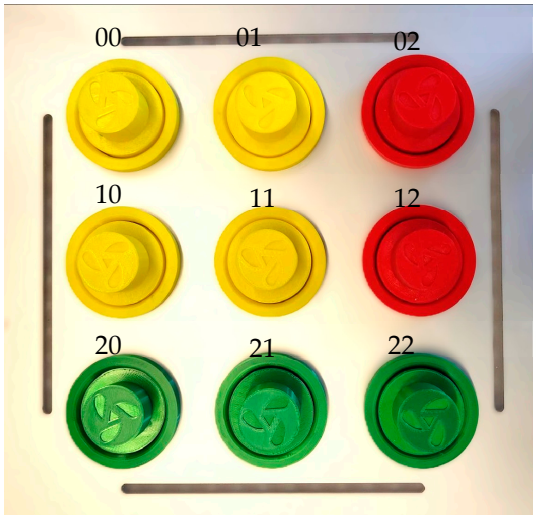


Figure 16. Goal state.

The goal state is given for all positions: (00, yellow circular-container, yellow cylinder), (01, yellow circular-container, yellow cylinder), (02, red circular-container, red cylinder), (10, yellow circular-container, yellow cylinder), (11, yellow circular-container, yellow cylinder), (12, red circular-container, red cylinder), (20, green circular-container, green cylinder), (21, green circular-container, green cylinder), and (22, green circular-container, green cylinder).

○ The Constrains

The robot can move one thing at a time, one cylinder can be placed in one circular-container, and a circular-container cannot be put on a cylinder. Circular-containers can be put in the holes on the grid; cylinders are not allowed to be put in holes.

- The PDDL (Planning Domain Definition Language) Definitions

The PDDL definition covers: domain description, problem description, predicates, and actions definitions. It includes the constraints described above, the ability to move cylinders between circular-containers, placing circular-containers on the grid, and ensuring color-matching between cylinders and circular-containers. The initial state and goal state are defined based on the scenario described earlier.

○ PDDL Domain Description

The PDDL code of the domain description is shown below. For the PDDL planning to work, more complex descriptions are needed. Here, a simplified version is presented.

```
(define (domain assembly-dom)
(:types
    hole cylinder circular-container bin color
)
(:predicates
    (hasColor ?obj - cylinder ?color - color)
    (contains ?bin - bin ?cylinder - cylinder)
    (empty ?bin - bin)
    (onGrid ?container - circular-container ?hole - hole)
    (placed ?cylinder - cylinder ?container - circular-container)
    (goal-state-reached)
)
(:action pick
    :parameters (?cylinder - cylinder ?bin - bin)
    :precondition (and (contains ?bin ?cylinder) (empty ?cylinder))
    :effect (and (not (contains ?bin ?cylinder)) (not (empty ?cylinder)))
)
(:action place
    :parameters (?cylinder - cylinder ?container - circular-container ?hole - hole)
    :precondition (and (empty ?container) (onGrid ?container ?hole) (not (placed ?cylinder
?container)))
    :effect (and (not (empty ?container)) (placed ?cylinder ?container)))
)
```

