Semantic Web 15 (2024) 389–428
DOI 10.3233/SW-233394
IOS Press

Enhancing awareness of industrial robots in collaborative manufacturing

Alessandro Umbrico *, Amedeo Cesta and Andrea Orlandini
Institute of Cognitive Sciences and Technologies, National Research Council, Rome, Italy

Editors: Bahar Aameri, University of Toronto, Canada; María Poveda-Villalón, Universidad Politécnica de Madrid, Spain; Emilio M. Sanfilippo, ISTC-CNR Laboratory for Applied Ontology, Italy; Walter Terkaj, STIIMA-CNR, Italy
Solicited reviews: Daniel Beßler, University of Bremen, Germany; Alessandro Mosca, Free University of Bozen-Bolzano, Italy; Baifan Zhou, University of Oslo, Norway; two anonymous reviewers

Abstract. The diffusion of Human-Robot Collaborative cells is prevented by several barriers. Classical control approaches seem not yet fully suitable for facing the variability conveyed by the presence of human operators beside robots. The capabilities of representing heterogeneous knowledge representation and performing abstract reasoning are crucial to enhance the flexibility of control solutions. To this aim, the ontology SOHO (Sharework Ontology for Human-Robot Collaboration) has been specifically designed for representing Human-Robot Collaboration scenarios, following a context-based approach. This work brings several contributions. This paper proposes an extension of SOHO to better characterize behavioral constraints of collaborative tasks. Furthermore, this work shows a knowledge extraction procedure designed to automatize the synthesis of Artificial Intelligence plan-based controllers for realizing flexible coordination of human and robot behaviors in collaborative tasks. The generality of the ontological model and the developed representation capabilities as well as the validity of the synthesized planning domains are evaluated on a number of realistic industrial scenarios where collaborative robots are actually deployed.

Keywords: Ontology, knowledge representation and reasoning, Human-Robot Collaboration, automated planning and scheduling, Artificial Intelligence

1. Introduction

Nowadays, robots are successfully deployed in a large spectrum of real-world applications. Nevertheless, robots require an increased level of autonomy and additional features to operate in "open environments" guaranteeing reliable and safe interactions. These constitute major scientific challenges and many research activities are ongoing to address them. In manufacturing, open research challenges concern in particular the design of control systems capable of robustly anticipating changes in production requirements and goals [98]. Higher levels of flexibility and adaptability of industrial robots are crucial to face the challenges of Industry 4.0 [17,34]. Industry 4.0 [37] is indeed pushing manufacturing systems towards customer-oriented and personalized production while trying to guarantee the advantages of mass production systems in terms of both productivity and costs [61]. Future manufacturing systems should in other words evolve towards flexible production embracing changes to the needs, requirements, and objectives of the factory [85]. Research in Human-Robot Collaboration (HRC) pursues these challenging objectives by investigating the tight and symbiotic collaboration between human workers and autonomous (or semi-autonomous)

*Corresponding author. E-mail: alessandro.umbrico@istc.cnr.it.

robots. Novel production paradigms that see humans and robots working side-by-side as interchangeable resources, thus combining the precision and tirelessness of the former with the problem-solving skills of the latter [32,97].

Classical control approaches usually rely on static models of robot skills and a static (or hard-coded) description of production requirements and objectives. Such technology is not fully able to support the level of (flexible) autonomy that future production environments need. This is especially true in HRC where robots and humans share the working space and physically interact together to achieve common objectives. Human actors introduce a significant source of *uncertainty* that robot controllers should properly take into account in order to safely cooperate with them while supporting production [54]. It is crucial to investigate novel control approaches implementing advanced *cognitive capabilities* and allow robots to achieve a higher level of *awareness* about themselves, their "peer companions" as well as the production context and related dynamics e.g., production procedures, task requirements, needed skills and capabilities of actors taking part to production processes.

Research in Artificial Intelligence (AI) designs and develops technologies that are suitable to realize the desired cognitive capabilities. The integration of AI and Robotics in particular [48,73] would allow (collaborative) robots to: (i) *perceive* the environment by correctly *interpreting* events and situations; (ii) *build* and maintain *knowledge* about the production context; (iii) *reason* about their own *capabilities*/*skills* and dynamically contextualize possible actions to the *perceived* state of a production scenario; (iv) *autonomously decide* how to act/interact with the environment and other "actors" (i.e., human operators but also other robots if necessary) to support production and; (v) *adapt* behaviors over time according to the "learned experience" and evolving production needs.

Our long-term research objective is to enrich robot controllers with an AI-based "perceive-reason-act" paradigm implementing advanced cognitive features. The envisaged cognitive control approach would allow a collaborative robot to be *aware* of the production context and autonomously decide which tasks are needed and how to execute them in order to collaborate with human workers and support production in the best way possible. For example, the way a procedure is executed may depend on several factors. The tool a worker needs for implementing the tasks could be damaged or not available. The needed resources e.g., bolts or other pieces could not be available or insufficient. The worker joining the production today could not know well the procedure because of low experience. In all these cases (and others), a robot with enhanced awareness of the production context would autonomously adapt its behavior and the execution of collaborative processes to different situations. Suppose for example that the worker cannot execute some tasks because the needed screwdriver is not available, the robot, *knowing* this *fact* and *knowing* that it can perform the same tasks using its own screwdriver (i.e., the robot *knows* its capabilities), it would autonomously synthesize a collaborative plan by assigning the tasks requiring the screwdriver to itself. This level of autonomy and flexibility could not be achieved by classic control approaches without interrupting production and manually fixing the control procedure of the robot.

To pursue this level of awareness and cognitive control, we investigate the integration of AI-based Knowledge Representation & Reasoning with Automated Planning and Execution. The combination of these AI technologies with Robotics has shown promising results in heterogeneous scenarios ranging from service and assistive robotics [4,22,53,90] to Reconfigurable Manufacturing Systems [7,13,30,77], improving flexibility of robot behaviors. Semantic technologies are crucial to endow robot controllers with the *theoretical context* necessary to *represent* (heterogeneous) information coming from different sources (e.g., deployed sensing devices or domain expert knowledge about production processes) and *reason* about the resulting *knowledge* in order to *understand* the state of a production environment and make *contextualized decisions*.

This paper advances a recent work by refining an ontological model for Human-Robot Collaboration in manufacturing [91] and investigating the integration between Knowledge Representation & Reasoning and Automated Planning in order to enhance *awareness*, *adaptability* and *flexibility* of collaborative robots. In particular, the work better explain the human factor ontological context and the use of *ontology design patterns* [40] as a means to facilitate the description of collaborative dynamics. The human factor context describes the concepts and properties that characterize skills and qualities of human workers. This knowledge is useful to adapt collaborative processes to the different features of human workers. Ontology patterns are defined according to consolidated schemes of interaction between humans and robots in HRC [44,59]. Similarly to software design patterns, ontology design patterns are used to narrow knowledge design choices and define sufficiently general and reusable concepts characterizing behavior constraints between a human and a robot when performing collaborative tasks [44,59]. Robot *awareness* is achieved

through designed knowledge extraction procedures that automatically synthesize (and online adapt) plan-based control models. Here ontological patterns are translated into sets of causal and temporal constraints that comply with the desired "shape" of the collaboration between the human and the robot.

The conjunction of AI-based planning and semantic technologies realizes *cognitive skills* suitable to enhance *autonomy* and *context awareness* of industrial (collaborative) robots as well as robustly deal with evolving production needs (e.g., changing production requirements, changing capabilities of a robotic platform or changing skills of human operators, etc.). The validity and generality of the proposed approach are evaluated on a number of realistic HRC scenarios, pilots of the EU H2020 research project Sharework.[1] These scenarios concern different types of production, involving different production entities, tools, objects, and procedures. The assessment shows that the ontology-based control approach effectively supports the definition of valid and complete *production knowledge*. The automatically synthesized task planning models have been concretely and effectively used to coordinate human and robot operations.

The paper is structured as follows. Section 2 discusses related works concerning the integration of knowledge reasoning and automated planning with robotics. It highlights how other researchers are investigating the integration of the mentioned technologies to enhance the flexibility, adaptability, and (social) context awareness of robot behaviors. Section 3 discusses the Sharework Ontology for Human-Robot Collaboration (SOHO) initially introduced in [91]. SOHO defines the representational space of the proposed approach and this section provides a complete and refined definition of its concepts and properties. Section 4 briefly introduces the timeline-based planning formalism and then describes the developed knowledge extraction procedure. This procedure is the central point linking knowledge to task planning. It supports the automatic update and adaptation of the plan-based control model to the contextualized knowledge of a (collaborative) production scenario. Section 5 evaluates the proposed approach on a number of realistic scenarios taken from the pilot cases of the EU H2020 research project Sharework. On the one hand, the evaluation shows the capability of capturing all relevant aspects of collaborative scenarios. On the other hand, it shows the feasibility and correctness of the knowledge extraction procedure. Finally, Section 6 summarizes the contribution of the paper pointing out possible directions for future developments.

2. Knowledge representation and reasoning in robotics

Robotics and Artificial Intelligence (AI) are two research areas that historically addressed the challenge (among others) of building embedded intelligent systems capable of acting in a real-world environment [73]. Recent advancements in Robotics and AI are pushing the design and deployment of autonomous robots in increasingly complex/unstructured environments. On the one hand, technological advancements concerning the increased reliability and efficiency of sensing devices, manipulation and navigation skills of robots as well as solving and predictive capabilities of AI technologies open new opportunities for the effective deployment of Robotics and AI solutions. On the other hand, the increased complexity of application scenarios raises new technical/methodological challenges.

A tight integration of Robotics and AI is crucial to enhance the autonomy and control capabilities of robots and allow them to safely and reliably *act* in the real-world [41,48]. In particular, robots acting in the real world should take into account a number of "non-functional" qualities that are crucial to realize behaviors that are safe and acceptable with respect to humans [15,28,75]. Robot controllers should therefore evolve towards an advanced "Perception, Reason, Act" paradigm implementing the cognitive capabilities needed to synthesize and execute flexible behaviors that are valid from both a technical and social point of view. To implement the envisaged cognitive control paradigm, we found particularly promising the integration of ontology-based reasoning with AI-based planning and robot control. The integration of semantic technologies with robot controllers has been widely studied in the literature [56].

Ontologies have been recognized as true enablers of adaptable and flexible systems compared to classic approaches [21,87]. Robot-integrated ontology-based reasoning has in particular shown effective results in the enhancement of robot flexibility and *awareness* [7,13,53]. This section discusses some relevant works in the literature

[1] https://sharework-project.eu

concerning the integration of ontology-based knowledge reasoning, planning, and robotics. It shows the enhanced flexibility and awareness of robots acting in different domains, thanks to the integration of the mentioned technologies.

2.1. Ontology in robotics and human-robot interaction scenarios

KnowRob [6,84] is a well-known framework supporting advanced Perception, Reasoning, and Control. The framework provides robots with a logical representation of a number of entities ranging from robotic parts and objects (with their composition and functionalities) to tasks, actions, and behaviors. This framework in particular focuses on manipulation tasks and allows robots to perceive objects of the environment, reason about their functionalities (e.g., formal description of affordances of objects [8]), and decide how to use them within planning actions [7,31]. Although general, KnowRob is mainly suitable to deal with scenarios where a single robot manipulates objects and interacts with an environment. The dyadic nature of HRC scenarios requires reasoning on simultaneous executions of actions and the synergetic combination of robotic and human actors.

An ontological model characterizing object manipulation tasks of robots has been also considered within the PMK framework [30]. Similar to KnowRob [6], PMK supports a standardized representation of the environment defining a common language to exchange information between a human and a robot. It characterizes causal information and constraints about manipulation tasks well and defines knowledge that is useful at both task and motion planning levels. The work [82] exploits a robot knowledge framework (OMRKF) consisting of a series of ontology layers, including a robot-centered and human-centered ontology. The system in particular relies on an object layer, a context layer, and an activity layer to abstract gathered sensor data. However, this framework lacks a foundational background which limits the reliability of inferred knowledge. Furthermore, it does not distinguish between activity and functionality resulting in a rigid characterization of robot capabilities and behaviors. For example *avoid obstacle* is not a behavior but a function that can be implemented in different ways like, e.g., moving away or turning around.

An interesting framework concerning the ontological description and integration of robotic skills is SkiROS [76]. Less general than KnowRob, this framework is designed on ROS with the objective of proposing an ontological model of robot skills. On top of this knowledge, action-based planning supports a dynamic combination of skills to realize complex behaviors. Similar to KnowRob [84], SkiROS focuses on the description/control of a single robot acting in the environment. Concerning HRC scenarios, this framework does not support an explicit representation of the skills of multiple agents, concurrency, time, and controllability issues.

The ORO framework [52] develops a knowledge reasoning framework endowing robots with common sense reasoning capabilities to autonomously operate in semantically-rich human environments. With respect to KnowRob, the ORO framework addresses the control problem from a cognitive perspective and realizes a general cognitive architecture deployed on different robotic platforms and assessed on different cognitive scenarios [52]. This architecture has been specifically developed to support advanced cognitive skills (e.g., *theory of mind* capabilities) and thus support increasingly flexible and adaptive human-robot interactions [53]. Considering an HRC perspective, ORO pursues a turn-based interaction approach where the human and the robot are supposed to perform one action at a time with the robot reacting to the observed behavior and inferred state of the human. This interaction mechanism is not fully effective in production scenarios that require cooperation and simultaneous action execution of the human and a robot to achieve shared objectives.

Concerning human-robot social interactions, knowledge representation, and reasoning have been used to realize socially compliant behaviors. Non-functional requirements like those regarding social norms are crucial to realize acceptable behaviors [75]. For example, the work [4] uses knowledge reasoning to represent social norms and allow a social robot to implement socially acceptable behaviors for social tasks. More specifically, the work proposes a formal description of the functional affordances of objects to reason about their possible use and thus infer those that are suitable to accomplish the requested social task (i.e., serving coffee to guests using the right object). The work [15] proposes the use of knowledge reasoning to adapt human-robot interactions to the cultural knowledge of different contexts and people. This is another example of how ontology-based reasoning can enhance *context awareness* of robots. In this case, reasoning capabilities evaluate non-functional qualities of human-robot interactions and synthesize socially compliant behaviors. Another interesting work is [22]. Similar to other works e.g., KnowRob [84], it proposes an ontological model characterizing users, the interacting environment, capabilities (not necessarily

correlated to the robot only), and tasks. On top of this model, the work instantiates a cognitive architecture realizing a perceive, reason, act control loop. The resulting "cognitive agent" incrementally decides which social task to perform according to the perceived state of the interaction context. An added value of [22] is the explicit representation of interacting users through user profiles that support personalized services.

Ontology-based reasoning has been used also in medical scenarios. The work [42] proposes the use of ontology in orthopedic surgery. The ontological model OROSU integrates and shows domain knowledge to different types of users uniformly e.g., surgeons, nurses, or technicians, during surgery. It relies on KnowRob [84] to describe surgical procedures concerning hip surgery. Finally, ontology-based reasoning has been used also to formally represent normative standards and evaluate compliance with them. An example is the work [74] where normative standards for indoor environmental qualities have been encoded into an ontological model. A social robot has been endowed with cognitive capabilities to ground detected quality conditions of an environment and automatically evaluate the compliance of a perceived environment to the normative standards.

2.2. *Ontology in manufacturing*

In manufacturing, ontologies have mainly focused on the manufacturing system as a whole or rather on specific production aspects e.g., [78,86,94]. Modeling procedures, capabilities of working entities, and possible interactions connected to production objectives are challenging. The description of so-called Cyber-Physical Systems (CPSs) (e.g., HRC Systems) requires modeling the dynamics of the involved agents from both a local perspective (i.e., the point of view of a specific agent) and global perspective (i.e., the point of view of the production) [11]. In this context, ontologies have been mainly applied with the aim of increasing flexibility in modeling and planning of, e.g., mechatronic devices [5], resources in collaborative environments, and the whole enterprise [80], collaborative robots [46] and navigation robots [16].

The work [5] uses an ontology to collect static and dynamic information relative to robots. The basic actions of a robot are hard-coded but the ontological system adds some flexibility like the possibility to learn articulated actions and to act with partial information, e.g., information about the location of the object to move. Besides the lack of functionality/activity distinction, the robot has very limited knowledge of the environment. The work [7] uses a well-structured ontological model to characterize product assembly tasks. Specifically, the work extends the KnowRob framework [84] by integrating inference rules necessary to reason about incomplete assemblies of different products. Based on the outcome of the implemented perception and reasoning processes they automatically plan the next action to be executed and incrementally assemble the desired products. The work [51] uses ontology to represent kit-building parts for assembly operations. Kit building parts are presented by means of XML descriptions whose schema (XSDL) is mapped to an ontological model providing a uniform logic-based representational space. The ontological model and the automatic generation of OWL descriptions from XML schema are then used within an agility framework to evaluate the agility performance of robotic systems.

Concerning collaborative manufacturing, the work [66] proposes an ontological model called OCRA which takes into account uncertainty and safety constraints. Interestingly, it characterizes reliable human-robot collaborations providing robots with a well-structured formalization suitable to reason on the execution perspective of their plans. The work [77] realizes an ontology-based multi-agent system integrated with a Business Rule Management System to define a language for the coordination of human and robotic agents. The ontological model does not rely on a structured theoretical background and the knowledge about tasks and agents' capabilities is hard-coded. Furthermore, it proposes a limited, and schematic description of a collaborative environment. For example, workers and cobots are represented with a simple schema describing just their location within the environment, no additional information about their capabilities, composition, or behavioral features is provided.

Works within the ROSETTA project [68] have also investigated the use of ontology-based reasoning to simplify the programming of industrial robots and interactions with humans. The work [81] uses and extends the SIARAS ontology to characterize knowledge about robot skills focusing in particular on manipulation skills and devices that may compose a manufacturing environment (e.g., gripper, fixture, the robot itself). The framework aggregates several ontological models that are relevant also from a human-robot collaboration perspective e.g., the *injury.owl* which characterizes the expected risk level of injury when a robot cooperates with a human or shares the same environment.

3. An ontology for Human-Robot Collaboration

Considering the discussed literature, it is still missing an ontological model capable of capturing the capabilities of different types of resources, actors with different interacting features, and production requirements. Enriching manufacturing systems with such a semantically rich model would: (i) increase their level of awareness about the state and needs of a production environment; (ii) autonomously interpret production events and properly coordinate acting resources e.g., collaborative robots and workers and; (iii) dynamically adapt production processes to the skills and working and/or health conditions of collaborating workers. The work [91] made a first step toward the definition of an ontological model specifically designed for Human-Robot Collaboration (HRC). SOHO (*Sharework Ontology for Human-Robot Collaboration*) is a *domain ontology* [43] aiming at characterizing HRC scenarios from different but synergetic levels of abstraction (contexts). The objective is to define a well-structured model of production environments, human, machine, and robot structures, capabilities, and functional operations.

SOHO pursues a flexible interpretation of these concepts in order to interpret production states/situations according to the specific needs of processes and features of the environment. For example, operational capabilities (or simply capabilities) of robots and workers intrinsically depend respectively on their structures (e.g., actuators and end-effector that are part of the robotic device – *embodiment*) and their skills or abilities (e.g., a worker can perform specific welding operations). These capabilities, combined with the specific needs and requirements of a production environment, would enable the execution of different (instances of) functions [12]. This section provides a complete and refined description of SOHO,[2] defined using Protégé [3] and the OWL language [3].

3.1. Foundations and contexts

Foundational ontologies aim at describing reality from a high-level perspective in order to define concepts that are general enough to be valid across many domains. Their use is generally recommended and represents a good design choice in order to base new (more specific) ontological models on well-structured semantics. These ontological models constitute a stable theoretical background of more specific ontologies and thus foster a clear structuring and disambiguation of domain concepts [49,58]. Several foundational ontologies have been introduced in the literature e.g., BFO [67], DOLCE [14] or SUMO [65]. Among these, SOHO relies on DOLCE [14] in order to support a flexible interpretation of temporally evolving entities and, also, to rely on a recognized standard representation framework (ISO 213838-3). DOLCE, therefore, represents a flexible model, well suited to support the interpretation of domain entities whose state depends on the context and may change/evolve over time.

In addition to DOLCE, SOHO is built on top of two other ontologies: (i) the CORA ontology [71] and; (ii) the SSN ontology [27]. CORA is an IEEE standard ontology for robotics and automation aiming at promoting a common language in the robotics and automation domain. It characterizes knowledge about robots and robot parts, robot positions and configurations, and groups of robots. This standard relies on SUMO [65] as a theoretical foundation and integrates the framework ALFUS [47] to define possible autonomy levels and related operative modes of a robot. SSN is a W3C standard ontology for IoT devices and sensor networks. It defines basic concepts and properties characterizing the capabilities of sensing devices, their deployment into a physical environment, and the outcome of sensing processes. SSN relies on DOLCE and defines a sufficiently general model to represent the physical properties of an environment and physical entities that can be observed or monitored over time.

