Deploying Ontology-based Reasoning in Collaborative Manufacturing

Alessandro Umbrico *, Amedeo Cesta and Andrea Orlandini
Institute of Cognitive Sciences and Technlogies, National Research Council, Rome, Italy
E-mail: alessandro.umbrico@istc.cnr.it

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 a 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, research activities are still ongoing to enable robots to autonomously operate in “open environments" for, e.g., understanding the actual situation, planning their tasks and acting to safely and effectively achieve some given goals. In particular in manufacturing, an open problem is the design of control systems that can robustly deal with quick and frequent changes of the production requirements. Higher levels of flexibility and adaptability of industrial robots is indeed crucial to face the challenges of Industry 4.0 [1, 2]. Modern manufacturing systems should evolve towards customer-oriented production as well as towards different production paradigms that see humans and robots working side by side as interchangeable production resources [3, 4].

Classical control processes that usually rely on static models of robot capabilities and production dynamics are today obsolete since they do not provide robotic systems (and more in general manufacturing systems) with the flexibility needed to effectively support dynamic production environments. This is especially true in Human-Robot Collaboration (HRC) where robots and humans share the working space and tightly interact together to achieve common (production) objectives. The symbiotic coexistence of human operators and robots raises several technological challenges because the behavior of human workers is neither predictable nor controllable. To realize safe, effective and efficient cooperation between humans and robots is therefore necessary to robustly deal with a significant

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

amount of *uncertainty* requiring higher levels of flexibility and adaptability of robot controllers [5]. HRC scenarios entail thus the integration of Artificial Intelligence (AI) and Robotics solutions to enrich collaborative robots with advanced *cognitive capabilities* [6, 7] and thus allow robots (and collaborative systems as a whole) to: (i) *perceive* the environment, correctly *interpret* occurring events and situations to properly build and maintain *knowledge* about the production context; (ii) reason about their own *capabilities*/*skills* and dynamically contextualize possible actions according to the *known* state of a production scenario and; (iii) autonomously decide how to act and interact with the environment and other "actors" (i.e., human operators but also other robots if necessary) in order to carry out (assigned) tasks and dynamically support production needs.

In this context, we are investigating the enrichment of collaborative robot controllers through a "perceive-reason-act" paradigm implementing advanced cognitive features. Such features would allow a robot to perceive the production environment, recognize events and activities, and dynamically adapt its behavior accordingly. In particular, the integration of AI-based Knowledge Representation & Reasoning technologies with Automated Planning and Execution technologies has shown to be effective in enhancing the flexibility and adaptability of *autonomous behaviors* of robots in a number of (heterogeneous) scenarios ranging from service and assistive robotics [8–10] to Reconfigurable Manufacturing Systems [11, 12]. Semantic technologies are crucial to realize the cognitive capabilities needed to provide robot controllers with the *semantics* 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 (autonomously) *understand* the state of a production environment and make *contextualized decisions*.

This paper advances a recent work extending an ontological model for Human-Robot Collaboration in manufacturing [13] and investigating the integration between Knowledge Representation & Reasoning and Automated Planning to enhance *awareness*, *adaptability* and *flexibility* of collaborative robots. On the one hand, the paper refines the context-based ontological model given in [13] to better characterize collaborative dynamics and contextualize them with respect to the production needs of a HRC scenario. On the other hand, the paper proposes *knowledge reasoning mechanisms* that contextualize *production knowledge* and synthesize plan-based control models suitable for the flexible and robust coordination of collaborative robots [14, 15]. More specifically, the paper first contribution is a refined *domain ontology* characterizing HRC scenarios from different synergetic perspectives (defined as *contexts*). The ontology defines the key concepts and properties necessary to describe and characterize human and robot capabilities, production objectives, production procedures, constraints and also collaboration modalities.

An original and novel aspect of the proposed approach is the use of *ontology design patterns* [16] as a mean to facilitate the description of HRC production processes by taking into account consolidated schema of interaction between humans and robots [17]. Similarly to software design patterns, ontology design patterns are here proposed to narrow user design choices and define sufficiently general and reusable concepts characterizing typical *collaboration dynamics*. Such patterns are thus suitable to characterize behavioral constraints of humans and robots, according to the (expected) types of collaboration required by tasks. Developed knowledge reasoning and knowledge extraction procedures automatically generate and validate planning models. They also show how ontological patterns are translated into behavioral constraints that generally characterize collaboration dynamics of humans and robots within the execution of production tasks. Such procedures are crucial to enable the dynamic synthesis and adaptation of plan-based controllers. The integration of planning and semantic technologies thus realize the *cognitive skills* necessary to support *production awarenss* and *reconfigurable capabilities*, necessary to guarantee higher levels of reliability and autonomy under evolving conditions and states of a production environment (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 real HRC production scenarios extracted from an EU H2020 research project, called Sharework[1]. These scenarios concern different types of production environments with different production entities, tools and objectives. The assessment shows that the proposed approach supports a proper definition of suitable *production knowledge* and the synthesis of valid plan-based controllers that can be concretely used to coordinate human and robot behaviors.

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

2. Ontology in Computer Science and Robotics

Ontologies can be seen as formal descriptions of objects, properties and relationships among objects collected in a particular data structure called Knowledge Base (KB). In computer science, ontologies have been defined in different ways by different scientists. Studer et al. [18] combined the definitions by Gruber [19] and Borst [20] stating that an ontology is "an explicit, formal specification of a shared conceptualization". The different characterizations of ontology are complementary and can be combined together. According to [21], it can be said that "an ontology is an artificial representation, that represents types or universals of a certain domain and the relations that hold according to a certain theory in a formal structure". Depending on the specific application needs, four types of ontology can be defined, supporting different levels of generality of underlying concepts and properties [22]: (i) *Top-level or Upper ontologies* describe very general concepts like e.g., space, time, event or action, that are independent from a particular problem or domain; (ii) *Domain ontologies* describe general concepts related to a specific domain; (iii) *Task ontologies* describe generic tasks or activities. (iv) *Application ontologies* characterize a specific application and describe concepts whose relevance is limited to a specific domain and task.

2.1. Foundations

Upper ontologies aim at describing reality from a quite general perspective in order to define very general concepts that are the same across all domains. The concepts and the properties defined by upper ontologies may seem too abstract and not really useful in concrete applications but the use of this kind of ontologies (also known as foundational ontologies) is generally recommended [21, 23]. The use of upper ontologies indeed represents a good design choice to build new domain ontologies. As shown in [21], these concepts represent a stable theoretical foundation fostering a clear structuring and disambiguation of new concepts and related relationships. According to [23], upper ontologies guide a correct classification of knowledge entities of a particular domain and facilitate interoperability among different ontologies.

A number of upper ontologies exist in the literature. SUMO, DOLCE and BFO are probably the most famous and used. Each upper ontology has its own basic assumptions that characterize general and abstract concepts. For example, DOLCE is an "ontology of particulars". It does have universal (classes and properties), but the claim is that they are only employed in the service of describing particulars. In contrast, SUMO could be described as an ontology of both particulars and universals. Also, DOLCE uses meta-properties as a guiding methodology, while SUMO pursues a formal definition of such meta-properties directly in the ontology itself (axiomatization). The work [23] gives a first comparison of these and other upper ontologies known in the literature.

Although similar, these ontologies cannot be directly integrated without introducing contradictions. Some works have focused on the definition of the so called upper-upper ontologies for the integration of different foundational ontologies [24]. Since CORA and SSN rely on two different upper ontologies and since upper-upper ontologies like e.g., COSMO (COmmon Semantic MOdel) have obtained poor practical results, a first design choice was the selection of an upper ontology for SOHO. Given the variety of information the system should deal with and the needed representation flexibility, we decided to use DOLCE [25] as theoretical background in order to support a flexible interpretation of temporally evolving entities and, also, to rely on a recognized standard representation framework (ISO 213838-3).

