CPROSA-Holarchy: An Enhanced PROSA Model to Enable Worker–Cobot Agile Manufacturing

Ahmed R. Sadik and Bodo Urban
Fraunhofer Institute for Computer Graphic Research-IGD, Rostock 18059, Germany
University of Rostock, Rostock 18057, Germany
Email: ahmed.sadik@igd-r.fraunhofer.de, bodo.urban@igd-r.fraunhofer.de

***Abstract*—This research combines two important concepts of intelligent manufacturing: agile manufacturing and collaborative robotics. On the one hand, agility in manufacturing is the capability of an industrial enterprise to respond rapidly and effectively to unanticipated changes that occur in the production. The aim of agile manufacturing is to proactively develop solutions to adapt to the customers' needs. These solutions are a result of collective decision-making that has been formed among the different entities of the agile manufacturing system. On the other hand, collaborative robotics is a new trend in industrial robotics, which involves a collaborative robot (cobot). A cobot is usually an industrial robot designed to operate safely in a shared work environment with human workers. This is in contrast with the conventional industrial robot that operates in isolation from the workers' workspace. One of the most important advantages of collaborative robotics is the increase of the agility of manufacturing. Therefore, in this research, we focus on developing a proper information control and communication solution to facilitate worker–cobot agile manufacturing. Furthermore, we introduce a case study of two workers in cooperation with one cobot to demonstrate the solution concept.**

***Index Terms*—Agile Manufacturing, Collaborative Robotics, Holonic Control Architecture, Ontology-Based Communication, Autonomous Reactive Agent, Multiagent System.**

I. Introduction

Human–Robot Interaction (HRI) is an attractive field of study for many scholars from different research domains. HRI focuses on understanding, designing, and evaluating robotic systems that involve humans as an essential element [1]. HRI has literature roots, as its fundamentals have been stated by the author Isaac Asimov in his novel "I, Robot" [2]. The first two fundamentals are as follows:

- A robot may not injure a human being or, through interaction, allow a human being to come to harm.
- A robot must obey the orders that are given to it by the human beings.

Those two fundamentals became later an inspiration for the scientific community to formulate the main HRI problems. Scientifically speaking, the first fundamental addresses the problem of a safe physical HRI, whereas the second fundamental addresses the problem of an HRI information communication and control.

In industry, safe physical HRI has attracted a great deal of attention in the last decades [3]. As a result of this attention, a new generation of safe cobots is now available on the commercial market. Examples of these cobots are KUKA Lightweight, Rethink-Baxter, YuMi-ABB, and Universal Robots. These cobots adopt different technologies and methodologies to ensure the safe cooperation with the human coworker [4]. On the contrary, the second HRI fundamental concerning information control still did not get the proper amount of attention, although it is as much important as the first fundamental rule. This motivated us to carry out more research on the second HRI fundamental during this work.

The next section of this paper highlights the research preliminaries. Those preliminaries are then used to formulate the research problems in Section III. Section IV introduces in details the solution concept; therefore, a case study is illustrated in Section V. At the end, Section VI discusses and summarizes the research to wrap it up with the conclusion and future work.

II. Research Preliminaries

A. Holonic Control Architecture (HCA)

In the late sixties, the term holon was introduced for the first time by philosopher Koestler [5]. Koestler developed this term as a basic unit in his explanation of the evolution of the biological and social structures. Based on his observations, organisms (e.g., biological cells) are autonomous self-reliant units, which have a certain degree of independent control on their actions, yet they are still subject to a higher level of control instructions. His conclusion was that any organism is a whole ("holos") and a part ("on") at the same time, which derived the term holon [6]. The concept of holon has been

Manuscript received June 2nd, 2017; revised December 30th, 2017.

doi: 10.18178/ijmerr.7.3.296-304

adopted in the early nineties by the Intelligent Manufacturing Systems (IMS) consortium to define a new paradigm for the factory of the future. The following terms have been defined by the IMS to provide a better understanding of the HCA [7]:

- Holon: an autonomous cooperative building block of the manufacturing system that can be used to transform, transport, store, and/or validate the information and the physical signals.
- Autonomy: the capability of the holon to create and control the execution of its own plans.
- Cooperation: a process in which a set of holons develop mutual plans and execute them together.
- Holarchy: a system of holons that cooperate to achieve a goal or an objective. The holarchy defines the basic rules for cooperation of the holons and thereby limits their autonomy.