Both CORA and SSN define concepts and properties relevant to HRC but they are not sufficient to describe production procedures and possible collaborations needed between human and robot agents. The scope of SSN is limited to the characterization of a physical environment in terms of properties that can be observed and sensing devices that carry out sensing processes. This ontology is quite "self-contained" and can be easily integrated with CORA to represent also robot interfaces and sensing parts. CORA instead has a broader scope. It focuses on robot parts, robot configurations, and levels of autonomy. However, CORA does not support the contextualization and interpretation of behaviors of robots and other autonomous agents (e.g., human operators) with respect to global production objectives and processes.

[2]The ontology is publicly available at the following GitHub repository – https://github.com/pstlab/SOHO.git.
[3]https://protege.stanford.edu

3.1.1. Qualities, norms, and events

Human-Robot Collaboration scenarios are a combination of *technical*, *physical*, and *social contexts* since the acting entities should physically interact while complying with a number of *rules* that guarantee correct and safe execution of production processes. Indeed, an HRC scenario is composed of a number of *physical entities* each characterized by different *features* and *qualities* that cooperate to achieve common (production) objectives. SOHO, therefore, interprets HRC scenarios as *social contexts* where the behavior of each acting entity affects the behavior of others, and coordination is necessary to correctly and safely carry out production processes. As such, any HRC scenario is subject to social structures known as norms [9] either implicit or explicit *rules*, that constrain the behavior of involved actors. To model concepts and properties that suitably capture such dynamics SOHO relies on the DOLCE+DnS Ultralite ontology (DUL) [4] which is a lightweight version of DOLCE suitable to model either physical or social contexts. DUL uses simplified constructs to represent temporal and spatial relations and supports modular, pattern-based, structures (*content ontology design pattern*).

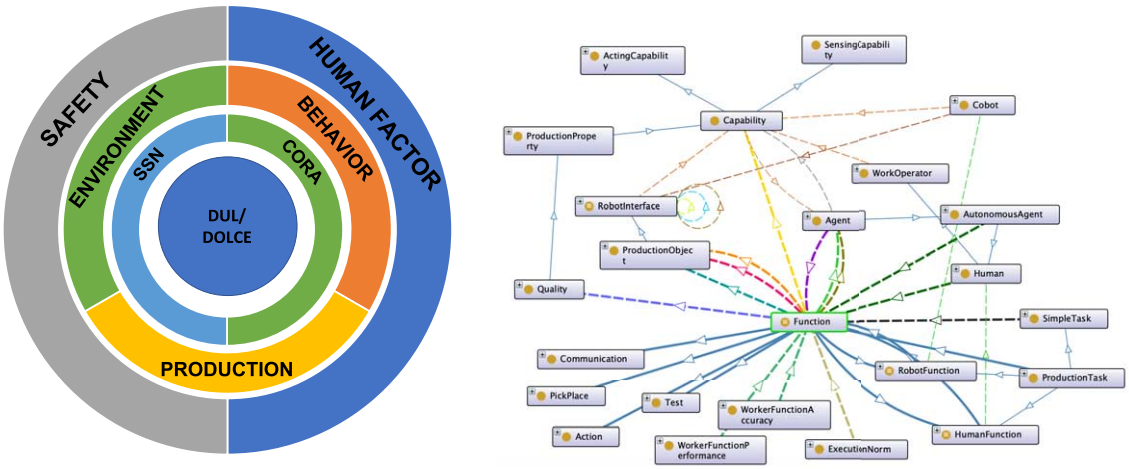
The concept `DUL:Object` models any physical, social, or mental object or a substance of the domain. SOHO in particular considers the sub-concept `DUL:Agent` and `DUL:PhysicalObject` to characterize respectively acting entities of the domain (i.e., collaborative robots and human workers) and passive physical elements that are part of a production environment (e.g., tools, robot parts, sensing devices, production resources, etc.). The concept `DUL:Agent` characterizes any agentive object either physical (e.g., a robot) or social (e.g., an institution) that behaves according to some logic/algorithm. The concept of `DUL:Agent` is equivalent to the concept `DUL:PhysicalAgent` that is a particular type of `DUL:PhysicalObject` which is in turn any object associated with a space region. The concept `DUL:PhysicalObject` is thus useful to characterize the structure of a production environment and the types of objects that could compose it. Each `DUL:PhysicalObject` is described by a set of *attributes* that characterize its specific features. DUL supports a flexible representation of such attributes and the way they can be measured and expressed through the distinction of `DUL:Quality` and `DUL:Region`.

According to the documentation, a `DUL:Quality` is any aspect of an entity (e.g., a `DUL:PhysicalObject`) that cannot exist without that entity. However, quality is not part of an entity. Rather it represents a particular attribute/aspect that is relevant to be expressed in the considered domain. The physical location, shape, or color of the surface of a `DUL:PhysicalObject` are examples of possible `DUL:Quality`. For each `DUL:Quality` there can be one or more `DUL:Region` expressing the *value* of the associated quality. Namely, a `DUL:Region` is any *dimensional space* which can be used as a value for a quality. Examples of (general) regions available within DUL are `DUL:TimeInterval` and `DUL:SpaceRegion` that are used to represent respectively *time* and object location, with respect to a particular dimensional space. Next sub-sections will show with further details how SOHO uses these classes (i.e., `DUL:PhysicalObject`, `DUL:Quality` and `DUL:Region`) to characterize different types of physical entities.

Another relevant concept used by SOHO is `DUL:Description` and its sub-concepts `DUL:Method`, `DUL:Goal`, and `DUL:Norm`. A `DUL:Description` is defined as a `DUL:SocialObject` representing a conceptualization. According to the documentation, it can be thought also as a descriptive context that creates a view of a relational context out of a set of data or observations. SOHO uses this concept to conceptualize production goals, procedures, and related constraints. In this regard, a `DUL:Method` is a `DUL:Description` that defines concepts to guide carrying out actions aimed at a solution with respect to a problem. It is worth noticing that a `DUL:Method` is different from a `DUL:Plan` since plans could be carried out in order to follow a method while a method can be followed by executing different plans. This concept is therefore well suited to characterize *production procedures* that can be instantiated into different collaborative plans entailing the execution of human and robot actions. A `DUL:Goal` is the description of a `DUL:Situation` that is desired by an `DUL:Agent` and usually associated with a `DUL:Plan` describing how to actually achieve it. SOHO extends this concept to model *social goals* and thus describes (production-related) situations that would be jointly achieved by more than one agent (i.e., a human and a robot in the specific case of an HRC production scenario).

As mentioned at the beginning of the section, SOHO interprets an HRC scenario as a social context where two or more agents should cooperate to achieve a common objective (`SocialGoal`). Involved agents should

[4] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite



(a) (b)

Fig. 1. Overview of SOHO: (a) general structure and defined contexts; (b) excerpt of concepts and properties.

therefore comply with a number of *rules* in order to satisfy requirements concerning for example safety, procedure consistency, production quality, etc. To describe such rules SOHO relies on the concept `DUL:Norm` which generally represents *social norms*. In particular, SOHO distinguishes between two types of norms: (i) norms determining how human and robot agents should behave while carrying out joint tasks (i.e., collaborative tasks) and; (ii) norms determining how tasks should be executed to satisfy production requirements. The next sub-sections describe with further detail how the concept `DUL:Norm` is specialized within SOHO.

Finally, the concept `DUL:Event` represents any physical, social, or mental process that can occur in a domain. SOHO uses the concept `DUL:Event` to characterize situations that can occur and be observed within the environment, and that can affect production. Namely, this concept is used to define exogenous events that should be taken into account during the life-cycle of a collaborative production cell and that may require the adaptation of implemented production procedures and collaborative plans. The concept `DUL:Action` represents a particular type of `DUL:Event` with at least one participating `DUL:Agent`. SOHO specifically uses this concept to represent actions physically executed either by a human, by a robot, or jointly by both of them. An action is thus seen as a temporal occurrence (i.e., *implementation*) of a `DUL:Description` about a task/function of a production procedure.

3.1.2. Integrating synergetic perspectives

Ontologies should be adequate to their domains, and domains come along on different granularity levels [49]. An ontology should account for all perspectives and levels of abstraction that are relevant to a domain. SOHO follows a context-based approach and organizes knowledge in a number of synergetic contexts, each describing the domain from a particular perspective. Contexts support a modular and *multi-perspective* representation of domain knowledge. Figure 1 shows the general structure of SOHO: (i) the *Environment Context*; (ii) the *Behavior Context* and; (iii) the *Production Context*. The *safety* and *human factor* contexts do not represent actual ontological contexts. Rather they define two "meta-perspectives" that must be uniformly considered at different levels of abstraction by all ontological contexts.

Environment context The environment context defines physical elements and general properties of an environment that can be observed. This context strongly relies on SSN which is crucial to characterize the sensing capabilities of available devices and the physical properties of domain entities they can observe. First, SOHO defines a concept to model objects that are part of a production environment by extending the concept `DUL:PhysicalObject`.

$$\texttt{ProductionObject} \sqsubseteq \texttt{DUL:PhysicalObject} \sqcap \exists \texttt{DUL:hasQuality.ObjectQuality} \tag{1}$$

where ObjectQuality is a subclass of ProductionProperty which in turn is a subclass of DUL:Quality. The concept of ObjectQuality thus describes the attributes of any DUL:PhysicalObject that specifically compose a production environment (i.e., a ProductionObject).

Some properties of the objects that constitute a production environment can change over time and it could be necessary to observe/monitor them in order to correctly carry out tasks. Examples are the position in space of objects necessary to perform a task, the position and occupancy state of an area where to place objects, or the state of a bolt, etc. SOHO relies on SSN to model sensing devices and the information they can gather through (implemented) sensing processes. SSN defines the concept SSN:FeatureOfInterest which generally describes aspects of an environment (e.g., properties of objects of a production environment) that are interesting to be observed through some sensing device. Observable properties may change according to the specific features (i.e., DUL:Quality) of the objects that compose the environment but also according to the sensing capabilities of deployed devices. The "observability" of one or more qualities of an object actually depends on the sensing capabilities of available devices and their deployment in the production environment. To characterize the *observable properties* of a production environment SOHO extends SSN by leveraging the concept DUL:Role. A DUL:Role is a concept used to *classify* an object of the domain and is thus useful to support a flexible classification of ProductionObject that can be actually observed in a production environment. SOHO defines the concept ObservableFeature as a DUL:Role that objects can play according to the available sensing devices.

$$\begin{aligned} \texttt{ObservableFeature} \sqsubseteq\ & \texttt{DUL:Role} \sqcap \texttt{SSN:FeatureOfInterest} \sqcap \\ & \exists \texttt{DUL:isRoleOf.ProductionObject} \sqcap \\ & \exists \texttt{hasObservableProperty.ProductionProperty} \sqcap \\ & \exists \texttt{isObservableThrough.SSN:Sensor} \end{aligned} \tag{2}$$

Behavior context The behavior context characterizes the behaviors of the acting entities of a production environment. The central concept is DUL:Agent representing the physical entities that actually *act* in the environment and carry out production processes. Given the focus on Human-Robot Collaboration, SOHO distinguishes two particular types of DUL:Agent that are: (i) Cobot and; (ii) Worker. SOHO interprets both agents as two physical *autonomous agents* that are associated with an *embodiment* which is in turn associated with a number of physical and behavioral qualities. The two concepts differ in terms of the specific types of *physical objects* that compose their embodiment, associated qualities that can be observed/monitored, and related capabilities. More specifically, a Cobot is defined as follows:

$$\begin{aligned} \texttt{Cobot} \sqsubseteq\ & \texttt{DUL:Agent} \sqcap \\ & \exists \texttt{DUL:hasPart.RobotInterface} \sqcap \\ & \exists \texttt{DUL:hasQuality.Autonomy} \sqcap \\ & \exists \texttt{DUL:hasQuality.Capability} \sqcap \\ & \exists \texttt{DUL:hasQuality.RobotProperty} \end{aligned} \tag{3}$$

The role DUL:hasPart associates a robot with a set of RobotInterface representing its *embodiment* i.e., the set of PhysicalObject composing the physical device. Being physical objects, each RobotInterface is associated with a number of DUL:Quality representing relevant attributes that can be measured and/or monitored (RobotProperty). Let us consider for example a wheeled base $wb1$ as a possible individual of RobotInterface being part of a collaborative robotic agent *cob* (individual of Cobot). The individual *wb* would then be associated with the qualities SpatialLocation and RobotSpeed describing respectively the known geometric position and speed.

A particular type of DUL:Quality is Capability. This concept characterizes the types of operations a DUL:Agent can support through its "functional parts". Namely, such qualities characterize general operations e.g., GraspingObject, UseTool or MoveObject that an acting entity can intrinsically perform according to its

physical/technical composition. A Cobot inherits the set of Capability supported by its parts. Let us consider for example a robot gripper *grip* as a possible individual of RobotInterface being part of *cob*. The description of *grip* associates the individual with the capability GraspObject. Consequently, the robot *cob* itself would inherit the same capability given its internal composition. In other words, a robot would be capable of grasping objects if and only if the agent is endowed with a physical interface capable of grasping objects (i.e., a *gripper*).

Another particular type of DUL:Quality is Autonomy which represents a specific *behavioral quality*. SOHO defines behavioral qualities of DUL:Agent introducing the concept AgentBehavior as subclass of DUL:Quality. This concept is then specialized into Autonomy and WorkerLevel to characterize knowledge about the expected behavior of the acting agent, respectively robots and workers. In the case of the robot, the quality Autonomy is expressed in a number of AutonomyLevel (subclass of DUL:Region) structured according to the ALFUS framework [47]. Different levels of autonomy determine different operation modalities and different safety constraints that should be considered when performing production tasks. It could be the case, for example, that some tasks can be performed only if a robot can operate in FullyAutonomy. Also, the production procedure implemented when a robot works in Teleoperation could be different from the procedure implemented when a robot works in SemiAutonomy of FullyAutonomy. This information is useful to parametrize production procedures and model conditions under which different procedures and/or tasks can be executed.

Similar to Cobot a human agent Worker is defined as follows:

$$\begin{aligned} \texttt{Worker} \sqsubseteq\ & \texttt{DUL:Agent} \sqcap \\ & \exists \texttt{DUL:hasPart.HumanBodyPart} \sqcap \\ & \exists \texttt{DUL:hasQuality.WorkerLevel} \sqcap \\ & \exists \texttt{DUL:hasQuality.Capability} \sqcap \\ & \exists \texttt{DUL:hasQuality.HumanBodyProperty} \end{aligned} \tag{4}$$

The general assumptions and structure described for Cobot hold also in the case of Worker. SOHO distinguishes these two concepts with respect to the specific types of DUL:Quality associated through their embodiment.

Behavioral knowledge is determined according to the specific features and internal composition (i.e., the *embodiment*) of the specific agents. The set of operations that can be actually implemented in a particular scenario (and "who" can implement them) can be dynamically inferred according to the known capabilities of an agent. To support this reasoning SOHO should generally characterize manufacturing operations and correlate them to the types of capabilities necessary for their execution. To this aim, SOHO integrates the *Taxonomy of Functions* defined in [12]. Low-level manufacturing operations are defined as Function that are classified according to the *effects* they have on the DUL:Quality of *target* objects. A Function represents a "primitive" ProductionTask that cannot be further decomposed in simpler operations. Each function is associated with the needed Capability and the affected DUL:Quality of the target ProductionObject.

$$\begin{aligned} \texttt{Function} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists \texttt{canBePerformedBy.DUL:Agent} \sqcap \\ & \exists \texttt{hasEffectOn.DUL:Quality} \sqcap \\ & \exists \texttt{requires.Capability} \sqcap \\ & \exists \texttt{hasTarget.ProductionObject} \end{aligned} \tag{5}$$

$$\begin{aligned} \texttt{ProductionObject}(o) \wedge \texttt{Function}(f) \wedge \\ \texttt{DUL:Agent}(a) \wedge \texttt{Capability}(c) \wedge \\ \texttt{DUL:hasQuality}(o, q) \wedge \texttt{hasEffectOn}(f, q) \wedge \\ \texttt{hasCapability}(a, c) \wedge \\ \texttt{requiresCapability}(f, c) \rightarrow \texttt{hasTarget}(f, o) \wedge \\ \texttt{canBePerformedBy}(f, a) \end{aligned} \tag{6}$$

Production context Considering the production perspective, SOHO defines concepts and properties that characterize production procedures in terms of objectives and operations necessary to successfully achieve them. A proper representation of this knowledge is crucial to establish human and robot *commitment* to production goals [1,18] and a level of *agreement* about the way the human and the robot together achieve these goals [25,79]. Furthermore, it is necessary to characterize *events* that may occur in a production environment and that are relevant with respect to the execution of production procedures. This perspective relies on the foundational concepts `DUL:Event` and `DUL:Description`. The former supports the description of temporal occurrences requiring the implementation of some production procedure. The latter supports the description of such procedures and the collaborative rules constraining the behaviors of the human and the robot.

SOHO defines the concept `ProductionGoal` as a particular type of `DUL:Goal` to characterize situations that agents should achieve to fulfill production requirements. Each `ProductionGoal` is associated with (at least) one `ProductionMethod` which is a particular type of `DUL:Method` describing valid procedures.

$$\begin{aligned} \texttt{ProductionGoal} \sqsubseteq \texttt{DUL:Goal} \sqcap \\ \exists \texttt{DUL:hasConstituent.ProductionMethod} \end{aligned} \tag{7}$$

Goals are achieved through plans each composed of a number of actions implementing a particular production method. SOHO defines the concept `ProductionPlan` to describe the way a certain `ProductionGoal` is achieved. A `ProductionPlan` describes a particular `ProductionProcess` which implements a particular `ProductionMethod` through the actual execution of a number of `ProductionAction`.

$$\begin{aligned} \texttt{ProductionPlan} \sqsubseteq \texttt{DUL:Plan} \sqcap \\ \exists \texttt{DUL:hasComponent.ProductionGoal} \sqcap \\ \exists \texttt{DUL:isDescribedBy.ProductionMethod} \sqcap \\ \exists \texttt{DUL:describes.ProductionProcess} \end{aligned} \tag{8}$$

A `ProductionProcess` is the description of a dynamic (physical) event involving the execution of a number of `ProductionAction` by participating agents. SOHO interprets this concept as a particular type of `DUL:Process` which is in turn a particular type of `DUL:Event`.

$$\begin{aligned} \texttt{ProductionProcess} \sqsubseteq \texttt{DUL:Process} \sqcap \\ \exists \texttt{DUL:hasPart.(ProductionAction} \sqcup \\ \texttt{ProductionRelatedEvent)} \sqcap \\ \exists \texttt{DUL:hasParticipant.DUL:Agent} \sqcap \\ \exists \texttt{DUL:isDescribedBy.ProductionPlan} \end{aligned} \tag{9}$$

A `CollaborativeProcess` then is a particular `ProductionProcess` where exactly one autonomous robot (`Cobot`) and one human worker (`WorkOperator`) jointly contribute to its execution.

The temporal and physical occurrence of production processes entails the execution of ProductionAction and the occurrence of a number of ProductionRelatedEvent. SOHO defines a ProductionAction as a particular type of DUL:Action describing the actual execution in time and space of production operations (i.e., instances of Function).

$$\begin{aligned} \texttt{ProductionAction} \sqsubseteq\ & \texttt{DUL:Action} \sqcap \\ & \exists \texttt{DUL:hasParticipant.DUL:Agent} \sqcap \\ & \exists \texttt{DUL:isDescribedBy.(InteractionModality} \sqcup \\ & \qquad \texttt{ProductionTask)} \end{aligned} \tag{10}$$

SOHO defines the concept ProductionRelatedEvent and EnvironmentRelatedEvent as general DUL:Event describing respectively facts correlated to the execution of actions (e.g., execution failures or results) and exogenous facts concerning the state of monitored features of the environment (e.g., the physiological state of the human worker).

The concept ProductionMethod is central to the description of procedures. It is a particular type of DUL:Method resulting from the composition of ProductionTask describing simple/primitive or complex operations to be performed by some agent.

$$\begin{aligned} \texttt{ProductionTask} \sqsubseteq\ & \texttt{ProductionMethod} \sqcap \\ & \exists \texttt{DUL:isDescribedBy.ProductionNorm} \end{aligned} \tag{11}$$

It is worth noticing that our definition of SOHO:ProductionTask falls under DUL:Description with the aim of defining a descriptive context of production procedures. Namely, the defined concepts would specify the general requirements of the tasks that could be performed within a collaborative environment. Thus, the defined SOHO:ProcutionTask represents a descriptive context for the concrete actions performed within the environment. A DUL:Action instead represents a temporal instantiation of SOHO:ProductionTask in the environment (for this reason we specialize the concept SOHO:ProductionAction as a specialization of DUL:Action). The concept SOHO:ProductionTask describes procedures that generally guide the execution of actions that aim at solving a specific production need that is modeled as a SOHO:ProductionGoal. Our interpretation of task is more general than DUL:Task which is classified as a type of event characterizing DUL:Action to be executed. In this regard, our interpretation of SOHO:ProductionTask is close to DUL:Method that we choose as theoretical foundation.