2.2. What is Missing for HRC?

In addition to DOLCE, SOHO is built on top of other two ontologies: (i) the CORA ontology [26] and; (ii) the SSN ontology [27]. CORA is an IEEE standard ontology for robotics and automation. It has been defined with the aim of promoting a common language in the robotics and automation domain. It proposes a semantics to formally characterize knowledge about robots and robot parts, robot positions and configurations and groups of robots. This standard relies on SUMO as theoretical foundation and integrates the framework ALFUS [28] to characterize the autonomy levels of a robot. SSN is a W3C standard ontology for IoT devices and sensor network. It defines basic concepts and properties characterizing the capabilities of sensing devices, their deployment into a physical environment and the physical phenomena such devices can observe. SSN relies on DUL (a subset of

DOLCE) as theoretical background. It extends abstract concepts like e.g., `DUL:Quality` and `DUL:Region` to represent respectively physical properties that can be observed and metrics that can be used to measure the outcome of sensing processes.

CORA and SSN are well structured ontologies defining concepts and properties that are relevant for HRC but they do not cover all the necessary information. 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 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 the global production objectives and processes. For example, CORA does not consider the Human as an autonomous agent operating in autonomy or in collaboration with robots to achieve a common (production) objective. Also, CORA does not support a structured description of production processes in terms of tasks, relationships among tasks, operational constraints and needed capabilities/skills for their execution. More specifically, three main limitations can be pointed out.

- *Functions, Tasks and Capabilities*. A detailed description of production processes and tasks that *agents* (e.g., human workers or robots) can perform is necessary to dynamically coordinate the available resources. Such a structured description is crucial to realize a flexible collaboration between humans and robots.
- *Humans as collaborative agents*. A detailed description of human operators in terms of capabilities and their "autonomy level" is crucial to dynamically adapt and coordinate collaborative processes. In this regard, it would be interesting to extend the ALFUS [28] model to human workers. This knowledge together with a model of possible collaboration modalities of tasks is necessary to reason about safety requirements and synthesize collaborative plans accordingly.
- *Intentions, commitment and coordination issues*. The envisaged system should be capable of recognizing human behaviors from sensor observations and contextualize them with respect to production objectives. It is necessary to represent and reason about abstract concepts like e.g., *human intentions* and link observed behaviors to (known) production processes in order to react or adapt planned operations accordingly.

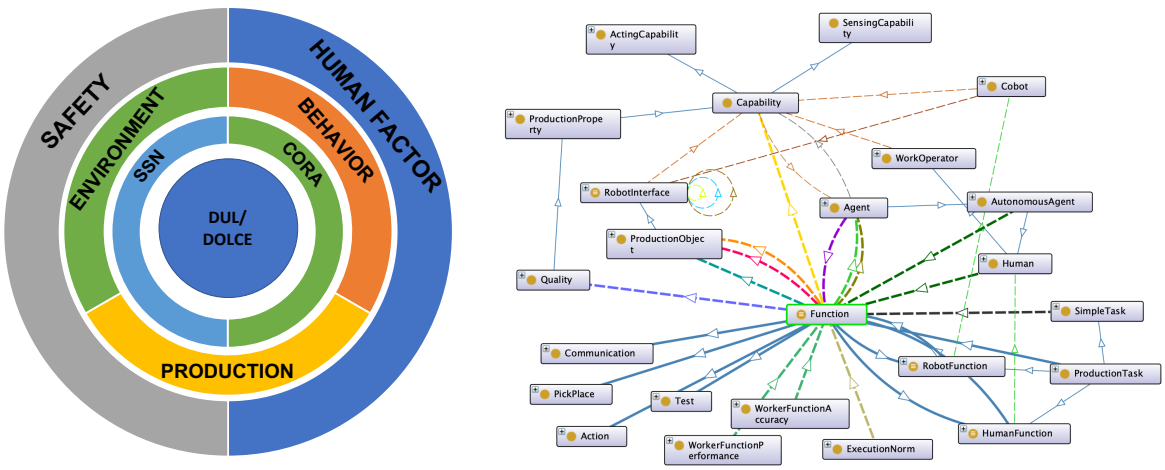
3. A Domain Ontology for HRC

In manufacturing, ontologies have mainly focused on (cyber-physical) production systems as a whole or rather on specific production aspects like e.g., [29–31]. Such works have not taken into account *collaboration dynamics* and symbiotic interaction for the achievement of shared (production) goals. In this paper, we propose a refinement of SOHO (*Sharework Ontology for Human Robot Collaboration*) to pursue a stricter focus on *collaboration dynamics*. An overview of the structure of SOHO and a brief description of main concepts and properties has been given in [13]. Figure 1 shows the general organization of SOHO which follows a context-based approach to characterize domain entities from different perspectives and abstraction levels. This flexibility is crucial to support a *multi-perspective* representation and *interpretation* of domain (production) knowledge.

SOHO considers three main contexts, i.e., the *environment*, the *behavior* and the *production contexts*. Broadly speaking, the *environment context* characterizes the physical configuration of a working environment and the "qualitative" properties of the elements that are part of it. The *behavior context* characterizes the "acting elements" of the environment (i.e., the worker and the robot) in terms of operational capabilities, performances and behavioral qualities. The *production context* characterizes goals, procedures and functional operations that should be performed and how "acting elements" could interact to correctly carry out them.

3.1. Capabilities, Functions and Production Requirements

The goal of SOHO is to define humans and robots capabilities as well as operations they should perform in a production environment. These "acting entities" are represented through the concepts `Cobot` and `HumanWorker` that are defined as specializations of the concept `DUL:Agent`. The acting qualities of each agent are represented



(a) (b)

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

by means of `Capability` and `Function`. Capabilities characterize the operations an agent can "intrinsically" carry out according to their structure and skills. For example, a human worker can perform welding operations only if she is skilled in that task. Similarly, a robot can perform "pick and place" of objects only if it is endowed with a gripper or other suitable functional parts.

While capabilities do not depend on the features of a production context, the concept of `Function` integrates the Taxonomy of Functions defined in [32] to characterize low-level production tasks that a human and a robot should perform in a manufacturing environment. In this taxonomy, different types of `Function` are defined according to the *effects* they have on the `DUL:Quality` of objects. The functions an (either human or robotic) agent can perform in a context can be dynamically inferred according to the actual capabilities of that agent. The separation between functions and capabilities supports contextual reasoning since functions contextualize general agents' capabilities with respect to the needs and features of a production scenario.

$$\begin{aligned} \texttt{Function} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\, \texttt{isDescribedBy.ProductionNorm} \sqcap \\ & \exists\, \texttt{canBePerformedBy.DUL:Agent} \sqcap \\ & \exists\, \texttt{hasEffectOn.DUL:Quality} \sqcap \\ & \exists\, \texttt{hasTarget.ProductionObject} \sqcap \\ & \exists\, \texttt{requires.ProductionObject} \sqcap \\ & \exists\, \texttt{requires.Capability} \end{aligned} \tag{1}$$

The description of a production process follows a task-oriented approach [33, 34]. The top-level element is the *goal* which defines the general objectives of a production context. Each `ProductionGoal` is associated with a number of `ProductionMethod` (at least one method for each goal is necessary) defining the rules that must be considered to successfully achieve the desired production goal. Each `ProductionMethod` always refers to one specific `ProductionGoal` and is composed by a hierarchical organization of `ProductionTask`. The ontology specifically defines three types of task: (i) `ComplexTask` (either *disjunctive* or *conjunctive*); (ii) `SimpleTask` and; (iii) `Function`. A `ComplexTask` is a `ProductionTask` (i.e., an instance of `DUL:Method`) representing

a compound logical operation. The hierarchical structure is enforced by the property `hasConstituent` which associates `ComplexTask` with either `SimpleTask` or other `ComplexTask`.

$$\begin{aligned} \texttt{ComplexTask} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\,(\texttt{hasConstituent.ComplexTask} \sqcup \\ & \quad \texttt{hasConstituent.SimpleTask}) \sqcap \\ & \exists\,\texttt{isConformTo.OperativeConstraint} \end{aligned} \tag{2}$$

A `SimpleTask` represents a *leaf* of the hierarchical structure of a `ProductionMethod`. This concept describes primitive production operations that could be carried out leveraging the functional capabilities of the agents. A `SimpleTask` requires thus the execution of a number of `Function` instances by the agents.

$$\begin{aligned} \texttt{SimpleTask} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\,\texttt{hasConstituent.Function} \sqcap \\ & \exists\,\texttt{hasConstituent.SimpleWorkpiece} \sqcap \\ & \exists\,(\texttt{isConformTo.InteractionModality} \sqcup \\ & \quad \texttt{isConformTo.OperativeConstraint}) \end{aligned} \tag{3}$$

The execution of tasks should comply with operational constraints that are represented as `ExecutionNorm`. Two main types of execution norms can be defined: the concept `OperativeConstraint` describes *norms* requiring the sequential or parallel execution of tasks; the concept `InteractionModality` instead characterizes *norms* about how agents should cooperate to carry out a task.

