Journal of Manufacturing Systems 70 (2023) 359–381

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Deploying OWL ontologies for semantic mediation of mixed-reality interactions for human–robot collaborative assembly

Joe David [a,b,*], Eric Coatanéa [a], Andrei Lobov [b]

[a] Tampere University, Korkeakoulunkatu 7, 33720 Tampere, Finland
[b] Norwegian University of Science and Technology, Richard Birkelands vei 2b, 7034 Trondheim, Norway

ARTICLE INFO

Keywords:
Human–robot collaboration
Mixed reality
Multi-agent systems
OWL ontology
SHACL
Belief–desire–intent
Digital thread

ABSTRACT

For effective human–robot collaborative assembly, it is paramount to view both robots and humans as autonomous entities in that they can communicate, undertake different roles, and not be bound to pre-planned routines and task sequences. However, with very few exceptions, most of recent research assumes static pre-defined roles during collaboration with centralised architectures devoid of runtime communication that can influence task responsibility and execution. Furthermore, from an information system standpoint, they lack the self-organisation needed to cope with today's manufacturing landscape that is characterised by product variants. Therefore, this study presents collaborative agents for manufacturing ontology (CAMO), which is an information model based on description logic that maintains a self-organising team network between collaborating human–robot multi-agent system (MAS). CAMO is implemented using the Web Ontology Language (OWL). It models popular notions of net systems and represents the agent, manufacturing, and interaction contexts that accommodate generalisability to different assemblies and agent capabilities. As a novel element, a dynamic consensus-driven collaboration based on parametric validation of semantic representations of agent capabilities via runtime dynamic communication is presented. CAMO is instantiated as agent beliefs in a framework that benefits from real-time dynamic communication with the assembly design environment and incorporates a mixed-reality environment for use by the operator. The employment of web technologies to project scalable notions of intentions via mixed reality is discussed for its novelty from a technology standpoint and as an intention projection mechanism. A case study with a real diesel engine assembly provides appreciable results and demonstrates the feasibility of CAMO and the framework.

1. Introduction

The fifth industrial revolution is neither a chronological addition nor an alternative to the fourth (better known as Industry 4.0) but rather one that extends its key traits to be human centric, resilient and sustainable [1,2]. Many key technologies of Industry 4.0 are thus expected to act as important enablers of this transition [2]. Human centricity can be instantiated as human-centric interactions with the manufacturing environment [3]. However, as noted by Li et al. [4], interactions in today's human–robot collaboration (HRC) systems predominantly suffer from two weaknesses. Firstly, these interactions do not embody adaptive robotics or the intuitive role of the human operator, as they are designed to follow operator commands, such as gestures or augmented-reality instructions. Secondly, their collaboration is largely unidirectional and restricted to a master/slave model, far from the adaptability and flexibility embodied in the social aspects of teamplay. Proactive HRC is proposed as a cognitive paradigm in this direction, as "a self-organising, bi-directional collaboration between human operators and robots in manufacturing activities, where they can proactively work for a common goal in every execution loop over time" [3]. In this study, we aim to implement an information system that can pave the way towards proactive HRC.

In the context of HRC for component assembly, a cognitive system and mixed reality (MR) are two of a quartet of enabling technologies envisioned by Wang [5] that enhance human abilities for human-centric assembly. The cognitive system is expected to aid the human operator in global decision making in the form of an intelligent multi-agent system to help with the operator's limited cognitive ability, while mixed reality is expected to assist the operator's fading memories of the past and unreliable predictions of the future [5]. The cognitive system itself is not expected to be a real–time system used for, say, robot path planning or collision avoidance, but rather for human–robot relationship management [4]. As such, it offers greater leeway for the choice of knowledge representation mechanisms from a performance

* Corresponding author at: Tampere University, Korkeakoulunkatu 7, 33720 Tampere, Finland.
E-mail addresses: joe.david@tuni.fi, joesd@stud.ntnu.no (J. David), eric.coatanea@tuni.fi (E. Coatanéa), andrei.lobov@ntnu.no (A. Lobov).

https://doi.org/10.1016/j.jmsy.2023.07.013
Received 30 November 2022; Received in revised form 8 May 2023; Accepted 25 July 2023
Available online 10 August 2023

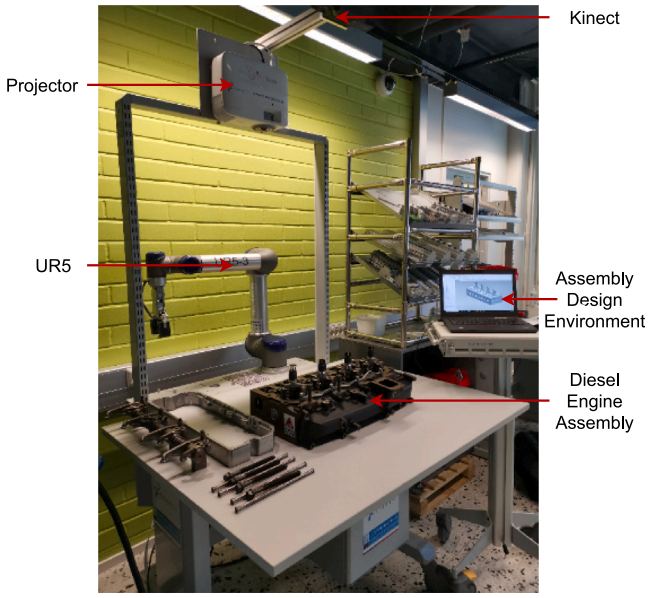


Fig. 1. Research setting.

standpoint [4]. As we see it, the cognitive system should have the means to represent knowledge that capture at least the production context and the mental attitudes (desires and intentions) of the agent with respect to the assembled components that synergistically support the collaborative processes.

The notion of intention is closely associated with cognition. Bi-directional intent[1] recognition is considered to be among the "deepest and most important" [6] research questions faced by HRC systems. Selection of roles for shared tasks at a high level or at a finer level, the goals or intended motion paths may be communicated as intentions [6]. Since the setting in which the work is carried out consists of a projector–camera setup (Fig. 1), of particular interest is the approach to communicate robot intentions as visual cues on the product or the part the operator is collaborating. This has the advantage that it is in the operator's field of view, has the operator's undivided attention, and bears no requirement on the operator's part (e.g., a wearable). Several approaches [7,8] for robot intention projection have been proposed in literature, but these seem to fall short of the self-organisation we intend to achieve as they rely on manual techniques for feature extraction.

As for *self-organisation*, its core tenet is to connect in situation-dependent ways, automatically, without external intervention [9]. From the standpoint of an information system,[2] for HRC, the main variable we intend to attain self-organisation for, is the component assembly (product) and consequently its associated process for assembly. In effect, this means that the cognitive system must support reusable concepts of different component assemblies, and together with the concepts defining the agent (such as desires and intentions), it must ensure smooth collaboration processes in tandem with the MR environment. To this end, the implementation integrates the assembly design environment via a digital thread framework so that the required degree of self-organisation can be achieved with respect to the product models and is what enables the study, in part, to make novel contributions.

As for knowledge representation, we aim to model collaborative assembly as a logical consequence of deliberation between agents. As such, the generalisable logic is to allow collaboration for different iterations of assembly to "play out" differently based on runtime deliberation between agents. In contrast to classical deep learning approaches, we do not use pre-existing data for a reason — to allow them to "play out" each time, such that it is well within the realms of logic to reason on data with well-defined semantics. The cognitive system in our work uses description logic (DL) for knowledge representation as it was deemed expressive enough for the application context while supporting relatively efficient inference. In particular, we use the Web Ontology Language (OWL) for maintaining the knowledge base of agents following a distributed architecture. The MR environment, however, is a projector-based, purpose-built web application that forms intuitive interpretations of the information model underlying the cognitive system and augments the operator's reality to assist with the assembly. It also enables communicating with agents via an underlying MAS framework. As for component- or product-aware visual cues as intentions, the system allows computation of bounding boxes that are overlaid on the physical component as visual cues during assembly using the MR application. The research question we aim to answer from an information model perspective is "What is an information model that maintains and converges upon the common mental attitudes of collaborative agents in a multi-agent system and associated production context?"

The remainder of the paper is organised as follows: Section 2 reviews the state of the art in ontologies, particularly for HRC. Section 3 presents the objectives, setting, and scope of the research. The theoretical foundations underpinning the implementation are presented in Section 4. In Section 5, the approach to knowledge convergence and representation is introduced. Section 6 documents the step-by-step results of following a methodology towards engineering the ontology and evaluating it based on a case study. Results of its application and deployment in an HRC framework in a case study are presented in Section 7. Section 8 analyses the work with respect to the objectives set out and discusses its limitations before presenting directions for future work. Section 9 presents a summary of the contributions of the study in conclusion.

2. Related work

The challenges pertinent to HRC are multi-disciplinary and have benefited from contributions from evolutionaries, bayesians, analogisers, connectionists, and symbolists alike. The work done as part of this study predominantly represents that of symbolists for knowledge representation and reasoning for (intelligent) agents in the manufacturing domain, particularly robotics and HRC. As such, it seems fitting to review (and limit ourselves to) works that use DL (specifically OWL) that cater to knowledge representation and reasoning for manufacturing as well as works of robotics, preferably in human–collocated settings, particularly those of collaborative environments between humans and robots.

Core ontology for robotics and automation (CORA) [10] is the IEEE standard ontology for robotics and automation that uses suggested upper merged ontology (SUMO) [11] as the upper ontology and autonomy levels for unmanned systems (ALFUS) [12] to characterise its autonomy level. RoboDB [13] is a database of descriptions of robot embodiment (body structure, capabilities, etc.) using semantic web (compatible) technologies.

The OpenRobots ontology (ORO) that enables artificial cognition for human–robot interactions has been presented in the work of Lemaignan et al. [14]. The ontology coverage encompasses statically asserted common sense knowledge that is aligned to the upper ontology OpenCyc, and entails rules and class expressions for inferences. Whilst sharing the same common sense knowledge, the robot maintains separate models of the beliefs of the human to enforce what the authors call as "perspective taking" (i.e., maintaining private beliefs of the human's perspective on

[1] Note the use intent and intention synonymously.

[2] Information system here and throughout the article is used as catchall for tools and technologies for accessing, collecting, processing, storing, and distribution of information that includes the information model, mixed-reality environment, ontology, etc.

the state of affairs). A memory profile is possibly attached to assertions in the knowledge base to mimic long- and short-term working memory. Three profiles, viz. short-term, episodic, and long-term, define the lifetime of statements as 10 s, 5 min, or unlimited, which, if elapse, result in their removal. The ORO knowledge base receives multimodal input from communication (via natural language processing) and perception modules that receive inputs from the operator and the environment, respectively, for use by the task planning, execution, and motion planning modules.

KnowRob [15] uses upper-level ontology concepts from OpenCyc [16] in part to develop knowledge bases for robots for the household domain (kitchen assistance robot) and allows for a description of events, spatio-temporal information, objects, actions, tasks, processes, components, and capabilities. The architecture uses the paradigm of the "world as a virtual knowledge base", where information is computed on demand rather than asserting everything known, as in ORO [14]. Three main types of inference exist in KnowRob: the run-of-the-mill DL inference, like in ORO, for deterministic information in the ontology, probabilistic inference using Bayesian logic networks for uncertain information, and "computables" to compute on-demand instances of target classes and the relationship between these instances from perception.

Of late, HRC has seen the use of DL for knowledge representation. Work has focused on creating new ontologies, such as Sharework Ontology for Human–Robot Collaboration (SOHO) [17] and Ontology for Collaborative Robotics and Adaptation (OCRA) [18], due to a lack of concepts that model collaboration in existing ontologies. SOHO builds on DUL, a subset of descriptive ontology for linguistic and cognitive engineering (DOLCE), as its foundation and incorporates SSN [19] and CORA [10] to extend along three contexts; the production, behaviour, and environment. SOHO was built as a domain ontology to formally conceptualise the relationships and properties of collaborative manufacturing scenarios. OCRA is also a domain ontology for collaborative scenarios, that builds on DUL, but has a primary focus on collaboration and plan adaptation.

However, these ontologies were not directly usable for our application for reasons described subsequently. The previously described "general" ontologies, such as CORA and RoboDB are devoid of concepts that describe collaboration. Even initiatives such as KnowRob were not targeted for collaborative environments and miss notions of joint intentions or collaborative (not cooperative) tasks where communication between agents and execution norms of tasks take the centre stage. Furthermore, industrial manufacturing and production settings were not necessarily the goal, and self-organisation from a product (assembly) or "recipe" variety point of view is not the focus. These are, however, an important research question for HRC in an industrial setting [20] and something we attempt to achieve. The notion of capabilities that theoretically motivate HRC is non-existent in OCRA and has rather shallow representations in SOHO and neither supports use in a distributed architecture. The former point is discussed in more detail in Section 8.

Furthermore, we aim to explore mixed reality as the modality of choice (vision) as opposed to speech which may not always be viable in industrial settings. We believe that carefully architected MR interfaces can prove to be effective channels of communication and have the potential to obviate the need for non-ergonomic wearables.

Lastly, except for ORO, they do not espouse a multi-agent philosophy as would economic and engineering rationality require for production environments for HRC [21], where autonomous agents maintain private distributed beliefs and come to a capability-based consensus for shared collaborative tasks through dynamic runtime communication. Just as behavioural and sensorimotor patterns emergent in human-human interaction cannot be reduced to individual accounts, HRC should transition towards a multi-agent approach with embodied intelligence that incorporates human-human interaction dynamics [22]. Many works base their approach on a purely communication standpoint, devoid of the social aspects of teamplay. In contrast, this study aims to address this by taking advantage of the inherent human ability to understand speech acts and emphasises dialogue between collaborating agents to negotiate turns during collaboration.

3. Research objectives, scope, and setting

Exploring dyadic configuration for logic-based formalisms in HRC presents not only a challenge from an ontology design pattern point of view, but its instantiation in a coherent multi-agent architecture also presents a technical and a software architectural challenge. Therefore, the study sets the following objectives:

1. to architect a logic-based information model for HRC for use in a distributed setup that captures representations of the manufacturing, agent and interaction contexts.
2. to develop and interface with 1, a component-driven MR application that supports the foregoing information model and provides the operator with a useful interface for inter-agent bi-directional communication.
3. to support principles of self-organisation and social aspects teamplay as described along the development processes of 1 and 2.
4. to implement a multi-agent system instantiation for HRC that supports 1, 2, and 3 and verify its utility by means of a case study.

The scope of this study is to realise this self-organising team information system between collaborating agents for component assembly by formalising the representation of knowledge for the agents. Thus, the scope is limited to information flows and information modelling. As such, the self-organisation from an information system standpoint is also at the semantic level of knowledge representation (i.e., concepts developed must support reusability across different use cases of the intended application to accomplish the set goals of collaborative assembly). Furthermore, the work does not deal with implementations of real-time systems, such as robot trajectory planning and collision avoidance, but the work may be considered an input to these systems, as justified in Section 1. For example, a task may be inferred as an intention, but it remains out of the scope of this study to check if and how the task is carried out. For the sake of completeness and carrying out a case study, we have implemented a module that executes predefined motion trajectories for the execution of the tasks. However, appropriate interfaces are provided so that they can be integrated with real-time systems that are envisioned to be of a broader architecture left as future work.

The research is carried out in a laboratory environment, as shown in Fig. 1. It consists of a DLP projector (1920 × 1080) and a Kinect camera (RGB-D) mounted atop, and thus have in their fields of view a height-adjustable table that acts as a collaborative working space between a table-mounted UR5 collaborative robot and a human operator. The component assembly use case is that of the assembly of a real diesel engine.

4. Theoretical foundations

4.1. A multi-agent view

Collaborative scenarios between robots and humans can be seen as a team working on a common task or towards a common goal [23]. These tasks may require various degrees of autonomy that are at or anywhere along a spectrum that spans between the ends of human-only and machine-only [21]. Such symbiosis between humans and robots that is characterised by autonomy fits well [21] into the abstractions of the distributed artificial intelligence paradigm of multi-agent systems [24] that forms a sound foundation for the design of intelligent

systems [25]. Woolridge and Jennings [26] highlight three desirable characteristics of intelligent agents, viz. *reactivity*, which allows a timely response to events, *proactiveness*, which allows the exhibition of goal-directed behaviour, and *social ability*, which allows interaction with other agents.

As evident from the literature in the previous section, most studies on multi-agent systems in the domain of HRC either ignore or downplay the social aspect as a mere communication problem that caters to the robot's reactivity to specific tasks or events. We posit, similar to Li [4], that attaining human centricity in collaborations involving humans would greatly benefit from greater social aptitude and a proactive attitude from the robot during interactions and is yet to receive due attention.

To this end, *self-organising teamwork* has been identified as the "central brain" [4] in enabling proactive HRC and helps the collaborating agents to assume roles based on their capabilities at runtime. This is accomplished via federated knowledge convergence mechanisms of data in decentralised HRC systems within and between factories [4]. As such, systems that accomplish this are not solely used for time-critical applications, such as robot path planning or decision making, but to serve as crucial inputs to these [4]. We dissect *self-organising teamwork* as two concepts of *self-organisation* and *teamwork* and treat them separately in our approach for HRC in the next section, based on existing theories in the literature that is subsequently presented.