○ PDDL Problem Description

The basic code of the problem description is shown below. This, too, is a simplified version of the problem description used to illustrate the ontology support for the PDDL.

```
(define (problem assembly-prob) (:domain assembly-dom)
(:objects
    h00 h01 h02 h10 h11 h12 h20 h21 h22 - hole
    c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 - cylinder
    cc00 cc01 cc02 cc10 cc11 cc12 cc20 cc21 cc22 - circular-container
    bin-red bin-yellow bin-green - bin
    red yellow green - color
)
(:init
    ; initial state of bins
    (contains bin-red c1) (hasColor c1 red)
    (contains bin-red c2) (hasColor c2 red)
    (contains bin-red c3) (hasColor c3 red)
    (contains bin-red c4) (hasColor c4 red)
    (contains bin-red c5) (hasColor c5 red)
    (contains bin-red c6) (hasColor c6 red)
    (contains bin-yellow c7) (hasColor c7 yellow)
    (contains bin-yellow c8) (hasColor c8 yellow)
    (contains bin-yellow c9) (hasColor c9 yellow)
```

```
    (contains bin-yellow c10) (hasColor c10 yellow)
    (contains bin-yellow c11) (hasColor c11 yellow)
    (contains bin-green c13) (hasColor c13 green)
    (contains bin-green c14) (hasColor c14 green)
    (contains bin-green c15) (hasColor c15 green)
    (contains bin-green c16) (hasColor c16 green)
    (contains bin-green c17) (hasColor c17 green)
    ; Initial state of circular containers on the grid
    (onGrid cc00 h00) (hasColor cc00 yellow)
    (onGrid cc01 h01) (hasColor cc01 yellow)
    (onGrid cc02 h02) (hasColor cc02 red)
    (onGrid cc10 h10) (hasColor cc10 yellow)
    (onGrid cc11 h11) (hasColor cc11 yellow)
    (onGrid cc12 h12) (hasColor cc12 red)
    (onGrid cc20 h20) (hasColor cc20 green)
    (onGrid cc21 h21) (hasColor cc21 green)
    (onGrid cc22 h22) (hasColor cc22 green)
    ; Initial state of cylinders in circular containers
    (placed c12 cc11) (placed c18 cc21)
    ; Initial state of bins
    (not (empty bin-red)) (not (empty bin-yellow)) (not (empty bin-green))
)
(:goal (and
    (placed c7 cc00) (placed c8 cc01) (placed c1 cc02)
    (placed c9 cc10) (placed c12 cc11) (placed c2 cc12)
    (placed c13 cc20) (placed c18 cc21) (placed c14 cc22)
    (goal-state-reached)
))
)
```

- Ontology Representation for PDDL Domain and Problem

To use the ontology to support the PDDL, a mapping between the ontology elements and the PDDL constructs must be established. This means defining the PDDL concepts with the ontological format. Such an ontology can support the PDDL by providing a structured and semantically rich representation of the domain concepts and relationships. The mapping from PDDL to ontology is described below:

- Types (cylinders, circular containers, bins, holes, colors, and hangers) in the PDDL are represented as classes in the ontology (Cylinder, CircularContainer, Bin, Hole, Color, and Hanger).
- Objects are instances of types in the PDDL, and they are mapped to ontology with individuals. For example, individual cylinders such as c1, c2, and c3 are instances of the class Cylinder, and individual circular containers such as cc1, cc2, and cc3 are instances of the class CircularContainer.
- PDDL uses predicates to define relationships between objects, and these are described as object properties in the ontology. For example, the object property inContainer relates cylinders to circular containers, the object property inBin relates cylinders to bins, and the object property onGrid relates circular containers to holes on the grid.
- Initial state specifies the state of the world before the start in the PDDL. Ontology uses object property assertions to represent these individual instances. Individual instances in the ontology are associated with their properties using object property assertions. For example, individual cylinders like c1, c2, and c3 are associated with their colors, placement in bins, and circular containers using object property assertions. It is the same for the goal state.
- Actions in the PDDL represent possible transitions between states. They can be created as classes or properties in ontology. For example, the Pick action in the PDDL can be defined as the Pick Class.

- Constraints are rules and limitations for actions in the PDDL. In ontology, they are described using additional axioms or properties. For example, the constraint that "the container cannot be put on the cylinder" could be represented as an axiom in the ontology.
- The PDDL applies reasoning to create plans, while the ontology reasoning engines can infer new information based on axioms and assertions. For example, "a container is empty" is specified in the ontology, so the reasoning engine can infer and derive that the container is not holding any cylinders.

The ontology representation of the PDDL domain and problem is shown in Figure 17. This is a specific application. This PDDL ontology will be extended to a generic ontology to support PDD planning in general. The final result will be presented when this application ontology is completed and tested.