3.2. Reification of Collaborative Modalities

Although the “boundaries" of the *representation space* are well delimited within a domain ontology [22] the definition of a Knowledge Base (KB) is not straightforward (i.e., the “instantiation" of an ontological model). Mapping a general ontology (TBox) to a concrete, and effective, model (ABox) (e.g., a HRC work-cell as in this case) entails a significant number of design choices. Ontology design patterns [16] can play a role in supporting knowledge definition. Patterns can indeed specialize an ontological model without losing generality but defining useful “structures" that may facilitate knowledge definition.

In the considered domain, a crucial point is the representation of the hierarchical structure of production processes and the correlations between production tasks and the functions the human and the robot can actually perform. Ontological patterns in this case characterize typical and/or recurrent associations between tasks and functions. We specifically propose an extension of SOHO by introducing “domain-level" patterns that characterize different ways of performing collaborative tasks. The concept `HRCTask` is introduced as a particular type of `SimpleTask`. The basic assumption is that a `HRCTask` requires a minimum of one `Function` and a maximum of two `Function` to be correctly performed. Each required `Function` should be performed by a `HumanWorker` or by a `Cobot`. When two instances of `Function` are required, exactly one function should be performed by a `HumanWorker` and exactly one function should be performed by a `Cobot`. Furthermore, all the considered functions should have effect on the same target entity of the environment (i.e., `ProductionObject`).

According to [17], the execution of a collaborative task (i.e., an individual of `HRCTask`) entails one of four different collaboration modalities: (i) *Independent*, human and robot perform their tasks on different work pieces without collaboration; (ii) *Simultaneous*, human and robot perform distinct tasks on the same work piece at the same time, still without physical interaction; (iii) *Supportive*, human and robot perform the same task on the same work piece and they work simultaneously and cooperatively on the same task. (iv) *Synchronous*, human and robot should complete sequential tasks on the same work piece. Figure 2 shows a graphical representation of these four types of collaborative tasks.

Following this classification, the concept `HRCTask` has been further specialized into four types of (collaborative) task. These four types are reified as four patterns characterizing specific knowledge structures in terms of associated

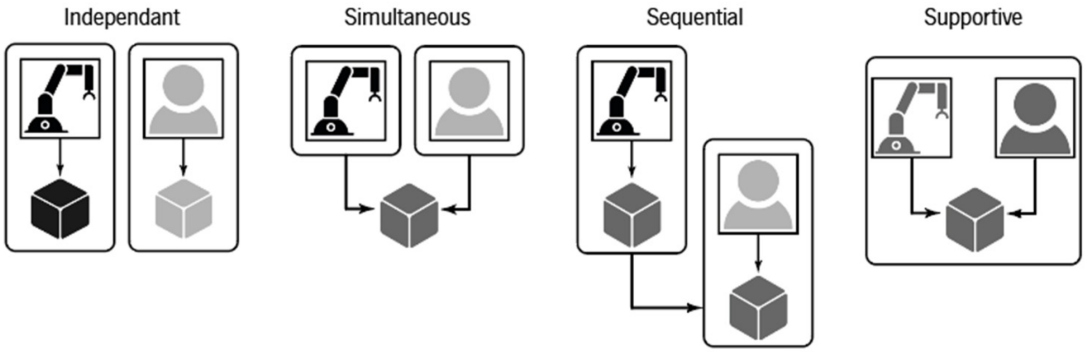


Fig. 2. Graphical representation of collaboration modalities

concepts and cardinality restrictions above related properties. Tasks of type `IndependentTask` are implemented by a single `Function` that can be performed by a `HumanWorker` or by a `Cobot`. A human/robot performs the function independently from the other.

$$\begin{aligned} \texttt{IndependentTask} \sqsubseteq\ & \texttt{HRCTask} \sqcap \\ & \exists! \, (\texttt{hasConstituent.HumFunction} \sqcup \\ & \quad \texttt{hasConstituent.RobFunction}) \sqcap \\ & \exists! \, \texttt{isConformTo.Independent} \end{aligned} \tag{4}$$

Tasks of type `SimultaneousTask` are implemented by exactly two instances of `Function`, one function performed by a `HumanWorker` (i.e., `HumanFunction`), another performed by a `Cobot` (i.e., `RobotFunction`). In this case, the human and the robot work on the same `WorkPiece` performing two different functions that can be carried out without any specific constraint.

$$\begin{aligned} \texttt{SimultaneousTask} \sqsubseteq\ & \texttt{HRCTask} \sqcap \\ & \exists! \, \texttt{hasConstituent.HumFunction} \sqcap \\ & \exists! \, \texttt{hasConstituent.RobFunction} \sqcap \\ & \exists! \, \texttt{isConformTo.Simultaneous} \end{aligned} \tag{5}$$

Tasks of type `SynchronousTask` are implemented by a `RobotFunction` and a `HumanFunction`. The pattern in this case forces the human and the robot at performing these two functions following a strict sequential order (this type of task is associated to the production norm `SequentialExecutionConstraint`).

$$\begin{aligned} \texttt{SynchronousTask} \sqsubseteq\ & \texttt{HRCTask} \sqcap \\ & \exists! \, \texttt{hasConstituent.HumFunction} \sqcap \\ & \exists! \, \texttt{hasConstituent.RobFunction} \sqcap \\ & \exists! \, \texttt{isConformTo.SequentialExec} \sqcap \\ & \exists! \, \texttt{isConformTo.Synchronous} \end{aligned} \tag{6}$$

Tasks of type `SupportiveTask` are implemented by a `RobotFunction` and a `HumanFunction`. In this case the pattern forces the human and the robot at performing two functions in parallel (this type of task is associated to the production norm `ParallelExecutionConstraint`) achieving the highest level of collaboration.

$$\begin{aligned} \texttt{SupportiveTask} \sqsubseteq\ & \texttt{HRCTask} \sqcap \\ & \exists! \texttt{hasConstituent.HumFunction} \sqcap \\ & \exists! \texttt{hasConstituent.RobFunction} \sqcap \\ & \exists! \texttt{isConformTo.ParallelExec} \sqcap \\ & \exists! \texttt{isConformTo.Supportive} \end{aligned} \tag{7}$$

The definitions (4), (5), (6) and (7) constitute representation schema that facilitate the definition of typical production tasks of HRC scenarios that should be performed following a well-defined procedure (execution modality). Such reusable structures thus can help the definition of KB as well as the definition process of control models such as, e.g., plan-based specification by providing a known, formal and well founded reference for the definition of operative and behavioral constraints.

4. Knowledge Definition and Automated Synthesis of Plan-based Control Models

A *production knowledge* defined according to the semantics proposed by SOHO characterizes a particular HRC scenario from different perspectives. The obtained knowledge represents a kind of “standard" model of “acting dynamics" that can be used to support control features. In this work, we are interested in investigating how the knowledge can be leveraged to autonomously synthesize plan-based control models and, thus, support dynamic reconfiguration and flexible control of industrial (collaborative) robots.

In this work, we are interested in investigating how this knowledge can be leveraged to automatically generate planning specifications and endow collaborative robots with control features to support dynamic reconfiguration and flexible control of industrial (collaborative) robots. We thus design a knowledge extraction procedure to automatically generate suitable planning domain specifications and dynamic configure a robot controller to support collaborative production processes. This procedure bridges the gap between (production) knowledge representation and robot control and enables the realization of a cognitive “perceive-reason-act" loop as shown in [11] for Reconfigurable Manufacturing Systems.

Although the production knowledge relies on standard semantic technologies and therefore is “planning agnostic", we here design a general procedure that complies production knowledge into a temporal planning model that complies with the timeline-based planning formalism [35].