4.2. Self-organisation

The basic tenet of self-organisation is increased autonomy and reduced dependencies, with the objective of fast response in dynamic contexts [9,27]. Lu et al. [9] define self-organisation as "...connecting in situation-dependent ways that can change their internal structure, organisation, and functions with minimum external intervention to achieve optimal manufacturing operations and system performance in response to unforeseen conditions and evolution along time".

From the targeted scope of an information system standpoint, our perspective on self-organisation tailored to the context of human–robot collaborative assembly is as follows. Robot agents are automatically configured to handle different component assemblies and adapt to them in a manner that satisfies the overall objectives of the said component assemblies. Such a requirement was even reported as necessary for collaborative assembly applications in a recent interview that was conducted with automation engineers and assembly operators [20]. This calls for the representation and reasoning of the underlying information system to be semantically and syntactically generalisable and accommodating to different products, processes, and resources. Operator assistance systems would similarly follow suit and adapt accordingly by modifying assembly instructions, for example. Such self-configuration also means that there must be a provision for agents (robot or human) to join and leave at will or unexpectedly with minimal consequence on the assembly process. As agents become available and unavailable, the team must self-optimise in that they should take stock of the capabilities of available agents, acknowledge the non-participation of unavailable or disagreeing agents, and accordingly manage the collaboration. However, since the implementation of real-time systems that monitor the execution of tasks and deviations is outside the scope of this study, we do not implement feedback systems that can detect anomalies to optimise the behaviour of collaborating agents, which admittedly would enable a higher degree of self-organisation. Agents are trusted with tasks and are expected to communicate truthfully regarding failures or completions.

4.3. Teamwork

Humans are social beings that sustain and thrive on inter-personal relationships with fellow peers, and few notable achievements, if any, have been accomplished without them. The case can be thought to be no different for human–robot collaborative relationships, and a sound motivation for HRC has already been presented in Section 1. However, to endow a sense of belongingness, which is needed in a human-centric manufacturing context [3], our approach takes the view that humans' collaboration with their machine counterparts should be designed as a social interaction. This social interaction, in principle, might not only foster a sense of understanding and safety and enable a closer relationship but might also consequently promote their wider acceptance.

To endow a sense of being part of a team, one approach is for these social interactions be expressive of tasks as *desires* based on the current *beliefs* of the agent that could potentially transform into *intentions* for execution. The expectation is that using notions of folk psychology, such as *beliefs*, *desires*, and *intentions* (BDI), to model the *mental attitude* towards collaborative assembly would be intuitive to humans and thus help reduce the ambiguities in communication involving them, especially when used with supportive interaction modalities. The BDI framework is a conceptual framework for developing intelligent agents and, in general, the intentional stance is an abstraction tool in computing to explain, understand, and, crucially, program complex computer systems [28, p. 31]. The paradigm has been in use since several decades since its inception. More mature implementations include the procedural reasoning system [29,30], which has been used to successfully implement systems that tackle air traffic congestion (OASIS [31]) and air mission modelling [32] (dMars [33]). The next subsection gives a brief description of the mental attitudes adopted in this study.

4.3.1. Beliefs, desires, and intentions

Beliefs are the "informative component of the system's state" [34]. The system includes the world that the agent inhabits and the agent itself. Beliefs generally include only the domain-dependent abstractions of the important properties of those aspects relevant to the agent's design objectives [35]. The notion of *capabilities* is closely associated with agency. The treatment of *capability* as a *mental attitude* is debatable [36]. However, an agent would not decide to pursue anything that it *believes* it is incapable of, which could be perceived as a *belief* in an agent's ability to be useful and is the direction chosen here.

Desires represent the motivational attitude of the agent regarding what it wishes to achieve or what states of the world it wishes to create. Hence, desires act as the driver for actions [35]. However, the agent may not necessarily be able to achieve everything it desires, either due to it not aligning with its prior commitments or due to the lack of capabilities required for it. The former can be considered the class of *conflicting desires*, while the latter, the class of *non-achievable desires*.

Intentions form the class of *achievable* and *non-conflicting* desires. In other words, although the agent *desires* to create different states of the world, it can pursue a sub-set of these as its *intentions* based on its capabilities and its runtime commitments. *Intentions* are treated as separate mental states and are those that the agent intends to pursue upon deliberation with the collaborating agent. Thus, communication acts as the activity that could potentially transform the desires of an agent into the intentions that it commits to.

4.3.2. Joint intentions, commitment, plans and action

The elegant structure and use of folk psychological notions of mental attitudes as beliefs, desires, and intentions are expected to make execution flow logically intuitive to the collaborating human operator, which could contribute to effective collaboration. The notion of intention plays a distinct role and is treated on par with beliefs and desires. This allows for modelling these *attitudes* based on the *deliberation* process (on what to achieve) of practical reasoning [37], the results of which are intentions that lead to actions.

We draw some inspiration for modelling *joint intentions* from Cohen and Levesque's joint intention theory [38], which informs joint intentions as a "joint commitment to perform a collective action, while

in a certain shared mental state". We see these shared mental states as *individual intentions* that represent the same task that the agents in question have mutually committed to and are thus *joint intentions*. *Commitments* are agreements modelled based on *communication*, i.e., the runtime messages exchanged that are pertinent to the (collaborative) task.

Once commitments are made, intentions are pursued by executing *actions* which are part of the body of a *plan* [34] that accomplishes an intended goal and thus drives means-end reasoning (the process of how to achieve intentions) [37]. The preconditions under which a plan may be adopted are defined by the head part of the plan [34]. Individual actions are representative of the collective plan that is fully defined with *collaboration modalities*. Each plan has actions that execute tasks in one of three collaboration modalities: *independent*, *synchronous*, or *concurrent*. Actions with the *independent* modality execute separate tasks without any collaboration. Actions of a *synchronous* modality execute tasks sequentially, i.e., one is done after another. *Simultaneous* modality actions execute tasks at the same time.

This section presented the theoretical foundations that underpin the concepts defining the attitude of the agents towards collaborative assembly as beliefs, desires, and (joint) intentions. The notions of commitments, plans, and actions are also discussed. We build on these foundations in Section 6 after presenting the overall approach in the next section.

5. Approach

Insights from literature [4] into attaining self-organising teamwork for proactive HRC guide us towards a two-step approach: (i) knowledge convergence from HRC systems and (ii) knowledge representation for non-real-time human–robot relationship management. The approaches for the two as adopted in this study are subsequently detailed.

5.1. Knowledge convergence by realising an HRC digital thread framework

Our approach towards attaining self-organisation for HRC (as described in Section 4.2) entails architecting and implementing a digital thread framework that is purposed with knowledge convergence from siloed information systems in the extended environment, which the collaborative environment is part of. Kraft's comprehensive definition of the digital thread is as follows "an extensible, configurable and agency enterprise-level analytical framework that seamlessly expedites the controlled interplay of authoritative data, information, and knowledge in the enterprise data- information-knowledge systems…to inform decision makers throughout a system's life cycle by providing the capability to access, integrate and transform disparate data into actionable information". Drawing a parallel to HRC, authoritative data in systems corresponds to product, process, and resource models; decision makers include the robot and operator; and the capability to access, integrate, and transform disparate data into actionable information describes the utility of the framework, the underlying information models that support seamless integration, and operator support systems that aid collaboration. Lu et al. [9] opine that such a manufacturing digital thread works upstream and downstream between phases of the product lifecycle (design, manufacturing, inspection, etc.) via standard interfaces and is expected to enable informed design and collaborative manufacturing.

Our earlier works [39,40] developed an agent-oriented digital thread framework for HRC. The framework exploits the flexibility offered by the paradigm or "technology" [41] of knowledge-based engineering systems to build a digital thread [42] between the assembly design and the physical assembly environment. Using the framework, the robot and operator agents have the means to retrieve the required information for the assembly dynamically and on demand at runtime. These may include assembly instructions and product and manufacturing (PMI) data, among others, and use the KBE software API to define

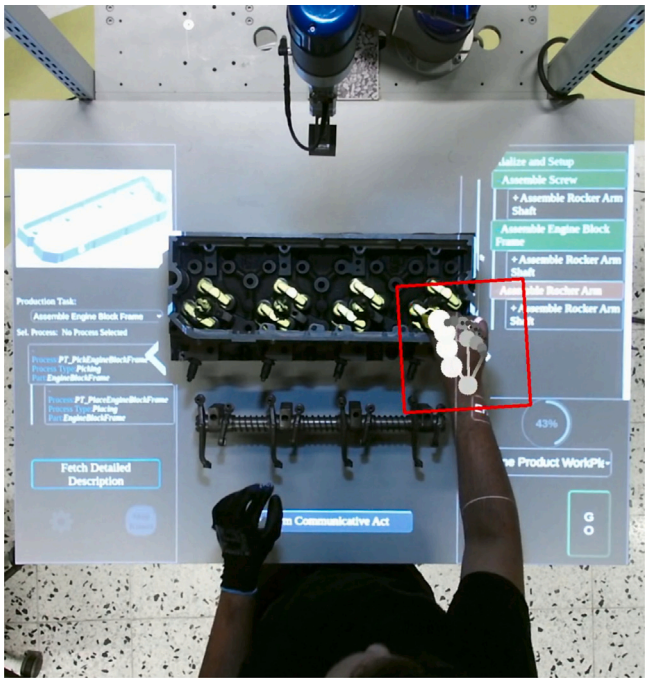


Fig. 2. Mixed-reality interaction with the digital thread framework.

software interfaces to the supportive MR model in a product-neutral (therefore accommodating product variety), or rather product-aware, manner that builds the foundation towards self-configuration as we first defined in Section 4.2. Furthermore, the Java Agent Development Framework (JADE) [43], an agent-oriented middleware, addresses the domain-independent issues of self-configuration in an agent-oriented environment that allows agents to join and leave at will while allowing them to be discovered through yellow/white pages services and communicate via standardised interaction protocols. Although, JADE was developed circa 2000, it is still maintained. It was the result of several years of development that has had four major releases and is therefore quite stable and allows for future maintenance via an LGPL licence.

The framework includes an MR application (Fig. 2) that builds on JADE to enable explicit real-time dynamic interactions between human and robot agents involved in collaborative assembly (Fig. 2) while also functioning as an operator assistance system. The MR environment prototyped as part of our earlier work [40] has had its existing functionality extended and improved as part of this study to support the rich vocabulary of the engineered ontology in this study, which was not envisioned at the time of its conception. The earlier developed application quickly grew unwieldy as the vocabulary and features became richer and more complex and state management became a problem of its own. Therefore, the source was rebuilt using a component-based architecture that uses the popular ReactJS [44] library to facilitate its sharing, reusability, and extensibility, while Redux [45] was used for component state management. In essence, the MR environment consists of a purpose-built web application that is projected onto the shared workspace between the operator and a robot. The operator interacts with the interface using a ring mouse that effects clicks on the interface components. A precomputed homography maps the operator's hands, which are detected using a machine-learning hand-detection solution [46] to mouse coordinates. More details on the MR interface are presented in Section 7.2, which details the case study where it is employed.

5.2. Knowledge representation

Our earlier works laid the technological foundation as infrastructure for information retrieval and access by means of a framework [39] and

an MR application [40]. However, attaining self-organisation for our use case in HRC collaborative assembly would not be possible without a concrete internal agent architecture that characterises social aspects of teamwork and manages the collaborative assembly of a product in a self-organised way. Therefore, this study contributes an information model that is instantiated in the software models that define the agents and allows it to be employed in a use case of a collaborative product assembly.

It is important to revisit the notion of self-organising teamwork and how it influences the choice of knowledge representation mechanisms. Firstly, it is not a real-time system but provides inputs to the real-time systems (e.g., robot motion planning) [4]. As for what these inputs are, our approach entails defining the state of the assembly, which would then serve the necessary production tasks, processes, and assembly part in question as inputs to the real-time systems. For example, a specific assembled state of the component assembly would dictate that the next possible state(s) correspond to the assembly of, say, exactly three parts following corresponding pick-and-place assembly tasks. The real time systems, for example, object-detection and robot-motion-planning modules, can then function with respect to these parts alone, and operator guidance systems may present these as "options" with corresponding instructions to the operator. The system was modelled as a finite-state machine. Specifically, we model the system as a subset of a Petri Net as described next. As with workflow nets, these PNs have a single input place (source) without a prior transition and a single output place (sink) with no subsequent transitions. Furthermore, it is *safe* (1-bounded, i.e., does not contain more than 1 token in all reachable markings) with each place having exactly one incoming arc and exactly one outgoing arc. This allows for *concurrency* but not *conflict* in PN terms. With such net systems, the sequential and parallel interdependencies of sub-assembly tasks can be naturally captured in a simple and intuitive manner and have already been widely used in literature to represent component assembly. These can be represented in a standardised syntax of Petri Net Markup Language (PNML) (PNML, ISO/IEC 15909 Part 2) [47]. Representations that follow a standard enable the interoperability that is required for self-configuration and self-organisation from a knowledge representation standpoint.

Furthermore, *self-organising teamwork* should allow the robot and operator to take up manufacturing tasks or activities if they are qualified to do so based on their capabilities, and it should allow changing roles on the fly [4]. For component assembly, such a representation should thus at least include representations of the tasks associated with the assembly of the sub-assembly parts (e.g., pick/place, grasp/release, etc.), necessary representations of the sub-assembly parts that feed any real-time systems (e.g., a point cloud profile that could be useful for object detection), a representation of agent capabilities that defines an agent's ability to do a task, and mental attitudes of the agents that are reflective of their behaviour and allow carrying out the tasks collaboratively towards accomplishing the overall goal of component assembly.

Robotics applications generally make use of imperative languages, such as C/C++, JAVA, etc., that code logic in traditional IF/THEN statements and loops. In such cases, the knowledge is usually "hard-wired" and lost between program code [48–50]. However, for collaborative multi-agent robotics, formally representing knowledge explicitly in a knowledge base (KB) has the advantage that this knowledge is not bound to a specific agent and may be reused for several agents possibly for different domains and consequently results in less overall program code and maintenance. Furthermore, the core tenet of self-organisation is to adapt to changing scenarios, and in this case, changing assemblies, components and behaviours of the collaborating agents, which thus requires the flexibility to create, update, and maintain knowledge as private beliefs in distributed knowledge bases. Thus, the flexibility offered by formal knowledge representation is better suited to the needs of self-organisation and is the direction chosen in our work.

In the domain of distributed multi-agent systems, *ontologies* play a key role in developing such knowledge bases by conceptualising a domain. "Conceptualisation" refers to an abstract model of the key concepts and their relationships that characterise a phenomenon of interest [51], and an ontology is "a formal explicit specification of a shared conceptualizaton" [52]. When the terminologies (concepts) of an ontology are augmented with assertional knowledge (individuals or instances) that instantiate a domain of interest, the touted "intelligent" agents materialise as the formal semantics allows for automated reasoning.

Logic-based formalisms, specifically description logics are preferred for authoring ontologies for the following reasons:

(i) Formal logic can be used to formulate relationships that describe the relationships within an assembly and model their interplay with the social aspects of teamwork that embody conversational elements, in a logically intuitive manner.
(ii) DL is formal with clear semantics and manages the trade-off between expressivity and reasoning complexity well. Although the study does not develop real-time systems, the system is to be reasonably performant.
(iii) Formal semantics allows for automated reasoning and the inference of implicit data from asserted facts. Inferring relevant states of affairs from asserted conversational elements may be achieved using the notion of classification (i.e., if a class is a sub-class of another) and subsumption (i.e., if an individual is a member of a class) by running inference procedures on logic-based representations. As such, the rational behaviour of collaborating agents may be imitated by checking the consistency of an agent's beliefs, commitments, and capabilities by DL reasoners when appropriately implemented.

This section presented our approach towards developing an information system for HRC that involves knowledge convergence via a digital thread framework and the use of ontologies for developing the knowledge bases of the agents. The next section details the results of an ontology engineering methodology that developed such an ontology for this study.

6. Ontology engineering

Engineering the knowledge base (KB) involves formally creating an explicit specification of the key concepts and their relationships, which are known as ontologies in computer science parlance, that make up the domain of multi-agent collaborative component assembly, as described previously. An ontology engineering methodology provides step-by-step guidelines for the development of such concepts and helps manage their complexity [53,54]. This study uses the methodology presented by Sure et al. [54] in developing the ontology and involves five phases: (i) feasibility, (ii) kickoff, (iii) refinement, (iv) evaluation, and (v) application and evolution. Their outcomes are detailed in the following subsections.

6.1. Feasibility study

The motivation that advocated the development of the ontology was detailed in Section 5.2. During the feasibility study, its scope from an information modelling point of view is defined based on the application scope defined earlier.

The ontology is developed in the context of a distributed human–robot environment for a collaborative component assembly use case with an emphasis on *self-organisation* and *teamwork*. As such, the scope from a knowledge representation viewpoint encompasses the representation of the component assembly, the assembly processes, and their interplay with social interactions of a human–robot team as logical definitions. The aim is to achieve self-organisation, and that means

that the ontology concepts entailed in the aforementioned representations must be independent of the dynamic components of an HRC environment, i.e., the product to be assembled and the behaviour of the collaborative agents. Thus, the product representations must follow a product-neutral and product-aware method that dovetails with the interplay between collaborative interactions and teamwork. Here, product neutrality means that the concepts of assembly must be generalisable to any component assembly, with respect to both the subassembly parts and the processes entailed in assembling them. Product awareness means that functions/activities that depend on the product features and/or geometry must work across all subassembly parts.