A ProductionNorm is a particular type of ExecutionNorm which is a sub-class of DUL:Norm. It describes general rules determining the way tasks (and resulting actions) should be executed by the participating agents.

$$\begin{aligned} \texttt{ProductionNorm} \sqsubseteq\ & \texttt{ExecutionNorm} \sqcap \\ & \exists \texttt{DUL:describes.ProductionTask} \sqcap \\ & \exists \texttt{constrains.ProductionTask} \end{aligned} \tag{12}$$

3.2. Human factor and worker profiles

A novel aspect of SOHO is the support to the explicit representation of the *human factor*. SOHO interprets the *human factor* as the set of physical or abstract features that characterize the expected behavior and skills of a worker and that can directly (or indirectly) affect the interactions with a robot and the whole production. The human factor is defined through a set of DUL:Quality that characterize the physical and behavioral qualities of workers and correlate them with production needs. Figure 2(a) shows the taxonomic structure defined for the behavioral features of workers, while Fig. 2(b) shows the taxonomical structure defined for the physical features of workers.

Concepts like WorkerTaskAccuracy, WorkerTaskPerformance, or WorkerLevel describe expected performance or expertise levels of human workers. Such concepts are useful for example to estimate the efficiency,

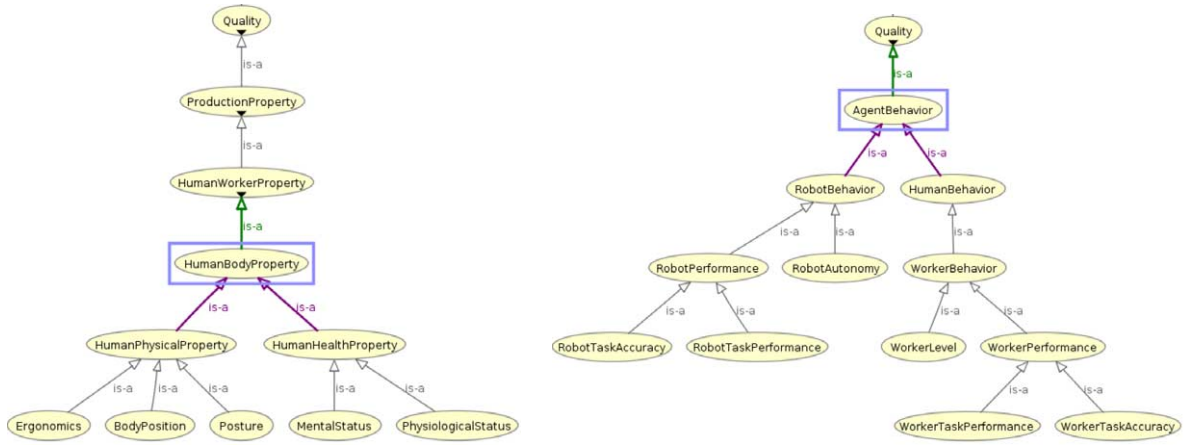


Fig. 2. Excerpt of SOHO concerning physical and behavioral qualities of human workers.

and accuracy of tasks performed by workers. The concept `WorkerLevel` represents a measure of the level of knowledge of a worker about a specific production scenario and the reliability of her performance. On the one hand, the expertise level determines the (sub)set of production tasks a human worker can carry out. For example, some tasks may require a certain minimum level of experience to be performed by a worker. On the other hand, the expertise level characterizes the expected *uncertainty* about the performance of a worker. Low experience determines a higher *variance* in the performance and thus a higher amount of *uncertainty* in terms of execution time and accuracy. Higher experience instead determines performance with lower variance in terms of "expected" execution time and achieved accuracy. Given the *behavioral model* of a worker, it would be possible then to automatically adapt collaborative processes, tailoring the level of assistance/support provided by a robot [92].

Concepts like `BodyPosition`, `MentalStatus`, or `PhysiologicalStatus` describe human body features that may affect the safety, performance, and quality of the resulting collaborative processes. These concepts define an interpretation space for specific perception data that can be integrated into collaborative control systems. A proper representation and monitoring of physical/health information could be useful to automatically detect hazards or particular situations that may require a quick and proactive adaptation of the behavior of a robot. For example, physiological data about heart rate or blood pressure could be processed to evaluate the stress level of a worker. Depending on the defined thresholds, results may determine a high level of stress for the worker. SOHO allows a collaborative control system to be *aware* of such a situation and proactively change collaborative dynamics by for example slowing down operations or assigning less cognitively demanding tasks to the humans.

3.3. Production procedures and interaction modalities

The definition of a `ProductionMethod` through a number of `ProductionTask` follows a hierarchical task-oriented approach [57,83]. The top-level element is the `ProductionGoal` that is associated with a number of `ProductionMethod` defining the rules that must be considered to successfully achieve a production goal. SOHO specifies the associations between goals, methods, and tasks using the property `DUL:hasConstituent` as non-transitive relation supporting a layered description of the procedure.

Following this layering of a procedure, SOHO defines three types of `ProductionTask`: (i) `ComplexTask` that can be either *disjunctive* or *conjunctive*; (ii) `SimpleTask` and; (iii) `Function`. Figure 3 shows an excerpt of the resulting taxonomical structure. In particular, the structure shows the integration of the Taxonomy of Functions introduced in [12] with the concept `Function` interpreted as a particular type of `ProductionTask`.

A `ComplexTask` is a `ProductionTask` (i.e., an instance of `DUL:Method`) representing a compound logical operation. Complex tasks are generally interpreted as `ConjunctiveComplexTask` meaning that associated

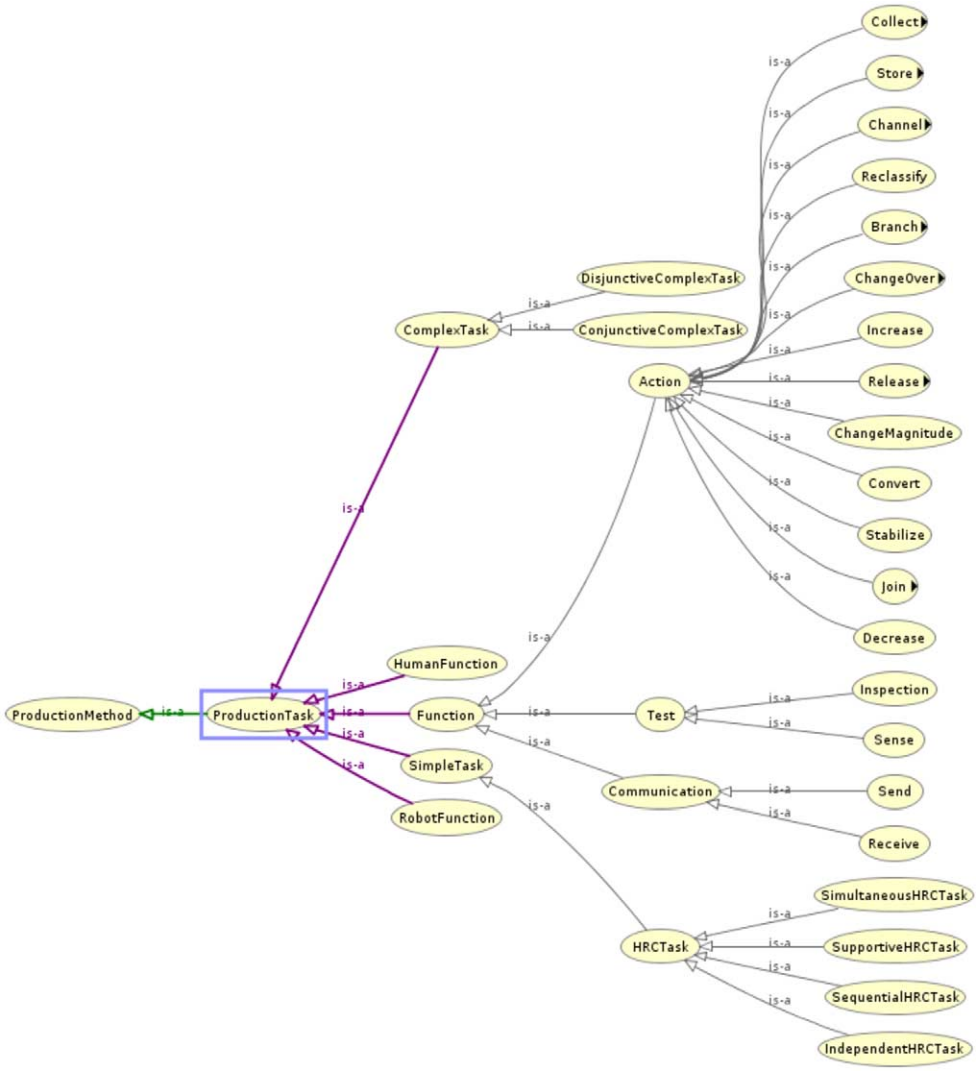


Fig. 3. Taxonomical structure of production tasks with the integrated Taxonomy of Function [12]. Please note that the concept `Action` is a particular type of `Function` and falls under the namespace of SOHO. This concept is therefore different from the concept `DUL:Action` defined within DUL.

tasks (i.e., production tasks associated through `DUL:hasConstitutent`) should all be part of a plan implementing it. Complex tasks could be also interpreted as `DisjunctiveComplexTask` meaning that associated tasks represent alternative ways of implementing it. Plans should consider only one of the modeled alternatives (i.e., only one of the production tasks associated through `DUL:hasConstituent`).

$$\begin{aligned} &\texttt{ComplexTask} \sqsubseteq \texttt{ProductionTask} \sqcap \\ &\qquad \exists \texttt{DUL:hasConstituent.}(\texttt{ComplexTask} \sqcup \texttt{SimpleTask}) \sqcap \\ &\qquad \exists \texttt{DUL:isDescribedBy.OperativeConstraint} \end{aligned} \tag{13}$$

A `SimpleTask` represents a *leaf* of the hierarchical structure of a `ProductionMethod`. This concept describes simple operations agents can directly implement through their functional capabilities. A `SimpleTask`

requires thus the execution of a number of Function by one or more participating agents.

$$\begin{aligned} \texttt{SimpleTask} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists \texttt{DUL:hasConstituent.}(\texttt{Function} \sqcup \texttt{ProductionObject}) \sqcap \\ & \exists \texttt{DUL:isDescribedBy.}(\texttt{InteractionModality} \sqcup \\ & \qquad \texttt{OperativeConstraint}) \end{aligned} \tag{14}$$

The concepts OperativeConstraint and InteractionModality are two types of ExecutionNorm (specialization of DUL:Norm) constraining the behaviors of acting entities when participating in some production process. The former describes rules that constrain the execution of two or more ProductionTask and can be further classified into PrecedenceConstraint and ParallelExecutionConstraint. The latter describes more rules constraining the behavior of a human and a robot when realizing collaborative tasks.

Although the "boundaries" of the *representation space* are well delimited within a domain ontology [43], there is a multitude of behaviors that can be described with a production scenario and a multitude of design choices to take into account. The correct definition of all necessary information and constraints is not always straightforward. Ontology design patterns [40] can play a role in supporting knowledge definition. Patterns can indeed specialize an ontological model without losing generality but defining useful structures that guide knowledge definition.

Ontological patterns in this case characterize typical and/or recurrent associations between tasks and functions. Namely, they define structures describing typical collaborative behaviors of human workers and robots in production scenarios. SOHO introduces HRC ontological patterns by taking into account interaction schema known in the literature [44]. First, SOHO defines the concept HRCTask as a particular type of SimpleTask requiring the tight interaction of a human worker (i.e., a WorkOperator) and a collaborative robot (i.e., a Cobot). The basic assumption is that a HRCTask entails the execution of a maximum of two Function. Each required Function should be performed by a WorkOperator or by a Cobot. If only one function is necessary then it can be performed either by a human worker or a robot. If two functions are necessary then one function should be performed by the human and the other by the robot.

Each HRCTask is associated with a InteractionModality specifying behavioral norms that constrain the way underlying Function are executed.

$$\begin{aligned} \texttt{InteractionModality} \sqsubseteq\ & \texttt{ProductionNorm} \sqcap \\ & \exists \texttt{DUL:describes.ProductionTask} \sqcap \\ & \leqslant 2 \texttt{constrains.Function} \end{aligned} \tag{15}$$

According to [44], the execution of a collaborative task (i.e., an individual of HRCTask) follows one of four different collaboration modalities: (i) *Independent*, humans and robots perform their tasks on different workpieces without collaboration; (ii) *Simultaneous*, human and robot perform distinct tasks on the same workpiece at the same time, still without physical interaction; (iii) *Supportive*, human and robot perform the same task on the same workpiece and they work simultaneously and cooperatively on the same task. (iv) *Sequential*, human and robot should complete sequential tasks on the same workpiece. Figure 4 shows a graphical representation of these four types of collaborative tasks.

SOHO thus defines four types of InteractionModality as four patterns characterizing specific knowledge structures in terms of associated concepts and cardinality restrictions. An interaction modality of type Independent requires a human or a robot working on a particular target object independently from each other but in a shared space. It describes one HRCTask and constrains one Function which can be either a HumanFunction (i.e., a Function that can be performed by a WorkOperator) or a RobotFunction (i.e., a Function that

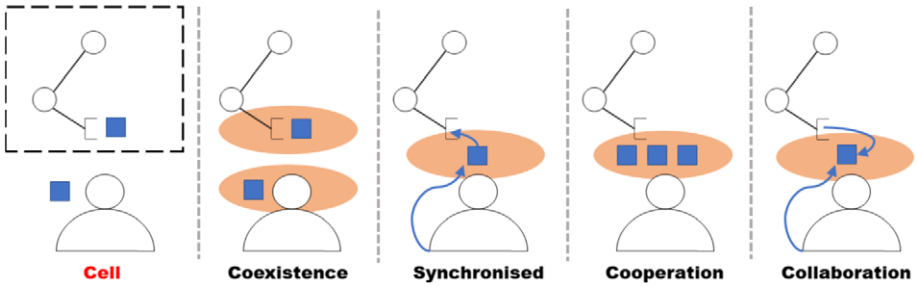


Fig. 4. General structure of an HRC cell and types of collaboration modalities as defined in [59]: (i) Coexistence/Independent; (ii) Synchronised/Sequential; (iii) Cooperation/Simultaneous; (iv) Collaboration/Supportive.

can be performed by a `Cobot`).

$$\begin{aligned} \texttt{Independent} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\texttt{constrains.(HumanFunction} \sqcup \\ & \qquad \texttt{RobotFunction)} \end{aligned} \tag{16}$$

An interaction modality of type `Simultaneous` requires a human and a robot to perform two different operations on the same target object at the same time. It describes a `HRCTask` requiring exactly one `RobotFunction` and one `HumanFunction` to be executed at the same time. Namely, the execution of the two operations may overlap in time and should not follow a specific ordering (e.g., precedence constraint).

$$\begin{aligned} \texttt{Simultaneous} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\texttt{constrains.HumanFunction} \sqcap \\ & \exists!\texttt{constrains.RobotFunction} \sqcap \end{aligned} \tag{17}$$

An interaction modality of type `Sequential` requires a human and a robot to perform operations on the same object according to a strict order. The pattern in this case forces the necessary `RobotFunction` and the `HumanFunction` to be executed according to a specified precedence constraint. It is therefore associated with one `PrecedenceConstraint` which specifies the desired ordering of the functions.

$$\begin{aligned} \texttt{Sequential} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\texttt{constrains.HumanFunction} \sqcap \\ & \exists!\texttt{constrains.RobotFunction} \sqcap \\ & \exists!\texttt{DUL:isDescribedBy.PrecedenceConstraint} \end{aligned} \tag{18}$$

An interaction modality of type `Supportive` requires a human and a robot to perform the same operation on the same object at the same time. The pattern in this case forces the execution of a `RobotFunction` and a `HumanFunction` to start and end at the same time. Although the operation is the same, it is necessary to model two distinct instances of human and robot `Function`. Namely, a human and a robot execute two distinct operations of the same type (i.e., two instances of the same type of `Function`) and they are executed in strict parallelism. The

required temporal constraint is described by the `ParallelExecutionContraint`.

$$\begin{aligned} \texttt{Supportive} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\texttt{constrains.HumanFunction} \sqcap \\ & \exists!\texttt{constrains.RobotFunction} \sqcap \\ & \exists!\texttt{DUL:isDescribedBy.ParallelExecutionConstraint} \end{aligned} \tag{19}$$

According to these four interaction modalities, SOHO defines four types of `HRCTask` each associated with a specific `InteractionModality`: (i) `IndependentHRCTask`; (ii) `SimultaneousHRCTask`; (iii) `SequentialHRCTask` and; (iv) `SequentialHRCTask`. These tasks provide designers with generic and reusable concepts suitable to characterize collaborative dynamics in manufacturing scenarios.

4. Knowledge definition and automated synthesis of plan-based control models

A SOHO-compliant knowledge base (ABox) characterizes an HRC scenario from different perspectives. The use of semantic technologies based on RDFS [29], OWL [3] and SPARQL [70] supports accessibility and interoperability between knowledge and production-related processes. We are especially interested in showing how the proposed semantics and related knowledge bases would contribute to the enhancement of *awareness*, *flexibility*, and *autonomy* of collaborative robots. This section shows in detail how knowledge is used to automatize the synthesis of task planning models and coordinate human and robot operations through deliberative plan-based control [41,48]. A knowledge extraction procedure bridges the gap between knowledge representation and task planning for robot control, supporting the realization of a cognitive "perceive, reason, act" loop. It is worth noticing that SOHO is *planning agnostic*. The defined knowledge bases thus describe therefore HRC scenario in general terms without considering the particular task planning formalism used for the actual coordination. This is a key point to "standardize" production knowledge and thus realize general services that can be used and combined with different control technologies (at different levels of abstraction).

Task planning generally relies on AI Planning and Scheduling technologies [33,36,39,60] allowing robots (or more in general artificial agents) to autonomously synthesize and execute plans that achieve some desired goal. Such plans are generally seen as sequences of actions to be executed starting from an initial state. Several planning formalisms exist in literature each supporting different reasoning capabilities e.g., causal reasoning [36,45,60], numeric and temporal reasoning [26,39] or hierarchical reasoning [10,64]. HRC requires reasoning on the simultaneous execution of human and robot actions, taking into account different qualities of the resulting collaborative processes e.g., cycle time, safety, and idle time of the robot. Furthermore, the human introduces a significant source of *uncertainty* from a control perspective. A plan-based controller should properly deal with this level of uncertainty in order to synthesize plans that are sufficiently robust at execution time [62,96].

For these reasons, this work specifically considers the *timeline-based planning formalism* and proposes a knowledge extraction procedure mapping production knowledge to timeline-based specifications. The formalism has been successfully applied in many real-world scenarios e.g., [20,50,63,72] and is quite expressive supporting concurrency, durative actions, (flexible) time as well as numeric and temporal constraints. The work [24] formalizes timeline-based planning by introducing the notions of *temporal uncertainty* and *controllability issues* [23]. Timelines have been applied to HRC thanks to the capabilities of dealing with human behavioral uncertainty [35,69]. Before entering into the details of the developed knowledge extraction procedure, the next sub-section briefly introduces the main concepts regarding timelines, as introduced in [24].

4.1. Plan-based control through timeline-based planning and execution

A timeline-based planning specification describes valid temporal behaviors of a number of domain features to be controlled. The planning process consists in synthesizing valid flexible behaviors (i.e., timelines) describing *how*

these features should evolve over time to achieve some given *objectives* (i.e., *which* states/actions assumes/executes and *when*). According to [24], *state variables* model domain features by specifying valid temporal behaviors in terms of allowed timed sequences of states/actions (generally denoted as *state variable values*).

Definition 1. A *State Variable* is a tuple $SV = \langle V, T, D, \gamma \rangle$ describing valid behaviors of a domain feature:

- V is a set of *values* $v_i \in V$ representing states of actions the feature can perform or assume over time.
- $T : V \rightarrow 2^V$ is a *state transition function* describing for each value $v_i \in V$ possible successors on a timeline and thus valid transitions.
- $D : V \rightarrow \mathbb{T} \times \mathbb{T}$ is a *duration function* specifying for each value $v_i \in V$ its expected duration bounds, expressed in some temporal domain $\mathbb{T}$ (typically $\mathbb{N}^+$).
- $\gamma : V \rightarrow \{c, pc, u\}$ is a *controllability tagging function* specifying the controllability property of a value.

Controllability properties characterize the execution of SVs' values with respect to the dynamics of the environment.