4.1. Timeline-based Approach to Planning

Task planning and scheduling capabilities rely on the timeline-based paradigm formalized in [35]. A timeline-based specification consists of a number of *state variables* that describe possible behaviors of domain features to be controlled over time. A state variable is defined as a tuple $SV = \langle V, T, D, \gamma \rangle$ where: (i) V is a set of *values* $v_i \in V$ representing states or actions the feature can assume or perform over time; (ii) $T : V \rightarrow 2^V$ is a transition function specifying valid sequences of values $v_i \in V$; (iii) $D : V \rightarrow \mathbb{R} \times \mathbb{R}$ is a duration function associating to each value $v_i \in V$ lower and upper bounds to its execution (i.e., duration bounds); (iv) $\gamma : V \rightarrow \{c, pc, u\}$ is the *controllability tagging function* specifying if the execution of a value $v_i \in V$ is *controllable* (c), *partially controllable* (pc) or *uncontrollable* (u).

Information about *controllability* is necessary to reliably deal with *temporal uncertainty* and uncontrollable dynamics of the environment during the execution of a (timeline-based) plan. This is known as the *controllability problem* [36] and is particularly relevant in scenarios like HRC where an artificial system like e.g., a collaborative robot, interacts with “unpredictable" agents like e.g., a human worker. Complex behaviors of a system (e.g., a HRC work-cell) are modeled by means of *synchronization rules* that constrain simultaneous behaviors of state variables

whose temporal evolution are the *timelines* of a plan. A rule is a kind of *logical entailment* specifying a behavioral dependency among timelines. Every time a value v_x is assumed by a variable SV_i a number of values v_y should be assumed by other state variables SV_j. The temporal occurrences of such values should satisfy the set of temporal constraints of the rule.

The task planning model follows a hierarchical decomposition methodology and (in the case of HRC scenarios) is generally structured as follows: (i) a state variable SV_G describes the high-level production goals that can be performed within the HRC work-cell; (ii) a number of state variables SV_L^i where $i = 0, ..., K$ describe the production tasks to be performed at a specific abstraction level i, where K is the number of hierarchy levels of the procedure; (iii) a state variable SV_R and a state variable SV_H describe the low-level operations (i.e., instances of `Function`) the robot and the human can actually perform; (iv) a set of synchronization rules $\mathcal{S}$ describes the procedural decomposition of high-level goals (i.e., values of state variable SV_G) into increasingly simpler production tasks (i.e., values of state variables SV_L^i), until they are associated to a number of functions (i.e., values of state variables SV_R and SV_H) the human and the robot should perform to complete production tasks and achieve high-level goals.

4.2. Knowledge Extraction and Model Synthesis

The designed knowledge processing mechanism relies on the extended version of SOHO to retrieve contextualized knowledge and also uses ontological patterns to model planning domain constraints. A pseudo-code description of the main steps composing this procedure is given in Algorithm 1.

Algorithm 1 Timeline-based specification extraction

Input: $\mathcal{KB}$
Output: $\mathcal{M}$

1: $S_G \leftarrow$ goals $(\mathcal{KB})$
2: $\{F_H, F_R\} \leftarrow$ functions $(\mathcal{KB})$
3: $\{SV_G, SV_H, SV_R\} \leftarrow$ createSVs $(\mathcal{M}, S_G, F_H, F_R)$
4: **for** $g \in S_G$ **do**
5: $\quad S_M \leftarrow$ methods $(g, \mathcal{KB})$
6: $\quad$ **for** $m \in S_M$ **do**
7: $\quad\quad \mathcal{G}_{g,m} \leftarrow$ decompositions $(g, m, \mathcal{KB})$
8: $\quad\quad \{\mathcal{T}_1, ..., \mathcal{T}_l\} \leftarrow$ hierarchy $(\mathcal{G}_{g,m})$
9: $\quad\quad$ **for** $\mathcal{T}_i \in \{\mathcal{T}_1, ..., \mathcal{T}_l\}$ **do**
10: $\quad\quad\quad SV_{m,i}^g \leftarrow$ createSV $(\mathcal{M}, \mathcal{T}_i)$
11: **for** $g \in S_G$ **do**
12: $\quad S_M \leftarrow$ methods $(g, \mathcal{KB})$
13: $\quad$ **for** $m \in S_M$ **do**
14: $\quad\quad \mathcal{T} \leftarrow$ tasks $(m, \mathcal{KB})$
15: $\quad\quad$ **for** $t \in \mathcal{T}$ **do**
16: $\quad\quad\quad \mathcal{T}^t \leftarrow$ decomposition $(t, m, \mathcal{KB})$
17: $\quad\quad\quad \mathcal{R}^t \leftarrow$ createRule $(\mathcal{M}, t, \mathcal{T}^t)$
18: **return** $\mathcal{M}$

First, the procedure extracts information about which production goals and functions can be actually performed by the human and the robot (rows 1-2). This information is easily retrieved by extracting individuals of `ProductionGoal` and individuals of `Function` that can be performed by the agents `HumanWorker` and `Cobot`. The procedure then defines the *state variables* characterizing possible goals of the planning model (SV_G) and possible (operational) behaviors of the human and the robot (SV_H and SV_R). The extracted individuals are then used to define the *values* of these state variables (row 3). Next steps define the state variables that model production processes (rows 4-17) and thus correlate production goals (i.e., possible *planning goals*) to the operations the human and the robot should perform over time (i.e., their functions). A specific state variable is defined for each hierarchical decomposition level of a production process. For each production goal the procedure retrieves the list of associated

ProductionMethod (row 5). For each method the procedure retrieves the described decomposition by extracting ProductionTask associated through the property hasConstituent (row 7). The decomposition is internally represented as a *graph* ($\mathcal{G}_{g,m}$) where a root node represents a ProductionGoal, the leaves represent Functions and intermediate nodes represent instances of ComplexTask or SimpleTask.

The graph $\mathcal{G}_{g,m}$ is acyclic by construction. A topological sort algorithm extracts hierarchical information as a partitioning of production tasks in a number of equivalent sets $\{\mathcal{T}_1, ..., \mathcal{T}_l\}$ (row 8). Ignoring the root and the leaves of $\mathcal{G}_{g,m}$, each equivalent set $\mathcal{T}_i \in \{\mathcal{T}_1, ..., \mathcal{T}_l\}$ represents a hierarchical level of the decomposition procedure. For each equivalent set of production tasks $\mathcal{T}_i$ the procedure defines a dedicated state variable $SV^g_{m,i}$ (rows 9-10). The values of such state variables represent ComplexTask or SimpleTask that belong to the same decomposition level of the (hierarchical) production process described by a ProductionMethod $m \in \mathcal{S}_M$.

When all the state variables have been defined, the procedure generates the synchronization rules (rows 11-17). For each goal $g \in \mathcal{S}_G$ and for each associated method $m \in \mathcal{S}_M$, the procedure retrieves the set of production tasks $\mathcal{T}$ (row 14). For each task $t \in \mathcal{T}$ the procedure retrieves its direct decomposition extracting the set of tasks $\mathcal{T}^t$ associated to t through the property hasConstituent (i.e., without transitivity). For each couple of task t and subtasks $\mathcal{T}^t$, the procedure creates a new synchronization rule R^t (rows 15-17). The reference task t is the head of the rule while tasks $\mathcal{T}^t$ compose the body. Temporal constraints are defined according to the type of the reference task t.

If t is a ComplexTask then the rule models a decomposition constraint by means of *contains* constraints between t and subtasks in $\mathcal{T}^t$ ($|\mathcal{T}^t| > 1$). Eventually, a number of *before* temporal constraints between subtasks $\mathcal{T}^t$ can be added for each OperativeConstraint. If t is a SimpleTask then the constraints of the rule follow ontological patterns. In case of IndependentTask the set of subtasks $\mathcal{T}^t$ is composed by only one robotic or human function ($|\mathcal{T}^t| = 1$) and a *contains* constraint is set between t and its subtask. In case of SimultaneousTask the set of subtasks $\mathcal{T}^t$ is composed by one robotic function and one human function ($|\mathcal{T}^t| = 2$). Two *contains* constraints are set between t and the two subtasks. In case of SupportiveTask the set of subtasks $\mathcal{T}^t$ is composed by one robotic function and one human function ($|\mathcal{T}^t| = 2$). Two *during* constraints between the two subtasks and the reference task t are set to enforce the simultaneous execution of subtasks. Finally, in case of SynchronousTask the set of subtasks $\mathcal{T}^t$ is again composed by one robotic function and one human function ($|\mathcal{T}^t| = 2$). In addition to the *contains* constraints between t and its subtasks, a *meets* temporal constraint is set between the two subtasks in order to satisfy the desired synchronism.