Therefore, at this stage, we define three contexts for the ontology: (i) the *manufacturing* context (MCx) that includes the product, process, and resource descriptions, (ii) the *agent* context (ACx) that defines the *mental attitude* which defines the agent's motivational and intentional stance with respect to the assembly tasks, and (iii) the *interaction* context (ICx) that defines the message exchanges and consequent commitments. The agent and interaction contexts together define the social aptitude of the agents, i.e., the "capacity to encode and interpret social cues to draw inferences about other people's beliefs and intentions" [55].

Here, we make a final note on the intended type of ontology to be developed based on Hejst et al.'s classification [56]. Ontologies may be classified along the "amount and type of structure" dimension as *terminological*, *information*, and *knowledge modelling* ontologies and along the "subject of the conceptualisation" dimension as *representation*, *generic*, *domain*, and *application* ontologies. It is the intention of the authors to develop an *application* ontology for the specific tools and technologies employed in the framework described in Section 5.1 and a *knowledge modelling* ontology with logic formalisms to capture necessary semantics. Although application ontologies are not intended to be directly reusable, we discuss how CAMO may be reused in Section 8.

6.2. Kickoff

The kickoff phase initiates the actual ontology development, starting with a description of what the ontology should support depending on its application context in the form of an *ontology requirements specification* [54]. This, as noted by Sure et al. [54], is expected to guide the authoring process in the inclusion and exclusion of concepts and relationships while looking for potentially existing reusable ontologies. The outcome of this stage *is a semi-formal description* of the ontology. In the following subsections, a detailed account of these two steps is given.

6.2.1. Ontology requirement specification

The requirements are further clarified by elucidating the following: (i) the aim and use of the ontology, (ii) the information provided by the ontology, (iii) the users of the ontology, and (iv) the questions that the ontology must be capable of answering, i.e., the competency questions.

(A) **Aim and use of the ontology:**
The aim and use of the ontology to be developed are best clarified by its use case. The ontology is intended to be used as an application ontology that is to be instantiated as a distributed knowledge base for a human–robot team that is characterised as a multi-agent system engaged in collaborative assembly within the context of an existing digital thread framework that integrates the assembly design environment. The ontology has the following aims:

(i) to form representations of agent beliefs that logically dovetail the manufacturing, agent, and interaction contexts as envisioned in the feasibility stage (Section 6.1).
(ii) to infer the social aptitude of the agent by reasoning over the represented knowledge in the foregoing contexts.
(iii) to support self-organisation as described in Section 4.2 in the context of human–robot collaborative assembly.

(B) **Information provided by the ontology:**
The information that can be obtained from the ontology by direct query or by running inference procedures is as follows:

(i) the assembled state of the component that defines the production tasks and entailed processes (MCx) for a given assembly.
(ii) the sequence of and inter-dependencies between these tasks and processes (MCx).
(iii) the informational, motivational, and intentional stance of the agents with respect to these tasks (ACx).
(iv) the capabilities of the agents and resources and those required by the tasks for their execution (MCx, ACx).
(v) information regarding plans, actions, and any collaboration modalities resulting from commitments made to execute the task (ICx).

(C) **Users:** The users of the ontology are divided into three categories:

(i) *Producers of assertional information*: The assertional knowledge (ABOX) of the product is expected to be populated automatically by external applications. Relevant GD&T information is obtained via "KBE technology" of the CAD software as presented in Section 5.1 and populated automatically. To facilitate easy population of the other information of the product work and process plans, the representational schema (TBOX) must be compatible with prevalent standards. With the existing framework infrastructure, it should allow for easy and automatic population by mapping process and resource information from enterprise systems to their corresponding resource and process information constructs within the knowledge base via suitable adapters, possibly with intuitive user interfaces.
(ii) *Consumers of the information*: The consumers of the information as envisioned are the collaborating agents, i.e., the robot, and the operator, and the product designers.
(iii) *Maintainers of the information*: The information model would ideally be maintained by the knowledge engineer in the organisation that extends the TBOX with concepts useful in the organisation, following a methodology similar to the one undertaken in this study.

(D) **Key competency questions:** The answers to the following questions are pivotal for the agent to perform its core functions and support the deployed mixed-reality environment. The developed information model is checked for its ability to answer these questions in the evaluation phase, which also documents the method of information retrieval (Table 1). For any question, the word "*given*" is used to define the input to the query and is something the agent requires information about. The rationale for the questions is provided along with the questions. To be concise, trivial questions are avoided.

(i) *What are the available tasks for an agent?*
Agents exhibiting goal-directed behaviour should be able to gather the tasks that they need to perform so that they can accomplish the overall goal of collaborative assembly.
(ii) *What is the motivational and intentional stance that defines the mental attitude of the agent towards the available tasks?*
Characterising the mental attitude of the agent with respect to the assembly tasks is important when coordinating the tasks between them and preventing turn-taking behaviour.

(iii) *What is the effect of carrying out a given task on the state of the assembly process?*
Once the agent has carried out a task, a belief update operation must yield the subsequent tasks to be performed, respecting their inter-task dependencies and pre-requisites.

(iv) *What are the primitive tasks (functions) that constitute a given process? In what order are they to be executed, and under what execution norms?*
The decomposition of a process into primitive tasks or functions is necessary for "division of labour" during collaboration. The sequence and norms of functions that a *given* process is made of are necessary for an agent to know to generate a plan that has actions that execute all primitive tasks that fully define the *given* process. Its knowledge also helps initiate appropriate communication.

(v) *What plans, if any, does an agent have for a given process?*
Plans contain actions that are needed to execute a *given* process.

(vi) *Is an agent capable of performing a given process? What capabilities are required?*
Every agent has resources that give it capabilities, and it is necessary for the agent to know about them so that it can commit to a task that requires a certain capability to perform it. For example, a robot has a gripper that gives it the capability to grasp and release certain kinds of objects for a pick-and-place task.

(vii) *Are all pre-requisites for a given production task fulfilled?*
Understanding the pre-requisites of a task is necessary to carry out the given task successfully. For example, regarding assembly, a part might need to be assembled on top of another sub-assembly part.

(viii) *What are the human-readable instructions for a task in a language familiar to the operator ?*
Collaboration with a novice operator would be more efficient and safer if the operator was better aware of his or her responsibility towards accomplishing the task.

6.2.2. Determining the essential concepts and their relationships

This step involves creating a conceptual model of the important concepts and their attributes and relationships for detailed refinement in the subsequent stage. The primary concepts belonging to the production, interaction, and agent contexts are introduced, building on the foundations introduced first. For better readability, Manchester OWL Syntax [57] is used.

(A) **Foundations:** The developed ontology builds on DUL [58], a lightweight ontology that combines theories from two foundational ontologies, namely DOLCE [59] and "description and situation"[60], along with concepts from other ontologies spanning concepts of plan, information and collection ontologies [58] . Foundational ontologies is used with the objective of facilitating reuse, enhancing reliability, and making them well organised [61]. The vocabulary for capability and processes uses existing works in literature [62].

(B) **Manufacturing context**
We introduce a product physical object (`ProductPO`) as a specialisation of the `DUL:PhysicalObject` to categorise all product-related concepts introduced herein. This also makes definitions of other manufacturing-related concepts more intuitive and necessarily restrictive, as will soon be shown.
A part (`Part`) is a "leaf" *physical product object* that cannot be decomposed in the context it is used in and thus is characterised only by its shape and size. For reasons that will be made clear in Sections 5.1 and 7.2, we are particularly interested in the minimum bounding box that encloses a part.

```
Class: Part

SubClassOf: ProductPO
SubClassOf: hasBoundingBox exactly 1 BoundingBox
```

An assembly is a *physical product object* that is composed of two things that is any combination of only another assembly or a part.

```
Class: Assembly

SubClassOf: ProductPO
SubClassOf: DUL:hasComponent min 2 ProductPO
SubClassOf: DUL:hasComponent only (Assembly
                                   or Part)
SubClassOf: DUL:hasComponent some (Assembly
                                   or Part)
SubClassOf: hasBoundingBox max 1 BoundingBox
```

A product is the produce of (value to) an enterprise that can be composed of at least of an assembly or a part. The realisation of a product follows a product workplan (`ProductWorkPlan`) (or work instructions), which is the sequence of steps that are production tasks (`ProductionTask`). These tasks contain one or more processes (`ptm:Process`) that are executed according to a process plan (`ProcessPlan`). Note the process vocabulary `ptm` is a process taxonomy model from literature [62].

```
Class: Product

SubClassOf: ProductPO
SubClassOf: DUL:hasComponent min 2 ProductPO
SubClassOf: DUL:hasComponent only (Assembly
                                   or Part)
SubClassOf: DUL:hasComponent some (Assembly
                                   or Part)
SubClassOf: hasProductWorkPlan min 1
                               ProductWorkPlan
```

The product workplan is modelled as (a subset of) the petri net that includes a set of *production tasks* and corresponding intermediary states. Fig. 3 shows the relationships between the core PN concepts discussed (in blue) and a petri net equivalent representation (Fig. 4).[3] Each "place" of the petri net is characterised by activities which are *production tasks*, represented by PN "transitions" that are connected to outgoing arcs from the said "place". Both outgoing and incoming arcs of transitions are modelled as simple OWL object properties (`leadsTo` and `includesActivity`, respectively) as they do not have any corresponding weights in PNs. *Production tasks* themselves are represented by a "nested" PN to define *processes* according to a *process plan* (`ProcessPlan`) with similar intermittent *process states*. A boolean attribute `hasToken` keeps track of the existence of a token in "places" that "enables" corresponding transition(s), be it either *production tasks* or entailed *processes*. The agent transitions between the product and process states by "carrying out" the production tasks and processes that are represented by these "transitions" and thereby effecting a change in the product's assembled state as denoted by the marking of the PN.

(C) **Interaction context:**
The interaction context represents communication (message exchanges) and commitments (agreements). The agents use the

[3] Due to a more intuitive representation, a similar visual representation is preferred over a graph of nodes and edges to present the case study in Section 6.4.1.

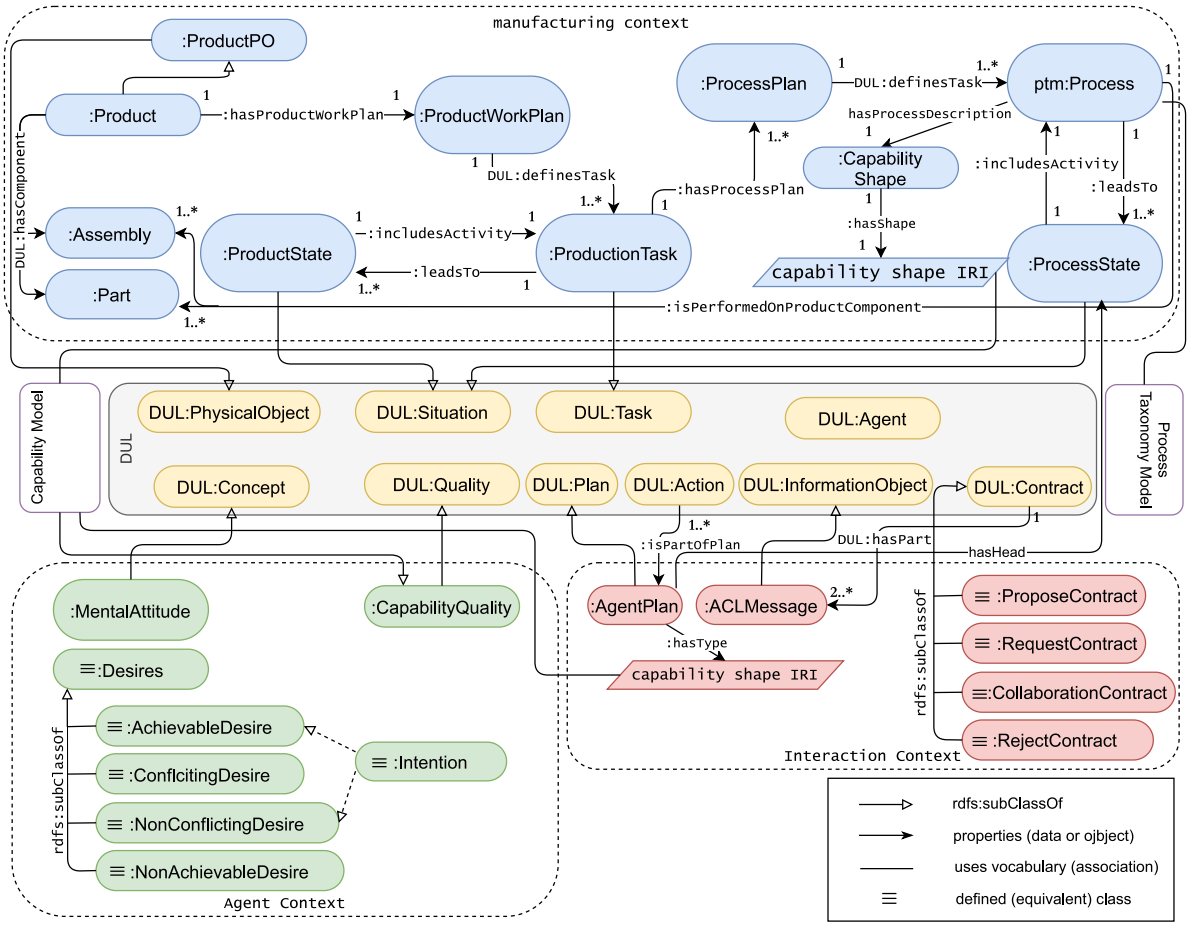


Fig. 3. Key concepts and relationships that models the manufacturing, agent, and interaction contexts during the kickoff and refinement phases of ontology engineering.

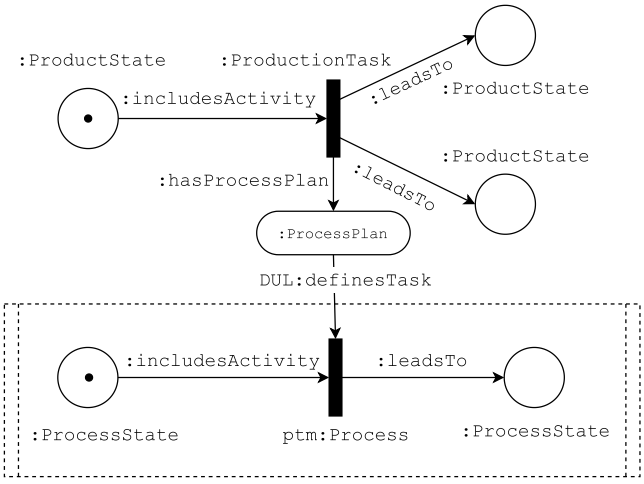


Fig. 4. Conceptual petri net equivalent representation. Production Tasks follow a process plan that defines processes and process states as "nested" nets.

JADE framework to communicate using standardised interaction protocols (IPs). These IPs define message sequences and, for each message, a communicative act (also known as performatives, such as *request*, *query*, *propose*, etc.) among others, as defined by FIPA [63] using an agent communication language (ACL) (`ACLMessage`). An `ACLMessage` is defined by eight parameters (OWL data properties): `sender`, `receiver`, `content`, `performative`, `in_reply_to`, `reply_with`, `protocol`, and `conversation_id` as defined by JADE.

A collection of message exchanges that are part of a conversation and follow a protocol are commitments made and stored as a *contract* (`DUL:Contract`). These specialise as defined classes `AcceptContract`, `ProposeContract`, `RequestContract`, `CollaborationContract` and `RejectContract` based on the performative and/or interaction protocol which the messages are a part. Note that disagreements (such as a *reject* or *cancel*) are also interpreted as contracts as an agreement to disagree.

When a contract is recorded, it is always defined with respect to a production task indicated with the `isTaskDefinedIn` object property. Since communication is about existing tasks, the concepts of the agent context can use the interaction context to model the mental attitudes that influence agent behaviour as shown next.

(D) **Agent context:**

Based on concepts of beliefs, desires, goals, and intentions, which were introduced in Section 4.3, the agent context describes the semantics of their instantiations in the knowledge base.

Beliefs are the "*informative component of the system's state*", according to the notions of Rao and Georgeff [34]. Since we follow a distributed architecture, each agent is considered to have a private belief about everything instantiated in the knowledge base. As such, the notion of belief is "passive" and contains beliefs about capabilities, plans, actions, and so on. That is, we do not use reification to model the propositions of belief that do or do not hold true at any given instant to limit the number of propositions that could otherwise affect reasoning performance.