Definition 2. A value $v_i \in V$ of a state variable $SV = \langle V, T, D, \gamma \rangle$ is:

- *Controllable* ($\gamma(v_i) = c$) if the control system can decide both the start and end times for its execution.
- *Partially-controllable* ($\gamma(v_i) = pc$) if it can only be started by the control system, while its end time and, i.e., its actual duration, can be only observed.
- *Uncontrollable* ($\gamma(v_i) = u$) the control system can neither decide its start nor its end times.

Information about controllability and temporal flexibility are crucial to solve planning problems while dealing with *temporal uncertainty* to support robust plan execution [19,62,95].

A *flexible timeline* for a state variable SV_i is a sequence of (flexible) temporal intervals called *tokens* that describe an *envelope* of valid temporal behaviors.

Definition 3. If $SV_i = (V, T, \gamma, D)$ is a state variable, a *token* x_j for the variable has the form:

$$x_j = \big(v_k, \big[e_j, e'_j\big], \big[d_j, d'_j\big], \gamma(v_k)\big)$$

where $v_k \in V$ is the value assumed by the token x_j, $[e_j, e'_j]$ is the end-time interval of x_j (with $e_j \leqslant e'_j$) and $[d_j, d'_j]$ is the minimum and maximum duration of x_j (with $d_j \leqslant d'_j$).

A token x_j represents a specific allocation of a value $v_k \in V$ over a certain (flexible) temporal interval. The duration of a token must be consistent with the duration bounds of the associated value v_k. A *timeline* is a *continuous sequence of tokens* x_i describing the behavior of a domain feature from a *temporal origin* to (at least) a *planning horizon* $\mathtt{H} \in \mathbb{T}$. The start-time interval of a token is not explicitly represented since it coincides with the end-time interval of the previous token in the timeline (the first token of a timeline starts at the temporal origin [0, 0]).

Definition 4. A *timeline* FTL_i for a state variable $SV_i = (V, T, D, \gamma)$ is a continuous and finite sequence of tokens of the form:

$$x_1 = \big(v_1, \big[e_1, e'_1\big], \big[d_1, d'_1\big], \gamma(v_1)\big), \ldots, x_m = \big(v_m, \big[e_m, e'_m\big], \big[d_m, d'_m\big], \gamma(v_m)\big),$$

where $v_1, \ldots, v_m \in V$ and for all $j = 1, \ldots, m-1$, $v_{j+1} \in T(v_j)$. Denoting with $start(x_j)$ the computed start time interval of a token x_j then, for all $j = 1, \ldots, m-1$, $[e_j, e'_j] = start(x_{j+1})$ (*i.e.*, a timeline is a continuous sequence of non-overlapping tokens).

Synchronization rules specify additional constraints that are necessary to synthesize timelines that achieve desired objectives (e.g., planning goals).

Definition 5. A *synchronization rule* has the form

$$a_0[SV_0 = v_0] \rightarrow a_1[SV_1 = v_1], \ldots, a_n[SV_n = v_n].\mathcal{C}$$

Fig. 5. Knowledge extraction pipeline for the automatic generation of timeline-based planning models.

where every $a_i[SV_i = v_i]$ is a *token variable* denoting a temporal interval in which a state variable SV_i assumes the value v_i. The left-hand part of the synchronization rule ($a_0[SV_0 = v_0]$) is called the *trigger of the rule. The set C specifies* temporal relations *between token variables.*

Synchronization rules with the same trigger are treated as disjunctions and represent alternative constraints that should hold between different sets of token variables.

A planning problem then consists of a set of partially instantiated timelines that specify known *facts* about the initial state of domain features (i.e., tokens specifying the values assumed by each state variable at plan origin) and *goals* constraining their temporal evolution (i.e.,. tokens specifying the desired values that one or more state variables should assume during certain temporal intervals). A planning process should synthesize valid and complete temporal behaviors of all (controllable) state variables (i.e., *timelines*) such that all *duration constraints*, *value transition constraints* and *temporal constraints* of applied synchronization rules are satisfied.

4.2. Model generation through knowledge extraction

The objective of the knowledge extraction procedure is to synthesize a valid timeline-based specification to coordinate human and robot agents through task planning. The procedure relies on SOHO to extract information from the knowledge base and instantiate timeline-based structures. Two types of structures compose a timeline-based model: (i) *state variables* and; (ii) *synchronization rules*. State variables describe possible behaviors of modeled domain features singularly (local perspective). Synchronization rules constrain such behaviors to coordinate state variables and carry out complex tasks.

Given these structures, it is necessary to reason about how to organize the task planning model in order to coordinate domain entities correctly (e.g., the human and the robot in HRC scenarios). A number of modeling decisions are necessary concerning the number of state variables (i.e., which and how many domain features must be modeled), and the number of decomposition and temporal constraints (i.e., which and how many synchronization rules must be modeled to effectively coordinate state variables).

The modeling of an effective task-planning domain is not trivial and should follow a number of well-defined steps. Broadly speaking we define a timeline-based model of an HRC scenario following a hierarchical decomposition methodology [35,69]. We define a number of logical state variables modeling production goals and production tasks that could be performed over time. A number of state variables model the low-level operations that agents could perform over time. A number of synchronization rules describe how production goals are decomposed into increasingly simpler tasks and how simple (or primitive) tasks are correlated with the low-level operations that modeled agents can perform over time. This methodology has been implemented in a knowledge extraction procedure capable of generating timeline-based models from SOHO-compliant production knowledge. Figure 5 shows the general structure of the developed procedure.

4.2.1. Knowledge extraction procedure

The procedure takes as input a production knowledge defined according to SOHO and generates as output a complete timeline-based model for task planning. It consists of a number of knowledge extraction steps that can be divided into two macro-steps of the procedure. The first macro-step comprises the steps needed to generate the state variables of the timeline-based model. The second macro-step comprises the steps needed to generate the synchronization rules.

State variables model at different levels of abstraction the tasks or low-level operations that could be executed in an HRC scenario to support production. The step *Goal-level Behavior* extracts knowledge useful to define the goal-level state variable $SV_G = \langle V_G, T_G, D_G, \gamma_G \rangle$. This state variable describes the high-level production goals

that could be performed in an HRC cell and thus require the execution of some collaborative procedure. It is not difficult to imagine that this state variable is generated by taking into account the individuals of `ProductionGoal` extracted from the knowledge base. Specifically, each state variable value of $v_{G,i} \in V_G$ of SV_G is associated with a distinct individual i of `ProductionGoal` found in the knowledge base.

The step *Agent-level Behavior* extracts knowledge useful to generate the state variable defining the low-level operations that agents can execute. The procedure would generate a state variable for each individual of `DUL:Agent` found in the knowledge base. In the specific case of the HRC scenario, two state variables are generated. A human state variable $SV_H = \langle V_H, T_H, D_H, \gamma_H \rangle$ associated with the individual of `WorkOperator`. A robot state variable $SV_R = \langle V_R, T_R, D_R, \gamma_R \rangle$ associated with the individual of `Cobot`. Each value of these state variables is associated with the individuals of `Function` the related agent can perform. For example, the values $v_{H,i} \in V_H$ are associated with the individuals of `Function` that `canBePerformedBy` the individual of `WorkOperator`, as inferred through Equation (6). Similarly, the values $v_{R,i} \in V_R$ are associated with the individuals of `Function` that `canBePerformedBy` the individual of `Cobot` as inferred through Equation (6).

It is worth noticing that values of the human state variable $v_{H,i} \in V_H$ are all modeled as *uncontrollable* (i.e., $\gamma(v_{H,i}) = u$). The task planner can only assume that the execution of these functions would follow a certain (expected) schedule but cannot actually control their duration. The values of the robot state variable $v_{R,i} \in V_R$ are instead modeled as *partially controllable* (i.e., $\gamma(v_{R,i}) = pc$). The task planner can decide when to start the execution of these operations but cannot control the end because of the co-existence with the human (e.g., the human can slow down or stop robot motions while executing an operation).

The step *Task-level Behavior* then extracts knowledge useful to generate the state variables defining the tasks modeling the production procedures. This step first analyzes the knowledge by "navigating" the described production procedures through the property `DUL:hasConstituent` which associates individuals of `ProductionGoal` with individuals of `ProductionMehtod` and in turn with individuals of `ProductionTask`. Considering production procedures modeled in L hierarchical levels, this step generates a task-level state variable SV_T^j for each hierarchical level $j = \{0, \ldots, L-1\}$. For example, the highest level of abstraction $j = 0$ generates the task-level state variable $SV_T^0 = \langle V_T^0, T_T^0, D_T^0, \gamma_T^0 \rangle$. Each value $v_{T,i}^0 \in V_T^0$ of the state variable SV_T^0 is associated with an individual of `ProductionTask` at the hierarchical level $j = 0$.

When all the state variables have been generated, the procedure generates the synchronization rules that constrain their values. The step *Procedural Decomposition* analyzes the property `DUL:hasConstituent` to define decomposition constraints correlating goals or high-level tasks to lower-level tasks. For example, a synchronization rule with a value $v_{G,i}$ of the goal state variable SV_G as the head would correlate it with a (sub)set of values $v_{T,k}^0 \in V_T^0$ of the task-level state variable SV_T^0 at hierarchy level $j = 0$. The rule thus specifies that the head value $v_{G,i}$ should be decomposed into the associated values $v_{T,k}^0$. Similarly, a synchronization rule with a value $v_{T,i}^j$ of the task-level state variable SV_T^j as the head would correlate it with a (sub)set of values $v_{T,k}^{j+1}$ (with $j < L-1$) of the task-level state variable SV_T^{j+1}. In all cases, decomposition relationships established through the property `DUL:hasConstituent` are encoded using the temporal constraint `CONTAINS` [2].

The (last) step *Task Implementation* extracts information about collaborative patterns in order to generate synchronization rules correlating values $v_{T,i}^{L-1} \in V_T^{L-1}$ of the state variable SV_T^{L-1} (i.e., the leaves of the production procedures) to the values $v_{H,j} \in V_H$ and $v_{R,k} \in V_R$ of the state variables SV_H and SV_R respectively.

This set of synchronization rules constrains possible task allocations and possible interactions between the worker and the robot. It is at this level that ontological patterns about known collaboration modalities (i.e., Equation (16), Equation (17), Equation (18) and Equation (19)) are used to define temporal constraints determining specific human-robot collaborative behaviors.

4.2.2. Generation of timeline-based structures in detail

This section describes in detail the implementation of the knowledge extraction steps of the methodology encoded with the pipeline of Fig. 5. Following the procedure, we show the auxiliary data structures created to support the methodology with the pieces of the generated timeline-based model. The knowledge base and the extraction proce-

dure have been developed in Java using the open-source library Apache Jena.[5] The knowledge base is specifically stored and manipulated as an RDF Knowledge Graph (KG). Triples of the graph are accessed/filtered through RDF pattern matching using the API of Apache Jena.

Algorithm 1 shows the implementation of the knowledge extraction pipeline of Fig. 5. Specifically, it takes as input a knowledge base $\mathcal{KB}$ in the shape of a knowledge graph. It implements a number of knowledge extraction steps based on the Apache Jena API. It then generates and returns a timeline-based model $\mathcal{M} = \langle \mathcal{SV}, \mathcal{R} \rangle$. The block of rows 1–12 concerns the steps necessary to generate the set of state variables $\mathcal{SV}$ while the block of rows 13–23 concerns the steps necessary to generate the set of synchronization rules $\mathcal{R}$.

Algorithm 1 Procedure for the generation of a timeline-based model $\mathcal{M}$ from the knowledge base ($\mathcal{KB}$)

Require: $\mathcal{KB}$ ▷ Production knowledge
Ensure: $\mathcal{M} : \langle \mathcal{SV}, \mathcal{R} \rangle$ ▷ Timeline-based model
1: $\mathcal{SV}, \mathcal{R} \leftarrow \{\}$
2: $SV_G \leftarrow$ `createGoalSV`($\mathcal{KB}$) ▷ Step "Goal-level Behavior" of Fig. 5
3: $\{SV_H, SV_R\} \leftarrow$ `createAgentSV`($\mathcal{KB}$) ▷ Step "Agent-level Behavior" of Fig. 5
4: $\mathcal{SV} \leftarrow \{SV_G, SV_H, SV_R\}$
5: $\mathcal{G} = \langle \mathcal{N}_\mathcal{T}, \mathcal{E}_\mathcal{T} \rangle \leftarrow$ `createDecompositionGraph`($\mathcal{KB}$) ▷ Extract decomposition dependencies among `ProductionTask`
6: $\mathcal{T} = \{\{t_1^0, \ldots, t_n^0\}, \ldots, \{t_1^{|\mathcal{T}|-1}, \ldots, t_m^{|\mathcal{T}|-1}\}\} \leftarrow$ `hierarchy`($\mathcal{G}$) ▷ Extract hierarchy through topological sorting
7: $l \leftarrow 1$ ▷ Set starting hierarchy level to generate task-level state variables
8: **while** $l < |\mathcal{T}|$ **do** ▷ Step "Task-level Behavior" of Fig. 5
9: $\quad \mathcal{T}^l \leftarrow \{t_0^l, \ldots, t_k^l\} \in \mathcal{T}$ ▷ Individuals of `ProductionTask` at hierarchical level l
10: $\quad SV_T^l \leftarrow$ `createTaskSV`($\mathcal{KB}, \mathcal{T}^l$)
11: $\quad \mathcal{SV} \leftarrow \mathcal{SV} \cup \{SV_T^l\}$
12: $\quad l \leftarrow l + 1$
13: $l \leftarrow 0$ ▷ Set starting hierarchy level to generate synchronization rules
14: **while** $l < |\mathcal{T}|$ **do**
15: $\quad \mathcal{T}^l \leftarrow \{t_0^l, \ldots, t_k^l\} \in \mathcal{T}$ ▷ Individuals of `ProductionTask` at hierarchical level l
16: $\quad$ **for** $t_i^l \in \mathcal{T}^l$ **do**
17: $\quad\quad$ **if** $t_i^l \in$ `ComplexTask` $\vee\, t_i^l \in$ `ProductionGoal` **then** ▷ Step "Procedural Decomposition" of Fig. 5
18: $\quad\quad\quad R_i^l \leftarrow$ `createDecompositionRules`($\mathcal{KB}, t_i^l$)
19: $\quad\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{R_i^l\}$
20: $\quad\quad$ **if** $t_i^l \in$ `SimpleTask` **then** ▷ Step "Task Implementation" of Fig. 5
21: $\quad\quad\quad R_i^l \leftarrow$ `createImplementationRules`($\mathcal{KB}, t_i^l$)
22: $\quad\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{R_i^l\}$
23: $\quad l \leftarrow l + 1$
24: **return** $\mathcal{M} : \langle \mathcal{SV}, \mathcal{R} \rangle$

Define state variables The procedure starts by generating the state variables modeling the temporal behaviors of domain features. The first state variable generated is the goal state variable SV_G (row 2) which describes the high-level goals that can be achieved within the considered HRC cell (i.e., the high-level collaborative tasks that can be executed). The sub-procedure `createGoalSV`($\mathcal{KB}$) generates the state variables by retrieving individuals of `ProductionGoal` from the input knowledge base $\mathcal{KB}$. Algorithm 2 shows the implementation of the procedure which extracts the set $\mathcal{I}_G$ of individuals of `ProductionGoal` from input $\mathcal{KB}$ (row 2). It specifically retrieves all the triples matching the pattern ⟨?g rdf:type ProductionGoal⟩. Such triples contain the RDF ID of the individuals of `ProductionGoal` in place of the variable ?g. Each individual $g_i \in \mathcal{I}_G$ represents a high-level planning request to the task planner for implementing a collaborative process. These individuals $\mathcal{I}_G$ are then used to generate the state variable SV_G (rows 4–8).

There is a default state $v_{G,0}$ of the state variable (e.g., an idle state denoting the fact that no production goal is being performed at a given interval of time) that is explicitly added to V_G (row 3). For each individual g_i (i.e., goal) then the procedure retrieves a label l_{g_i} used to uniquely identify the individual in the knowledge base $\mathcal{KB}$ (row 4).

[5] https://jena.apache.org

Algorithm 2 `createGoalSV`: procedure for the generation of the goal state variable

Require: $\mathcal{KB}$ ▷ Production knowledge
Ensure: $SV_G : \langle V_G, T_G, D_G, \gamma_G \rangle$ ▷ Goal State Variable
1: $V_G, T_G, D_G, \gamma_G \leftarrow \{\}$
2: $\mathcal{I}_G \leftarrow \{g : g \in$ `ProductionGoal`$\}$ ▷ Retrieve individuals `ProductionGoal` from $\mathcal{KB}$
3: $V_G \leftarrow \{v_{G,0}\}$ ▷ Set the default *Idle* state of $\mathcal{SV}_\mathcal{G}$
4: **for** $g_i \in \mathcal{I}_G$ **do**
5: $v_{G,i} \leftarrow \{l_{g_i} : \langle g_i, l_{g_i} \rangle \in$ `DUL:hasLabel`$\}$ ▷ Retrieve a label identifying the individual $g_i \in$ `ProductionGoal` within $\mathcal{KB}$.
6: $V_G \leftarrow V_G \cup \{v_{G,i}\}$
7: $T_G(v_{G,i}) \leftarrow \{v_{G,0}\}$
8: $T_G(v_{G,0}) \leftarrow T_G(v_{G,0}) \cup \{v_{G,i}\}$
9: **return** $SV_G : \langle V_G, T_G, D_G, \gamma_G \rangle$

The label lg_i is used to define the state variable value $v_{G,i}$ associated with the goal g_i and added to the set of values V_G (row 5). The procedure then set the transition constraints T_G of the state variables. The goal state variable SV_G allows transitions from a value $v_{G,i}$ associated with the goal $g_i \in \mathcal{I}_G$ to the default value $v_{G,0}$ only (row 7). While it allows transitions from the default value $v_{G,0}$ to all other values $v_{G,i} \in V_G$ (row 8). The duration function D_G and the controllability tagging function γ_G of SV_G are then defined using default values. Each value $v_{G,i}$ is associated with a default duration bound $(1, +\infty)$ and tagged as *controllable*, $\gamma_G(v_{G,i}) = c$.

Going back to Algorithm 1, the procedure continues with the generation of the state variables of the agents of the scenario (rows 3). In the specific case of HRC, two state variables are generated. One state variable describes the low-level operations the human can perform over time SV_H. Another state variable describes the low-level operations the robot can perform over time SV_R. These state variables are generated by retrieving individuals of `Function` that are associated with agents (i.e., individuals of `DUL:Agent`). Algorithm 3 further describes how agent state variables are generated. The procedure extracts the set of individuals I_A of `DUL:Agent` matching the pattern $\langle ?a$ `rdf:type DUL:Agent`$\rangle$ (row 1). For each individual of agent $a_i \in I_A$, the procedure retrieves the set of individuals I_F of `Function` that can be performed by the agent a_i (row 4), as inferred by Equation (6). The procedure retrieves only the (sub)set of individuals $f_{a,i}$ of `Function` that match the pattern $\langle ?f_{a,j}$ `canBePerformedBy` $?a_i \rangle$. These individuals are then used to generate the state variable $SV_{A,i}$ of the agent a_i (rows 6–14).

Specifically, the set of values $V_{A,i}$ of the state variable is set according to the names of the functions $f_{i,j} \in \mathcal{I}_F$ the agent a_i can perform (rows 7–8). Transition constraints are set following the same simple structure described for the goal state variable i.e, transitions are allowed from a default idle state $v^0_{A,i}$ to all other values $v^j_{A,i}$, and from a "function" value $f^j_{A,i}$ to the default idle value $v^0_{A,i}$ (rows 9–10). Each state variable value associated with a concrete function is associated with a flexible duration bound (rows 11–13). Lower and upper bounds are set by retrieving data about the nominal duration (row 11) and the expected uncertainty of the execution of the functions (row 12). Finally, the procedure sets the controllability information of the generated state variable values (row 14). In the case of HRC, the values of the human state variable SV_H are tagged as *uncontrollable* (u), while the values of the robot state variable SV_R are tagged as *partially controllable* (pc).

Define task-level state variables The last set of state variables created by Algorithm 1 concerns the set of `ProductionTask` (rows 5–12) defining production operations at different levels of abstraction and supporting decomposition of high-level goals (i.e., *ProductionGoal*) into low-level operations performed by the human and the robot (i.e., `Function`). Production procedures are generally encoded in a hierarchical way into the knowledge base through the property `DUL:hasConstituent`. To generate the set of task state variable SV^l_T the procedure extracts a *decomposition graph* $\mathcal{G} = \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ (row 5) to capture the hierarchical relationships among `ProductionTask` $\mathcal{T}$ (row 6).