5. Representation and Reasoning Assessment on Real-World Scenarios

The feasibility of the proposed ontology-based representation and reasoning approach has been assessed on a set of real HRC manufacturing scenarios, elicited from a H2020 research project, called Sharework. The ontology and production knowledge have been defined in Protégé. The reasoning mechanisms and the knowledge extraction procedure of Algorithm 1 have been developed in Java using Apache Jena[2]. Furthermore, these functionalities have been integrated into ROS[3] through ROSJava[4] and supporting the dynamic configuration of the task planning module developed within the project.

The envisaged evaluation considers a number of 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. These four scenarios are extracted from the pilots of the projects and thus characterize actual production processes. They constitute a realistic benchmark to assess the proposed the reasoning capabilities of the developed AI-based technologies against the flexibility required by real production scenarios.

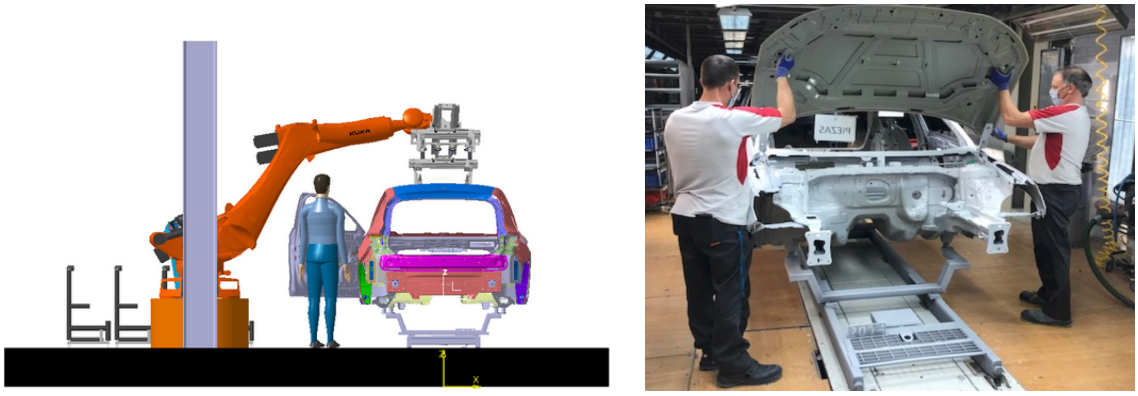
[2]https://jena.apache.org
[3]Distribution ROS Melodic - http://wiki.ros.org/melodic
[4]http://wiki.ros.org/rosjava

In addition, we consider an advanced (and futuristic) productive scenario where task allocation and possible alternative behaviors of the human and the robot are largely more variable. This scenario is called MOSAIC and takes inspiration from a typical collaborative assembly scenario [37]. Although it does not correspond to a concrete production process, MOSAIC describes a highly flexible (collaborative) production process entailing the representation of various possible (alternative) behaviors of the robot and the human into the knowledge and the resulting task planning model.

5.1. Industrial Scenarios

5.1.1. The AUTOMOTIVE Scenario

This scenario takes into account a specific station of an assembly line of vehicles. The considered collaborative process specifically focuses on a door assembly task of chassis on the conveyor of the production line. 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 3 shows some pictures of the layout of the working-are and “mount point" of the front door on the chassis.



(a) (b)

Fig. 3. Design of the collaborative cell for the AUTOMOTIVE scenario (a) and structure of the production line for the assembly of the chassis (b).

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. There is a variety of `Function` the human and the robot should perform in this scenario. The robot (a robotic arm), according to its capabilities, 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 carries out the actual assembly operations and should therefore perform functions of different type like e.g., `Assemble`, `ManualGuidance`, `ChangeOver` and `Screw` on the `vehicle body` (`WorkPiece`). These types of `Function` are defined in SOHO as a “specialization" of the integrated Taxonomy of Functions [32] (e.g., `Assemble` and `Join` are specialization of `Join`).

In this case, all the production tasks the human and the robot can perform through their functions are modeled as `IndependentTask`. This means that each task requires a single `Function` the human or the robot carries out autonomously and *independently* from each other (from an operational perspective). For example the robotic task of moving a door to the front assembly area of the layout is modeled as and independent collaborative task (i.e., instance of `IndependentTask`) and implemented by a pick-place operation of the robot (i.e., an instance of `PickPlace` function).

Figure 13(a) shows the hierarchical structure of the planning model automatically generated from the knowledge base of the described production process. The top-level element of the hierarchy is associated to predicates

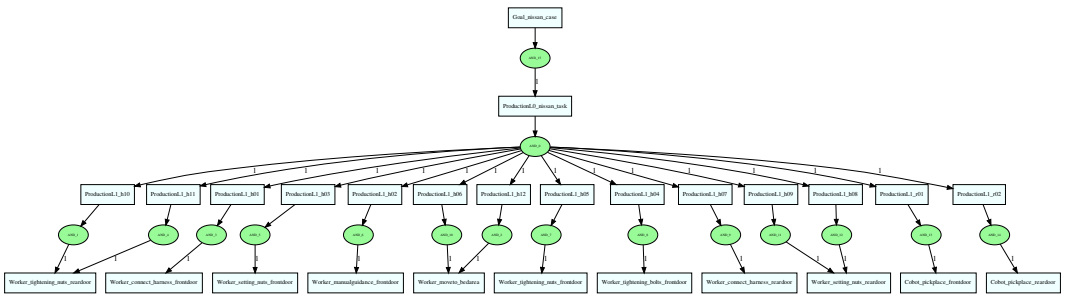


Fig. 4. Decomposition graph of the production process of AUTOMOTIVE. The top-level goal is decomposed into a single `ComplexTask` which is in turn decomposed into a number of `IndependentTask` that compose the state variable `ProductionL1` shown in the hierarchy of Figure 13(a). Each `IndependentTask` entails the execution of a `Function` by the human or by the robot.

associated to the instances of `ProductionGoal` extracted from the knowledge base. The low-level elements of the hierarchy corresponds to the predicates associated to the functions the human and the robot can perform. The intermediate levels are instead associated to the predicates (and state variables) that characterize the hierarchical decomposition of complex tasks (i.e. production goals) into more and more simple tasks.

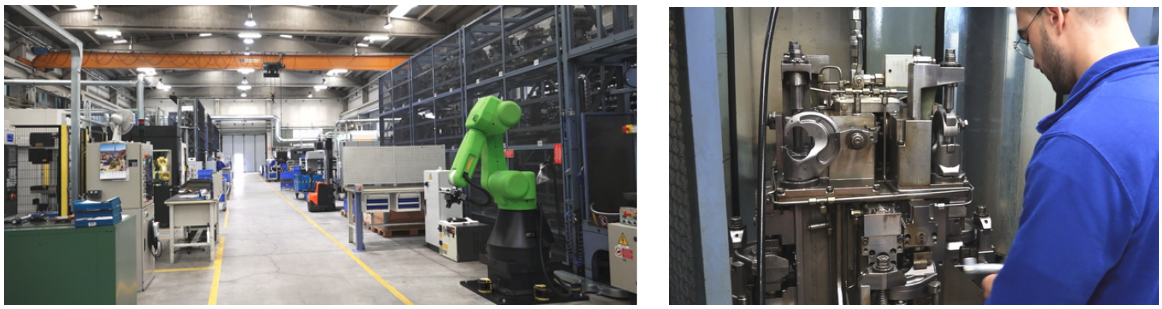
More in details Figure 4 shows the task decomposition graph extracted from the knowledge base to build the (timeline-based) planning model through Algorithm 1. The levels of the graph correspond to the hierarchical levels of Figure 13(a). As can be seen the process obtained for this scenario is quite simple, requiring a direct decomposition of a single `ComplexTask` in a number of `SimpleTask` that should be performed sequentially by the human and the robot. Each simple task is specifically modeled as `IndependentTask` each requiring the execution of a specific `Function` by the human or by the robot (as stated in 4). The task planning model encapsulates this kind of collaborative behavior through a single *contains relation* [35] requiring the predicate associated to the `Function` of the human or the robot to be executed during the execution of the predicate associated to the (simple) production task.