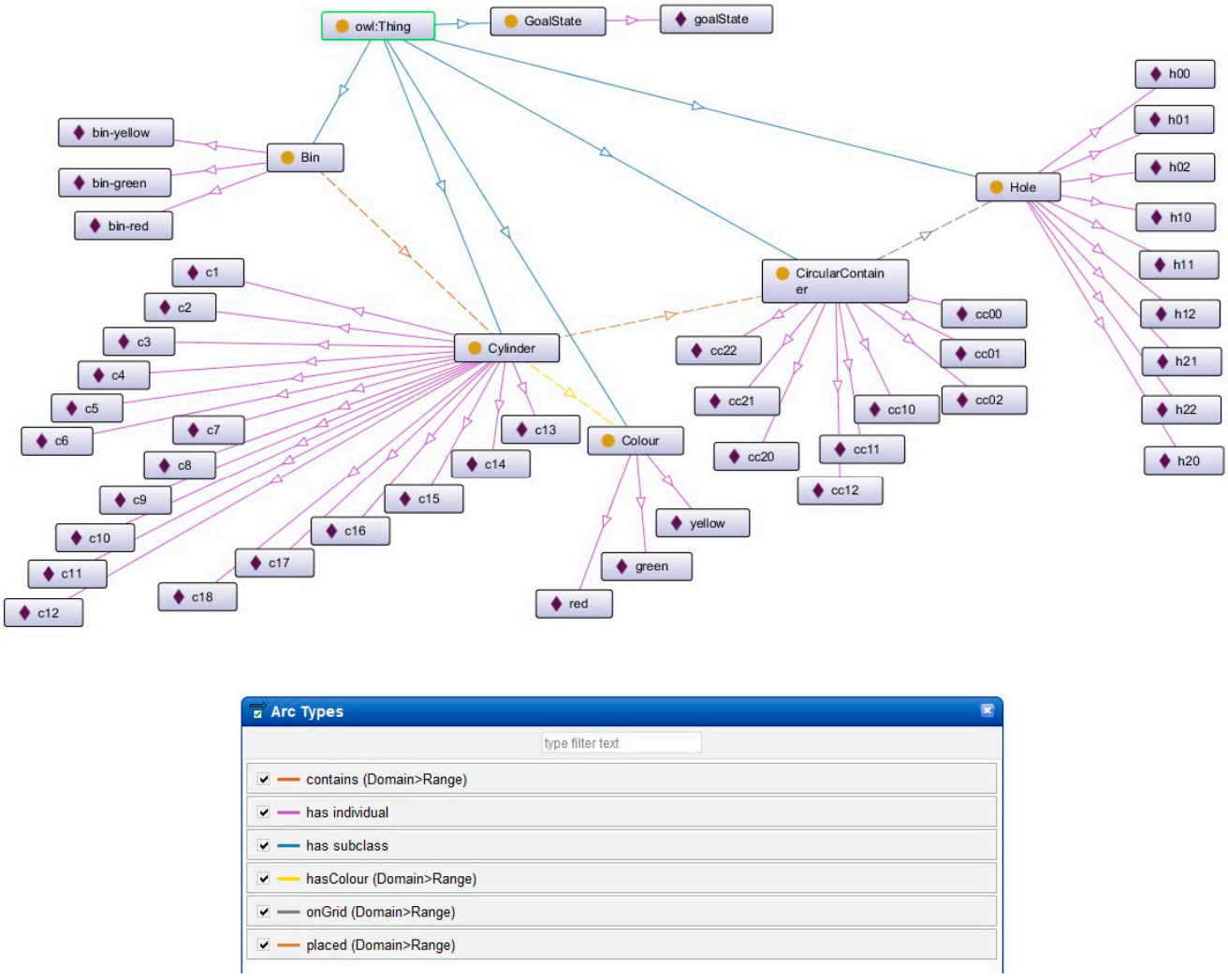


Figure 17. PDDL Ontology Relation Graph.

Aligning the PDDL and ontology allows for a more semantically meaningful representation of the plan domain, and therefore enables more intelligent planning systems supported by ontological reasoning capabilities. More complex planning scenarios need more expressive and detailed ontology. This scenario does not cover complex scenarios—for instance, allow cylinders placed in the wrong direction (top to bottom) in the initial state, allow random cylinders placed in bins (on top of each other), or allow human robot collaboration.

5. Conclusions

Robotic process management is no easy task as it involves numerous tasks. It is advantageous if a robot's knowledge representation combines different types of knowledge with reasoning capabilities, enabling it to devise the optimal plan of action to perform its duties [35]. While a few papers did focus on ontology-based task planning for robotics, further work is required to cover the full agile manufacturing domain. For ontology-based systems to be effective, standardization of the domain is key [23]. In this paper, the authors provide a state-of-art analysis of the existing ontology approaches and based on the results, provide a standardization of the robotic task planning domain. The major gap identified in the literature is the need to bridge the higher-level robotic process planning and the lower-level robotic control. Therefore, an ontology for robotic task planning (ORPP) was proposed. ORPP covers both industrial and collaborative robots in agile manufacturing. The major contributions of ORPP are as follows: (1) it details the concepts in the robotic manufacturing process and structures the concepts from the process level to the robot control level; (2) it creates traceability of requirements to KPI measurements of the process; (3) it extends the task layer to three task types (Robot Task, Human Task, and HRC Task); (4) it supports the PDDL task planning; (5) it is modular, reusable, and extensible, which facilitates knowledge sharing and reasoning; (6) it aligns with upper-level ontology, extends domain-level ontology, and adds application-level ontology.