Such propositions automatically and quite conveniently rely on the rules of the petri net formalism (e.g., an enabled transition). *Desires* represent the motivational state of the agent and are those tasks that the agent would like to perform but has yet to commit to. To this end, we want the agents to desire to do the available *production tasks* associated with the assembly.

```
Class: Desire

EquivalentTo:

    ProductionTask and (isActivityIncludedIn some
    (ProductState and (hasToken value true)))
```

Drawing inspiration from [50], there are four sub-classes of desires modelled. These are `AchievableDesire`, `ConflictingDesire`, `NonAchievableDesire` and `NonConflictingDesire`. The achievability of a desire is premised on the agent's capability to do a task. Thus, we have desires that are achievable with a capability defined as achievable desires.

```
Class: AchievableDesire

EquivalentTo:

    Desire and
    (isAchievableWithCap min 1 CapabilityQuality)
```

The above `isAchievableWithCap` axiom is populated at runtime to account for changing capabilities. More on the capabilities of an agent and what is required to do a task will be dealt with in the subsequent refinement stage (Section 6.3).
When agents communicate, they store an `ACLMessage` *individual* for every message exchanged, which is part of a type of `Contract` based on the protocol used. Knowledge of the well-defined semantics that the protocols follow, such as the type (performative) and sequence of message exchanges, allows us to define the descriptions of classes that define other subclasses of *desires*. For `NonConflictingDesire`, the definition is as follows:

```
Class: NonConflictingDesire

EquivalentTo:

    Desire and (isTaskDefinedIn some
    (ProposeContract and (hasPart some
    (ACLMessage and ((performative value
                          "ACCEPT-PROPOSAL")
    and (receiver value "Robot"))))))

EquivalentTo:

    Desire and (isTaskDefinedIn some
    (RequestContract and (hasPart some
    (ACLMessage and ((performative value
    "AGREE") and (sender value "Robot"))))))

EquivalentTo:

    Desire and (isTaskDefinedIn some
    (RequestContract and (hasPart some
    (ACLMessage and ((performative value
    "REFUSE") and (receiver value "Robot"))))))

EquivalentTo:

    Desire and (isTaskDefinedIn some
    (RequestContract and (hasPart some
    (ACLMessage and ((performative value
    "FAILURE") and (receiver value "Robot"))))))

EquivalentTo:

    Desire and (isTaskDefinedIn some
    (ProposeContract and (hasPart some
    (ACLMessage and ((performative value
    "REJECT-PROPOSAL") and
    (sender value "Robot"))))))
```

In layperson terms, this means that *non-conflicting desires* are desires (desirable production tasks by definition) which the agent in question has proposed that it perform or has accepted another agent's request to do so. Conversely and intuitively, if there is a refusal or failure to honour a request or a rejection of a proposal that is initiated by the agent, this too qualifies as a non-conflicting desire. Note that since non-conflicting desires are, in principle, desires, based on the previous definition, they only apply to the currently available production tasks that respect the rules of PNs (i.e., production tasks represented by transitions that are enabled). Furthermore, note that the description entails value restrictions along the property `receiver` and `sender`, and each agent has its own agent identifier as its value (the case above is shown for the robot). The reasoning engine that each agent contains separate instantiations of, can then infer subjectively for each agent, different other subclasses of desires and, further, *intentions*, as will soon be shown.
When we scale to additional robots and operators, it is easy to see that the core principles still work. Each agent will have a private knowledge base with the above definition (for the concept `NonConflictingDesire`), with the agent identifier as the value for sender and receiver in the aforementioned definition. For example, if a single robot calls for a proposal from two operators (O1 and O2) and both operators agree, the robot then chooses between the two based on some internal logic (e.g., total operator workload) and approves the proposal of one (O1) and rejects the other (O2). Consequently, the presence of the "ACCEPT-PROPOSAL" (as an attribute of the message exchanged with the robot) in the knowledge base of O1 causes the corresponding task to be inferred as a non-conflicting desire via logical reasoning. The opposite is true in the case of O2 where the task is inferred as a conflicting desire due to the presence of "REJECT-PROPOSAL" (as an attribute of the message exchanged with the robot).
Similar descriptions exist for `ConflictingDesire` and `NonAchievableDesire` and have been omitted for brevity. The former is defined as desires that the agent has *requested* of another agent to be done or that were *proposed* by another agent to be done. The latter is defined as desires that are complete (each agent sends a message with the performative *inform* to inform of task completion) and thus are no longer achievable.
Intentions specialise desires as those that are *achievable* and *non-conflicting*. The definition is as follows:

```
Class: Intention

EquivalentTo:

    AchievableDesire and NonConflictingDesire
```

In summary, although the agent desires to carry out production tasks that lead to component assembly, it can pursue a subset of these as its intentions based on its capabilities and its runtime commitments.

```
camo:SHP_MOV_RockerArmShaft
    a sh:NodeShape;

    # targets instances of the specific capability
    sh:targetClass cm:Moving;

    # capability property shapes
    sh:property camo:PayloadShape;
    sh:property camo:HumanGuidance;

    # instance property shape
    sh:property [
        a sh:PropertyShape;
        sh:path rdf:type ;
        sh:hasValue cm:Moving ;
        sh:order 2;
        sh:group camo:SimultaneousModality;
        sh:name "Approach";
        sh:description "Move the part to the desired
        location"@en-US;
    ] .

    camo:PayloadShape
        a sh:PropertyShape;
        sh:path cm:payload ;
        sh:minInclusive 4.

    camo:HumanGuidance
        a sh:PropertyShape;
        sh:path camo:type;
        sh:description "defines the requirement of the
        operator guiding a robot into place while on
        force mode"@en-us;
        sh:hasValue "Human Guidance".


camo:SHP_MOV2_RockerArmShaft
    a sh:NodeShape;

    sh:targetClass cm:Moving;

    sh:property camo:PayloadShape;
    sh:property [
        a sh:PropertyShape;
        sh:path rdf:type ;
        sh:hasValue cm:Moving ;
        sh:order 4;
        sh:group camo:IndependentModality;
        sh:name "Extract";
        sh:description "Move the part to the desired
        location"@en-US;
    ] .

camo:SHP_GRA_RockerArmShaft
    a sh:NodeShape;

    sh:targetClass cm:FingerGrasping;

    sh:property camo:PayloadShape;
    sh:property [
        a sh:PropertyShape;
        sh:path cm:gripperType ;
        sh:hasValue "parallel";
    ] ;
    sh:property [
        a sh:PropertyShape;
        sh:path rdf:type ;
        sh:hasValue cm:FingerGrasping ;
        sh:order 1;
        sh:group camo:IndependentModality;
        sh:name "Brace";
        sh:description "Grasp or Brace the Part"
        @en-US;
    ] .

camo:SHP_REL_RockerArmShaft
    a sh:NodeShape;

    sh:targetClass cm:Releasing;
    sh:property camo:PayloadShape;
    sh:property [
        a sh:PropertyShape;
        sh:path cm:gripperType ;
        sh:hasValue "parallel";
    ] ;
    sh:property [
        a sh:PropertyShape;
        sh:path rdf:type ;
        sh:hasValue cm:Releasing ;
        sh:order 3;
        sh:group camo:IndependentModality;
        sh:name "Release";
        sh:description "Release the Part"@en-US;
    ] .

camo:IndependentModality
    a sh:PropertyGroup ;
    rdfs:label "The class of actions that must be
    performed independently." .

camo:SynchronousModality
    a sh:PropertyGroup ;
    rdfs:label "The class of actions that must be
    performed sequentially, one after another.

camo:SimultaneousModality
    a sh:PropertyGroup ;
    rdfs:label "The class of actions that must be
    performed at the same time." .
```

Fig. 5. SHACL shape that defines the process of placing rocker arm shaft.

6.3. Refinement

The kickoff phase modelled the key concepts in the manufacturing, agent, and interaction contexts and laid the foundation for developing the detailed ontology of the knowledge base for attaining the requirements we set out with. At this point, we are faced with the important decision of choosing a knowledge representation language, as this could influence the further development of the information model. The languages of the Semantic Web stack were selected for the implementation because of the following reasons:

(i) The Web Ontology Language (OWL) as a knowledge representation language is based on descriptive logic that is formal with clear semantics and was deemed expressive enough to model the needed constructs while also being able to support inference using rules that govern the "encyclopedic" and "common-sense" knowledge.

(ii) OWL is widely popular as the language of choice for authoring ontologies, especially in production systems, with several developed in the literature that are available for reuse. Its use obviates the need to model all concepts from the ground up.

(iii) The languages of the Semantic Web stack support both notions of the open world (e.g., OWL) and the closed world (e.g., the Shapes Constraint Language (SHACL)). Their compatibility with each other makes it possible to synergistically model real-world use cases of HRC by exploiting both of these opposite notions.

(iv) As a technology originally built for the web, it is possible to perform federated queries that return results based on knowledge from multiple sources. Doing this would be advantageous in a heterogeneous environment that is characterised by multiple agents.

(v) There is active development in and around the technologies that support OWL (reasoners, triplestores, other compatible languages, etc.) and an active community that maintains them. Therefore, developers have ample choices and community support while building such systems. Most of these tools or technologies have seen several decades of improvement, with many receiving the recommendation status of the World Wide Web Consortium (W3C). OWL has been used in a variety of case studies in systems deployed within organisations that are currently used in production environments, as documented in an online repository [64] and thus can be considered mature technology capable of realising implementations with a higher technology readiness level (TRL).

Having selected the knowledge representation language, the refinement phase consisted of (i) task description, (ii) collaboration, and (iii) task execution.

(A) **Task Description:**
As introduced in Section 6.2, *production tasks* and constituent *processes* exist. Both of these are represented by PN transitions expressed in OWL. Their segregation makes for a convenient abstraction for adoption as a desire or intention. Their corresponding PN places denote partial product states and process

states, respectively. The definition of what constitutes a production task is left to the process engineer to generalise. In any case, the central idea is that a complex workplan for assembly leads to increasingly simpler production tasks containing processes that require certain capabilities on the part of the agents performing them. These capability requirements are represented by the `hasProcessDescription` property, which defines a resolvable URI of a SHACL graph. SHACL [65] is a language used for validating RDF graphs by defining constraints on the content, structure, and meaning of an ontology, and we use it primarily for two reasons:

(a) decomposing a process into *primitive tasks* or *functions*. Such a decomposition is considered necessary for "division of labour", i.e., assigning agents' roles during collaborative processes.
(b) representing the capability required to perform each of these primitive tasks and, therefore, the entire process.

To understand better, we examine the representation in greater detail by using an example of the process of placing a rocker arm shaft (Fig. 5). We use the process taxonomy model and the capability model vocabularies [62] with namespace prefixes `ptm` and `cm`, respectively.

Primitive Tasks or Functions: Primitive tasks are represented by SHACL node shapes. Node shapes are used to validate RDF terms, that become "focus nodes" in SHACL terminology during a validation, and here we use them to validate the capability required to perform the said primitive task. Property shapes of a node shape are used for the parametric validation of capabilities by specifying constraints that must be met.

In the example graph shown in Fig. 5, there are four such node shapes, each corresponding to four primitive tasks: `SHP_MOV_RockerArmShaft`, `SHP_MOV2_RockerArmShaft`, `SHP_GRA_RockerArmShaft` and `SHP_REL_RockerArmShaft`. These node shapes reflect the primitive task that entail a placing process, i.e., a bracing function, two moving functions, one each to approach and depart, and a release function. Each of these node shapes contains two kinds of property shapes. One or more *capability property shapes* validate the capabilities possessed by the agent at the parametric level required by the function represented by the node shape that contains the said property shape. A single *instance property shape* is used to represent "meta" properties of the primitive task. The two are discussed separately next.

Capability property shape: The capability property shape represents the capability (or capabilities) required by the primitive tasks that the said shape is part of (node shape). As such, it targets the capabilities of the capability model instantiated in the KB of the agent with the necessary parameters, i.e., it imposes constraints on them as required by the primitive task. An example of the capability parameters defined as property restrictions is shown for `cm:Moving` in Fig. 6. Furthermore, each node shape, since each node shape represents an atomic primitive task, it targets one capability using the predicate `sh:targetClass` and validates it against the one or more property shapes (of the node shape). The property shapes define constraints on the parameters of the instantiated capability individual, which the node shape targets as required by the primitive task the node shape represents.

In Fig. 5, e.g., the `SHP_MOV_RockerArmShaft` node shape defines a property shape (`camo:PayloadShape`) that checks for `cm:payload` data property being greater than or equal to `4` for the instantiated individual(s) belonging to the capability class `cm:Moving` (property restrictions as shown in Fig. 6). This means that the task requires that the robot be capable of moving a payload of 4 kg. Assuming the correct instantiation of the capabilities, a UR3 (payload 3 kg) robot would fail the validation, while a UR5 (payload 5 kg) would not. Several of these checks may be carried out by more property shapes for the node shape if needed, depending on the task.

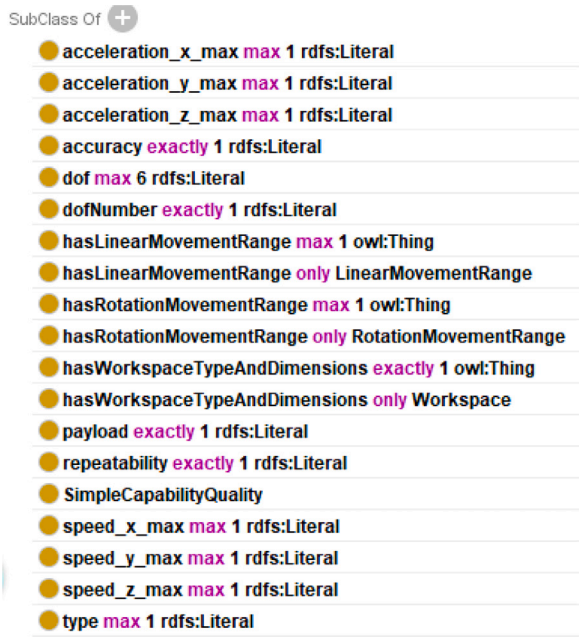


Fig. 6. Property restrictions defined for the "Moving" Capability as visualised from Protege ontology editor, vocabulary from [62].

The aforementioned instantiation of capabilities is also dynamic to account for changing capabilities, such as tool changes. For example, every gripper change (or initialisation) effects an update of the payload parameter to the capabilities provided by it (the gripper) by adjusting for its own weight using a SPARQL UPDATE query. For example, if the robot is rated for a payload of 5 kg and the gripper (rated payload 4 kg) weighs 1.2 kg then the robot payload-gripper weight difference (5–1.2) of 3.8 kg is compared with the gripper's rated payload of 4 kg. Since the difference is not greater than the griper payload, the difference, i.e., 3.8 kg, is the combined payload (even though the gripper is rated for 4 kg). On the contrary, if the gripper weighed 0.8 kg, the difference would be 4.2 kg, and as this difference is greater than the rated payload of the gripper, the combined payload is the rated payload of the gripper, i.e., 4 kg (even though the robot is rated for a payload of 5 kg).

Fig. 7 illustrates the validation report for the aforementioned UR3 robot case. Note how the validation report contains useful information (using vocabulary defined by the SHACL standard) on the failure, i.e., which capability individual (along `sh:focusNode`), which parameter (along `sh:resultPath`), the value that caused the failure (along `sh:value`), the property shape that failed (along `sh:sourceShape`), the constraint component (along `sh:sourceConstraintComponent`) and a user-defined message (along `sh:resultMessage`). As the validation reports are basically RDF graphs themselves, can be parsed to extract these important values and deal with such failures appropriately to communicate with another agent regarding the specific parameter of the specific capability, as an example.

Instance Property Shape: Besides representing and validating the capabilities required for primitive tasks, there should be means to represent other properties, such as the order in which they are to be executed, their execution modalities, etc. The challenge we

```
[ rdf:type      sh:ValidationReport ;
  sh:conforms   false ;
  sh:result     [
    rdf:type                sh:ValidationResult ;
    sh:focusNode            camo:CAP_Moving ;
    sh:resultMessage        "Data value \"3.0\"^^xsd:float is
                            not greater than or equal to 4" ;
    sh:resultPath           capabilityModel:payload ;
    sh:resultSeverity       sh:Violation ;
    sh:sourceConstraintComponent
                            sh:MinInclusiveConstraintComponent ;
    sh:sourceShape          camo:PayloadShape ;
    sh:value                "3.0"^^xsd:float
    ]
] .
```

Fig. 7. Result of SHACL validation as a validation report serialised in turtle format as reported by SHACL Jena Implementation (v4.5.0) [66].

face here is one that is created by the choice of SHACL graphs for the representation of process descriptions. While SHACL graphs themselves are RDF graphs and may be extended by custom-defined properties, we cannot take for granted that the SHACL processors that implement the SHACL standard and parse these graphs, will ignore these. To this end, a reliable workaround would mean finding means to represent the information we need within the bounds allowed by the SHACL standard. Favourably, SHACL implements what are known as "non-validating properties" for property shapes that are ignored by SHACL processors and are not subject to any formal interpretation by them. Conveniently, we have four such attributes (`sh:name`, `sh:description`, `sh:order`, and `sh:group`) that, by a matter of pure coincidence, have relatable names given the context of the properties of the primitive tasks we aim to represent. Since they are to be used in property shapes, we need to define a shape that will pass validation as they are used alongside other shapes that validate the capability required to perform the task. This we define as an instance property shape that checks the instance type to be the class the node shape has as its target declaration (hence the name instance property shape). Note that this way, the instance property shape is always bound to conform (pass validation), and as such it makes for a convenient syntax to represent the non-validating properties of the primitive task.

The name and description of the task are provided in `sh:name` and `sh:description` attributes. The order in which the tasks may be executed is represented in the `sh:order` attribute. Drawing inspiration from Helms et al. [67], the `sh:group` attribute represents the execution norm of the task and takes as a value (range) a `sh:PropertyGroup`. Tasks with the *independent* norm execute separate tasks. Tasks of a *synchronous* norm execute tasks sequentially, i.e., one done after another. *Simultaneous* norm tasks are to be executed at the same time. For more complex representations of modalities, for example, the presence of multiple different simultaneous modality tasks, property groups themselves can be of a type maintained along `rdf:type`. Such complex cases are not considered currently but can be accommodated. These SHACL shape graphs are parsed and queried with SPARQL (Table 1 iv) to return them in the order of execution and grouped by execution norms using the SPARQL ORDER BY and GROUP BY clauses, respectively. The agent can then use this information, along with coded execution logic, to carry out processes independently or collaboratively.