5.1.2. The METAL *Industrial Scenario*

This scenario takes into account the logistic station of the manufacturing system of 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 scenarios 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 a 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 and 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 the work of operators and speedup the production, i.e., increasing the throughput.

Figure 5 shows some pictures of the layout of the designed collaborative 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 at both sides of the conveyor and works simultaneously on the products. Products are placed and move on the central conveyor which represents the shared working area where operations take place and where the human and the robot physically interact.

The considered production process is characterized by different types of operations depending on the specific types of work-piece entering the collaborative cell. In all the cases however the structure of the process is similar in terms of task allocation choices since the human and the robot can perform the same types of task (e.g., screw, unscrew, pick, place, etc.). The worker and the robot represent two autonomous and "functionally equivalent" actors that work *independently* on each task. The synthesis of collaborative processes thus concerns the correct allocation of tasks to these two *resource* (i.e., autonomous agents).



(a) (b)

Fig. 5. Structure of the shop-floor of the METAL industrial scenario (a) and the fixturing system where collaboration takes place (b).

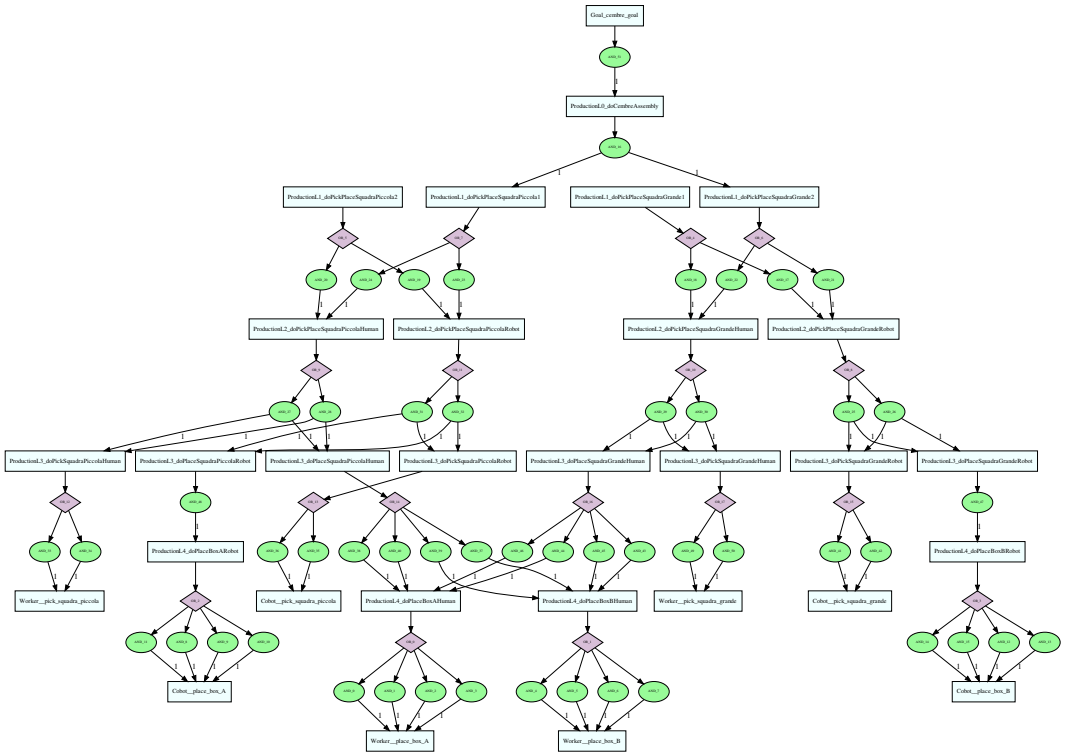


Fig. 6. Decomposition graph of the production process of METAL. The top-level goal is decomposed into a single `ComplexTask` which is further decomposed into a number of `IndependentTask` that compose the state variable `ProductionL1` shown in the hierarchy of Figure 13(a). Each `IndependentTask` entails the execution of a `Function` by the human or by the robot.

Figure 6 shows an excerpt of the *inferred* task decomposition graph concerning a process of work-pieces that mainly entails *pick and place* operations. The considered process concerns the *substitution* of a `WorkPiece` from the "base" moving on the conveyor with another `WorkPiece` available into the collaborative cell. The substitution is realized through pick & place operations that can be performed by both the worker and the robot. The worker 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`).

In this case functions of type `Pick` and `Place` are represented separately (i.e., not as atomic instances of `PickPlace`) in order to capture operational requirements and properly characterize possible planning choices

(e.g., which box to use to place a work-piece). The decomposition graph of Figure 6 thus contains several `DisjunctiveComplexTask` representing decomposition choices. For example, the excerpt of Figure 6 shows four *pick & place* tasks to be performed. Each task can be performed either by the worker or by the robot. Such tasks are represented as `DisjunctiveComplexTask` and decomposed into human and robotic *pick & place* `ComplexProductionTask` through OR nodes. Such tasks are further decomposed into two distinct *pick* and *place* `ProductionTask` to characterize the specific `Pick` and `Place` functions the worker and the robot should perform. Tasks that concern *pick* operations are generally represented as `IndependentTask` and directly mapped to the instances of `Pick` functions of the worker and the robot. Tasks that concern *place* operations are instead represented as `DisjunctiveComplexTask` and associated to alternative decomposition through OR nodes. Each decomposition represents an alternative human or robotic task that places a `WorkPiece` into one of the available boxes (i.e., `Box-A` and `Box-B`). Each task is then represented as an `IndependentTask` and directly linked to the `Place` function of the worker or the robot

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 specifically considers a servo rotary table that is assembled in seven fixed assembly stations. In each station there is an operator performing a specific task of 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 operator and the robot concerns three out of seven tasks of the rotary table assembly process. The introduction of collaborative robots in this scenario represents an example of how such technology can support operators workers in handling heavy parts and generally facilitate operations by passing parts to be assembled or tools to be used. In the current assessment we specifically consider the task *bolt tightening and torque measuring*. Figure 7 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 determines their position and dimension through perception modules developed with the project. Information about detected bolts is then used by the robot to automatically screw them.

(a) (b)

Fig. 7. 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.

The scenario has been designed as “reactive" where the robot uses *perception capabilities* to autonomously react to the observed behavior of the worker (“bolt placing"). However, to evaluate the representation capability of the ontological model and synthesis of the task planning model, we have modeled the whole production process as collaborative. Namely, we have considered the process of screwing a number $N = 8$ of bolts. The worker places the bolts on the holes of the rotary table and the robot screw 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 production task of “screwing a bolt" should be represented as `SynchronousTask` since required human and robotic functions have the same target (i.e., a particular *hole-X* that can be seen as a `SimpleWorkPiece` composing the *CompoundWorkPiece rotary table*) and should follow a strict temporal ordering. In this case the `Screw` function of the robot must always be performed *after* the `PickPlace` function of the human has been executed.

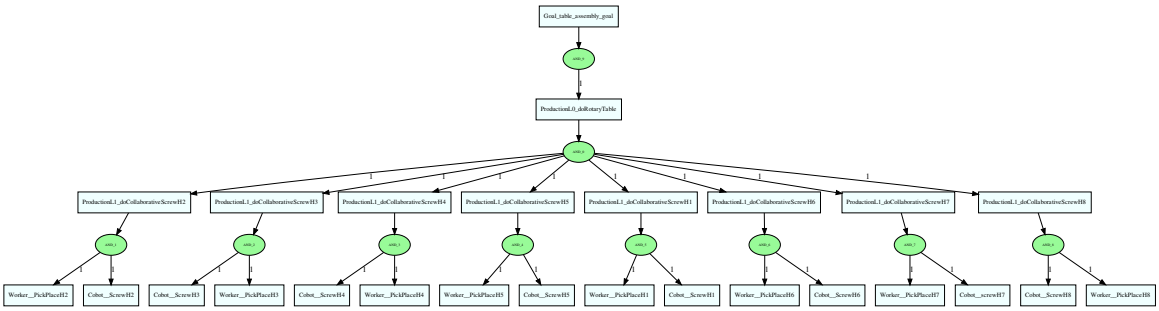


Fig. 8. Decomposition graph of the production process of CAPITAL-GOODS.