6. Discussion

This ontology framework will be utilized and validated in the ACROBA project. However, the testing of the ontology can only be conducted in a limited way. The limitations are due to the partial implementation of the application ontologies. The PDDL ontology only covers one scenario and needs to be extended to cover other scenarios. The ACROBA, RTMN, and PDDL application ontologies are not implemented yet. For these reasons, the complete validation of the ontology needs to be performed when the whole integration is finished.

Additional usage of the ontology might be required in this period. When the authors finish the implementation, the authors will perform full validation tests on the five use cases of ACROBA.

Future research will be focused on Human–Robot Collaboration concepts that are now only covered in the ORPP ontology at a basic level. Richer knowledge about human safety and human factors will be added to extend the current ontologies.

Author Contributions: Conceptualization, C.Z.S., J.A.C.R. and N.U.B.; methodology, C.Z.S., J.A.C.R. and N.U.B.; validation, C.Z.S.; formal analysis, C.Z.S.; resources, C.Z.S.; data curation, C.Z.S.; writing—original draft preparation, C.Z.S.; writing review and editing, C.Z.S., J.A.C.R. and N.U.B.; visualization, C.Z.S.; supervision, J.A.C.R. and N.U.B.; project administration, C.Z.S. and N.U.B.; funding acquisition, J.A.C.R. and N.U.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by ACROBA. The ACROBA project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 101017284. Juan Antonio Corrales Ramón was funded by the Spanish Ministry of Universities through a "Beatriz Galindo" fellowship (Ref. BG20/00143) and by the Spanish Ministry of Science and Innovation through the research project PID2020-119367RB-I00.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: The authors would like to thank the ACROBA project for providing the opportunity and funding for this research. Special thanks to the ACROBA consortium for the contributions and support. The authors would like to extend their thanks also to Katrina Mugliett for her excellent assistance in reference management and text writing and formatting.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A List of Terms

Term	Source	Description
Physical	SUMO	An entity that has a location in space-time
Abstract	SUMO	Properties or qualities as distinguished from any particular embodiment of the properties/qualities in a physical medium. Instances of Abstract can be said to exist in the same sense as mathematical objects such as sets and relations, but they cannot exist at a particular place and time without some physical encoding or embodiment
Process	SUMO	The class of things that happen and have temporal parts or stages. Examples include extended events like a football match or a race, actions like pursuing and reading, and biological processes. The formal definition is anything that occurs in time but is not an object
Object	SUMO	Corresponds roughly to the class of ordinary objects. Examples include normal physical objects, geographical regions, locations of processes, and the complement of objects in the physical class
Region	SUMO	A topographic location. Regions encompass surfaces of objects, imaginary places, and geographic areas. Note that a region is the only kind of object that can be located at itself
Collection	SUMO	Collections have members like classes, but, unlike classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the collection
Agent	SUMO	Something or someone that can act on its own and produce changes in the world
Artifact	SUMO	An object that is the product of a making
Group	SUMO	A collection of agents
Device	SUMO	A device is an artifact whose purpose is to serve as an instrument in a specific subclass of a process
Quantity	SUMO	Any specification of how many or how much of something there is. Accordingly, there are two subclasses of quantity: number (how many) and physical quantity (how much)
Attribute	SUMO	Qualities that we cannot or choose not to reify into subclasses of object
Set or Class	SUMO	The set or class of sets and classes—i.e., any instance of abstract that has elements or instances
Relation	SUMO	The class of relations. There are three kinds of relation: predicate, function, and list. Predicates and functions both denote sets of ordered n-tuples. The difference between these two classes is that predicates cover formula-forming operators, while functions cover term-forming operators. A list, on the other hand, is a particular ordered n-tuple.
Proposition	SUMO	Propositions are abstract entities that express a complete thought or a set of such thoughts
Device	CORA(X)	Device is partitioned to Robot, Robot Part, Robot Interface, and Electric Device. Robots have other devices as parts
Robot	CORA(X)	a robot is a Device (SUMO term) that participates as a tool in a process. A robot is also an Agent which is something that can act on its own and produce changes. Robots perform tasks by acting on the environment/themselves
RobotPart	CORA(X)	RobotPart is further divided into robotActuatingPart, robotCommunicatingPart, robotProcessingPart and robotSensingPart
RobotInterface	CORA(X)	Robots interact with the world surrounding it through an interface. The RobotInterface is a device composed by other devices that play the roles of sensing device, actuating device, and communicating device. Through the interface, the robot can sense and act on the environment as well as communicate with other agents. Therefore, the robot interface can be viewed as a way to refer to all the devices that allow the robot to interact with the world. Every robot interface must have a part that is either a robot sensing part, a robot actuating part, or a robot communicating part
RobotGroup	CORA(X)	A robot is an agent, and agents can form social groups. According to SUMO, a group is "a collection of agents", like a pack of animals, a society, or an organization. A RobotGroup is a group whose only members are robots
ArtificialSystem	CORA(X)	ArtificialSystem is an Artifact formed by various devices (and other objects) that interact in order to execute a function. For any part of an artificial system, there is at least one other part it interacts with