(B) **Collaboration:** The motivation for collaboration comes from the partial capability to perform a process of a production task identified because of a corresponding failed SHACL validation. Coded execution logic initiates communication with existing agents using the FIPA contract-net protocol [68]. Note that the FIPA contract-net protocol was originally designed for task delegation, where a coordinator agent delegates a task to an agent from

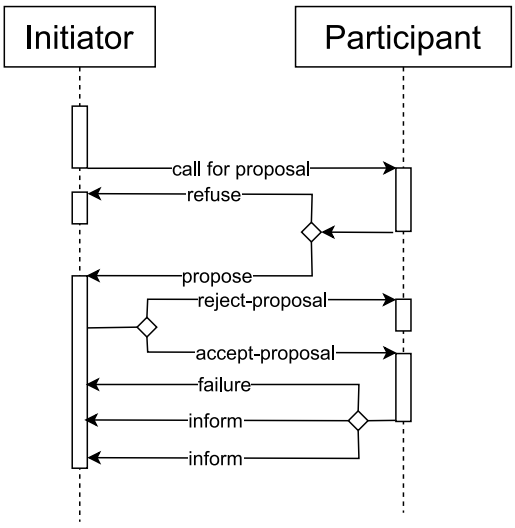
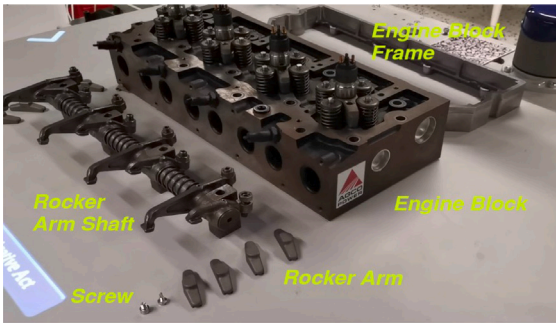


Fig. 8. The FIPA Contract-Net Interaction Protocol.

a pool of agents capable of it. Since FIPA does not have an interaction protocol specifically for collaborative use cases, we adapt the contract-net protocol as message exchange sequences for task delegation and collaboration logically overlap (Fig. 8). The first message in the protocol uses the "call for proposal" performative that the agent uses to inform available agents of the need for collaboration. The message contains information about the incapable process, the production task it is part of, and the capability class that failed (obtained from the validation report as just mentioned). Agents that receive this message describe their capabilities for the failed capability (via SPARQL DESCRIBE), internally validate the process in question, and if it conforms, send a proposal to collaborate using the "propose" performative with a description of the capability in question in RDF/OWL in the content field. The initiator agent validates this capability based on its initial knowledge of what it is incapable of and, if it conforms, accepts the proposal and records the messages exchanged as a collaboration contract (i.e., an agreement for collaboration). Since the collaboration was designed between a single human operator and a single agent, as mentioned in Section 3 where the research scope and setting were outlined, we use a subset of the contract-net protocol, the propose protocol instead. The propose protocol is similar to the contract-net protocol, without the initial unnecessary call for proposals, given the dyadic human-robot configuration of the research setting.

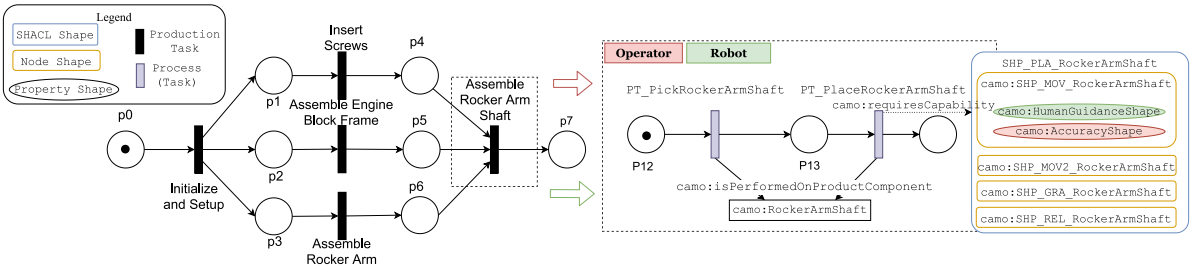
(C) **Task Execution:**

We draw inspiration from multi-agent system literature [37] to move from intentions to actions that execute the task. Intentions are pursued by executing *actions* that are part of a *plan* that accomplishes the goal and thus drives means-ends reasoning. Intentions also constrain future deliberations [37]. So far, we have modelled intentions as achievable and non-conflicting desires that are production tasks (not processes). Their definitions entail descriptions of the message exchanges about the available production tasks that are inferred at runtime and consequently influence their behaviour. Each production task has processes and process states that follow a process plan for the production task (Fig. 4). The agents' plans are represented at this level i.e., the process level (not production task). The class `AgentPlan` (⊆`DUL:Plan`) acts as a library of agent plans that are instantiated as its individuals at runtime. Actions (`DUL:Action`) that belong to a plan are indicated via an (`isPartOfPlan`) object property. The capability required to



Production Task	Process	Part	Primitve Tasks	(Distinct) Capability Classes	Capability possessed by (R, O, R||O, R&&O)
Initialize and Setup	Setup		Moving, Expanding Gripper	cm:Moving, cm:FingerGrasping	-
Insert Screws	Pick	Screw1, .. Screw4	Approach, Grasping, Extract	cm:Moving, cm:FingerGrasping	O
	Screwing		Approach, Screwing	cm:Moving, cm:Screwing	O
Assemble Engine Block Frame	Picking	Engine Block Frame	Approach, Grasping, Depart	cm:Moving, cm:FingerGrasping	R||O
	Placing		Bracing, Approach, Releasing, Depart	cm:Moving, cm:FingerGrasping	R||O
Assemble Rocker Arm	Picking	RockerArm1, ..RockerArm8	Approach, Grasping, Depart	cm:Moving, cm:FingerGrasping	O
	Placing		Bracing, Approach, Releasing, Depart	cm:Moving, cm:FingerGrasping	O
Assemble Rocker Arm Shaft	Picking	Rocker Arm Shaft	Approach, Grasping,	cm:Moving,	R
	Placing		Bracing, Approach, Releasing, Depart	cm:Moving, cm:FingerGrasping	R&&O
Complete	Enter Safe Idle Mode	-	-	-	-

(a) Physical parts used in the test case (left), and a table representing the ABOX assertions in the ontology for the test case (right).



(b) Petri Net equivalent of the test case represented by the ontology.

Fig. 9. Relevant information associated with the assembly test Case.

execute an action is maintained via a `hasCapabilityClass` attribute that defines a URI of the capability (from the capability model ontology) required to perform the intended task. These attributes are necessary to model deterministic actions based on specific capability classes. As an example, the unique URI `camo:HumanGuidance` can be interpreted as requiring a simultaneous modality action on the part of both agents, an upward force action by the robot that compensates the weight of the part, and a move action for the operator to effortlessly guide the part into place. A schematic of the key concepts is shown in Fig. 3.

Plans are generated dynamically once a commitment (Contract) regarding the execution of a production task is made based on runtime communication between agents. One of two types of plans, namely "collaborative" and "non-collaborative", are generated depending on the capability checks at runtime and are maintained by the `camo:hasType` data property. These plans help execute different program logic during plan execution, as is shown in Section 6.4. The preconditions under which a plan may be adopted are defined in the head part of the plan [34]. Since processes are modelled as PN transitions, the enabling of the transition can be modelled as the pre-condition (concept of Situation in DUL) under which the plan for a process task may be considered for adoption. A `hasHead` attribute of `Plan` associates with the respective `ProcessState`. Note that each plan can have one or more actions depending on the process description (of primitive tasks), and although the semantics allow for several plans to exist as options in a process state, the scope of the current work limits us to working with a single plan and reserves the question of multiple plans and one being better than another for future work.

In conclusion, the engineered ontology as the information model aims to effect a logically consistent interplay between the concepts of desires, intentions, plans, and actions between collaborating agents so that they are suitable for use by the real-time systems when queried. Therefore, an external robot module interfaced directly with the robot controller can query for plans, their corresponding processes, the part on which a process is performed, and the actions of the plan without contradicting the operator's private beliefs. For example, the availability of a non-collaborative plan to pick an engine block frame, when adopted, can initiate robot motion and grasping actions according to the generated plan, where the robot is solely responsible for task execution as it is capable. This would be either upon the operator's request or proactively by the robot itself, but only after the operator has acknowledged the robot's desire for the same and agreed to it, leading to robot's intention. Since the scope of the work does not cover the real-time systems responsible for pose estimation and robot path planning, for this study, the location of parts is assumed to be known and robot waypoints and trajectories are hard coded. As such, the robot timely contributes to collaborative assembly with the help of the control logic discussed in Section 7.3.

6.4. Evaluation

Before the ontology was employed in the target environment, it was evaluated for its ability to answer the competency questions defined in the kickoff stage. To validate the answers to the competency questions, we populate the ABOX of the ontology with a test assembly case used in the upcoming application phase, as described next.

6.4.1. Test case:

The test case involves the assembly of a real diesel engine that consists of five parts: (i) engine block (x1), (ii) screws (x4), (iii) engine block frame (x1), (iv) rocker arm (x8), and (vi) rocker arm shaft (x1), as shown in Fig. 9a (left). The equivalent PN representation of the *product workplan* is shown in Fig. 9b, using the convention presented in Section 6.2.2. It consists of five *production tasks* that entail corresponding *processes* (performed on parts). While the PN equivalent representation of all *production tasks* as processes are not shown due to lack of space, they mostly resemble the case for the "Assemble Rocker Arm Shaft" that is shown.

Fig. 9a (right) tabulates key terms of the instantiation of the diesel engine assembly (ABOX) in CAMO. Capability descriptions instantiated in the ontology render the agents (operator and robot) capable or

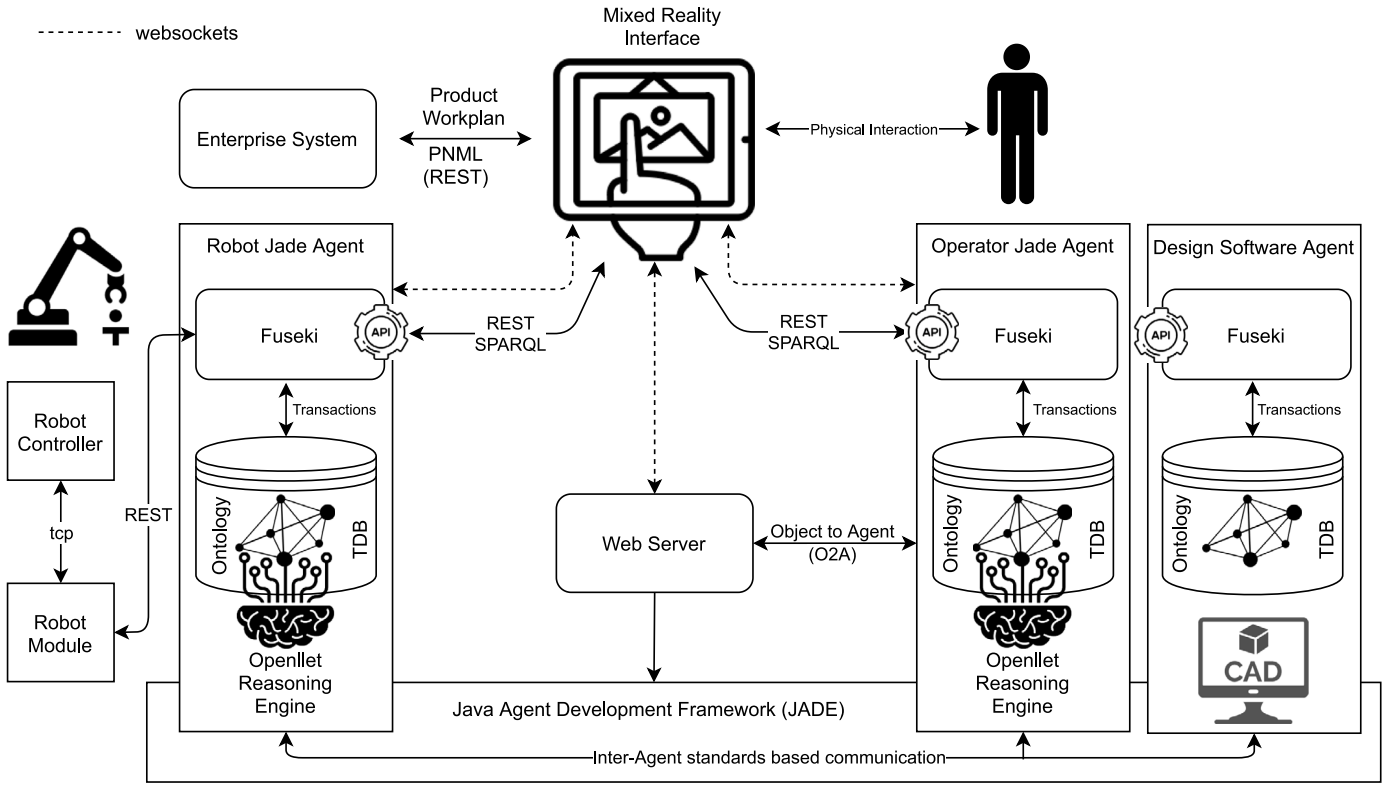


Fig. 10. A schematic representation of the implementation.

incapable of a process. This is indicated by the last column in the table in Fig. 9a as follows: robot-only (R), operator-only (O), both robot and operator capable separately (R||O), both robot and operator capable collaboratively (R&&O).[4] These are modelled in a way that they reflect the capability of the gripper attached to the robot and the preferences of the operator that are instantiated as individuals of the capability classes of the capability model or lack thereof. Next, we briefly present these semantic descriptions underpinning the capabilities for this specific test case.

The lack of capability to carry out a *process* can be understood semantically in two ways, and an example of each is intentionally present in the test case. First, the absence of the instantiation of the capability class that is required for the task in question, that is due to the absence of a resource that provides the agent with the capability. For example, this is the case of the "screwing" *process* as the robot is not equipped with a suitable gripper to perform a screwing process. Secondly, such an incapability may also be understood by the failure of the parametric SHACL validation of the capabilities instantiated in the agent KB against the shape that represents the capabilities required to do the task in question. For example, this is the case for the *pick* and *place* processes of the rocker arm, as it is modelled that the gripper is suitable for picking parts with a minimum dimension that is greater than the relatively small rocker arms. Note that other picking and placing processes exist of which the robot is capable.

It is possible to define special property shapes in CAMO to effect deterministic behaviours of the agents. An example of this is `camo:HumanGuidance` property shape of the "Place Rocker Arm Shaft" process (Fig. 9b (right)). The failure of this shape during validation indicates that the action to move needs to be performed in simultaneous modality with that of the operator's action for the said process. This will then result in corresponding communicative acts to make sure that the agents are aware of the modality, as is shown in the next section.

The answers to the competency questions are obtained by queries that answer or graphs that validate the agent's knowledge base and are tabulated in Table 1, and are verified for the test case presented.

7. Application: Case study – deployment in an HRC framework

7.1. Tools and technology

The engineered ontology is deployed in the setting (Fig. 1) with the HRC framework that integrates the assembly design software and incorporates the mixed-reality interface (Fig. 2), which was introduced in Sections 3 and 5. A schematic of the framework instantiation is shown in Fig. 10.[5] The framework builds on JADE, to enable agent-oriented interactions between the collaborating agents via standardised interaction protocols (FIPA [63]). Both the robot and the operator are instantiated as JADE agents in the framework with each maintaining its private knowledge base using CAMO. We use the popular JAVA Apache JENA library [66] (v4.5.0) to work with OWL (Jena Ontology API), SPARQL (Jena ARQ API) and SHACL (Jena SHACL API) programmatically. Jena Fuseki provides a RESTful interface to the knowledge base of the agents for non-conversational information retrieval via SPARQL 1.1 query protocols. The back end of the knowledge base is implemented with the help of Jena TDB, which provides a robust, transactional persistent storage layer for the knowledge base. The inference engine is provided by Openllet (v2.6.5) [69], an open-source OWL2 DL reasoner.

[4] To be clear, this convention is used only as an indication to the reader and in reality they are a result of a combination of the capabilities instantiated in the knowledge base of the agents using the capability model vocabulary and the requirements of the process described as SHACL shapes as defined in Section 6.3(A).

[5] For a detailed software architecture, see [39].

Table 1
Evaluation of the ontology based on the defined competency questions.

CQ No.	Methodology of Information Retrieval	Query / Shape (*Prefix names are omitted for brevity*)	Single / Multiple SPARQL | Reasoning used | SHACL needed	Remark

(i)	Query for the enabled transitions that represent production tasks, i.e., check for transitions that have a marking via the boolean `hasToken` property set to "true" for all its input places and query its `includesActivities` object property.	`SELECT ?task1 ?id WHERE {`
		`?task1 a camo:ProductionTask. ?task1 camo:UID ?id.`
		`?state camo:includesActivities ?task1. ?state camo:hasToken true.`
		`MINUS {`
		`?state1 camo:hasToken true. ?state2 camo:hasToken false.`
		`?state1 camo:includesActivities ?task1.`
		`?state2 camo:includesActivities ?task2.`
		`filter ((?task1=?task2)) } }`
			Single | No | No	The same format of the query is applied to query the active process tasks of a production task. Note that this does not factor in the agent's capability to do the task.