Figure 8 shows the obtained decomposition graph of the described production process. The high-level production goal `rotary-table-assembly` is decomposed into a number of (complex) `ConjunctiveProductionTask`, one for each *hole* of the rotary table to be screwed. Each *screw* task is represented as `SynchronousTask` 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 like e.g., `screw-H3` requiring the operator to screw the bolt placed by the worker. Notably, the ontological pattern used to describe this type of collaborative behavior is translated into a well-defined structure of temporal constraints into the task planning model. In this case for example the *pattern* determines a *synchronization rule* requiring; (i) the state variable of the worker to assume the value associated to function of picking and placing a bolt on `hole-X`; (ii) the state variable of the robot to assume the value associated to the function of screwing a bolt on `hole-X`; (iii) to schedule (and thus execute) the value of the human *before* the value of the robot.

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 by 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 tram-way's windows and door frames. Among the tasks involved into 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 weights up to 5 kg.

(a) (b)

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

Figure 9 shows the physical layout of the shop-floor of the scenario 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 condition and reduce the risk of injuries.

A collaborative robot is thus supposed to work close to the window frame of Figure 9(b) and tightly collaborate with the human worker to carry out the pre-assembly tasks of window frames. The collaboration especially 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 insert 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. Such task will be represented as instances of `SynchronousTask` i.e., collaborative tasks (`HRCTask`) entailing a `Synchronous` execution modality).

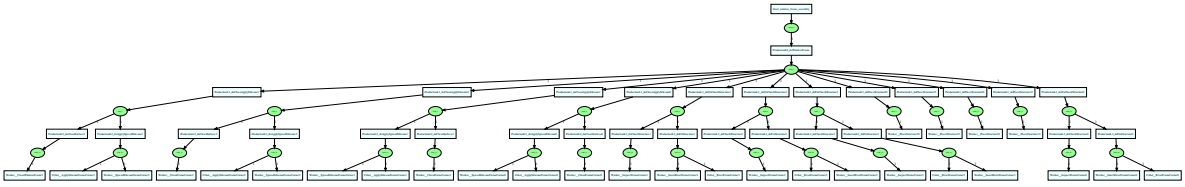
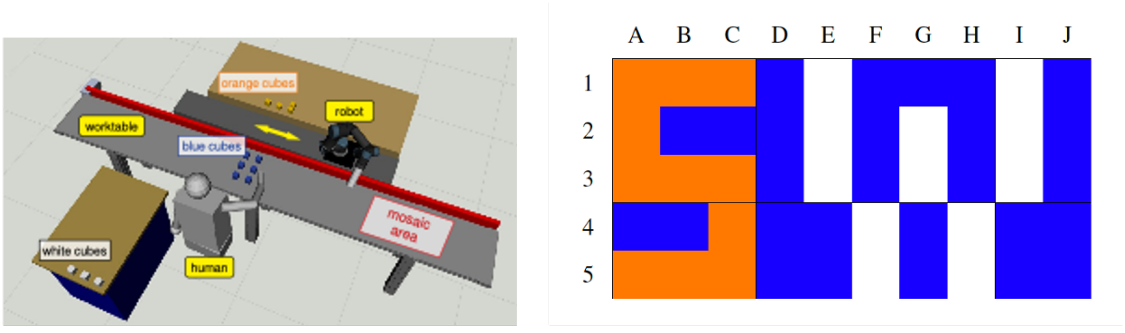


Fig. 10. Decomposition graph of the production process of the RAILWAYS scenario.

Figure 10 shows the decomposition graph of the considered production process. Also in this case there are not disjunctive tasks to be considered into the decomposition since the roles and "responsibilities" of the worker and the robot are quite fixed. The key aspect in this case similar to the CAPITAL-GOOD scenario is the use of `SynchronousTask` to represent *riveting tasks*. Such tasks are indeed associated to 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 is constrained by a *before temporal constraint* and thus constrain the robot to wait for the completion of the function of the worker. As seen for the CAPITAL-GOODS scenario, such *behavioral pattern* is properly encoded into the task planing model to correctly synthesize and execute the collaborative process.

5.1.5. *The* MOSAIC *Collaborative Scenario*

This scenario considers a general collaborative assembly of a compound work-piece. The layout of the work-cell is characterized by a shared central space where the work-piece is placed and where the human and the robot simultaneously carry out assembly operations. Figure 11 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 by simpler parts like e.g., *rows* (still `CompoundWorkPiece`) and *cells* (`SimpleWorkPiece`).



(a) (b)

Fig. 11. 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

The collaborative process consists in the execution of a set of *pick & place* operations. Each *pick & place* is performed by a single *agent* but their execution (and assignment) should satisfy some physical constraints. *Pick &*

place operations whose targets are coloured 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.

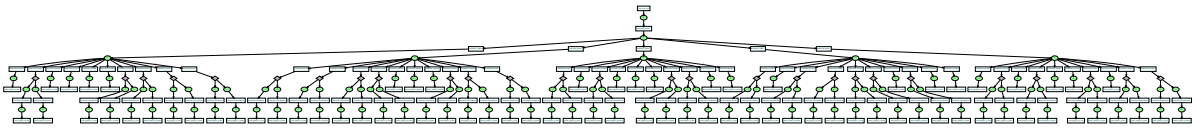


Fig. 12. Decomposition graph of the production process of MOSAIC. The top-level goal `mosaic_goal` is decomposed into a number of `ComplexTask` hierarchically organized as shown in Figure 13(b). State variable `ProductionL0` concerns a single `ComplexTask doMosaic`. This task is further decomposed into a number of row-level `ComplexTask doRow1`, ..., `doRow5` that are part of state variable `ProductionL1`. Each row-level task is further decomposed into a number of `IndependetTask` like e.g., `doCellI3` and *disjunctive* `ComplexTask` like e.g., `doCellJ3`. These tasks composed the state variable `ProductionL2`. The state variable `ProductionL3` thus contains `IndependnetTask` representing alternative decomposition of disjunctive cell-level tasks. Each `IndependentTask` entails the execution of a specific `PickPlace` by the human or by the robot.

The ASSEMBLY scenario is characterized by a more structured process where 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 by 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 have the same capabilities and should perform the same type of `Function` i.e., `PickPlace`. Each `PickPlace` moves a particular instance `wp1`, ..., `wpN` of `WorkPiece` 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 `IndependentTask` which should be associated to a `PickPlace` function of the human or a `PickPlace` function of the robot (see Eq. 4).

The resulting production process should contain a disjunction of tasks every time the needed pick-and-place functions can be performed by both agents. For example, a disjunctive `ComplexTask doCell(1,1)` would be associated to (alternative) `IndependentTask human-doCell(1,1)` (entailing `PickPlace pickplace-wp1-cell(1,1)-human`) and `IndependentTask robot-doCell(1,1)` (`PickPlace pickplace-wp1-cell(1,1)-robot`). No disjunction should be considered if a particular task can be performed only by one agent. Suppose for example that `cell(5,10)` cannot be reached by the robot, only the human would be able to perform that task. The production process will therefore specify only a `IndependentTask doCell(5,10)` (`PickPlace` function `pickplace-wp1-cell(5,10)-human`).

Figure 13(b) shows the hierarchical structure of the designed production process. The top-level element of the hierarchy describes the production goals of the scenario. The low-level elements of the hierarchy corresponds to the description of the pick-and-place functions the human and the robot should perform. The intermediate levels describe the hierarchical decomposition of complex tasks in simple tasks. More in details Figure 12 shows the task decomposition graph extracted from the knowledge and used to build the (timeline-based) planning model through Algorithm 1.

5.2. Generation of Planning Specification from Knowledge

Table 1 summarizes and compare data about the defined knowledge bases, the characteristics of the synthesized planning models and 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, it can be observed a different number of *individuals* for each of them and a different number of constraints and predicates in the resulting 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 time. The planning model obtained for

Table 1
Data about the size of knowledge bases and planing models and performance of model synthesis

Domain	Knowledge Base			Planning Model				Model Generation
	#Classes	#Properties	#Individuals	#SVs	#Predicates	#Synchs	#Cons	Time (msecs)
AUTOMOTIVE	284	186	61	5	34	15	28	4189
METAL	284	186	45	8	38	57	68	7885
CAPITAL-GOODS	284	186	42	5	31	9	16	4165
RAILWAYS	284	186	75	6	64	29	48	8879
MOSAIC	284	186	215	7	195	137	186	20408

AUTOMOTIVE for example is characterized by a lower number of predicates and synchronization rules/constraints. The planning model obtained for MOSAIC instead, given its structural augmented 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 that 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 does not introduce latency that may affect the efficiency of the system.