HCA is a distributed control and communication topology that divides the manufacturing process tasks and responsibilities over different holon categories. Product-Resource-Order-Staff-Architecture (PROSA) is the most known HCA model [8]. PROSA's basic holons are as follows:

- Product Holon (PH): responsible for processing and storing the different production plans required to insure the correct manufacturing of a certain product.
- Order Holon (OH): responsible for composing and managing the product orders. Furthermore, in a small-scale enterprise, it should assign the tasks to the existing operating resources and monitor the execution status of the assigned tasks.
- Operational Resource Holon (ORH): a physical entity within the manufacturing system, which can represent a robot, a machine, a worker, and so forth.

B. Autonomous Reactive Agent

A software agent is a computer system that is situated in a specific environment that is capable of performing autonomous actions in this environment in order to meet its design objective [9]. An agent is autonomous by nature; this means an agent operates without a direct intervention of the humans and has a high degree of controlling its actions and internal states. To achieve this autonomy, an agent must fulfill the following features:

- Responsive: ability to perceive the environment and respond in a timely fashion to the changes occurring in it.
- Proactive: ability to exhibit opportunistic, goal-directed behaviors and to take initiatives.
- Social: ability to interact with other artificial agents or humans within its environment in order to solve a problem.

Conceptually, an agent is a computing machine that is given a specific problem to solve [10]. Therefore, it chooses a certain set of actions and formulates the proper plans to accomplish the assigned task. The set of actions that are available to be performed by the agent is called a behavior. The agent's behaviors are mainly created by the agent programmer. An agent can execute one or more behaviors to reach its target. The selection of an execution behavior among others is based on a certain criterion that has been defined by the agent programmer. Building an execution plan highly depends on the information that the agent infers from its environment, including the other agents. A Multiagent System (MAS) is a collective system that is composed of a group of artificial agents, teaming together in a flexible distributed topology, to solve a problem beyond the capabilities of a single agent.

JAVA Agent Development Environment (JADE) is a distributed MAS middleware framework [11]. Each JADE instance is an independent thread that contains a set of containers. A container is a group of agents run under the same JADE runtime instance. Every platform must contain a main container. The main container contains two necessary agents: an Agent Management System (AMS) and a Directory Facilitator (DF). The AMS provides a unique ID for every agent under its platform to be used as an agent communication address, whereas the DF announces the services that agents can offer under its platform, to facilitate the agents' services exchange, so that every agent can reach its specific goal [12]. JADE applies the reactive agent architecture, which complies with the Foundation for Intelligent Physical Agents (FIPA) specifications and provides a graphical interface to deploy and debug an MAS [13]. FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies [14]. JADE agents use FIPA-Agent Communication Language (FIPA-ACL) to exchange messages either inside its own platform or with another platform in a distributed MAS.

III. Problem Formulation

The PROSA model is a common solution for small-scale agile manufacturing systems. Yet, the PROSA model cannot achieve worker–cobot cooperative agile manufacturing, for the following reasons:

- Although the PROSA model aims to adapt to the fast changeability in the market demands, it does not address any holon for the customer, which isolates the current model from the direct problem that it tries to solve. Hence, the model must be able to continuously comprehend the variations in the required production customization level and rate.
- The PROSA model discusses very briefly the functionality of its basic holons. However, it does not explicitly describe the nature and the mechanism of the interaction between the holons. In a worker–cobot interaction scenario, there is a need to provide the metadata required for supporting the cooperation. Descriptive metadata is needed to give a meaning for the

tasks and the operations, which can be done during the cooperation. Structural metadata is required to indicate how to compound new objects from existing objects. Finally, administrative metadata is required to control the task assignment and the cooperation planning, management, and execution.