(ii)	Achievable desires are asserted with `isAchievableWithCap` object property dynamically at runtime with a SPARQL Update Query. The other subclasses of desires and intentions are automatically inferred at runtime based on the defined class definitions. A simple SPARQL query (using negation) can be used to query intentions that are not non-achievable or conflicting desires.	`SELECT ?int WHERE {`
		`?int a camo:Intention.`
		`FILTER NOT EXISTS {`
		`?int a camo:NonAchievableDesire.`
		`?int a camo:ConflictingDesire.`
		`}`
		`}`
			Single | Yes | No	Desires and its subclasses represent the motivational stance while intentions represent the intentional stance.

(iii)	An update query to effect the firing of a transition, i.e., remove tokens from incoming places of a *given* transition representing the (production or process) task to outgoing places of the transition.	`DELETE`
		`{ camo:P1 camo:hasToken "true"^^xsd:boolean.`
		`camo:P2 camo:hasToken "false"^^xsd:boolean. }`
		`INSERT`
		`{ camo:P2 camo:hasToken "true"^^xsd:boolean.`
		`camo:P1 camo:hasToken "false"^^xsd:boolean. }`
		`WHERE {?places camo:hasToken ?hasSourceToken.}`
			Multiple | No | No	Note that the query is constructed dynamically by querying the input and output places of a *given* task. In this case separate queries have returned results that P1 and P2 are input and output places of the given task. The pattern is common for transitions representing both production tasks and processes.

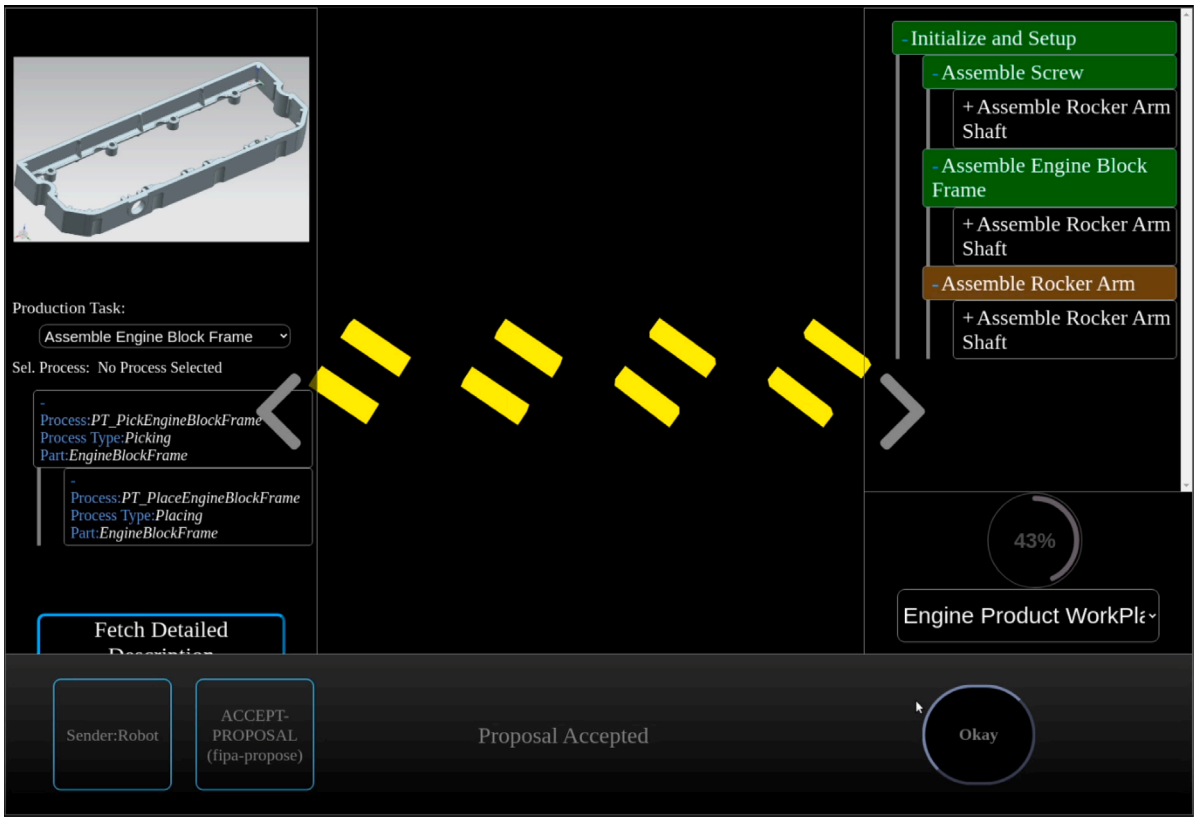
(iv)	The SHACL graph that maintains task description is first queried and resolved via the `hasProcessDescription` property. Then a second query (shown), queries the description for the primitive tasks, involving aggregates and solution sequence modifiers returns the sequence of tasks, possibly grouped by execution norms if needed.	`SELECT ?actionClass ?group ?order ?desc ?name ?context WHERE {`
		`?nodeShape a sh:NodeShape.`
		`?nodeShape sh:targetClass ?actionClass.`
		`?nodeShape sh:property ?propertyShape.`
		`?propertyShape sh:value ?actionClass.`
		`?propertyShape sh:order ?order.`
		`?propertyShape sh:group ?group.`
		`?propertyShape sh:description ?desc.`
		`?propertyShape sh:name ?context.`
		`} ORDER BY asc(?order)`
			Multiple | No | No	Solution sequence modifiers (`ORDER BY`) and Aggregation (`GROUP BY`) may be nested or used interchangeably in any combination depending on the returned result.

(v)	The plan for a process task is maintained via the `hasHead` attribute with the incoming place (process state) to the transition that represents the given process task.	`SELECT ?plan`
		`WHERE {`
		`?plan a camo:AgentPlan.`
		`?plan camo:hasHead ?processState.`
		`?processState camo:includesActivity camo:PT_PlaceRockerArmShaft.`
		`}`
			Multiple | Yes | No	All active plans are automatically inferred by the reasoner using a defined class expression. See concept `ActivePlan`.

(vi)	Query the description of a *given* task expressed as a SHACL graph maintained via the `hasProcessDescription` attribute. Read the returned graph as regular rdf and query target class (`sh:targetClass`, shown) of the node shape(s) that represent primitive tasks that make up the process task. Describe the capabilities of the agent with respect to that target class and run a shacl validation of the described capabilities against the SHACL graph.	*shacl shape as shown in Fig. 5*
		`SELECT ?targetClass WHERE {`
		`?shape a sh:NodeShape.`
		`?shape sh:targetClass ?targetClass.`
		`}`
			Multiple | No | Yes	Describe queries are generated dynamically for all returned target classes.

(vii)	Describe the *given* production task (transition) for the incoming places (`ProductState`). SHACL Validation of the returned graph (from the DESCRIBE query) using a node shape (shown) that has a property shape for the path (`hasToken`) that uses the value constraint to be "true" for all incoming places.	`camo:isEnabled`
		`a sh:NodeShape;`
		`sh:targetClass camo:ProductState ;`
		`sh:property [ a sh:PropertyShape;`
		`sh:path camo:hasToken ;`
		`sh:hasValue "true"^^xsd:boolean;] .`
			Multiple | No | Yes	What this essentially does in the PN formalism is checking if all incoming places for the transition that represents the task has a token.

(viii)	Query the `hasHRDescription` attribute of the sub-task of the task.	`SELECT ?Desc WHERE {`
		`camo:Place_Engine_Frame camo:hasHRDescription ?Desc.`
		`FILTER (lang(?Desc) = 'fi')}`
			Single | No | No	-



(a) Screenshot of the mixed-reality interface when the robot sends a "propose" message for the production task to assemble the Engine Block Frame.

(b) Interaction panel of the MR application.

Fig. 11. Components of the mixed-reality interface.

7.2. Mixed-reality interface

The MR interface (Fig. 11) consists of five different components that are tasked with different functions, albeit with the overarching purpose of providing an interface for the operator to interact with the collaborating agent (robot) and representing and making accessible to the operator the underlying information models. These five components are:

1. *Product Workplan Panel:* This is a collapsible drawer of adjustable width, drawn from the right (Fig. 11b (right drawer)) that allows the operator to select the workplan associated with the assembly and render it as a collapsible tree from its petri net markup language (PNML) (PNML, ISO/IEC 15909 Part 2) [47] representation. Each element of the tree that represents a production task is represented using different background colours that indicate the status of the task and/or draw the attention of the operator (Fig. 11a (right drawer)). For example, the screenshot of the MR interface (Fig. 11a) shows when three tasks are completed, which is indicated by the green background, while the task "Assemble Rocker Arm" blinks in dark orange to support the message sent by the robot (Fig. 11a (bottom)) with contextual information. It also consists of a progress wheel that represents the percentage completion of the tasks in the work plan.
2. *Process Plan Panel:* This is a collapsible drawer of adjustable width that is drawn from the left (Fig. 11b (left drawer)), which allows the operator to select a production task to view its process plan, which is rendered as a tree from its PNML representation. The drawer may also be drawn open by another agent to, for example, provide contextual information for a message. A process may be selected to view a detailed description of the primitive tasks by clicking a "Fetch Detailed Description" button, which subsequently draws open another left drawer to neatly display them as a stack of cards. The left drawer also displays any accompanying figures that are relevant to the intended process.
3. *Message Panel:* This is a "message box" that appears at the bottom (Fig. 11a (bottom)), which notifies the operator of messages from other agents in the framework (Fig. 11a (bottom)). It consists of a sender field, that is populated with the JADE agent identifier (AID), message type field that indicates the performative (communicative act) and interaction protocol the conversation belongs to, a message field that displays the content of the message, and 1–3 buttons (depending on the performative as defined by FIPA) that allow the operator to respond to the message.
4. *Interaction Panel:* The interaction panel (Fig. 11b) is a drawer that can be opened by clicking the button "Perform Communicative Act" in the layout (not visible in the figure) to interact with the robot. The panel allows the operator to select the

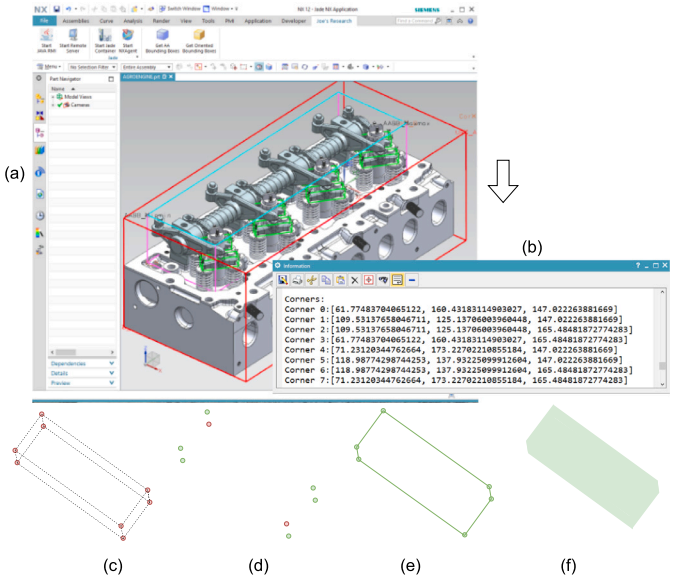


Fig. 12. Transformation of bounding box coordinates from CAD model.[6]

performative and the task from clickable buttons as necessary for communication.

5. *Canvas:* The canvas is implemented using the HTML 5 Canvas API and, allows for free-form drawing. The canvas is used by the agents to either communicate their intentions by spatially projecting shapes on the physical assembly or provide visual guidance for the operator. This, as shown in Fig. 12a-c , is extracted from the CAD model of the part at runtime and uses a pre-computed transform (direct linear transform) to map coordinates defining the bounding boxes of subassembly parts to projector pixels. The Jarvis algorithm is then used to compute the convex hull of the transformed points (Fig. 12d). Lines are drawn between the points defining the convex hull and filled with colour using the canvas API (Fig. 12e-f). The result for the case of rocker arms can be seen in yellow in the centre in Figs. 2 and 11.

These components interact with the knowledge base of the agents via RESTful message exchanges (Fig. 10) in real time to achieve their intended functionalities. For example, the Process Plan and Interaction Panels allow you to query and interact, respectively, with production tasks that are only currently available (enabled transitions in the equivalent PNs) by querying the KB with the SPARQL query (CQ No. (i) in Table 1).

7.3. Communication patterns and control logic

The process of collaborative assembly starts when the operator selects the product workplan from a list of available ones and requests collaborative assembly (done on the *product workplan panel*). The system supports four basic communication patterns that conform to their designated use as FIPA standards.

1. The robot-push pattern uses the FIPA-Request protocol to request that the operator performs a task.
2. The robot-pull pattern uses the FIPA-Propose (or FIPA-ContractNet) protocol to propose to the operator that the robot does a task (possibly with the need for collaboration).
3. The operator-push pattern uses the FIPA-Request protocol to request that the robot performs a task.
4. The operator-pull pattern uses the FIPA-Propose (or FIPA-ContractNet) protocol to propose to the robot that the operator does a task.

Furthermore, the robot is configured to operate either in a proactive mode (using primarily robot-push and robot-pull patterns), which involves picking up available tasks and initiating communication with the operator,[7] or reactive mode, which executes tasks upon operator request (using operator-push and operator-pull patterns). For the robot in proactive mode as employed in the case study, the control flow is as shown in Fig. 13. Only the key steps are shown to keep the flow chart readable and concise. The control logic of the robot is divided over two JADE behaviours, namely `Desire` and `Intention`. Elements that are part of a collaborative workflow are drawn in purple.

The `Desire` behaviour is responsible for traversing the PN, picking up production tasks, checking capabilities, and initiating conversations with the operator via the MR interface. The inference engine classifies the tasks after the conversation with the operator as subclasses of desires (*achievable* and *non-achievable*; *conflicting* and *non-conflicting*) and consequently as intentions (*achievable* and *non-conflicting* desires), as defined in the agent context of the KB (Section 6.2.2). In our case study, between agent conversations for reasoning about desires and intentions, we found it took between 100ms-1500 ms. As the size of the ontology grew the performance dipped. Although, it was unnecessary for the performed case study, in principle, we would be able to contain the size of the ontology dynamically (for example, by discarding older conversations about completed tasks after storing them) to keep the reasoning both computationally tractable and performant.

Furthermore, plans are generated dynamically, and each contains actions based on the corresponding process the plan has been created for in accordance with the process descriptions (in SHACL). These actions contain as attributes the fields of the instance property shape (Fig. 5) along with a performer field that indicates which agent is responsible for it based on the agreement in contracts that is the consequence of the runtime deliberation between the agents. They also contain a status field used to record the execution of the task. If an agent has agreed to do a task that it is wholly capable of (i.e., non-collaborative), all actions are the responsibility of the agent and will be filled with the AID of the said agent. Plans are generated for all processes in the production task. Once plans are generated, an `Intention` behaviour is spawned for the respective production task to carry out the process plan of the production task adopted as the intention. Note that at this stage, the agent is aware of all the processes of the production task that is adopted as the intention, the entailed processes, and the capabilities to perform each of them.

The `Intention` behaviour, similar to `Desire`, plays by the rules of the token game of the PN, representing the process plan, and explores the available processes of the production task the intention is about. An external *Robot Controller Module* interfaced spontaneously queries for the plans and any associated part and process information. This is accomplished via REST interface provided by Fuseki. The Robot Controller Module translates the knowledge of the process (picking, placing, screwing, etc.) and the part the process is performed on to actual robot motions based on pre-defined motion paths and actions. The robot module subsequently updates the status of the actions of the plan, in the KB of the robot upon execution (again via Fuseki). Collaborative processes have a slightly different execution flow in the `Intention` behaviour as shown in Fig. 13. This difference is predominantly to make the operator aware, via the MR interface, of the collaborative nature of the process and the specific sequence and execution norms of

[6] c-f are drawn manually using a drawing tool for the sake of explanation.