Term	Source	Description
RoboticSystem	CORA(X)	Robots and other devices can form a RoboticSystem. A RoboticSystem is an artificial system formed by robots and devices intended to support robots to carry on their tasks. Robotic systems might have only one or more than one robot
Interaction	CORA(X)	An interaction is a Process in which two agents participate. It is composed of two subprocesses defining action and reaction. The action subprocess initiated by agent x on a patient agent y causes a reaction subprocess having y as an agent and x as a patient
RobotProcess	ORPP	RobotProcess is a Process. It has at least one Robot as an Agent participant. A RobotProcess consists of at least one RobotTask. It can be composed of HumanTasks, RobotTasks, and HumanRobotCollaborationTasks
Task	ORPP	Task is a Process. It is part of a RobotProcess. It has subclasses, RobtTask, HumanTask, and HumanRobotColloaborationTask
RobotTask	ORPP	RobotTask is a Task. It is part of a RobotProcess. It has at least one Robot as Agent. It cannot have Human as Agent. It consists of RobotSkills
HumanTask	ORPP	HumanTask is a Task. It is part of a RobotProcess. It has at least one Human as Agent. It cannot have Robot as Agent
HumanRobot CollaborationTask (HRC Task)	ORPP	HRCTask is a Task. It is part of a RobotProcess. It has at least one Human and one Robot as Agent. There are five types of HRC tasks: Coexistence Fence (CF), Sequential Cooperation SMS (SS), Teaching HG (TH), Parallel Cooperation SSM (PS), and Collaboration PFL (CP).
RobotSkill	ORPP	RobotSkill is a Process. It is part of a RobotTask. It can also be part of a RobotSkill. It consists of Primitives and/or other RobotSkills. RobotSkill has Robot as its Agent. Human cannot be the Agent of RobotSkill.
RobotPrimitive	ORPP	RobotPrimitive is a Process. It is part of a RobotSkill. RobotPrimitive is atomic. RobotPrimitives has Robot or RobotPart as Agent. Human cannot be the Agent of RobotPrimitive
HumanRobot Group	ORPP	A Robot or a Human is an Agent, and agents can form social groups. According to SUMO, a group is "a collection of agents". A HumanRobotGroup is a group that has human and robots as members. It participates in HumanRobotCollaborationTasks
Requirement	ORPP	Requirement is a subclass of Proposition. Requirements evaluate Tasks or Processes. Requirement has KPI to specify it. A Task or Process can have multiple Requirements
KPI	ORPP	KPI is a subclass of Proposition. KPIs specify Requirements. A Requirement can have multiple KPIs. KPI and Requirement are a disjoint subclass of Proposition
Human	ORPP	Human is an Agent who can act on his own and produce changes. It participates in HumanTask or HumanRobotCollaborationTask. Human performs these tasks by acting on the environment/themselves