The graph $\mathcal{G}$ is an AND/OR graph composed of three types of nodes: (i) *task nodes*; (ii) *AND nodes* and; (iii) *OR nodes*. A *task node* is connected to a *AND node*, to a *OR node*, or to no nodes (leaves). Edges represent decomposition relationships between nodes. If a node n_i is connected to an AND node n^{AND}_i then all connected *task nodes* n_k (i.e., the task nodes n_k reachable from n_i through the *AND node* n^{AND}_i) represent a *conjunctive decomposition* of n_i. Namely, all `ProductionTask` t_k associated with the task node n_k must be executed to implement

Algorithm 3 `createAgentSV`: procedure for the generation of the agent state variables

Require: $\mathcal{KB}$ ▷ Production knowledge
Ensure: $\mathcal{SV}_A$ ▷ List of Agent State Variables
1: $\mathcal{I}_A \leftarrow \{a_i : a_i \in$ `DUL:Agent`$\}$ ▷ Retrieve individuals of `DUL:Agent` from $\mathcal{KB}$
2: **for** $a_i \in \mathcal{I}_A$ **do**
3: $SV_{A,i} \leftarrow \langle V_{A,i}, T_{A,i}, D_{A,i}, \gamma_{A,i} \rangle$ ▷ Initialize state variable $SV_{A,i}$ for agent a_i
4: $\mathcal{I}_F \leftarrow \{f_{i,j} : \langle f_{i,j}, a_i \rangle \in$ `canBePerformedBy` $\wedge f_{i,j} \in$ `Function`$\}$ ▷ Retrieve individuals of `Function`
5: $V_{A,i} \leftarrow \{v^0_{A,i}\}$
6: **for** $f_{i,j} \in \mathcal{I}_F$ **do**
7: $v^j_{A,i} \leftarrow \{l_{i,j} : \langle f_{i,j}, l_{i,j} \rangle \in$ `hasProcedureName`$\}$ ▷ Retrieve the name of $f_{i,j}$ through data property `hasProcedureName`
8: $V_{A,i} \leftarrow V_{A,i} \cup \{v^j_{A,i}\}$
9: $T_{A,i}(v^j_{A,i}) \leftarrow \{v^0_{A,i}\}$
10: $T_{A,i}(v^0_{A,i}) \leftarrow T_{A,i}(v^0_{A,i}) \cup \{v^j_{A,i}\}$
11: $d^j_{A,i} \leftarrow \{d_{i,j} : \langle f_{i,j}, d_{i,j} \rangle \in$ `hasDuration`$\}$ ▷ Retrieve nominal duration of $f_{i,j}$ through data property `hasDuration`
12: $\delta^j_{A,i} \leftarrow \{\delta_{i,j} : \langle f_{i,j}, \delta_{i,j} \rangle \in$ `hasUncertainty`$\}$ ▷ Retrieve uncertainty of $f_{i,j}$ through data property `hasUncertainty`
13: $D_{A,i}(v^j_{A,i}) \leftarrow (d^j_{A,i} - \delta^j_{A,i}, d^j_{A,i} + \delta^j_{A,i})$
14: $\gamma_{A,i}(v^j_{A,i}) \leftarrow pc \vee u$ ▷ Set the function $f_{i,j}$ as partially controllable (pc) or uncontrollable (u)
15: $\mathcal{SV}_A \leftarrow \mathcal{SV}_A \cup \{SV_{A,i}\}$ ▷ Add agent state variable $SV_{A,i}$
16: **return** $\mathcal{SV}_A$

the parent`ProductionTask` t_i or `ProductionGoal` g_i correctly. If a node n_i is connected to a *OR node* $n^{\text{OR}}_{i,j}$ all connected *task nodes* n_k (i.e., the task nodes n_k reachable from n_i through the OR node $n^{\text{OR}}_{i,j}$) represent the alternative decomposition (i.e., procedural disjunctions).

Algorithm 4 shows how the graph $\mathcal{G} = \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ is actually built extracting information from the knowledge base $\mathcal{KB}$. The procedure first extracts the individuals of `ProductionGoal` $\mathcal{I}_G$ from $\mathcal{KB}$ (row 2). Starting from these goals, the graph $\mathcal{G}$ is incrementally refined by exploring possible decomposition through the property `DUL:has-Constituent` (rows 3–30).

For each individual $g_i \in \mathcal{I}_G$, the procedure creates and adds to the graph a task node n_i (rows 4–5). For each goal $g_i \in \mathcal{I}_G$, the procedure retrieves the set of individuals of `ProductionMethod` $\mathcal{I}_{M,i}$ finding triples that match the pattern $\langle ?g_i$ dul:hasConstituent $?m_{i,j} \rangle$ (row 6). If more than one method is found (i.e., $|\mathcal{I}_{M,i}| > 1$) then each method $m_{i,j} \in \mathcal{I}_{M,i}$ describes an alternative procedure associated with `ProductionGoal` g_i (rows 7–20). If only one method is found ($|\mathcal{I}_{M,i}| == 1$) then a single decomposition is associated with `ProductionGoal` g_i (rows 21–30).

- In the first case, the procedure creates an OR node n^{OR}_i and associates it to the task node n_t through the directed edge $\langle n_i, n^{\text{OR}}_i \rangle$ (rows 8–10). The OR node n^{OR}_i should in turn be connected to the alternative decomposition associated with the individual of `ProductionMethod` found $\mathcal{I}_{M,i}$. For each method $m_{i,j} \in \mathcal{I}_{M,i}$ the procedure creates a AND node $n^{\text{AND}}_{i,j}$ connected to n^{OR}_i through the edge $\langle n^{\text{OR}}_i, n^{\text{AND}}_{i,j} \rangle$ (rows 11–14). Given a particular individual $m_{i,j} \in \mathcal{I}_{M,j}$ of `ProductionMethod` the procedure retrieves all associated individuals of `ProductionTask` $\mathcal{I}_{T,j}$ matching the triple pattern $\langle ?m_{i,j}$ dul:hasConstituent $?t_k \rangle$ (row 15). For each task $t_k \in \mathcal{T}_{T,j}$ the procedure creates a new task node n_k and connected to the AND node $n^{\text{AND}}_{i,j}$ through the edge $\langle n^{\text{AND}}_{i,j}, n_k \rangle$ (rows 16–19). At this point, the graph $\mathcal{G}$ is refined by going deeper into the procedural decomposition of `ProductionTask` t_k calling the recursive procedure `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_k)$ described by Algorithm 5 (row 20).
- In the second case, only one method $m_{i,j} \in \mathcal{I}_{M,j}$ is found and there is no need to encode a disjunctive decomposition of `ProductionGoal` g_i into the graph $\mathcal{G}$. The procedure first creates a AND node n^{AND}_i associated with the task node n_i of the `ProductionGoal` g_i through the edge $\langle n_i, n^{\text{AND}}_i \rangle$ (rows 22–24). It then extracts all the triples matching the pattern $\langle ?m_{i,j}$ dul:hasConstituent $?t \angle$ to find the set of individuals $\mathcal{I}_{T,j}$ of `ProdcutionTask` decomposing `ProductionGoal` g_i (row 25). For each task $t_k \in \mathcal{I}_{T,j}$, the procedure creates a task node n_k and associates it with the AND node n^{AND}_i through the edge $\langle n^{\text{AND}}_i, n_k$ (rows 26–29). At

Algorithm 4 `createDecompositionGraph`: procedure building an AND/OR graph encapsulating task decomposition of `ProductionTask`

Require: $\mathcal{KB}$ ▷ Production knowledge
Ensure: $\mathcal{G} : \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ ▷ Directed graph representing modeled production procedures
1: $\mathcal{N}_T, \mathcal{E}_T \leftarrow \{\}$
2: $\mathcal{I}_G \leftarrow \{g_i : g_i \in$ `ProductionGoal`$\}$ ▷ Retrieve individuals of `ProductionGoal`
3: **for** $g_i \in \mathcal{I}_G$ **do**
4: $n_i \leftarrow$ `createTaskNode`(g_i) ▷ Create a root node of the graph associated with the `ProductionGoal` g_i
5: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_i\}$
6: $\mathcal{I}_{M,i} \leftarrow \{m_{i,j} : \langle g_i, m_{i,j} \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Retrieve individuals of `ProductionMethod` associated with g_i
7: **if** $|\mathcal{I}_{M,i}| > 1$ **then** ▷ Disjunctive decomposition of g_i through multiple methods $m_{i,j} \in \mathcal{I}_{M,i}$
8: $n_i^{\text{OR}} \leftarrow$ `createORNode()`
9: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_i^{\text{OR}}\}$
10: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i, n_i^{\text{OR}} \rangle\}$
11: **for** $m_{i,j} \in \mathcal{I}_{M,i}$ **do**
12: $m_{i,j}^{\text{AND}} \leftarrow$ `createANDNode()`
13: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_{i,j}^{\text{AND}}\}$
14: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i^{\text{OR}}, n_{i,j}^{\text{AND}} \rangle\}$
15: $\mathcal{I}_{T,j} \leftarrow \{t_k : \langle m_{i,j}, t_k \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Retrieve `ProductionTask` t_k that are constituent of g_i through $m_{i,j}$
16: **for** $t_k \in \mathcal{I}_{T,j}$ **do**
17: $n_k \leftarrow$ `createTaskNode`(t_k)
18: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_k\}$
19: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_{i,j}^{\text{AND}}, n_k \rangle\}$
20: `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_k)$ ▷ Recursive refinement of the graph, see Algorithm 5
21: **else** ▷ Direct decomposition of g_i through a single method $m_{i,j} \in \mathcal{I}_{M,i}$ i.e., $|\mathcal{I}_{M,i}| == 1$
22: $n_i^{\text{AND}} \leftarrow$ `createANDNode()`
23: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_i^{\text{AND}}\}$
24: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i, n_i^{\text{AND}} \rangle\}$
25: $\mathcal{I}_{T,j} \leftarrow \{t_k : \langle m_{i,j}, t_k \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Retrieve `ProductionTask` t_k that are constituent of g_i through $m_{i,j}$
26: **for** $t_k \in \mathcal{I}_{T,j}$ **do**
27: $n_k \leftarrow$ `createTaskNode`(t_k)
28: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_k\}$
29: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i^{\text{AND}}, n_k \rangle\}$
30: `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_k)$ ▷ Recursive refinement of the graph, see Algorithm 5
31: **return** $\mathcal{G} : \langle \mathcal{N}_T, \mathcal{E}_T \rangle$

this point, the graph $\mathcal{G}$ is refined by going deeper into the procedural decomposition of `ProductionTask` t_k calling the recursive procedure `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_k)$ described by Algorithm 5 (row 30).

Algorithm 5 recursively refines the graph $\mathcal{G}$ by exploring the hierarchical structure of `ProductionTask` through the property `DUL:hasConstituent`. Individuals match triple pattern $\langle ?t_i$ dul:hasConstituent $t_{i,j} \rangle$. The resulting set of individuals of `ProductionTask` $I_{T,j}$ represent the decomposition of t_i (row 1). The decomposition is interpreted according to the type of the input `ProductionTask` t_i.

- If t_i is a `DisjunctiveTask` (row 2) then the graph $\mathcal{G}$ is refined interpreting sub-tasks $t_{i,j} \in \mathcal{I}_{T,i}$ as disjunctive decomposition of t_i (rows 2–10). The procedure creates a OR node n_i^{OR} connected to the task node n_i of the input task t_i through the edge $\langle n_i, n_{i,j}^{\text{OR}} \rangle$ (rows 3–5). For each sub-task $t_{i,j} \in \mathcal{I}_{T,j}$ a new task node $n_{i,j}$ for task the $t_{i,j}$ is created and connected to the OR node n_i^{OR} through the edge $\langle n_i^{\text{OR}}, n_{i,j} \rangle$ (rows 6–9). A recursive call then refines the graph $\mathcal{G}$ by deeper navigating the procedural description through the sub-task $t_{i,j}$ (row 10).
- If t_i is a `ConjunctiveTask` (row 11) then the graph $\mathcal{G}$ is refined interpreting sub-tasks of $t_{i,j} \in \mathcal{I}_{T,i}$ as conjunctive decomposition of t_i (rows 12–19). The procedure creates a AND node n_i^{AND} connected to the task node n_i of the input task t_i through the edge $\langle n_i, n_{i,j}^{\text{AND}} \rangle$ (rows 12–14). For each sub-task $t_{i,j} \in \mathcal{I}_{T,j}$ a task node $n_{i,j}$ for the task $t_{i,j}$ is created and connected to the AND node n_{AND}^t through the edge $\langle n_i^{\text{AND}}, n_{i,j} \rangle$ (rows

Algorithm 5 `graphRefinement`: recursive refinement of the decomposition graph navigating the task procedural implementation into the knowledge base

Require: $\langle \mathcal{KB}, \mathcal{G}, t_i \rangle$ ▷ Production knowledge, decomposition graph and the `ProductionTask` t_i to decompose
1: $\mathcal{I}_{T,i} \leftarrow \{t_{i,j} : \langle t_i, t_{i,j} \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Retrieve individual of `ProductionTask` decomposing input task t_i
2: **if** $t_i \in$ `DisjunctiveTask` **then** ▷ Interpret the decomposition as disjunction
3: $n_i^{\text{OR}} \leftarrow$ `createORNode()`
4: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_i^{\text{OR}}\}$
5: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i, n_i^{\text{OR}} \rangle\}$
6: **for** $t_{i,j} \in \mathcal{I}_{t,i}$ **do**
7: $n_{i,j} \leftarrow$ `createTaskNode`$(t_{i,j})$
8: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_{i,j}\}$
9: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i^{\text{OR}}, n_{i,j} \rangle\}$
10: `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_{i,j})$ ▷ Recursive call to navigate the decomposition deeper
11: **else if** $t_i \in$ `ConjunctiveTask` **then** ▷ Interpret the decomposition as conjunction
12: $n_i^{\text{AND}} \leftarrow$ `createANDNode()`
13: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_i^{\text{AND}}\}$
14: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i, n_i^{\text{AND}} \rangle\}$
15: **for** $t_{i,j} \in \mathcal{I}_{T,i}$ **do**
16: $n_{i,j} \leftarrow$ `createTaskNode`$(t_{i,j})$
17: $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \{n_{i,j}\}$
18: $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\langle n_i^{\text{AND}}, n_{i,j} \rangle\}$
19: `graphRefinement`$(\mathcal{KB}, \mathcal{G}, t_{i,j})$ ▷ Recursive call to navigate the decomposition deeper
20: **else**
21: ▷ individual of `SimpleTask` are leaves of the graph and no further decomposition is necessary

16–18). A recursive call then refines the graph $\mathcal{G}$ by deeper navigating the procedural description through the sub-task $t_{i,j}$ (row 19).

– If the task t_i is a `SimpleTask` (rows 20) no further refinement of the graph $\mathcal{G}$ is necessary. This is the base case stopping the recursion.

Going back to Algorithm 1, the graph $\mathcal{G}$ generated by Algorithm 4 characterizes the decomposition of `ProductionGoal` into `ComplexTask` and `SimpleTask` through associated `ProductionMehtod` (row 5). The sub-procedure `hierarchy`$(\mathcal{G})$ extracts the hierarchical layering of `ProductionTask` $\mathcal{T}$ by running a *topological sort algorithm* on $\mathcal{G}$ (row 6). These two auxiliary data structures then support the definition of the remaining task state variables and synchronization rules of the timeline-based model (respectively rows 7–12 and rows 13–23).

The hierarchical layering of `ProductionTask` $\mathcal{T}$ partitions the set of tasks in $|\mathcal{T}|$ sets each containing `ProductionTask` at the same level of abstraction. The higher hierarchical level of the layering (i.e., the sub-set at $l = 0$) corresponds to the root nodes of $\mathcal{G}$ and the individuals of `ProductionGoal` extracted from the $\mathcal{KB}$. This set of goals is already encoded into SV_G and therefore the construction of task-level state variables starts from the next layering level $l \leftarrow 1$ (row 7). For each layer $l \leqslant |\mathcal{T}|$, the procedure creates a dedicated task-level state variable SV_T^l (rows 9–12). Specifically, the set of individuals of `ProductionTask` $\mathcal{T}^l = \{t_0^l, \ldots, t_k^l\} \in \mathcal{T}$ used to create the task-level state variable SV_T^l through the procedure `createTaskSV` working in a similar way to Algorithm 2.

Define synchronization rules The definition of the synchronization rules starts with defining the rules that constrain the decomposition of goals (i.e., values of SV_G) into production tasks (i.e., values of SV_L, with growing values of L). The procedure iterates over the hierarchical layering of $\mathcal{T}$ extracted from $\mathcal{G}$ (rows 13–23). At each hierarchical level, $l < |\mathcal{T}|$ the procedure retrieves the set of individuals of production tasks or goals $T^l = \{t_0^l, \ldots, t_k^l\}$ that characterize the production procedure at the hierarchical level l (row 15). For each task of the level $t_i^l \in \mathcal{T}^l$ the procedure generates the needed synchronization rules with t_i^l as a trigger (rows 17–22). If the task t_i^l is a `ComplexTask` or a `ProductionGoal` a number of rules constraining possible decomposition of the task t_i^l are generated through Algorithm 6. If the task t_i^l is a `SimpleTask` a number of rules constraining possible (collaborative) implementations of the task t_i^l are generated through Algorithm 7.

Algorithm 6 `createDecompositionRule`: definition of synchronization rules decomposing an input task

Require: $\langle \mathcal{KB}, t_i \rangle$ ▷ Production knowledge, production task to be decomposed
Ensure: $\mathcal{R}$ ▷ Set of synchronization rules modeling possible decomposition
1: $\mathcal{I}_i \leftarrow \{t_{i,j} : \langle t_i, t_{i,j} \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Individuals representing decomposition of t_i
2: **if** $t_i \in$ `ConjunctiveTask` **then** ▷ The input task t_i is conjunctive
3: $R_H \leftarrow$ `createRuleHead`(t_i)
4: **for** $t_{i,j} \in \mathcal{I}_i$ **do** ▷ Create a rule body with all tasks $t_{i,j} \in \mathcal{I}_i$
5: $a_{i,j} \leftarrow$ `createTokeVariable`$(t_{i,j})$
6: $R_B \leftarrow R_B \cup \{a_{i,j}\}$
7: $I_c \leftarrow \{c_i : \langle t_i, c_i \rangle \in$ `ProductionNorm` $\wedge\, c_i \in$ `OperativeConstraint`$\}$
8: **for** $c_i \in \mathcal{I}_c$ **do**
9: $r_{i,j} \leftarrow$ `createTemporalRelation`(c_i)
10: $R_C \leftarrow R_C \cup \{r_{i,j}\}$
11: $\mathcal{R} \leftarrow \{\langle R_H, R_B, R_C \rangle\}$
12: **else** ▷ The input task t_i is disjunctive
13: **for** $t_{i,j} \in \mathcal{I}_i$ **do** ▷ Create a distinct rule for each task $t_{i,j} \in \mathcal{I}_i$
14: $R_H \leftarrow$ `createRuleHead`(t_i)
15: $a_{i,j} \leftarrow$ `createTokenVariable`$(t_{i,j})$
16: $R_B \leftarrow \{a_{i,j}\}$
17: $\mathcal{R} \leftarrow \mathcal{R} \cup \{\langle R_H, R_B, \emptyset \rangle\}$
18: **return** $\mathcal{R}$

Algorithm 6 shows the procedure generating synchronization rules for the decomposition of a `ProductionTask` t_i into simpler `ProductionTask` $t_{i,j}$. The procedure retrieves the set of individuals of `ProductionTask` $\mathcal{I}_i$ associated with the input task t_i through property `DUL:hasConstituent` (row 1). The generation of the decomposition rules then depends on the type of the input task t_i.

- In the case of a `ConjunctiveTask` $t_i \in$ `ConjunctiveTask` (row 2), the procedure creates a single synchronization rule constraining all the associated sub-tasks $t_{i,j} \in \mathcal{I}_i$ (rows 4–11). Specifically, the head of the rule is associated with input task t_i (row 3). For each sub-task $t_{i,j} \in \mathcal{I}_i$, the procedure creates a token variable $a_{i,j}$ associated with the task (row 5) and adds it to the body of the rule R_B (row 6). The set of temporal relations R_C of the rule is then generated by retrieving information about individuals of `ProductionNorm` I_c associated with the input task t_i (rows 7–10). Temporal relations are generally of type `CONTAINS` constraining the input task t_i and the sub-tasks $t_{i,j}$, and of type `BEFORE` possibly specifying ordering between sub-tasks $t_{i,j}, t_{i_k} \in \mathcal{I}_i$
- In the case of a `DisjunctiveTask` $t_i \in$ `DisjunctiveTask` (row 12), the procedure creates a distinct synchronization rule for each sub-task $t_{i,j} \in \mathcal{I}_i$ (rows 13–17). For each sub-task $t_{i,j} \in \mathcal{I}_i$, the procedure creates a rule with head t_i (row 14), a token variable $a_{i,j}$ associated with the sub-task $t_{i,j}$ as body (rows 15–16) and empty constraints (row 17).