- The PROSA model does not take into consideration the fact that, in case of worker cooperation with a cobot, a common language that can be understood by both the human and the machine must be considered as the proper way of communication.
- The PROSA model does not provide a solution for the stochastic variation in the time needed from the worker to complete a specific task. This stochastic time variation comes from the fact that the very same worker will always take a different time to complete the very same task. This factor will dramatically increase if we take into consideration different workers who perform the same task.

Therefore, during this research, we focus on enhancing the PROSA model by solving the previously mentioned problems. Hence, the enhanced PROSA model can fulfill the requirements of the worker–cobot cooperative agile manufacturing.

IV. Solution Concept

A. CPROSA-Holarchy

In the context of cooperative agile manufacturing, we found that it is necessary to modify the PROSA model [15]. Therefore, a new holon is introduced by this research, which is the Customer Holon (CH). The CH is deployed on the customer platform to provide a User Interface (UI) for the customer to select and customize the product order. Furthermore, it interacts with the PH to trigger a new production order. Therefore, we will refer to our modified PROSA holarchy as CPROSA, which can be seen in Fig. 1.

While the HCA is a conceptual model that focuses on the holons' functionalities and responsibilities, it does not specify a certain technology to apply that concept. The autonomous agent technology is a general-purpose solution that can implement the HCA [16]. Thus, during this research, the JADE framework has been selected to implement the proposed CPROSA-holarchy. JADE empowers object oriented concepts such as abstraction and inheritance, which makes it very suitable for applying the CPROSA-holarchy. For example, a Worker Holon (WH) can have many different instances originating from it, yet every WH instance can act differently from the others. Figure 1 illustrates the main concept to implement an agile cooperative manufacturing workcell, which can contain different workers and cobots as operational resources; the manufacturing workcell can simultaneously process different customized orders from various customers.

Using JADE to implement the CPROSA-holarchy empowers another very strong concept, which is the agent communication via ontology [17]. The CPROSA-holarchy implements many different objects. This not only obligates the holons to send or receive messages with objects' content but also obligates them to express relations between these objects and perform actions over them [18].

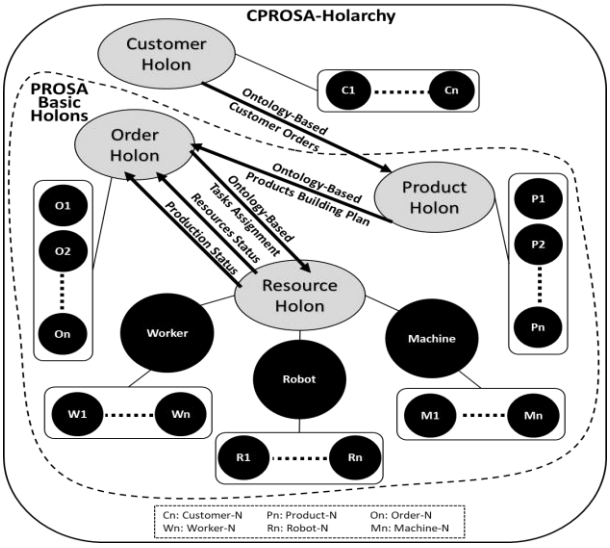

Figure 1. CPROSA-holarchy.

The term "ontology" can be considered sometimes vague and not precise; therefore, we state below the most suitable definitions of ontologies for our research [19].