[7] Note that nothing prevents the operator from exercising operator-push and operator-pull patterns in the robot proactive mode, and the discussed semantics underlying intentions automatically prevent turn-taking behaviour

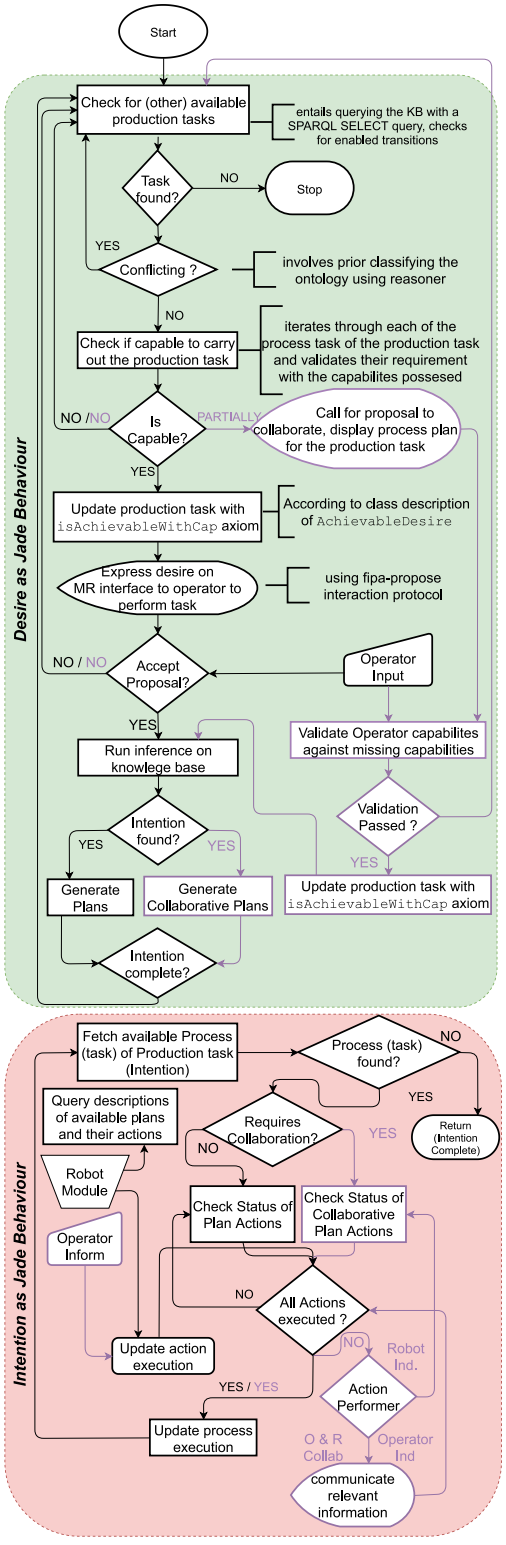


Fig. 13. The control flow defining proactive robot behaviour between `Desire` and `Intention` Jade behaviours.

the entailed actions. For example, for a simultaneous modality action (e.g., placing the rocker arm shaft in the case study), the operator has the possibility to indicate when the operator is ready to act and when the action is completed.

The assembly was done for the test case presented in the evaluation phase, and CAMO was found to support it well. The robot was configured in the proactive mode, but the operator also makes use of the operator-pull pattern. A video recording of the case study is available online.[8] In it, the assembly of both collaborative and non-collaborative tasks are shown from five concurrent views. Three views from cameras show the Kinect view output from the hand-tracking ML pipeline, an assembly view of the assembled diesel engine and a regular camera view of the workspace showing the MR environment in action. Two screen recordings also capture the JADE sniffer tool that records the messages exchanged in the JADE framework and the MR application as viewed from the web-browser, one each separately.

8. Discussion

This article presents the development of an application ontology and its instantiation as distributed knowledge bases in a multi-agent HRC framework. The approach entailed integrating the ubiquitous product design software, leveraging its programable CAD kernel (the "technology" of knowledge-based engineering according to Rocca [41]) to define appropriate interfaces to get relevant information about the assembly at runtime. Next, we discuss the significance of the approach, the developed information model, and the MR application in light of the research objectives outlined in Section 3.

8.1. Developed artefacts

The development of CAMO follows a sound ontology engineering methodology and builds upon DUL, a lighter axiomatisation version of a foundational ontology (DOLCE) with more intuitive terminology and additional supporting theories [58]. As such, it has sound logical theories as its foundation, and due to its broad foundational coverage, it may be easily reused or extended. Furthermore, despite being an application ontology, the design patterns used may be generalised for reuse in another multi-agent context. Specifically, the logical formalisms underpinning the concepts of the agent context (desires and intentions) can inspire similar approaches based on the performatives of agent interactions, as there is little, if any, application of multi-agent systems that do not demand inter-agent communication. This is besides the obvious fact that these concepts may be reused out of the box for systems developed in accordance with the FIPA standards. In these ways, the information model developed in this study supports generalisability for similar applications.

A review of related work on logic-based architectures for HRC exposed that they either lacked concepts modelling collaboration, ignored the multi-agent aspect, had centralised knowledge bases or downplayed their interaction as a mere communication problem, i.e., some ad-hoc way to get the message across. Such solutions from an information system standpoint, are difficult to scale to adapt to product variety, as an example. In this article, we have shown how a logic-based architecture can model collaborative interactions along with the associated manufacturing context should interactions adhere to a standard. We have attempted a human-centric approach by modelling interaction sequences following folk psychology that has been conveniently presented via mixed reality along with performatives based on speech-act theory [43,70]. These performatives include expressives, which express attitudes, exercitives, which ask for actions to be performed, and interrogatives, which query for information akin to human–human interactions. These provide contextually relevant information for the operator along with message content as text. For example, a glance at "request" performative may lead the operator to understand that

[8] https://youtu.be/bgB00fFKgG8

they are being requested to do something that they may choose to conveniently ignore, without understanding what, if busy.

Another observation could be made on how the concept of capabilities is treated. For example, Umbrico et al. [17] use rules to model the capabilities as follows:

```
DUL: Agent(a) ∧ Function(f) ∧ hasCapability(a,c)
∧ requires(f,c) →  canPerform(a,f)
```

This translates to "*if a function f requires a capability c and an agent has that capability c, then the agent can perform the function*". An analogous rule-based approach for determining the capabilities of manufacturing resources using SPARQL Inferencing Notation (SPIN) has been proposed in earlier work [62]. While such rule-based definitions work in principle, they do not allow for understanding what exactly causes incapability, should it occur. This is most likely a direct consequence of the open-world assumption of OWL, where no assumptions are made on the absence of statements, i.e., something not known to be true is unknown. In HRC use cases, where the theoretical motivation itself is premised on the combined capabilities of collaborating agents, rule-based and consistency-based capability checking may fall short of conveniently identifying missing or non-conforming capabilities. Using SHACL for capability checking, as we have in this study, enforces the closed world assumption as constraints on capabilities represented in OWL to identify missing capabilities.

Often, capabilities are defined by a whole host of parameters (an example of which can be seen in Fig. 6). Performing a semantic validation using a standard such as SHACL can help with the parametric validation of capabilities, such as the payload being too heavy, as discussed previously. The result of such *parametric* validation of capabilities can provide meaningful results using supported constraint components (value type, cardinality, value range, etc. [65]) that the agent can use to navigate around such failure (e.g., by considering switching to a suitable gripper at a gripper station). Note that the results of the validation are governed by the SHACL standard [65], whose values adhere to the standardised vocabulary of the SHACL namespace. This allows for the building of scalable situation-independent routines for capability validation without much programming overhead for the application developer and makes for an elegant and maintainable approach for the representation of capabilities and validating requirements. The alternative to this using prevalent approaches would be to exhaustively check the parameters individually, but that would be tedious (even programmatically) and far from optimal from a performance standpoint. In summary, knowledge of why a capability does not meet the criteria put forth by a task, besides acknowledging its absence, can be crucial in effectively modelling processes, such as HRC, that are motivated by complementing individual capabilities of agents to begin with. A standards-based approach only furthers this by making it scalable.

The mixed-reality application provides a convenient interface for the operator to interact with the developed information models. It does the job of abstracting away from the details of the underlying net formalisms, product work plans, entailed process plans, part geometry, etc. expressed in OWL/SHACL, STEP, etc. to provide their more intuitive representations in JavaScript as trees, cards, colour shades, and free-form drawings, sometimes neatly tucked away in optionally visible drawers. This is done with the goal of reducing the cognitive load on the operator while increasing his/her situational awareness. From a novelty standpoint, our work introduces web technologies to the task of architecting extended reality interfaces for a typically constrained collaborative industrial environment. Its significance lies in greatly reducing the barrier to working with scripting languages (JavaScript) that are easier and faster to develop and maintain [71] than their compiled counterparts, such as C# for Unity as an example. The web browser, as a mature platform, works cross-platform and has seen more than a couple of decades of improvement and standardisation (HTML, CSS). This means a wide range of 3rd party libraries to choose from both for added functionality (e.g., 3D STL — three.js, for visualisation of part geometry) and for development (ReactJs, Angular, Vue, etc.), and therefore "reduces" its development problem to a front-end web development problem, a problem that in today's day and age has solution architects aplenty.

In Sections 1 and 3, we set out the objective of attaining a respectable degree of self-organisation for the developed artefacts from an information system standpoint. As for the developed ontology, the notions of product work plans and entailed process plans follow net formalisms that have been a widely popular approach to modelling distributed systems. We posit that the possibility of a bijective correspondence between a versatile and standardised (PNML, ISO/IEC 15909 Part 2 [47]) net formalism and its equivalent OWL representation may be seen as an advantage and contribute towards CAMO's acceptability in industrial informatics. The benefit of using a standard was seen during the application's development itself. It was possible to create the product workplan and the process plans using the user-friendly WolfGang Petri Net editor [72] that conformed to the standard. As such, modelling collaborative processes is use case-agnostic following the pattern described that self-configures for situation-dependent use cases, and allows for the adoption of these collaborative processes as desires and intentions by collaborating agents. In the MR application, the *product workplan panel* and *process plan panel* are also accepting of PNML formats to visually represent product work plans and process plans as trees.

The visual communication of intentions via mixed reality is made possible by the integration of "KBE technology" [41] that features a programmable CAD kernel that we used to traverse the assembly tree to calculate bounding box coordinates of part components. Again here, the assembly can change, and its CAD model can be in a whole host of formats supported by NX (STL, 3MF, IGES, STEP, etc.) that can be interpreted in a way that populates the ABOX of CAMO correctly with BBOX coordinates, and thus the approach supports self-configuration as we first defined. This is in contrast with existing works in literature [7,8,73] that use manual approaches for feature extraction for intent projection. Lastly, the use of SHACL allows for a standards-based, scalable approach for parametric validation of agent capabilities. Therefore, we see the system as capable of self-optimising, i.e., taking stock of available agent capabilities and initiating collaborations with respective agents for collaborative assembly, thus contributing to self organisation as defined in Section 4.2.

8.2. Limitations and future work

First of all, it is important to note that the approach presented in the study alone may not be used to realise an end-to-end deployment of an HRC assembly system in practice, as evident in Section 3, which outlines the scope of the work. The system developed in the form of a self-organising team network plays the important role of assigning responsibilities through deliberation, which is needed to prevent turn-taking behaviour and achieve the outlined objectives. We envision that our contribution would come just before the real-time systems (logically, not based on priority) of pose estimation, trajectory planning, and collision avoidance.

As for this study, we are aware of the following shortcomings. Firstly, for the information model, the current ontology lacks a representation of concepts in the temporal dimension. In principle, it might be useful to schedule or coordinate shared tasks for the future, defer tasks for a fixed time, introduce time-based collaboration modalities, or simply log events. Currently, all tasks are discussed and executed "in the now". To allow for a greater "degree of freedom" for flexible interaction, temporal concepts will be necessary, and this is something we consider as future work to further improve the applicability of the developed work. Next, the decomposition of processes into primitive tasks is inevitable to decide who does what in collaborative scenarios, but it is not always a straightforward process. For this, we propose as

future work developing a library of process templates derived iteratively from successful implementations and the developing a GUI to aid their easy deployment. Likewise, for the description of production tasks and processes, we would like to investigate if we can programmatically extract their sequence for conversion into PNML using KBE technology. Currently, in the study, although the model supports the semantics of OWL, it is manually constructed using Protege, an OWL editor. It lacks a PNML to OWL translator, a straightforward piece of software that would map each PNML concept to its corresponding concept in OWL, that remains a future work. The workflow would then be to (i) construct work plans using user-friendly GUIs, (ii) translate from PNML to OWL, optionally validate the plans either using a GUI manually or automatically using SHACL constraints validation, and (iii) import them into the knowledge base of agents. Thus, the cost and time of modelling the tasks and processes are minimal once the required translator is developed as future work.

Another possible future direction of work for the information model is supporting concepts of design for collaborative assembly (DCA). We believe it would be beneficial for the operator to record his/her experience of the assembly to further improve the safety, physical and cognitive ergonomics, and assembly efficiency as per the design guidelines and principles presented by Gualtieria et al. [74]. This (experience) would then be presented to the designer in a custom-built KBE application to close the loop between manufacturing and design, realising the potential of the digital thread.

The aforementioned developments as concepts in the ontology must be reciprocated with corresponding developments in the MR application that supports their use. For example, component(s) in the application that enable the operator to record the DCA terminologies and component(s) that allow the operator to respond to deferred requests must be considered. Apart from this, in its current state, we acknowledge that the MR interface bears the requirement of a flat surface for operation and that it is not indicative of all collaborative assembly environments. However, table-top assembly environments in industrial settings are quite common. Furthermore, bounding box approximations as intentions bear the requirement of a CAD model and might not be the best approximation in niche cases for certain part geometries. Nevertheless, they will fully encompass the intended part shape.

From the standpoint of the developed framework, its dependence on proprietary software as the KBE system may not be appealing to some, and the generalisability of the approach may be of concern. The former may be justified by the value it adds to the system. The ability to map between various complex product standards (e.g., STEP) and proprietary models is a feat of software engineering to behold in itself and can be expected to require several thousand person-hours. Abstracting these models into a common object model and making it accessible by well-defined, albeit (necessarily) verbose APIs that form the basis for self-configuration understandably come at a cost. As for generalisability, "KBE technology" is growing in popularity and is finding its way to mainstream CAD/CAE software vendors. Examples of these software are Solidworks by Dassault Systems and AutoCAD by AutoDesk, as detailed in the work of Rocca [41]. The expectation is that as more works demonstrate the utility of "KBE technology", its absence will be seen as the lack of a versatile and useful feature, and consequently motivate CAD vendors to adopt more open architectures. As for bounding boxes, their intersection is a common and performant means of collision detection that most, if not all, 3D geometry software comes with the ability of its computation, as does NX.

While this work presented the information model and its validation by means of competency questions, case study, and informed arguments, in future work, the advantages of the proposed approach from the standpoint of the process of HRC (e.g., process performance and/or operator satisfaction) will also need to be quantifiable. As for what kind of improvements are expected, we currently have the following candidate hypotheses: (i) the use of the system reduces the workload of the operator during product changeovers and enables shorter task completion time; (ii) the system improves system usability and user experience when compared to hand-held and/or wearable augmented reality devices during assembly and during product changeovers; and (iii) the system allows collaborating with a robot effectively, thereby reducing workload on the operator and increasing robot utilisation time.

The final step of ontology engineering is its evolution over time [54]. The ontology must be maintained and extended when necessary to suit a specific context or application. Careful thought has been put into the development of the information model and MR application to facilitate its reuse, as has already been discussed. The ontology is shared publicly under the Creative Commons Attribution-ShareAlike 4.0 International Licence (CC BY-SA 4.0) online.[9]

9. Conclusion

As HRC gains traction and becomes more commonplace, the need for associated information systems that gain the operator's trust and approval will likely grow. These information systems are expected to facilitate the intuitive role of humans by incorporating social aspects of teamplay while also allowing for flexible and adaptable process flows, and are reported needed by the industry [20]. The core element of such an information system is an information model which models the key concepts of the domain, their relationships, constraints, and rules that describe the semantics of the production and agent contexts. The constrained nature of typical HRC environments also rules out the traditional interfaces employed by these information systems. Therefore, mixed-reality interfaces have been proposed to cope with fading memories of the past and unreliable predictions of the future.

The work presented in this study develops one such information system for human–robot component assembly. We put forth the following as the contributions of the study:

1. The work develops CAMO, an information model for distributed human–robot collaborative multi-agent systems that models the manufacturing, agent, and interaction contexts. The distributed nature of the application of the ontology adds to its novelty as, to the best of our knowledge, existing ontologies for HRC target a centralised architecture. The need for information systems to espouse multi-agent philosophy is prevalent in literature [21,22] and discussed in this study. The design pattern to represent the mental attitudes based on interactions mimics social aptitude, as in human-human interaction, maintains an appreciable degree of self-organisation, and is one that is novel and can be reproduced in similar contexts using logical formalisms.
2. Just as the theoretical motivation of HRC lies in complementing capabilities, as novel elements, the developed work introduces parametric validation of individual capabilities via SHACL shapes that motivate collaborations and maintain an appreciable degree of self-organisation. This is in contrast with existing works in literature that rely on rules and/or consistency checks or ignore them overall, which can be unsuitable for collaborative use cases.
3. In their very recent work, Hayes and Scassellati [6] put forth a question for intention conveyance for HRC: "How can a robot leverage channels of communication that humans understand, despite dissimilar physical forms or capabilities?" This work develops a web-based, component-driven, projector-based mixed-reality application[10] that supports the foregoing information model and takes into account different agent capabilities. Although a proof of concept was developed as part of our earlier work, this study shows the technical feasibility of the solution

[9] https://permanent.link/to/jd-doctoral-dissertation/camo
[10] https://permanent.link/to/jd-doctoral-dissertation/interaction-ui

by means of a case study that a large group of application developers known as ''front-end developers'' are adept at, using scripting languages that report overall faster development time and easier maintenance.

4. Although bounding boxes as an information construct for agent intentions was proposed in our prior work [75], their runtime use for collaborative interactions using the developed MR model is made possible as part of the work reported in this study. The demonstration shows the utility of the developed digital thread framework, which was purposed for knowledge convergence via enterprise systems, for use by collaborative agents for component assembly, and showcases the benefits of integrating product life-cycle data for manufacturing, particularly HRC.
5. From a methodological perspective, this study is an example of how ontology engineering methodologies may be used to build an application ontology for a distributed application. With the exception of a few, most papers in the manufacturing domain present only the result as the developed ontology. Here, methodological steps are documented in detail, which helps with understanding design decisions and rationale.