Algorithm 7 generally describes the procedure that creates the synchronization rules constraining the implementation of `SimpleTask` into `Function` performed by `DUL:Agent`. Constraints are set according to the set of `InteractionModality` $\mathcal{I}_c$ associated with the task t_i (row 6). In the case of collaborative tasks, ontological patterns define different types of tasks `HRCTask` determining different combinations of temporal relations between the associated `Function` (i.e., the sub-tasks $t_{i,j} \in \mathcal{I}_i$).

- A task t_i of type `IndependentHRCTask`, according to Equation (16), is associated with one individual of `Function` only (i.e., $|\mathcal{I}_i| == 1$). The set of constraints of the rule R_C contains only one `CONTAINS` relation constraining the trigger of the rule (i.e, the state variable value associated with t_i) and the value of the human or robot state variable associated with $t_{i,j} \in \mathcal{I}_i$.
- A task t_i of type `SimultaneousHRCTask`, according to Equation (17), is associated with one individual of `Function` that can be performed by the `WorkOperator`, and one individual of `Function` that can be performed by the `Cobot` (i.e., $|\mathcal{I}_i| == 2$). The set of constraints R_C contains in this case two `CONTAINS` relations constraining the trigger of the rule (i.e., the state variable value associated with the task t_i) and both

Algorithm 7 `createImplementationRules`: definition of synchronization rules implementing an input task in (patterns of) functions

Require: $\langle \mathcal{KB}, t_i \rangle$ ▷ Production knowledge, production task to be implemented
Ensure: $\mathcal{R} : \langle R_H, R_B, R_C \rangle$ ▷ A synchronization rule implementing the input task
1: $\mathcal{I}_i \leftarrow \{t_{i,j} : \langle t_i, t_{i,j} \rangle \in$ `DUL:hasConstituent`$\}$ ▷ Individuals representing decomposition of t_i
2: $R_H \leftarrow$ `createRuleHead`(t_i)
3: **for** $t_{i,j} \in \mathcal{I}_i$ **do**
4: $\quad a_{i,j} \leftarrow$ `createTokenVariable`$(t_{i,j})$
5: $\quad R_B \leftarrow R_B \cup \{a_{i,j}\}$
6: $\mathcal{I}_c \leftarrow \{c_i : \langle t_i, c_i \rangle \in$ `ProductionNorm` $\wedge\, c_i \in$ `InteractionModality`$\}$
7: **for** $c_i \in \mathcal{I}_c$ **do**
8: $\quad r_{i,j} \leftarrow$ `createTemporalRelation`(c_i)
9: $\quad R_C \leftarrow R_C \cup \{r_{i,j}\}$
10: **return** $\mathcal{R} : \langle R_H, R_B, R_C \rangle$

values of the human and the robot state variables associated with $t_{i,j} \in \mathcal{I}_i$. No additional constraints between the two functions are needed.

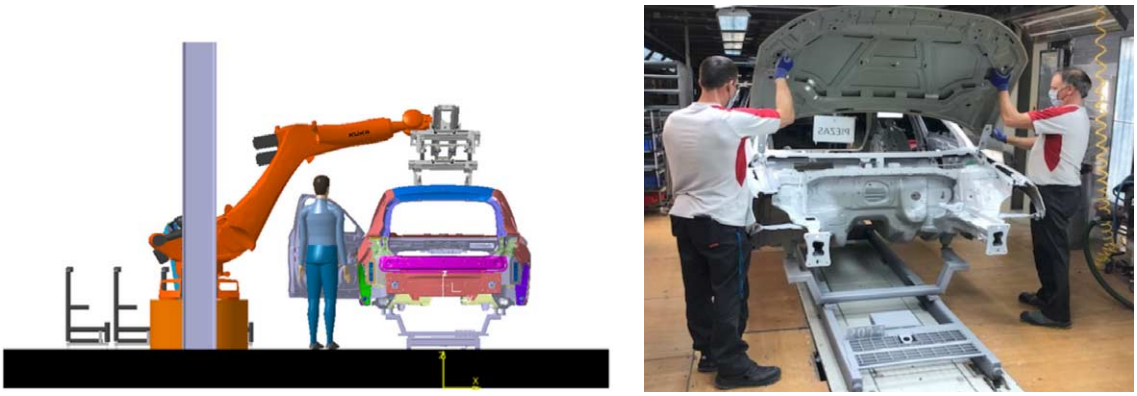
- A task t_i of type `SequentialHRCTask`, according to Equation (18), is associated with one individual of `Function` that can be performed by the `WorkOperator`, and one individual of `Function` that can be performed by the `Cobot` (i.e., $|\mathcal{I}_i| == 2$). Furthermore, the task t_i is associated with an interaction modality $c_i \in \mathcal{I}_c$ of type `PrecedenceConstraint` which specifies an execution order between the two functions $t_{i,j} \in \mathcal{I}_i$. In addition to the `CONTAINS` relations, the set of constraints R_C contains a `BEFORE` relations constraining one function to be executed as *first* (through the property `isFirstTask`) and the other function to be executed as *second* (through the property `isSecondTask`).
- A task t_i of type `SupportiveHRCTask`, according to Equation (19), is associated with one individual of `Function` that can be performed by the `WorkOperator`, and one individual of `Function` that can be performed by the `Cobot` (i.e., $|\mathcal{I}_i| == 2$). The two functions are in this case forced to be executed at the very same time and thus with strong parallelism. The task t_i is indeed associated with a `InteractionModality` of type `ParallelExecutionConstraint` requiring the human and robot functions to start and end at the same time. In addition to the `CONTAINS` relations, the set R_C contains two `EQUALS` relations, constraining the state variable value associated with the trigger t_i and the state variable values associated with the human and robot functions $t_{i,j} \in \mathcal{I}_i$. Such constraints enforce the human and robot functions to be executed during the same (flexible) temporal interval.

5. Assessment on real-world scenarios

We assess SOHO and the developed timeline-based model generation procedure on a number of real HRC scenarios from the pilot use cases of the EU H2020 Sharework project .[6] The ontology has been defined in OWL [3], using Protégé,[7] and is publicly available on a GitHub repository [8] For each use case, a dedicated knowledge base has been defined by gathering information from production engineers and domain experts. Specifically designed forms have been designed and administrated to domain experts in order to collect information about production procedures and desired collaborative dynamics. Collected information has been used by a knowledge engineer to concretely build the knowledge bases (ABoxes) of the different scenarios, using Protégé.[9]

The reasoning mechanisms and the knowledge extraction procedure of Algorithm 1 have been developed in Java using Apache Jena.[10] Representation and reasoning functionalities have been integrated into ROS [11] through

[6] https://sharework-project.eu
[7] https://protege.stanford.edu
[8] https://github.com/pstlab/SOHO.git
[9] The knowledge bases can be found on the GitHub repository of SOHO under the folder "instances".
[10] https://jena.apache.org
[11] Distribution ROS Melodic – http://wiki.ros.org/melodic.



(a) (b)

Fig. 6. Design of the collaborative cell for the AUTOMOTIVE scenario (a) and the production line for the assembly of the chassis (b).

ROSJava[12] to support deployment on (industrial) robots and implement the envisaged cognitive control loop. ROS modules and services developed within Sharework are publicly available on GitHub.[13] The evaluation considers different collaborative scenarios representing realistic production situations, needs, and constraints. Such scenarios are well suited to assess the generality of the proposed ontological model as well as its efficacy in capturing the requirements of real-world applications and synthesizing valid task-planning models. The scenarios are the following: (i) AUTOMOTIVE; (ii) METAL; (iii) CAPITAL-GOODS; (iv) RAILWAYS; (v) MOSAIC. These four scenarios constitute realistic benchmarks, suitable to assess the representation and reasoning capabilities of the developed AI-based technologies.

5.1. Industrial scenarios in detail

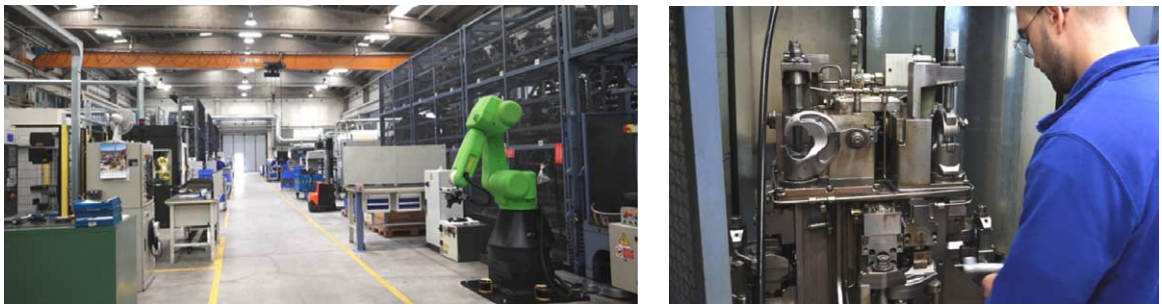
5.1.1. The AUTOMOTIVE *scenario*

This scenario concerns a specific station of an assembly line of vehicles. The collaborative process specifically focuses on a door assembly task of chassis. The collaborative robot is in charge of moving and holding the heavy parts of the vehicle (i.e., *pick-and-place* of front and rear doors to be assembled on the chassis) while the human carries out assembly tasks in the same working area of the robot (i.e., fix the doors to the body of the vehicle). Figure 6 shows some pictures of the layout of the working area with the mount point of the front door on the chassis.

This scenario is characterized by a flat production process where the human and the robot play different roles and carry out tasks autonomously but following a strict order. Door assembly is correctly performed only if the robot and the human execute their task at the correct time (e.g., the human cannot start her task if the robot does not place the door in the expected position). Roles are not interchangeable therefore it would not be possible to carry out a collaborative process without the correct coordination of the two actors. More specifically, the robot (a robotic arm) can perform only `PickPlace` functions on the `rear` and `front` doors of the vehicles (i.e., the `front` and `rear` doors are two `WorkPiece` of the environment). The robot (individual of `Cobot`) can perform two instances of `PickPlace`, one function on the target door `front` and the other one on the target door `rear`. The human instead carry out the actual assembly operations and should therefore perform functions of different type e.g., `Assemble`, `ManualGuidance`, `ChangeOver` and `Screw` on the `vehicle body` (`WorkPiece`).

All `SimpleTask` entailing the direct involvement of human and robot operations are modeled as individuals of `IndependentHRCTask` since both agents can carry out their functions autonomously. For example, the robotic task of moving a door to the front assembly area of the layout is modeled as an independent collaborative task (i.e.,

[12] http://wiki.ros.org/rosjava
[13] https://github.com/pstlab/sharework_knowledge.git



(a) (b)

Fig. 7. Structure of the shop-floor of the METAL industrial scenario (a) and the fixturing system where collaboration takes place (b).

individuals of `IndependentHRCTask`) and implemented by a pick-place operation of the robot (i.e., an instance of `PickPlace` function). However, `ProductionNorm` constraining some `ProductionTask` are necessary to correctly coordinate robot and human operations. The `Assemble` functions of the human that physically mount the doors on the chassis can be performed only if the robot has correctly placed the door in the expected position. Individuals of `PrecedenceConstraint` are thus necessary to constrain the execution of `PickPlace` functions of the robot to occur before the execution of `Assemble` functions of the human.

The production process of this scenario is simple from a control perspective since the possible behaviors of the human and the robot are fixed and there is no need for optimization. It just requires the unfolding of a single `ComplexTask` into a number of `IndependentHRCTask` the human and the robot perform sequentially. However, the integration of developed representation and planning capabilities contributes to facilitating human-robot interactions. The designed cognitive control approach would provide the human worker with detailed information about requested tasks when dispatched by the task planner. The system would inform the worker about tasks assigned to the robot and when they are going to be executed. This information allows the worker to know his/her plan and the current (and planned) behavior of the robot. For example, workers with low expertise may benefit from receiving frequent and detailed information about tasks and the collaborative plan. Expert workers instead may find an excessively frequent and detailed level of information annoying. Information about tasks and the requested feedback can thus change according to the qualities of participating workers [92,93].

5.1.2. The METAL *industrial scenario*

This scenario concerns the logistic station of a manufacturing system for electrical connectors. The workshop for the assembly of pallets and fixtures in load/unload stations is divided into two main areas: (i) a transporter panel buffer, where pallets are stored and moved, and; (ii) some CNC (Computerized Numerical Control) machines, where the pallets are moved to perform the machining operations. In this scenario, operators are generally responsible for transporting pallets and components to be mounted in a *tombstone* that goes inside the Flexible Manufacturing System where each part is machined. The scenario is characterized by high variability of parts to be produced. Operators therefore should be highly trained in order to correctly perform the suitable assembly procedure for each different product as well as perform the quality inspection on the pallets before/after machining. The collaborative robot is in charge of assisting operators when moving across the station, understanding operator's behavior, and anticipating tasks in order to facilitate their work and speed up the production, i.e., increasing the throughput.

Figure 7 shows some pictures of the layout of the logistic station. The working area is characterized by a central/shared conveyor where different types of products are loaded and processed in order to be machined. The worker and the robot (a UR10 robotic arm) are placed on two distinct sides of the conveyor and work simultaneously on the products. Products are placed and moved on the central conveyor which represents the shared working area where operations take place and where the human and the robot physically interact. The production process is characterized by different types of operations depending on the specific types of workpieces entering the collaborative cell. From a task planning perspective, the structure of the assembly/disassembly process is similar in all the cases, since the human and the robot can have the same capabilities (e.g., screw, unscrew, pick, place, etc.). Unlike the AUTOMOTIVE scenario, here the worker and the robot are two interchangeable agents capable of performing the

same tasks. The human and the robot represent therefore two autonomous peer actors that work simultaneously on the workpieces performing assembly/disassembly tasks. The synthesis of collaborative processes thus concerns the correct allocation of tasks to these two *resource* (i.e., autonomous agents).

The process consists in picking and placing workpieces transported by a pallet running over a conveyor. Each workpiece is placed over a pallet entering the conveyor from an initial position (`position0`). Broadly speaking, the procedure consists in replacing the workpiece transported by the pallet. There are then two working positions where the pallet can be moved (`position1` and `position2`). The human moves the pallet to one of the two working positions. Here the pallet is first unmounted in order to release the worked workpiece which is then placed into a dedicated box. Then, a new raw workpiece is picked from another box and mounted into the pallet. At this point, the human moves the pallet to the last position of the conveyor (`position3`) where it can be sent to other stations on the shop floor to be machined. Although the general procedure is the same for all types of workpieces, the geometry and the pallets themselves are different. For this reason, each type of workpiece is associated with a dedicated `ProductionGoal` and thus different individuals of `ProductionTask` and `Function`.

The replacement of a workpiece on a pallet is realized through pick & place operations that can be performed by both the human and the robot. The human or the robot `Pick` the current `WorkPiece` from the base and `Place` it into available containers e.g., `Box-A`, `Box-B` that are modeled as `CapacityResource` (i.e., they can store a limited number of `WorkPiece`). Functions of type `Pick` and `Place` are represented separately (i.e., not as atomic instances of `PickPlace`) to capture operational requirements and properly characterize possible planning choices (e.g., which box to use to place a workpiece). The knowledge base and the associated decomposition graph contain several `DisjunctiveTask` (OR nodes) representing alternative decomposition choices. Each task can be performed either by the worker or by the robot. Such tasks are represented as `DisjunctiveTask` and decomposed into human and robotic pick & place `ComplexTask` through OR nodes. Such tasks are further decomposed into `ProductionTask` characterizing the specific `Pick` and `Place` functions the worker or the robot should perform. Tasks that concern pick operations are generally represented as `IndependentHRCTask` implemented by instances of `Pick`. Tasks that concern place operations are represented as `DisjunctiveTask`. Each decomposition represents an alternative choice of assigning the task of placing a `WorkPiece` into one of the available boxes (i.e., `Box-A` and `Box-B`). Such tasks are represented as `IndependentHRCTask` and implemented by individuals of `Place` that can be performed by the human or by the robot.

The integration of developed representation and planning capabilities contributes to the optimization and safe coordination of human and robot behaviors. The designed cognitive control approach would automatically optimize human and robot behaviors through the generated (timeline-based) task planning model. Similar to AUTOMOTIVE, the cognitive control loop would provide workers with customized information about the tasks he/she should perform. This is especially relevant in this scenario involving a high variety of workpieces and pallets, each with different geometries and low-level operations. For example, the way `Assemble` or `Disassemble` functions are actually implemented by the worker (or by the robot) may significantly change according to the specific shape of the workpiece/pallet. Detailed information enriching dispatched tasks and showing for example contextualized instructions about required manipulation operations would help the human in doing her/his jobs.

5.1.3. The CAPITAL-GOODS *industrial scenario*

This scenario takes into account the shop floor of a company offering differential and global solutions in power transmission and spraying components. This scenario considers a servo rotary table that is assembled in seven fixed assembly stations. In each station, there is an operator performing a specific task in the assembly process. All tasks are carried out manually, just using cranes and lifters to transport the heavy components from one station to another. The collaboration between the human and the robot concerns three out of seven tasks of the rotary table assembly process. In the current assessment, we specifically consider the task *bolt tightening and torque measuring*. Figure 8 shows the designed physical environment with the rotary table equipped with the collaborative robot (a UR10 robotic arm). The operator applies adhesives on the bolts of the rotary table to allow the robot to simultaneously determine its position and dimension through perception modules developed with the project. Information about detected bolts is then used by the robot to automatically screw them.

The developed cognitive approach supports a *reactive* behavior where the robot relies on *perception capabilities* [55] to autonomously recognize human tasks (i.e., "bolt placing" tasks) and act accordingly. The developed knowl-

(a) (b)

Fig. 8. Working area of the CAPITAL-GOODS scenario (a) and structure of the workpiece (b) where a collaborative robot is deployed to support workers in repetitive screwing/unscrewing tasks.

edge representation capabilities indeed allow the robot to recognize new events concerning the placement of bolts by the human and interpret these events as signals triggering the execution of robot tasks (i.e., `ProductionGoal`). On the one hand, knowledge reasoning supports the continuous adaptation of robot behaviors by triggering planning requests based on the observed task performed by humans. On the other hand, knowledge reasoning validates perception outcomes by checking the validity of a detected human task. It could be the case for example that a perception module (wrongly) detects a placing task of a bolt already placed. In such a case the knowledge base would detect this inconsistency when interpreting the observation and no `ProductionGoal` would be triggered to the task planner.

The described reactive behavior represents the actual way the developed knowledge representation and reasoning modules have been deployed into this pilot case [88]. To evaluate the representation capabilities of the ontological model and the task planning model, we here model the whole production process assuming a deliberative approach similar to other use cases. Namely, we do not consider perception outcomes and assume the planner should synthesize screwing tasks of all bolts of the workpiece. The process thus consists of screwing a number $N = 8$ of bolts distributed over a rotary table. We keep the joint collaboration implemented in the real scenario and thus assume that a worker places the bolts on the holes of the rotary table and a robot screws them simultaneously. Although simple, an interesting aspect of this scenario is the *synchronization* required between the human and the robot. The robot can start screwing a bolt only after the worker has placed the bolt on one of the holes of the rotary table. This means that the task of "screwing a bolt" is modeled as a `SequentialHRCTask` requiring: (i) human and robotic functions to have the same target i.e., a particular hole that can be seen as a `SimpleWorkPiece` composing the *CompoundWorkPiece* (the rotary table); (ii) robot and human operations to follow a strict temporal ordering. In this case, the `Screw` function of the robot must always be performed *after* the complete execution of the`PickPlace` function of the human.

A high-level production goal `rotary-table-assembly` is decomposed into a number of (complex) `ConjunctiveTask`, one for each *hole* of the rotary table to be screwed. Each simple task (i.e., screwing tasks) is represented as `SequentialHRCTask` and thus decomposed into two functions. A `PickPlace` function like e.g., `pick-place-H3` requiring the operator to pick and place a bolt on a particular hole (e.g., `H3` represented as `SimpleWorkPiece`). A `Screw` function e.g., `screw-H3` requires the operator to screw the bolt placed by the worker.

5.1.4. The RAILWAYS *industrial scenario*

This scenario takes into account the shop floor of a railways transportation company supplying rolling stocks, services, and system infrastructure. The workshop is composed of six main stations each one dedicated to a specific set of operations concerning the assembly of trains. The project specifically focuses on the pre-assembly process of tramways windows and door frames. Among the tasks involved in the considered processes, *riveting* represents a repetitive and demanding task for human workers. It consists of the insertion of rivets in drilled holes along the metal pieces of window frames. This task is especially critical from *safety perspective* since it may cause significant injuries after a prolonged utilization of the riveting tool which weighs up to 5 kg.

(a) (b)

Fig. 9. Design of the collaborative cell of the RAILWAYS scenario to support workers in the assembly of door frames.