Results have been obtained by configuring the Apache Jena framework with an ontological model compliant with OWL-DL semantics [5]. This semantics supports many OWL features and represents a good trade-off between 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 using PLATINUm[6], a timeline-based planning and scheduling framework [38]. Figure 13 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 in 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 and according to [35], each hierarchy level is described through a dedicated *state variable*. Values of such state variables (i.e., the *predicates* of Table 1) describe tasks that should be performed at the associated abstraction level of the production procedure. State variables of the last hierarchical layer describe the tasks (i.e., the concrete operations) the worker and the operator 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 a 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 [14, 15, 37], describe the planned decomposition of modeled production procedures and planned *temporal behaviors* of collaborating actors (i.e., the worker and the robot). Following [35, 39] then, each token instantiates a value of a state variable to a flexible execution interval (the intervals associated to 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 aspects 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] More specifically, we have defined a basic ontological model with OWL-DL-MEM specification combined with an rule-based inference model based on OWL-MEM-MICRO-RULE-INF which encapsulates optimized rule-based reasoner with OWL rules. - https://jena.apache.org/documentation/ontology

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

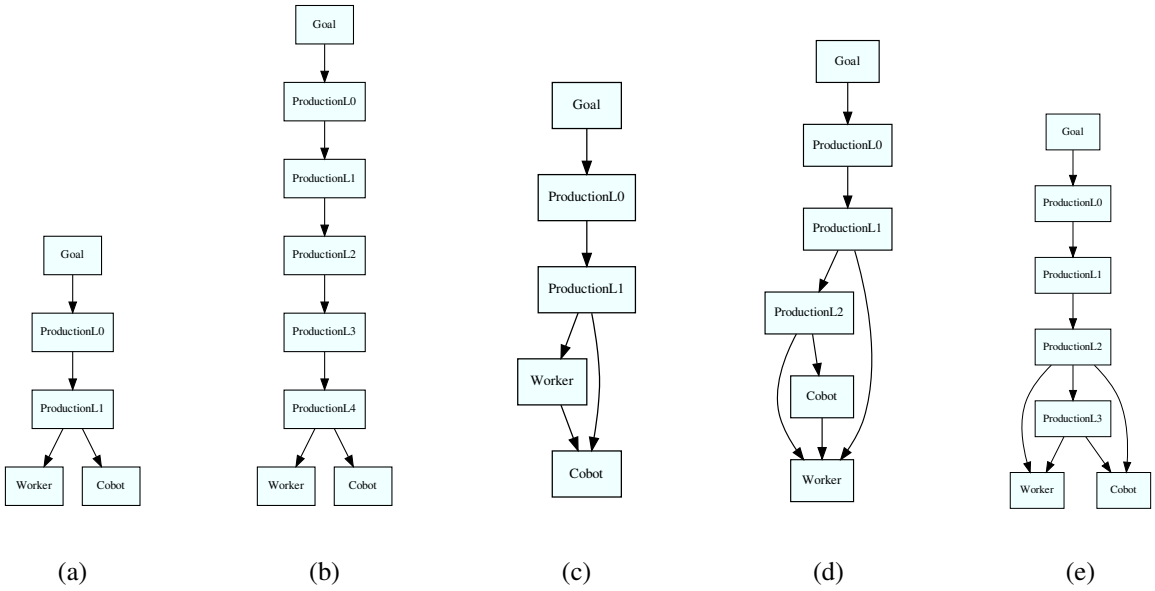


Fig. 13. Hierarchical structure of the generated planning models. The two graphs show the inferred hierarchical relationships between the *state variables* generated for: (a) AUTOMOTIVE; (b) METAL; (c) CAPITAL-GOODS; (d) RAILWAYS; (e) MOSAIC

6. Final Remarks 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 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 [14, 37]. 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, physical and cognitive conditions of human workers that are crucial to serve 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 [40].

Acknowledgement

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.

References

[1] 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.

[2] 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.

[3] 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.

[4] B. Dworschak and H. Zaiser, Competences for Cyber-physical Systems in Manufacturing – First Findings and Scenarios, *Procedia CIRP* **25** (2014), 345–350, 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

[5] S. Makris, *Cooperating Robots for Flexible Manufacturing*, Vol. 1, Springer International Publishing, 2021.

[6] K. Rajan and A. Saffiotti, Towards a science of integrated AI and Robotics, *Artificial Intelligence* **247** (2017), 1–9.

[7] F. Ingrand and M. Ghallab, Deliberation for autonomous robots: A survey, *Artificial Intelligence* **247** (2017), 10–44, Special Issue on AI and Robotics.

[8] 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.

[9] 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.

[10] 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.

[11] S. Borgo, A. Cesta, A. Orlandini and A. Umbrico, Knowledge-based Adaptive Agents for Manufacturing Domains, *Engineering with Computers* **35**(3) (2019), 755–779.

[12] S. Borgo, A. Cesta, A. Orlandini and A. Umbrico, A Planning-based Architecture for a Reconfigurable Manufacturing System, in: *The 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 2016, pp. 358–366.

[13] A. Umbrico, A. Orlandini and A. Cesta, An Ontology for Human-Robot Collaboration, *Procedia CIRP* **93** (2020), 1097–1102.

[14] 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. To appear.* (2017).

[15] A. Cesta, A. Orlandini and A. Umbrico, Fostering Robust Human-Robot Collaboration through AI Task Planning, *Procedia CIRP* **72** (2018), 1045–1050, 51st CIRP Conference on Manufacturing Systems.

[16] 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, Berlin, Heidelberg, 2005, pp. 262–276. ISBN 978-3-540-32082-1.

[17] 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.

[18] R. Studer, V.R. Benjamins and D. Fensel, Knowledge engineering: Principles and methods, *Data & Knowledge Engineering* **25**(1) (1998), 161–197.

[19] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing?, *International Journal of Human-Computer Studies* **43**(5) (1995), 907–928.

[20] B. Willem Nico, Construction of Engineering Ontologies for Knowledge Sharing and Reuse, PhD thesis, Centre for Telematics and Information Technology (CTIT), 1997. ISBN 90-365-0988-2.

[21] 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.

[22] N. Guarino, *Formal Ontology in Information Systems*, IOS Press, 1998.

[23] V. Mascardi, V. Cordì and P. Rosso, A Comparison of Upper Ontologies, in: *Workshop from Objects to Agents (WOA)*, 2007, pp. 1–10.

[24] A. Kiryakov, K.I. Simov and M. Dimitrov, OntoMap: Portal for Upper-Level Ontologies, in: *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, Association for Computing Machinery, 2001, pp. 47–58.

[25] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari and L. Schneider, Sweetening Ontologies with DOLCE, in: *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, A. Gómez-Pérez and V.R. Benjamins, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 166–181. ISBN 978-3-540-45810-4.

[26] 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.

[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, The SSN Ontology of the W3C Semantic Sensor Network Incubator Group **17**(C) (2012), 25–32–.

[28] H.-M. Huang, K. Pavek, B. Novak, J.S. Albus and E.R. Messina, A Framework For Autonomy Levels For Unmanned Systems (ALFUS), in: *Association of the Unmanned Vehicle System International (AUVSI)*, 2005.

[29] E.M. Sanfilippo, Y. Kitamura and R.I.M. Young, Formal ontologies in manufacturing, *Applied Ontology* **14** (2019), 119–125.

[30] 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.

[31] 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.
[32] 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.
[33] 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.
[34] J.T.C. Tan, Feng Duan, Ye Zhang and Tamio 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.
[35] M. Cialdea Mayer, A. Orlandini and A. Umbrico, Planning and execution with flexible timelines: a formal account, *Acta Inf.* **53**(6–8) (2016), 649–680.
[36] 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.
[37] 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.
[38] 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.
[39] 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.
[40] 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.