In this work, we choose to approach HRC from the standpoint of a team working in tandem towards the goal of assembly, considering the needs of the current manufacturing landscape. We allow for the collaboration to "play out" and not be based on pre-planned routines but rather on agent capabilities and runtime message exchanges pertinent to the assembly tasks.

Any attempt to detail the work that made the foregoing contributions in the limited space of an article would be challenging. Nonetheless, we hope the article suffices to appreciate the subtle interplay between philosophy, multi-agent systems, and logic modelling net systems, to effect what may be perceived as intelligence and make the foregoing contributions. In conclusion, as future products and production systems become more complex, information systems will take on greater responsibility to compensate for the inherent limits of the human working memory and enable the transition towards human-centred manufacturing, the likes of which today are labelled as Operator 4.0 and Industry 5.0. The expectation is that information systems research such as the reported work can help take significant strides forward in this direction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The research leading to these results has been funded by the Doctoral School of Engineering Sciences (TTITO) of Tampere University, Finland, to whom the authors are grateful. We would also like to express our sincere gratitude to Dr. Eeva Järvenpää, for her valuable inputs during the initial phases of this study. We also thank the anonymous referees for their useful suggestions.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jmsy.2023.07.013.

References

[1] European Commission and Directorate-General for Research and Innovation, Breque M, De Nul L, Petridis A. Industry 5.0 : towards a sustainable, human-centric and resilient European industry. Publications Office; 2021, http://dx.doi.org/10.2777/308407.
[2] Xu X, Lu Y, Vogel-Heuser B, Wang L. Industry 4.0 and Industry 5.0—Inception, conception and perception. J Manuf Syst 2021;61:530–5. http://dx.doi.org/10.1016/j.jmsy.2021.10.006.
[3] Lu Y, Zheng H, Chand S, Xia W, Liu Z, Xu X, et al. Outlook on human-centric manufacturing towards Industry 5.0. J Manuf Syst 2022;62:612–27. http://dx.doi.org/10.1016/j.jmsy.2022.02.001.
[4] Li S, Wang R, Zheng P, Wang L. Towards proactive human–robot collaboration: A foreseeable cognitive manufacturing paradigm. J Manuf Syst 2021;60:547–52. http://dx.doi.org/10.1016/j.jmsy.2021.07.017.
[5] Wang L. A futuristic perspective on human-centric assembly. J Manuf Syst 2022;62:199–201. http://dx.doi.org/10.1016/j.jmsy.2021.11.001.
[6] Hayes B, Scassellati B. Challenges in shared-environment human-robot collaboration. Learning 8:9.
[7] Uva AE, Gattullo M, Manghisi VM, Spagnulo D, Cascella GL, Fiorentino M. Evaluating the effectiveness of spatial augmented reality in smart manufacturing: a solution for manual working stations. Int J Adv Manuf Technol 2018;94(1):509–21. http://dx.doi.org/10.1007/s00170-017-0846-4.
[8] Andersen RS, Madsen O, Moeslund TB, Amor HB. Projecting robot intentions into human environments. In: 2016 25th IEEE international symposium on robot and human interactive communication. 2016, p. 294–301. http://dx.doi.org/10.1109/ROMAN.2016.7745145.
[9] Lu Y, Xu X, Wang L. Smart manufacturing process and system automation – A critical review of the standards and envisioned scenarios. J Manuf Syst 2020;56:312–25. http://dx.doi.org/10.1016/j.jmsy.2020.06.010.
[10] Prestes E, Carbonera JL, Rama Fiorini S, M. Jorge VA, Abel M, Madhavan R, et al. Towards a core ontology for robotics and automation. Robot Auton Syst 2013;61(11):1193–204. http://dx.doi.org/10.1016/j.robot.2013.04.005, Ubiquitous Robotics.
[11] Niles I, Pease A. Towards a standard upper ontology. In: Proceedings of the international conference on formal ontology in information systems - volume 2001. New York, NY, USA: Association for Computing Machinery; 2001, p. 2–9. http://dx.doi.org/10.1145/505168.505170.
[12] Huang H-M, Messina E, Albus J. Toward a generic model for autonomy levels for unmanned systems (alfus). In: Proceedings of the performance metrics for intelligent systems (PerMIS) workshop. 2003, p. 7.
[13] Juarez A, Bartneck C, Feijs L. Using semantic technologies to describe robotic embodiments. In: Proceedings of the 6th international conference on human-robot interaction. New York, NY, USA: Association for Computing Machinery; 2011, p. 425–32. http://dx.doi.org/10.1145/1957656.1957812.
[14] Lemaignan S, Warnier M, Sisbot EA, Clodic A, Alami R. Artificial cognition for social human–robot interaction: An implementation. Artificial Intelligence 2017;247:45–69. http://dx.doi.org/10.1016/j.artint.2016.07.002, Special Issue on AI and Robotics.
[15] Tenorth M, Beetz M. KNOWROB — knowledge processing for autonomous personal robots. In: 2009 IEEE/RSJ international conference on intelligent robots and systems. 2009, p. 4261–6. http://dx.doi.org/10.1109/IROS.2009.5354602.
[16] Lenat DB. CYC: A large-scale investment in knowledge infrastructure. Commun ACM 1995;38(11):33–8. http://dx.doi.org/10.1145/219717.219745.
[17] Umbrico A, Orlandini A, Cesta A. An ontology for human-robot collaboration. Procedia CIRP 2020;93:1097–102. http://dx.doi.org/10.1016/j.procir.2020.04.045, 53rd CIRP Conference on Manufacturing Systems 2020.
[18] Olivares-Alarcos A, Foix S, Borgo S, Guillem Alenyà. OCRA – An ontology for collaborative robotics and adaptation. Comput Ind 2022;138:103627. http://dx.doi.org/10.1016/j.compind.2022.103627.
[19] Compton M, Barnaghi P, Bermudez L, García-Castro R, Corcho O, Cox S, et al. The SSN ontology of the W3C semantic sensor network incubator group. J Web Semant 2012;17:25–32. http://dx.doi.org/10.1016/j.websem.2012.05.003.
[20] Gustavsson P, Syberfeldt A. The industry's perspective of suitable tasks for human-robot collaboration in assembly manufacturing. IOP Conf Ser Mater Sci Eng 2021;1063(1):012010. http://dx.doi.org/10.1088/1757-899x/1063/1/012010.
[21] Kemény Z, Váncza J, Wang L, Wang XV. Human–robot collaboration in manufacturing: A multi-agent view. In: Wang L, Wang XV, Váncza J, Kemény Z, editors. Advanced human-robot collaboration in manufacturing. Cham: Springer International Publishing; 2021, p. 3–41. http://dx.doi.org/10.1007/978-3-030-69178-3_1.
[22] Bütepage J, Kragic D. Human-robot collaboration: From psychology to social robotics. 2017, http://dx.doi.org/10.48550/ARXIV.1705.10146.
[23] Bauer A, Wollherr D, Buss M. Human–robot collaboration: A survey. Int J Hum Robotics 2008;05(01):47–66. http://dx.doi.org/10.1142/S0219843608001303.
[24] Monostori L, Váncza J, Kumara S. Agent-based systems for manufacturing. CIRP Ann 2006;55(2):697–720. http://dx.doi.org/10.1016/j.cirp.2006.10.004.
[25] Calegari R, Ciatto G, Mascardi V, Omicini A. Logic-based technologies for multi-agent systems: a systematic literature review. Auton Agents Multi-Agent Syst 2020;35(1):1. http://dx.doi.org/10.1007/s10458-020-09478-3.

[26] Wooldridge M, Jennings NR. Intelligent agents: theory and practice. Knowl Eng Rev 1995;10(2):115–52. http://dx.doi.org/10.1017/S0269888900008122.

[27] Zhang Y, Qian C, Lv J, Liu Y. Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor. IEEE Trans Ind Inf 2017;13(2):737–47. http://dx.doi.org/10.1109/TII.2016.2618892.

[28] Wooldridge M. An introduction to multiagent systems. John wiley & sons; 2009.

[29] Ingrand F, Georgeff M, Rao A. An architecture for real-time reasoning and system control. IEEE Expert 1992;7(6):34–44. http://dx.doi.org/10.1109/64.180407.

[30] Georgeff M, Lansky A. Procedural knowledge. Proc IEEE 1986;74(10):1383–98. http://dx.doi.org/10.1109/PROC.1986.13639.

[31] Ljungberg M, Lucas A. The OASIS air-tra c management system. In: Proceedings of the second pacific rim international conference on artificial intelligence, vol. 92. 1992, p. 185–90.

[32] Tidhar G, Heinze C, Selvestrel M. Flying together: Modelling air mission teams. Appl Intell 1998;8(3):195–218. http://dx.doi.org/10.1023/A:1008271016283.

[33] d'Inverno M, Kinny D, Luck M, Wooldridge M. A formal specification of dMARS. In: Singh MP, Rao A, Wooldridge MJ, editors. Intelligent agents IV agent theories, architectures, and languages. Berlin, Heidelberg: Springer Berlin Heidelberg; 1998, p. 155–76.

[34] Rao AS, Georgeff MP. BDI agents: From theory to practice. In: ICMAS. 1995.

[35] Braubach L, Pokahr A, Lamersdorf W. Jadex: A BDI-agent system combining middleware and reasoning. In: Unland R, Calisti M, Klusch M, editors. Software agent-based applications, platforms and development kits. Basel: Birkhäuser Basel; 2005, p. 143–68.

[36] Shoham Y. Agent-oriented programming. Artificial Intelligence 1993;60(1):51–92. http://dx.doi.org/10.1016/0004-3702(93)90034-9.

[37] Weiss G. Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press; 1999 [Chapter 1].

[38] Cohen PR, Levesque HJ. Teamwork. Nous 1991;25(4):487–512.

[39] David J, Lobov A, Järvenpää E, Lanz M. Enabling the digital thread for product aware human and robot collaboration - An agent-oriented system architecture. In: 2021 20th international conference on advanced robotics. 2021, p. 1011–6. http://dx.doi.org/10.1109/ICAR53236.2021.9659352.

[40] David J, Järvenpää E, Lobov A. A web-based mixed reality interface facilitating explicit agent-oriented interactions for human-robot collaboration. In: 2022 8th International conference on mechatronics and robotics engineering. 2022, p. 174–81. http://dx.doi.org/10.1109/ICMRE54455.2022.9734094.

[41] Rocca GL. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Adv Eng Inform 2012;26(2):159–79. http://dx.doi.org/10.1016/j.aei.2012.02.002, Knowledge based engineering to support complex product design.

[42] David J, Järvenpää E, Lobov A. Digital threads via knowledge-based engineering systems. In: 2021 30th conference of open innovations association FRUCT. 2021, p. 42–51. http://dx.doi.org/10.23919/FRUCT53335.2021.9599986.

[43] Bellifemine FL, Caire G, Greenwood D. Developing multi-agent systems with JADE. Wiley series in agent technology, Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2007.

[44] React – A JavaScript library for building user interfaces. 2022, https://reactjs.org/. [Accessed 25 November 2022].

[45] Redux - A predictable state container for JavaScript apps. | Redux. 2022, https://redux.js.org/. [Accessed 25 November 2022].

[46] Zhang F, Bazarevsky V, Vakunov A, Tkachenka A, Sung G, Chang C-L, et al. MediaPipe hands: On-device real-time hand tracking. 2020, http://dx.doi.org/10.48550/ARXIV.2006.10214.

[47] ISO/IEC 15909-2:2011(en). Systems and software engineering — High-level Petri nets — Part 2: Transfer format. Standard, International Organization for Standardization; 2011.

[48] Tenorth M. Knowledge processing for autonomous robots [Dissertation], München: Technische Universitünchen; 2011.

[49] Obitko M, Mařík V. Adding OWL semantics to ontologies used in multi-agent systems for manufacturing. In: Mařík V, McFarlane D, Valckenaers P, editors. Holonic and multi-agent systems for manufacturing. Berlin, Heidelberg: Springer Berlin Heidelberg; 2003, p. 189–200.

[50] Chen H, Perich F, Finin T, Joshi A. SOUPA: standard ontology for ubiquitous and pervasive applications. In: The First annual international conference on mobile and ubiquitous systems: Networking and services, 2004. 2004, p. 258–67. http://dx.doi.org/10.1109/MOBIQ.2004.1331732.

[51] Studer R, Benjamins VR, Fensel D. Knowledge engineering: principles and methods. Data Knowl Eng 1998;25(1–2):161–97.

[52] Gruber TR. A translation approach to portable ontology specifications. Knowl Acquis 1993;5(2):199–220. http://dx.doi.org/10.1006/knac.1993.1008.

[53] Iqbal R, Murad MAA, Mustapha A, Sharef NM, et al. An analysis of ontology engineering methodologies: A literature review. Res J Appl Sci Eng Technol 2013;6(16):2993–3000.

[54] Sure Y, Staab S, Studer R. Handbook on ontologies, no. II. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009, p. 135–52 http://dx.doi.org/10.1007/978-3-540-92673-3_6, [Chapter Ontology Engineering Methodology],

[55] Benarous X, Mikita N, Goodman R, Stringaris A. Distinct relationships between social aptitude and dimensions of manic-like symptoms in youth. Eur Child Adolesc Psychiatry 2016;25(8):831–42.

[56] van Heijst G, Schreiber A, Wielinga B. Using explicit ontologies in KBS development. Int J Hum-Comput Stud 1997;46(2):183–292. http://dx.doi.org/10.1006/ijhc.1996.0090.

[57] Horridge M, Drummond N, Goodwin J, Rector AL, Stevens R, Wang H. The manchester OWL syntax. In: OWLed, vol. 216. 2006.

[58] Hitzler P, et al. Dolce+ D&S Ultralite and its main ontology design patterns. Ontog Eng Ontog Des Patterns Found Appl 2016;25:81.

[59] Borgo S, Ferrario R, Gangemi A, Guarino N, Masolo C, Porello D, et al. DOLCE: A descriptive ontology for linguistic and cognitive engineering1. Appl Ontol 2022;17(1):45–69. http://dx.doi.org/10.3233/AO-210259.

[60] Gangemi A, Mika P. Understanding the semantic web through descriptions and situations. In: Meersman R, Tari Z, Schmidt DC, editors. On the move to meaningful internet systems 2003: CoopIS, DOA, and ODBASE. Berlin, Heidelberg: Springer Berlin Heidelberg; 2003, p. 689–706.

[61] Borgo S, Carrara M, Garbacz P, Vermaas PE. A formal ontological perspective on the behaviors and functions of technical artifacts. Artif Intell Eng Des Anal Manuf 2009;23(1):3–21. http://dx.doi.org/10.1017/S0890060409000079.

[62] Järvenpää E, Siltala N, Hylli O, Lanz M. The development of an ontology for describing the capabilities of manufacturing resources. J Intell Manuf 2019;30(2):959–78.

[63] O'Brien PD, Nicol RC. FIPA—towards a standard for software agents. BT Technol J 1998;16(3):51–9.

[64] W3C. Semantic web case studies and use cases. 2023, https://www.w3.org/2001/sw/sweo/public/UseCases/. [Accessed 28 April 2023].

[65] W3C. Shapes Constraint Language (SHACL). Recommendation, World Wide Web Consortium; 2017, https://www.w3.org/TR/shacl/.

[66] Apache Software Foundation. Apache jena. 2022, https://jena.apache.org/. [Accessed 25 November 2022].

[67] Helms E, Schraft R, Hagele M. Rob@work: Robot assistant in industrial environments. In: Proceedings. 11th IEEE international workshop on robot and human interactive communication. 2002, p. 399–404. http://dx.doi.org/10.1109/ROMAN.2002.1045655.

[68] FIPA contract net interaction protocol specification. 2022, http://www.fipa.org/specs/fipa00029/SC00029H.html. [Accessed 26 November 2022].

[69] Galigator. Openllet: An open source OWL DL reasoner for Java. 2022, GitHub repository, GitHub, https://github.com/Galigator/openllet. [Accessed 27 November 2022].

[70] Searle JR, Searle JR. Speech acts: An essay in the philosophy of language, vol. 626. Cambridge University Press; 1969.

[71] Wrigstad T, Nardelli FZ, Lebresne S, Östlund J, Vitek J. Integrating typed and untyped code in a scripting language. SIGPLAN Not 2010;45(1):377–88. http://dx.doi.org/10.1145/1707801.1706343.

[72] iig-uni-freiburg. WOLFGANG - Petri net editor. 2022, GitHub repository, GitHub, https://github.com/iig-uni-freiburg/WOLFGANG. [Accessed 27 November 2022].

[73] Kalpagam Ganesan R, Rathore YK, Ross HM, Ben Amor H. Better teaming through visual cues: How projecting imagery in a workspace can improve human-robot collaboration. IEEE Robot Autom Mag 2018;25(2):59–71. http://dx.doi.org/10.1109/MRA.2018.2815655.

[74] Gualtieri L, Monizza GP, Rauch E, Vidoni R, Matt DT. From design for assembly to design for collaborative assembly - Product design principles for enhancing safety, ergonomics and efficiency in human-robot collaboration. Procedia CIRP 2020;91:546–52. http://dx.doi.org/10.1016/j.procir.2020.02.212, Enhancing design through the 4th Industrial Revolution Thinking.

[75] David J, Coatanéa E, Lobov A. Projecting product-aware cues as assembly intentions for human-robot collaboration. In: Kim K-Y, Monplaisir L, Rickli J, editors. Flexible automation and intelligent manufacturing: The human-data-technology nexus. Cham: Springer International Publishing; 2023, p. 146–59.