Figure 9 shows the physical layout of the shop floor and the structure of the window frames that are the *target* of the considered production process. The introduction of collaborative robots into the production line is designed to relieve human workers from physically demanding tasks like e.g., the *riveting task* in order to improve their working conditions and reduce the risk of injuries. A collaborative robot is thus supposed to work close to the window frame of Fig. 9(b) and tightly collaborate with the human worker to carry out the pre-assembly tasks. The collaboration takes place within the *riveting task*. The human operator is in charge of spreading the silicone over the corners of the frame structure then, the robot inserts the rivets using a *riveting tool*. It can be observed that the *riveting task* entails a synchronous behavior of the two actors since the robot can insert the rivets only after the worker has correctly applied the silicone. Similar to the screwing tasks of CAPITAL-GOODS, these tasks are represented as instances of `SequentialHRCTask`.

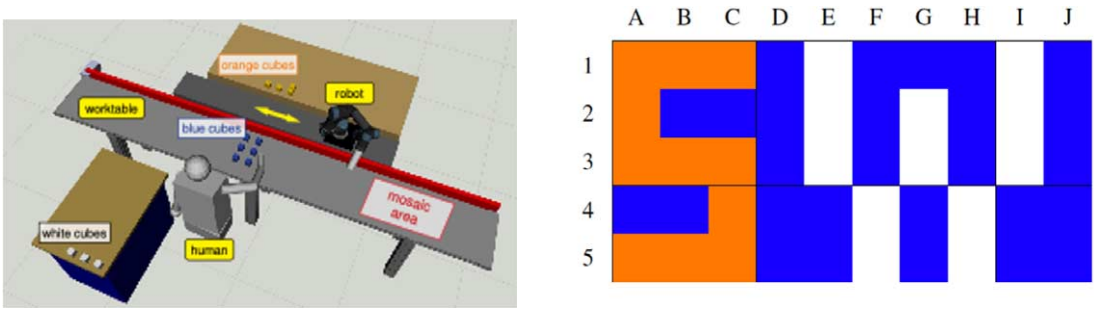
Also in this case there are no disjunctive tasks to be considered in the decomposition since the roles and responsibilities of the worker and the robot are quite fixed. The key aspect is the use of `SequentialHRCTask` to represent riveting tasks. Such tasks are indeed associated with a `Join` function of the worker, representing the application of the silicone over a particular corner of the frame structure, and, to a `Join` function of the robot, representing the use of the *rivet* `Tool` to insert the rivets. Given the needed synchronization the execution of these two functions constrains the robot to wait for the complete execution of the function of the worker.

5.1.5. The MOSAIC *collaborative scenario*

This scenario considers a general collaborative assembly of a compound workpiece. The layout of the work cell is characterized by a shared central space where the workpiece is placed and where the human and the robot simultaneously carry out assembly operations. Figure 10 shows the layout of the designed collaborative environment and the layout of the *mosaic*. The *mosaic* is modeled as a `CompoundWorkPiece` since it can be seen as composed of simpler parts like e.g., *rows* (still `CompoundWorkPiece`) and *cells* (`SimpleWorkPiece`).

The collaborative process consists of the execution of a set of pick & place operations. Each pick & place is performed by a single agent autonomously but their execution (and assignment) should satisfy some physical constraints. Pick & place operations whose targets are colored objects placed in different areas: *blue objects* can be handled by both the human and the robot; *orange objects* for short) can be performed by the robot only; *white area* (i.e., *white objects* for short) can be performed by the human only. Although this scenario does not correspond to a concrete production process of Sharework, it describes a highly flexible collaborative process representing alternative behaviors of the robot and the human into the knowledge and the task planning model [35].

The human and the robot play the same role and carry out tasks autonomously without a specific order. The process can be seen as the problem of moving some `WorkPiece` from an initial location to a desired one in order to form a desired shape. The structure of the process is given by the structure of the desired shape. The shape can be seen as a *mosaic* composed of a certain number of *rows* (e.g., 5 rows) and a certain number of columns (e.g., 10 columns). Each specific location of the mosaic (i.e., *cell(1,1)*,..., *cell(5,10)*) represents a specific destination of a `PickPlace` function, moving a specific `WorkPiece`. The human and the robot can thus perform the same type of `Function` i.e., `PickPlace`. Each `PickPlace` moves a particular instance `wp1`,..., `wpN` of `WorkPiece`



(a) (b)

Fig. 10. Configuration of the MOSAIC use case: (a) structure of the ROS-based simulation of the collaborative cell; (b) layout of the mosaic collaboratively assembled by the human and the robot.

Table 1

Data about the knowledge quality, size of knowledge bases and planning models, and performance of Algorithm 1. Coherence and consistency of the knowledge bases have been evaluated using the protégé debug tool- https://protegewiki.stanford.edu/wiki/OntoDebug

Domain	Knowledge base					Planning model				Model synthesis
	Coherent	Consistent	#Classes	#Properties	#Individuals	#SVs	#Values	#Rules	#Constraints	Time
AUTOMOTIVE	✓	✓	284	186	61	5	34	15	28	≈ 4 seconds
METAL	✓	✓	284	186	45	8	38	57	68	≈ 8 seconds
CAPITAL-GOODS	✓	✓	284	186	42	5	31	9	16	≈ 4 seconds
RAILWAYS	✓	✓	284	186	75	6	64	29	48	≈ 9 seconds
MOSAIC	✓	✓	284	186	215	7	195	137	186	≈ 20 seconds

to a specific location of the mosaic like e.g., `pickplace-wp1-cell(1,1)-human` and `pickplace-wp1-cell(1,1)-robot`. Each `SimpleTask` of the production process is defined as an instance of `IndependentHRCTask` which should be associated with a `PickPlace` function of the human or a `PickPlace` function of the robot.

5.2. *Generation of planning specification from knowledge*

Given the knowledge bases of the different scenarios, this section assesses the technical feasibility of the implemented representation and reasoning processes. This section in particular evaluates the validity of the semantic models and the resulting timeline-based specifications. For each scenario, it shows the time needed for the synthesis of the task planning models in order to measure the introduced production latency. Table 1 summarizes and compares data about the defined knowledge bases, the characteristics of the synthesized planning models, and the reasoning time of Algorithm 1. As a general comment, the obtained knowledge bases have the same number of defined classes and properties since they share the same ontological framework but for each of them, it can be observed a different number of *individuals* and a different number of synchronization rules, temporal constraints, and state variable values in the resulting timeline-based planning models.

The structure and the size of the obtained planning models indeed differ significantly from each other (see "Planning Model" columns in Table 1) leading to different model generation times. For example, The planning model generated for AUTOMOTIVE is characterized by a lower number of predicates and synchronization rules/constraints. The planning model generated for MOSAIC instead, given its structural complexity entails a higher number of predicates (195) as well as synchronization rules and constraints (respectively 137 and 186). In all cases, the ontology was capable of capturing all the needed information and the obtained planning models were valid and feasible for a correct deployment of the task planning module in the associated scenario. The performances of the generation process are reasonable (e.g., 4 seconds for AUTOMOTIVE and 20 seconds for MOSAIC) demonstrating the feasibility of the proposed approach. The model generation time should be seen as a "constant" *setup time* of the task planner.

Although it may vary significantly according to the size of the knowledge base, this cost is negligible with respect to the online functioning of a task planner since it is compatible with the usual control latency and it does not affect the overall efficiency of the system.

The results have been obtained by configuring the Apache Jena framework with an ontological model compliant with OWL-DL semantics.[14] This semantics supports many OWL features and represents a good trade-off between the expressiveness and efficiency of the resulting functionalities. Examples are *disjoint classes* and *all different axioms* that allow the reasoner to correctly interpret individuals of the same class (*siblings*) as *unique* knowledge entities. Generated planning models have been validated with PLATINUm,[15] a timeline-based planning and scheduling framework [89]. Figure 11 shows the hierarchical structures of the different planning models, automatically synthesized through Algorithm 1.

Each level represents a specific abstraction level of the defined hierarchical production procedures. The highest level of the hierarchy characterizes the high-level production goals that are incrementally decomposed into lower-level tasks (i.e., production operations). The lowest level of the hierarchy characterizes the tasks the worker and the robot can perform to support the associated production procedures. Following this hierarchy, as shown in Section 4.2.2, each hierarchy level is associated with a dedicated *state variable*. Values of such state variables describe tasks that should be performed at the associated abstraction level of the production procedure. State variables of the last hierarchical layer describe the concrete operations (i.e., functions) the worker and the human can perform over time and thus define their possible behaviors within a specific production scenario.

Given a timeline-based model, plans specify for each state variable of the model sequences of *tokens* determining the production tasks performed and the low-level production operations carried out by the worker and the operator. Such sequences of tokens (i.e., *timelines*), as shown in other works [35,69], describe the planned decomposition of modeled production procedures and planned *temporal behaviors* of collaborating actors (i.e., the worker and the robot). Following [23,24] then, each token instantiates a value of a state variable to a flexible execution interval (the intervals associated with each token represent respectively the end-time interval and the duration interval of the execution). Timelines, therefore, are said to encapsulate *envelopes* of temporal behaviors. One interesting aspect to point out is how the defined *ontological patterns* that formally characterize the representation structure of collaborative tasks, entail a clear and well-defined structure of the defined synchronization constraints that implement that *collaborative behavior*.

5.3. Discussion of results and impact

The developed representation and reasoning capabilities impact different aspects concerning the design and deployment of collaborative robots. The proposed approach would facilitate modeling, maintenance, and adaptation of control dynamics to the different (evolving) requirements of industrial scenarios. Table 2 lists the aspects concerning HRC production systems that are supported by the proposed approach. The table in particular shows the relevance of each aspect taking into account the production and interaction features of each pilot. In this way, it points out the flexibility of the approach according to the requirements of different scenarios.

The ontology proposes a kind of standardization of production knowledge and describes collaborative scenarios based on a clear and well-structured formalism. Defined concepts and properties characterize production requirements, interacting features, and skills of robotic and human actors according to clear semantics. The assessment shows that SOHO is suitable to describe both scenarios requiring simple and strict interactions between e.g., AUTOMOTIVE or `capital-goods`, and scenarios requiring more complex constraints and decomposition procedures e.g., METAL, RAILWAYS and MOSAIC.

The integrated representation of the *human factor* allows SOHO and the developed ontology-based control approach to contextualize production dynamics to the known features and skills of human workers. This knowledge

[14] More specifically, we have defined a basic ontological model with OWL-DL-MEM specification combined with a rule-based inference model based on OWL-MEM-MICRO-RULE-INF which encapsulates optimized rule-based reasoner with OWL rules. See the official Apache Jena documentation for further details – https://jena.apache.org/documentation.

[15] https://github.com/pstlab/PLATINUm.git

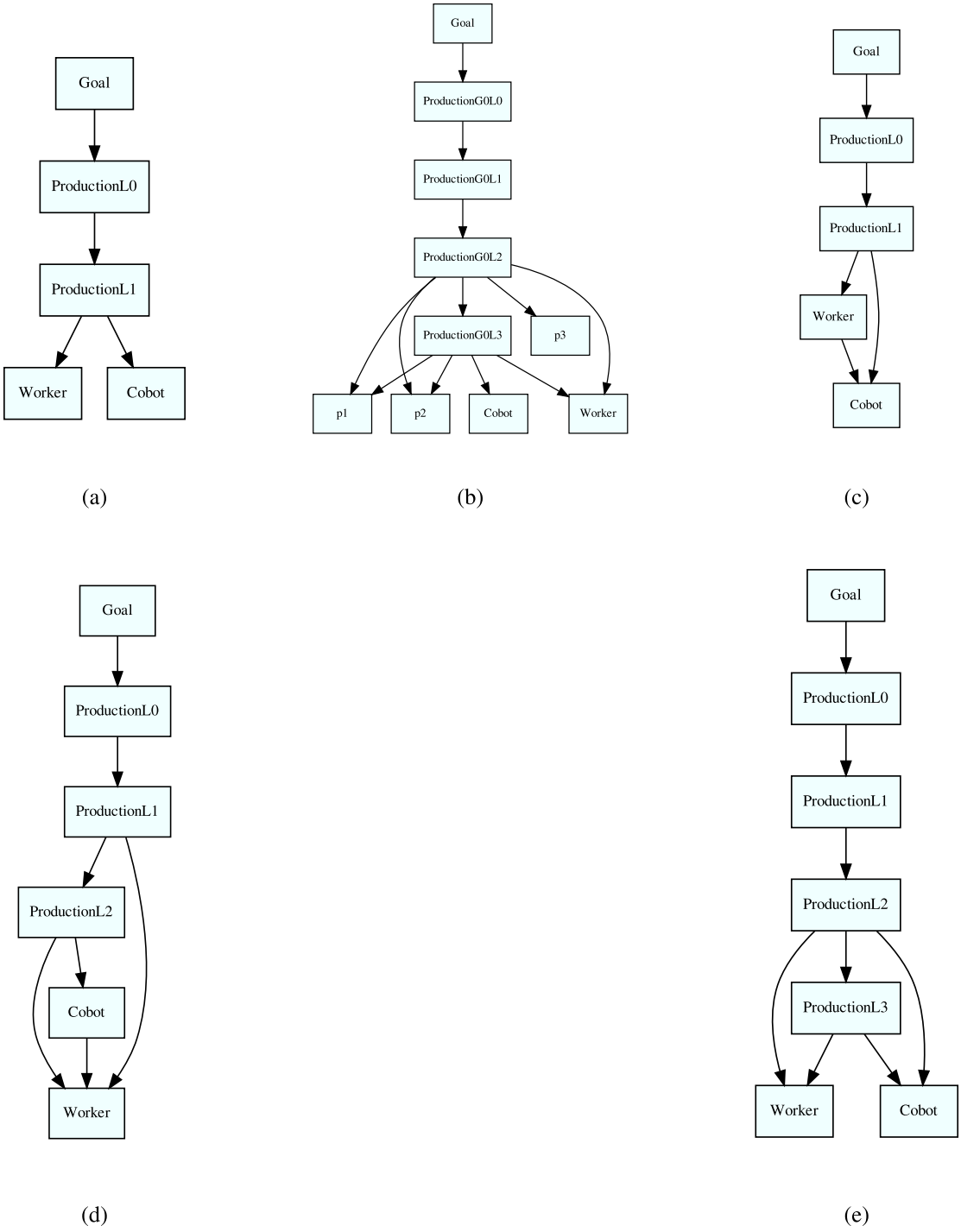


Fig. 11. Hierarchical structure of the generated planning models. The two graphs show the inferred hierarchical relationships between the *state variables* generated for: (a) AUTOMOTIVE; (b) METAL; (c) CAPITAL-GOODS; (d) RAILWAYS; (e) MOSAIC.

Table 2
Aspects concerning the design and deployment of collaborative robots that are supported by the proposed ontology-based approach, and their impact on the considered HRC scenarios

	AUTOMOTIVE	METAL	CAPITAL-GOODS	RAILWAYS	MOSAIC
Knowledge Engineering	M	H	M	H	H
Personalization and Human Factor	L	H	M	L	H
Multi-objective Optimization	L	M	L	M	H
Awareness and Proactive Support	L	H	H	M	H
Modular Robot Programming	M	H	M	M	M

H = high relevance, M = medium relevance, L = low relevance.

combined with task planning supports the synthesis of personalized collaborative plans adapted to the expected performance (e.g., expected average time of task execution) and expertise (e.g., worker experience) [92]. This knowledge and related reasoning capabilities are especially important in scenarios characterized by a high variety of workpieces and operations e.g., METAL or MOSAIC. The automatically configured task planner would thus assign tasks and dispatch instructions that are suitable and contextualized to the known *profile* of a worker. For example, the task planner would provide workers with a low level of experience with a higher number of instructions in order to better support the execution of assigned tasks. Furthermore, the tight integration with task planning technologies supports the automatic synthesis of *optimized* collaborative plans. Timeline-based planning capabilities in particular support multi-objective optimization by simultaneously reasoning on different metrics concerning, for example, the cycle time of a collaborative process, the risk of collisions, and the workload distribution between the human and the robot [69]. In this regard, the automatic synthesis of (valid) task planning models supports a direct encoding of domain expert knowledge into the robot controller, reducing modeling effort and the risk of inconsistencies. An optimization approach to the synthesis of collaborative plans is crucial to effectively combine human and robot skills in scenarios characterized by a high number of task allocation choices (e.g., MOSAIC) as well as multiple sets of operations and workpiece (e.g., METAL).

Another important aspect is the enhanced awareness of robot controllers. Developed knowledge representation and reasoning capabilities allow robot controllers to build and maintain an updated description of the production dynamics and observed state of the production environment. The semantics of SOHO in particular supports the abstraction and contextualization of sensing data that would be useful to recognize relevant production situations and proactively trigger robot actions. Considering for example the scenario `capital-goods` these capabilities allow the developed controller to properly interpret perception outcome and contextualize human activities (i.e., bolt positioning) with respect to the known production procedures and autonomously trigger suitable task planning goals. These events are indeed automatically translated into robot tasks necessary to proactively support the human worker in the collaborative task of screwing bolts on the rotary table.

More in general the developed approach supports a modular description of robot capabilities and production requirements that can be easily extended and refined over time. For example, the knowledge base can be enriched with additional robot capabilities, human skills, production goals, and procedures. This new knowledge would automatically be contextualized with respect to production requirements and thus integrated into the reasoning and task-planning processes. From the robot perspective, the ontology-based approach supports modular programming allowing roboticists to focus on the definition of new capabilities/skills (i.e., `Function`) without considering more general production aspects or defining complex (and static) behaviors. The synthesis of such behaviors would be indeed the responsibility of the integrated task planner that would automatically evaluate new skills/capabilities and dynamically synthesize (optimized) collaborative plans.

6. Conclusions and future works

SOHO (*Sharework Ontology for Human-Robot Collaboration*) is a novel domain ontology for Human-Robot Collaboration defined within the Sharework H2020 research project. It formally characterizes HRC manufacturing

scenarios by considering different perspectives. Indeed, its main original feature relies on the use of a context-based approach to ontology design, supporting the flexible representation of collaborative production processes.

This paper proposes an extension of SOHO by defining *ontology design patterns* that formally characterize collaboration dynamics that are typical in many Human-Robot Collaboration manufacturing scenarios. Furthermore, we have defined a general knowledge extraction procedure that relies on the *semantics* proposed by SOHO to analyze *production knowledge* and automatically synthesize timeline-based plan-based controllers that are suitable to effectively coordinate human and robot behaviors [35,69]. An experimental evaluation of the developed representation and reasoning technology shows the efficacy of properly capturing the complexity of real industrial collaborative scenarios and the capability of automatically "compiling" such knowledge into suitable planning domains.

Future research directions will focus on further extensions of SOHO to better characterize the *human factors* and support the representation of preferences, expertise levels, and physical and cognitive conditions of human workers that are crucial to serving advanced *personalization* and finer *adaptation* features in collaborative processes. In addition, we plan to integrate developed ontology-based representation and reasoning into a *knowledge engineering tools* to facilitate domain experts in the design of collaborative process as well as in the deployment of AI-based task planning technologies for the coordination of collaborative cells [38].

Acknowledgements

Authors are partially supported by the European Commission within the Sharework project (H2020 Factories of the Future GA No. 820807). Authors are grateful to EURECAT, STIIMA-CNR, STAM, MCM, CEMBRE, Goizper and ALSTOM, partners in Sharework, and sharing their knowledge and experience about the productive scenarios that inspired our work. Authors are also partially supported by the EU project TAILOR "Foundations of Trustworthy AI - Integrating Learning, Optimisation and Reasoning" G.A. 952215.

References

[1] R. Alami, A. Clodic, V. Montreuil, E.A. Sisbot and R. Chatila, Toward human-aware robot task planning, in: *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006, pp. 39–46.

[2] J.F. Allen, Maintaining knowledge about temporal intervals, *Commun. ACM* **26**(11) (1983), 832–843. doi:10.1145/182.358434.

[3] G. Antoniou and F. van Harmelen, Web ontology language: OWL, in: *Handbook on Ontologies*, S. Staab and R. Studer, eds, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 67–92. ISBN 978-3-540-24750-0. doi:10.1007/978-3-540-24750-0_4.

[4] I. Awaad, G.K. Kraetzschmar and J. Hertzberg, The role of functional affordances in socializing robots, *International Journal of Social Robotics* **7**(4) (2015), 421–438. doi:10.1007/s12369-015-0281-3.

[5] S. Balakirsky, Ontology based action planning and verification for agile manufacturing, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 21–28, Special Issue on Knowledge Driven Robotics and Manufacturing. doi:10.1016/j.rcim.2014.08.011.

[6] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A.K. Bozcuoğlu and G. Bartels, Know Rob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 512–519. doi:10.1109/ICRA.2018.8460964.