References

1. Diab, M.; Pomarlan, M.; Borgo, S.; Beßler, D.; Rossel, J.; Bateman, J.; Beetz, M. FailRecOnt-An Ontology-Based Framework for Failure Interpretation and Recovery in Planning and Execution. In Proceedings of the 2nd International Workshop on Ontologies for Autonomous Robotics, Bolzano, Italy, 11–18 September 2021.
2. Pantano, M.; Eiband, T.; Lee, D. Capability-based Frameworks for Industrial Robot Skills: A Survey. In Proceedings of the 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 20–24 August 2022; pp. 2355–2362.
3. Manzoor, S.; Rocha, Y.G.; Joo, S.H.; Bae, S.H.; Kim, E.J.; Joo, K.J.; Kuc, T.Y. Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications. *Appl. Sci.* **2021**, *11*, 4324. [CrossRef]
4. Aguado, E.; Sanz, R.; Rossi, C. Ontologies for Run-Time Self-Adaptation of Mobile Robotic Systems. In Proceedings of the XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA), Málaga, Spain, 22–24 September 2021.
5. Pang, W.; Gu, W.; Li, H. Ontology-Based Task Planning for Autonomous Unmanned System: Framework and Principle. *J. Phys. Conf. Ser.* **2022**, *2253*, 012018. [CrossRef]
6. ACROBA. Available online: https://acrobaproject.eu/project-acroba/ (accessed on 28 February 2022).
7. *IEEE Std 1872-2015*; IEEE Standard Ontologies for Robotics and Automation. IEEE: Piscataway, NJ, USA, 2015; p. 45.
8. Niles, I.; Pease, A. Towards a Standard Upper Ontology. In Proceedings of the Second International Conference on Formal Ontology in Information Systems, Ogunquit, ME, USA, 17–19 October 2001; pp. 2–9. [CrossRef]