[7] D. Beßler, M. Pomarlan and M. Beetz, OWL-enabled assembly planning for robotic agents, in: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'18, International Foundation for Autonomous Agents and Multiagent Systems*, Richland, SC, 2018, pp. 1684–1692.

[8] D. Beßler, R. Porzel, M. Pomarlan, M. Beetz, R. Malaka and J. Bateman, A formal model of affordances for flexible robotic task execution, in: *ECAI 2020*, IOS Press, 2020, pp. 2425–2432.

[9] C. Bicchieri, *The Grammar of Society: The Nature and Dynamics of Social Norms*, Cambridge University Press, 2005.

[10] A. Bit-Monnot, M. Ghallab, F. Ingrand and D.E. Smith, FAPE: A constraint-based planner for generative and hierarchical temporal planning, *CoRR* (2020), arXiv:2010.13121.

[11] S. Borgo, An ontological view of components and interactions in behaviorally adaptive systems, *Journal of Integrated Design and Process Science* **23**(1) (2019), 17–35. doi:10.3233/JID190013.

[12] S. Borgo, M. Carrara, P. Garbacz and P.E. Vermaas, A formal ontological perspective on the behaviors and functions of technical artifacts, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **23**(1) (2009), 3–21. doi:10.1017/S0890060409000079.

[13] S. Borgo, A. Cesta, A. Orlandini and A. Umbrico, Knowledge-based adaptive agents for manufacturing domains, *Engineering with Computers* **35**(3) (2019), 755–779. doi:10.1007/s00366-018-0630-6.

[14] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E.M. Sanfilippo and L. Vieu, DOLCE: A descriptive ontology for linguistic and cognitive engineering, *Applied Ontology* **17** (2022), 45–69. doi:10.3233/AO-210259.

[15] B. Bruno, C.T. Recchiuto, I. Papadopoulos, A. Saffiotti, C. Koulouglioti, R. Menicatti, F. Mastrogiovanni, R. Zaccaria and A. Sgorbissa, Knowledge representation for culturally competent personal robots: Requirements, design principles, implementation, and assessment, *International Journal of Social Robotics* **11**(3) (2019), 515–538. doi:10.1007/s12369-019-00519-w.

[16] D. Calisi, L. Iocchi, D. Nardi, C.M. Scalzo and V.A. Ziparo, Context-based design of robotic systems, *Robotics and Autonomous Systems* **56**(11) (2008), 992–1003, Semantic Knowledge in Robotics. doi:10.1016/j.robot.2008.08.008.

[17] N. Carvalho, O. Chaim, E. Cazarini and M. Gerolamo, Manufacturing in the fourth industrial revolution: A positive prospect in sustainable manufacturing, *Procedia Manufacturing* **21** (2018), 671–678, 15th Global Conference on Sustainable Manufacturing. doi:10.1016/j.promfg.2018.02.170.

[18] C. Castelfranchi, Commitments: From individual intentions to groups and organizations, in: *Proceedings of the First International Conference on Multiagent Systems*, 1995, pp. 41–48.

[19] A. Cesta, A. Finzi, S. Fratini, A. Orlandini and E. Tronci, Analyzing flexible timeline plan, in: *ECAI 2010*, Proceedings of the 19th European Conference on Artificial Intelligence, Vol. 215, IOS Press, 2010.

[20] A. Cesta, A. Oddi and S.F. Smith, A constraint-based method for project scheduling with time windows, *Journal of Heuristics* **8**(1) (2002), 109–136. doi:10.1023/A:1013617802515.

[21] S.K. Chandrasegaran, K. Ramani, R.D. Sriram, I. Horváth, A. Bernard, R.F. Harik and W. Gao, The evolution, challenges, and future of knowledge representation in product design systems, *Computer-Aided Design* **45**(2) (2013), 204–228, Solid and Physical Modeling 2012.

[22] D.S. Chang, G.H. Cho and Y.S. Choi, Ontology-based knowledge model for human-robot interactive services, in: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, Association for Computing Machinery, New York, NY, USA, 2020, pp. 2029–2038. ISBN 9781450368667.

[23] M. Cialdea Mayer and A. Orlandini, An executable semantics of flexible plans in terms of timed game automata, in: *The 22nd International Symposium on Temporal Representation and Reasoning (TIME)*, IEEE, 2015.

[24] M. Cialdea Mayer, A. Orlandini and A. Umbrico, Planning and execution with flexible timelines: A formal account, *Acta Informatica* **53**(6–8) (2016), 649–680. doi:10.1007/s00236-015-0252-z.

[25] A. Clodic, R. Alami and R. Chatila, Key elements for human-robot joint action, in: *Sociable Robots and the Future of Social Relations*, 2014.

[26] A.J. Coles, A. Coles, M. Fox and D. Long, Forward-chaining partial-order planning, in: *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010*, Toronto, Ontario, Canada, May 12–16, 2010, 2010, pp. 42–49.

[27] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W.D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth and K. Taylor, in: *The SSN Ontology of the W3C Semantic Sensor Network Incubator Group 17(C)*, 2012, pp. 25–32.

[28] K. Dautenhahn, M. Walters, S. Woods, K.L. Koay, C.L. Nehaniv, A. Sisbot, R. Alami and T. Siméon, How may I serve you? A robot companion approaching a seated person in a helping context, in: *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, HRI'06*, Association for Computing Machinery, New York, NY, USA, 2006, pp. 172–179. ISBN 1595932941. doi:10.1145/1121241.1121272.

[29] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, The semantic web: The roles of XML and RDF, *IEEE Internet Computing* **4**(5) (2000), 63–73. doi:10.1109/4236.877487.

[30] M. Diab, A. Akbari, M. Ud Din and J. Rosell, PMK – a knowledge processing framework for autonomous robotics perception and manipulation, *Sensors* **19**(5) (2019).

[31] M. Diab, M. Pomarlan, D. Beßler, A. Akbari, J. Rosell, J. Bateman and M. Beetz, SkillMaN – a skill-based robotic manipulation framework based on perception and reasoning, *Robotics and Autonomous Systems* **134** (2020), 103653. doi:10.1016/j.robot.2020.103653.

[32] B. Dworschak and H. Zaiser, Competences for cyber-physical systems in manufacturing – first findings and scenarios, *Procedia CIRP* **25** (2014), 345–350.

[33] K. Erol, J. Hendler and D.S. Nau, HTN planning: Complexity and expressivity, in: *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence, AAAI'94*, AAAI Press, 1994, pp. 1123–1128.

[34] P. Fantini, M. Pinzone and M. Taisch, Placing the operator at the centre of industry 4.0 design: Modelling and assessing human activities within cyber-physical systems, *Computers & Industrial Engineering* **139** (2020), 105058. doi:10.1016/j.cie.2018.01.025.

[35] M. Faroni, M. Beschi, S. Ghidini, N. Pedrocchi, A. Umbrico, A. Orlandini and A. Cesta, A layered control approach to human-aware task and motion planning for Human-Robot Collaboration, in: *IEEE Int. Conf. on Robot and Human Inter. Comm.*, Naples (Italy), 2020.

[36] R.E. Fikes and N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial intelligence* **2**(3–4) (1971), 189–208. doi:10.1016/0004-3702(71)90010-5.

[37] M. Flatscher and A. Riel, Stakeholder integration for the successful product–process co-design for next-generation manufacturing technologies, *CIRP Annals* **65**(1) (2016), 181–184. doi:10.1016/j.cirp.2016.04.055.

[38] E. Foderaro, A. Cesta, A. Umbrico and A. Orlandini, Simplifying the A.I. planning modeling for Human-Robot Collaboration, in: *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, 2021, pp. 1011–1016.

[39] M. Fox and D. Long, PDDL2.1: An extension to PDDL for expressing temporal planning domains, *J. Artif. Intell. Res. (JAIR)* **20** (2003), 61–124. doi:10.1613/jair.1129.

[40] A. Gangemi, Ontology design patterns for semantic web content, in: *The Semantic Web – ISWC 2005*, Y. Gil, E. Motta, V.R. Benjamins and M.A. Musen, eds, Springer, Berlin Heidelberg, 2005, pp. 262–276. ISBN 978-3-540-32082-1. doi:10.1007/11574620_21.

[41] M. Ghallab, D. Nau and P. Traverso, The actor's view of automated planning and acting: A position paper, *Artificial Intelligence* **208** (2014), 1–17. doi:10.1016/j.artint.2013.11.002.

[42] P.J.S. Gonçalves and P.M.B. Torres, Knowledge representation applied to robotic orthopedic surgery, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 90–99, Special Issue on Knowledge Driven Robotics and Manufacturing. doi:10.1016/j.rcim.2014.08.014.
[43] N. Guarino, *Formal Ontology in Information Systems*, IOS Press, 1998.
[44] E. Helms, R.D. Schraft and M. Hagele, rob@work: Robot assistant in industrial environments, in: *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002, pp. 399–404. doi:10.1109/ROMAN.2002.1045655.
[45] J. Hoffmann, FF: The fast-forward planning system, *AI Magazine* **22**(3) (2001), 57–62.
[46] A. Hristoskova, C.E. Agüero, M. Veloso and F.D. Turck, Heterogeneous context-aware robots providing a personalized building tour, *International Journal of Advanced Robotic Systems* **10**(1) (2013), 14. doi:10.5772/54797.
[47] H.-M. Huang, K. Pavek, B. Novak, J.S. Albus and E.R. Messina, A framework for autonomy levels for unmanned systems (ALFUS), 2005, in: Association of the Unmanned Vehicle System International (AUVSI).
[48] F. Ingrand and M. Ghallab, Deliberation for autonomous robots: A survey, *Artificial Intelligence* **247** (2017), 10–44, Special Issue on AI and Robotics. doi:10.1016/j.artint.2014.11.003.
[49] L. Jansen and S. Schulz, The ten commandments of ontological engineering, in: *Proceedings of the 3rd Workshop of Ontologies in Biomedicine and Life Sciences*, 2011, pp. 1–6.
[50] A.K. Jonsson, P.H. Morris, N. Muscettola, K. Rajan and B. Smith, Planning in interplanetary space: Theory and practice, in: *AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling*, 2000.
[51] Z. Kootbally, T.R. Kramer, C. Schlenoff and S.K. Gupta, Implementation of an ontology-based approach to enable agility in kit building applications, *International Journal of Semantic Computing* **12**(01) (2018), 5–24. doi:10.1142/S1793351X18400019.
[52] S. Lemaignan, R. Ros, L. Mosenlechner, R. Alami and M. Beetz, ORO, a knowledge management platform for cognitive architectures in robotics, in: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 3548–3553, ISSN 2153-0858.
[53] S. Lemaignan, M. Warnier, E.A. Sisbot, A. Clodic and R. Alami, Artificial cognition for social human-robot interaction: An implementation, *Artificial Intelligence* **247** (2017), 45–69, Special Issue on AI and Robotics. doi:10.1016/j.artint.2016.07.002.
[54] S. Makris, *Cooperating Robots for Flexible Manufacturing*, Vol. 1, Springer International Publishing, 2021.
[55] N. Mandischer, T. Huhn, M. Hüsing and B. Corves, Efficient and consumer-centered item detection and classification with a multicamera network at high ranges, *Sensors* **21**(14) (2021).
[56] S. Manzoor, Y.G. Rocha, S.-H. Joo, S.-H. Bae, E.-J. Kim, K.-J. Joo and T.-Y. Kuc, Ontology-based knowledge representation in robotic systems: A survey oriented toward applications, *Applied Sciences* **11**(10) (2021).
[57] J.A. Marvel, J. Falco and I. Marstio, Characterizing task-based Human-Robot Collaboration safety in manufacturing, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **45**(2) (2015), 260–275. doi:10.1109/TSMC.2014.2337275.
[58] V. Mascardi, V. Cordì and P. Rosso, A comparison of upper ontologies, in: *Workshop from Objects to Agents (WOA)*, 2007, pp. 1–10.
[59] E. Matheson, R. Minto, E.G.G. Zampieri, M. Faccio and G. Rosati, Human–robot collaboration in manufacturing applications: A review, *Robotics* **8**(4) (2019). doi:10.3390/robotics8040100.
[60] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld and D. Wilkins, PDDL – The planning domain definition language, Technical Report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.
[61] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn and K. Ueda, Cyber-physical systems in manufacturing, *CIRP Annals* **65**(2) (2016), 621–641. doi:10.1016/j.cirp.2016.06.005.
[62] P.H. Morris, N. Muscettola and T. Vidal, Dynamic control of plans with temporal uncertainty, in: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, pp. 494–502.
[63] N. Muscettola, *HSTS: Integrating Planning and Scheduling, in: Intelligent Scheduling*, Morgan Kauffmann, 1994.
[64] D. Nau, T.C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu and F. Yaman, SHOP2: An HTN planning system, *Journal on Artificial Intelligence Research* **20** (2003). doi:10.1613/jair.1141.
[65] I. Niles and A. Pease, Towards a standard upper ontology, in: *Proceedings of the International Conference on Formal Ontology in Information Systems*, Vol. 2001, Association for Computing Machinery, 2001, pp. 2–9. ISBN 1581133774.
[66] A. Olivares-Alarcos, S. Foix, S. Borgo and G. Alenyà, OCRA – an ontology for collaborative robotics and adaptation, *Computers in Industry* **138** (2022), 103627. doi:10.1016/j.compind.2022.103627.
[67] J.N. Otte, J. Beverley and A. Ruttenberg, BFO: Basic formal ontology, *Applied Ontology* **17** (2022), 17–43. doi:10.3233/AO-220262.
[68] R. Patel, M. Hedelind and P. Lozan-Villegas, Enabling robots in small-part assembly lines: The "ROSETTA approach" – an industrial perspective, in: *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1–5.
[69] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico and T. Tollio, Motion planning nad scheduling for human and industrial-robot collaboration, *CIRP Annals – Manufacturing Technology* (2017), to appear.
[70] J. Pérez, M. Arenas and C. Gutierrez, Semantics and complexity of SPARQL, *ACM Trans. Database Syst.* **34**(3) (2009).
[71] E. Prestes, J.L. Carbonera, S.R. Fiorini, V.A.M. Jorge, M. Abel, R. Madhavan, A. Locoro, P. Goncalves, M.E. Barreto, M. Habib, A. Chibani, S. Gérard, Y. Amirat and C. Schlenoff, Towards a core ontology for robotics and automation, *Robotics and Autonomous Systems* **61**(11) (2013), 1193–1204. doi:10.1016/j.robot.2013.04.005.
[72] F. Py, K. Rajan and C. McGann, A systematic agent framework for situated autonomous systems, in: *AAMAS-10*, Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems, 2010.
[73] K. Rajan and A. Saffiotti, Towards a science of integrated AI and robotics, *Artificial Intelligence* **247** (2017), 1–9. doi:10.1016/j.artint.2017.03.003.
[74] P. Ribino, M. Bonomolo, C. Lodato and G. Vitale, A humanoid social robot based approach for indoor environment quality monitoring and well-being improvement, *International Journal of Social Robotics* **13**(2) (2021), 277–296. doi:10.1007/s12369-020-00638-9.

[75] S. Rossi, F. Ferland and A. Tapus, User profiling and behavioral adaptation for HRI: A survey, *Pattern Recognition Letters* **99** (2017), 3–12. doi:10.1016/j.patrec.2017.06.002.
[76] F. Rovida, M. Crosby, D. Holz, A.S. Polydoros, B. Großmann, R.P.A. Petrick and V. Krüger, SkiROS – a skill-based robot control platform on top of ROS, in: *Robot Operating System (ROS): The Complete Reference (Volume 2)*, A. Koubaa, ed., Springer International Publishing, Cham, 2017, pp. 121–160. ISBN 978-3-319-54927-9. doi:10.1007/978-3-319-54927-9_4.
[77] A.R. Sadik and B. Urban, An ontology-based approach to enable knowledge representation and reasoning in worker–Cobot Agile manufacturing, *Future Internet* **9**(4) (2017). doi:10.3390/fi9040090.
[78] E.M. Sanfilippo, Y. Kitamura and R.I.M. Young, Formal ontologies in manufacturing, *Applied Ontology* **14** (2019), 119–125. doi:10.3233/AO-190209.
[79] E.A. Sisbot, A. Clodic, L.F. Marin-Urias, M. Fontmarty, L. Brethes and R. Alami, Implementing a human-aware robot system, in: *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006, pp. 727–732.
[80] L. Solano, P. Rosado and F. Romero, Knowledge representation for product and processes development planning in collaborative environments, *International Journal of Computer Integrated Manufacturing* **27**(8) (2014), 787–801. doi:10.1080/0951192X.2013.834480.
[81] M. Stenmark and J. Malec, Knowledge-based instruction of manipulation tasks for industrial robotics, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 56–67, Special Issue on Knowledge Driven Robotics and Manufacturing. doi:10.1016/j.rcim.2014.07.004.
[82] I.H. Suh, G.H. Lim, W. Hwang, H. Suh, J.-H. Choi and Y.-T. Park, Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence, in: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 429–436.
[83] J.T.C. Tan, F. Duan, Y. Zhang and T. Arai, Task decomposition of cell production assembly operation for man-machine collaboration by HTA, in: *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 1066–1071. doi:10.1109/ICAL.2008.4636309.
[84] M. Tenorth and M. Beetz, Representations for robot knowledge in the KnowRob framework, *Artificial Intelligence* **247** (2017), 151–169, Special Issue on AI and Robotics. doi:10.1016/j.artint.2015.05.010.
[85] W. Terkaj, T. Tolio and A. Valente, A review on manufacturing flexibility, in: *Design of Flexible Production Systems*, Springer, 2009, pp. 41–61. doi:10.1007/978-3-540-85414-2_3.
[86] W. Terkaj and M. Urgo, Ontology-based modeling of production systems for design and performance evaluation, in: *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 748–753. doi:10.1109/INDIN.2014.6945606.
[87] P. Turaga, R. Chellappa, V.S. Subrahmanian and O. Udrea, Machine recognition of human activities: A survey, *IEEE Transactions on Circuits and Systems for Video Technology* **18**(11) (2008), 1473–1488. doi:10.1109/TCSVT.2008.2005594.
[88] A. Umbrico, M. Anasagasti, S.-O. Bezrucav, F. Canale, A. Cesta, B. Corves, N. Mandischer, M. Mondragon, C.N. Rappis and A. Orlandini, Enhanced cognition for adaptive Human-Robot Collaboration, in: *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–7.
[89] A. Umbrico, A. Cesta, M. Cialdea Mayer and A. Orlandini, PLATINUm: A new framework for planning and acting, in: *AI*IA 2017 Advances in Artificial Intelligence*, 2017, pp. 498–512. ISBN 978-3-319-70169-1. doi:10.1007/978-3-319-70169-1_37.
[90] A. Umbrico, A. Cesta, G. Cortellessa and A. Orlandini, A holistic approach to behavior adaptation for socially assistive robots, *International Journal of Social Robotics* **12**(3) (2020), 617–637. doi:10.1007/s12369-019-00617-9.
[91] A. Umbrico, A. Orlandini and A. Cesta, An ontology for Human-Robot Collaboration, *Procedia CIRP* **93** (2020), 1097–1102. doi:10.1016/j.procir.2020.04.045.
[92] A. Umbrico, A. Orlandini, A. Cesta, M. Faroni, M. Beschi, N. Pedrocchi, A. Scala, P. Tavormina, S. Koukas, A. Zalonis, N. Fourtakas, P.S. Kotsaris, D. Andronas and S. Makris, Design of advanced human-robot collaborative cells for personalized human-robot collaborations, *Applied Sciences* **12**(14) (2022).
[93] A. Umbrico, A. Orlandini, A. Cesta, S. Koukas, A. Zalonis, N. Fourtakas, D. Andronas, G. Apostolopoulos and S. Makris, Towards user-awareness in Human-Robot Collaboration for future cyber-physical systems, in: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.
[94] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case and J.A. Harding, Towards a formal manufacturing reference ontology, *International Journal of Production Research* **51**(22) (2013), 6553–6572. doi:10.1080/00207543.2013.801570.
[95] T. Vidal, A unified dynamic approach for dealing with temporal uncertainty and conditional planning, in: *AIPS-00*, Proc. of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling, 2000, pp. 395–402.
[96] T. Vidal and H. Fargier, Handling contingency in temporal constraint networks: From consistency to controllabilities, *JETAI* **11**(1) (1999).
[97] L. Wang, M. Törngren and M. Onori, Current status and advancement of cyber-physical systems in manufacturing, *Journal of Manufacturing Systems* **37** (2015), 517–527. doi:10.1016/j.jmsy.2015.04.008.
[98] H.-P. Wiendahl, H.A. ElMaraghy, P. Nyhuis, M.F. Zäh, H.-H. Wiendahl, N. Duffie and M. Brieke, Changeable manufacturing – classification, design and operation, *CIRP Annals* **56**(2) (2007), 783–809. doi:10.1016/j.cirp.2007.10.003.