9. Pedersen, M.R.; Nalpantidis, L.; Andersen, R.S.; Schou, C.; Bøgh, S.; Krüger, V.; Madsen, O. Robot Skills for Manufacturing: From Concept to Industrial Deployment. *Robot. Comput.-Integr. Manuf.* **2016**, *37*, 282–291. [CrossRef]
10. Olivares-Alarcos, A.; Beßler, D.; Khamis, A.; Goncalves, P.; Habib, M.K.; Bermejo-Alonso, J.; Barreto, M.; Diab, M.; Rosell, J.; Quintas, J.; et al. A Review and Comparison of Ontology-Based Approaches to Robot Autonomy. *Knowl. Eng. Rev.* **2019**, *34*, e29. [CrossRef]
11. Nilsson, A.; Muradore, R.; Nilsson, K.; Fiorini, P. Ontology for Robotics: A Roadmap. In Proceedings of the 2009 International Conference on Advanced Robotics, Munich, Germany, 22–26 June 2009.
12. Buisan, G.; Sarthou, G.; Bit-Monnot, A.; Clodic, A.; Alami, R. Efficient, Situated and Ontology Based Referring Expression Generationfor Human-Robot Collaboration. In Proceedings of the 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Naples, Italy, 31 August–4 September 2020; ISBN 9781728160757.
13. Olivares-Alarcos, A.; Foix, S.; Borgo, S.; Alenyà, G. OCRA—An Ontology for Collaborative Robotics and Adaptation. *Comput. Ind.* **2022**, *138*, 103627. [CrossRef]
14. Umbrico, A.; Orlandini, A.; Cesta, A. An Ontology for Human-Robot Collaboration. *Procedia CIRP* **2020**, *93*, 1097–1102. [CrossRef]
15. Umbrico, A.; Cesta, A.; Orlandini, A. Deploying Ontology-Based Reasoning in Collaborative Manufacturing. 2022. Available online: https://www.semantic-web-journal.net/system/files/swj3024.pdf (accessed on 12 July 2024).
16. Sarthou, G.; Clodic, A.; Alami, R. Ontologenius: A Long-Term Semantic Memory for Robotic Agents. In Proceedings of the 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 14–18 October 2019.
17. Olszewska, J.I.; Barreto, M.; Bermejo-Alonso, J.; Carbonera, J.; Chibani, A.; Fiorini, S.; Goncalves, P.; Habib, M.; Khamis, A.; Olivares, A.; et al. Ontology for Autonomous Robotics. In Proceedings of the RO-MAN 2017 26th IEEE International Symposium on Robot and Human Interactive Communication, Lisbon, Portugal, 28 August–1 September 2017; pp. 189–194. [CrossRef]
18. Booch, G.; Jacobson, I.; Rumbaugh, J. *The Unified Modeling Language for Object-Oriented Development*; Documentation Set Version 0.91Addendum UML Update; Rational Software Corporation: Santa Clara, CA, USA, 1996.
19. Yuguchi, A.; Nakamura, T.; Toyoda, M.; Yamada, M.; Tulathum, P.; Aubert, M.; Garcia Ricardez, G.A.; Takamatsu, J.; Ogasawara, T. Toward Robot-Agnostic Home Appliance Operation: A Task Execution Framework Using Motion Primitives, Ontology, and GUI. *Adv. Robot.* **2022**, *36*, 548–565. [CrossRef]
20. Beetz, M.; Bessler, D.; Haidu, A.; Pomarlan, M.; Bozcuoglu, A.K.; Bartels, G. Know Rob 2.0—A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2018; pp. 512–519.
21. Tenorth, M.; Beetz, M. KNOWROB—Knowledge Processing for Autonomous Personal Robots. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, St. Louis, MO, USA, 10–15 October 2009; pp. 4261–4266. [CrossRef]
22. Weser, M.; Bock, J.; Schmitt, S.; Perzylo, A.; Evers, K. An Ontology-Based Metamodel for Capability Descriptions. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; ISBN 9781728189567.
23. Perzylo, A.; Grothoff, J.; Lucio, L.; Weser, M.; Malakuti, S.; Venet, P.; Aravantinos, V.; Deppe, T. Capability-Based Semantic Interoperability of Manufacturing Resources: A BaSys 4.0 Perspective. *IFAC-PapersOnLine* **2019**, *52*, 1590–1596. [CrossRef]
24. Schäfer, P.M.; Steinmetz, F.; Schneyer, S.; Bachmann, T.; Eiband, T.; Lay, F.S.; Padalkar, A.; Sürig, C.; Stulp, F.; Nottensteiner, K. Flexible Robotic Assembly Based on Ontological Representation of Tasks, Skills, and Resources. In Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, Online, 3–12 November 2021.
25. Saxena, A.; Jain, A.; Sener, O.; Jami, A.; Misra, D.K.; Koppula, H.S. RoboBrain: Large-Scale Knowledge Engine for Robots. 2014. Available online: https://ozansener.net/papers/robobrain.pdf (accessed on 12 July 2024).
26. Falbo, R.d.A. SABiO: Systematic Approach for Building Ontologies. ONTO.COM/ODISE@FOIS 2014. Available online: https://dblp.org/rec/conf/fois/Falbo14.html (accessed on 12 July 2024).
27. Caro, M.F.; Cox, M.T.; Toscano-Miranda, R.E. A Validated Ontology for Metareasoning in Intelligent Systems. *J. Intell.* **2022**, *10*, 113. [CrossRef] [PubMed]
28. Zhang Sprenger, C.; Ribeaud, T. Robotic Process Automation with Ontology-Enabled Skill-Based Robot Task Model and Notation (RTMN). In Proceedings of the IEEE International Conference on Robotics, Automation and Artificial Intelligence (RAAI 2022), Singapore, 9–11 December 2022.
29. Zhang Sprenger, C.; Corrales Ramón, J.A.; Urs Baier, N. RTMN 2.0—An Extension of Robot Task Modeling and Notation (RTMN) Focused on Human–Robot Collaboration. *Appl. Sci.* **2024**, *14*, 283. [CrossRef]
30. Ghallab, M.; Knoblock, C.A.; Wilkins, D.E.; Barrett, A.; Christianson, D.; Friedman, M.; Kwok, C.; Golden, K.; Penberthy, S.; Smith, D.E.; et al. PDDL-The Planning Domain Definition Language. Domain-Specific Insight Graphs (DIG) View Project Ability-Based Design View Project SEE PROFILE PDDL | The Planning Domain Deenition Language. 1998. Available online: https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf (accessed on 12 July 2024).
31. Borgo, S.; Cesta, A.; Orlandini, A.; Umbrico, A. Knowledge-Based Adaptive Agents for Manufacturing Domains. *Eng. Comput.* **2019**, *35*, 755–779. [CrossRef]

32. Schou, C.; Skovgaard Andersen, R.; Chrysostomou, D.; Bøgh, S.; Madsen, O. Skill-Based Instruction of Collaborative Robots in Industrial Settings. *Robot. Comput.-Integr. Manuf.* **2018**, *53*, 72–80. [CrossRef]
33. Topp, E.A.; Stenmark, M.; Ganslandt, A.; Svensson, A.; Haage, M.; Malec, J. Ontology-Based Knowledge Representation for Increased Skill Reusability in Industrial Robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
34. Protégé. Available online: https://protege.stanford.edu/ (accessed on 9 May 2023).
35. Paulius, D.; Sun, Y. A Survey of Knowledge Representation in Service Robotics. *Robot. Auton. Syst.* **2019**, *118*, 13–30. [CrossRef]