- An ontology defines the terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions for this vocabulary [20].
- An ontology is a formal, explicit specification of a shared conceptualization [21].
- An ontology is a logical theory accounting for the intended meaning of a formal vocabulary [22]; that is, "it is a commitment to a particular conceptualization of the world."
- An ontology [23] provides the meta-information to describe the data semantics, represent knowledge, and communicate with various types of entities (e.g., software agents and humans).
- An ontology can be described as a "means of enabling communication and knowledge sharing by capturing a shared understanding of terms that can be used both by humans and by machine software" [24].

All the previous definitions draw the complete picture of the ontology's meaning in the context of our research. Thus, an ontology is a conceptual tool to represent and create a common understanding for the manufacturing workcell entities (i.e., holons). Furthermore, this common understanding would enable the exchange, reuse, and extension of the manufacturing knowledge. JADE

supports an ontology-based MAS by defining three types of schemas:

- Terms: indicate entities that exist in the MAS and that agents may reason about. Terms can be seen as primitives, which are atomic data types such as strings or integers, and concepts, which are complex structures such as objects.
- Predicates: describe the status of the world and the relationships between the concepts.
- Actions: describe mechanisms or operations that can be executed by an agent.

B. CPROSA-Holarchy Interaction Model

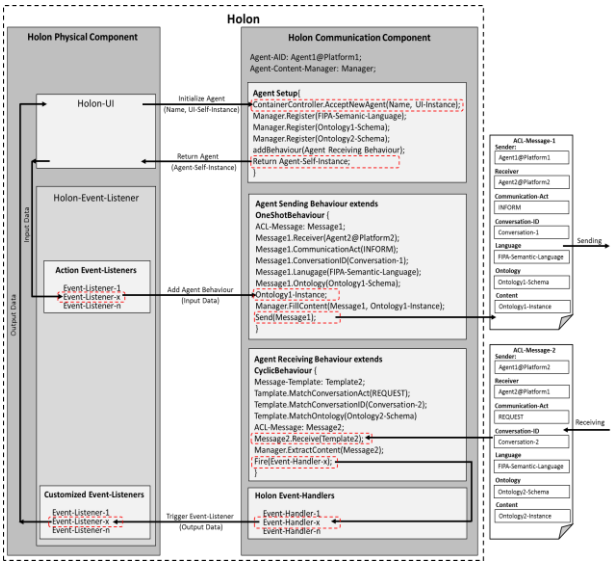

Figure 2. CPROSA-holarchy interaction model.

Figure 2 shows the interaction model that has been used in the CPROSA-holarchy. Each holon is composed of two components. The first component is the physical component, which is responsible for dealing with the input/output (I/O) data and events. The second component is the communication, which is responsible for sending, receiving, and processing information. The physical component can be presented as a UI, whereas the communication component can be implemented as a JADE agent [25]. Every JADE agent must have an Agent Identification (AID). The AID is a unique agent name over a specific platform, and it is used as an address for the agent to send or receive the ACL messages. In addition, every agent must have a setup method. The setup method is automatically triggered after the agent creation. The main function of the setup method is to initialize the required parameters and behaviors needed to perform the agent tasks.

In order to complete a communication process between two agents, one agent must have a behavior that is responsible for constructing and sending an ACL message, and the other agent must have a behavior that is responsible for receiving and using the ACL message. An ACL message is composed of a variety of fields, and it must have at least a sender and a receiver AID. Other fields such as the Communication-Act and the Conversation-ID are necessary to distinguish the message at the receiver side. The Communication-Acts define the message in terms of standard FIPA actions or functions; for instance, a Communication-Act field can contain INFORM, REQUEST, CONFIRM, and so forth. The Conversation-ID is a unique string to distinguish or record a specific conversation topic or thread between the agents.

In a simple agent conversation, the content field of an ACL message contains a string data type. However, in a complex agent conversation, an ontology-based content will be the proper method. In order to communicate via agent ontologies, every agent must deploy a content manager. The content manager registers a common language of conversation between the agents. The FIPA Semantic Language (FIPA-SL) is not mandatory but preferable in a complex JADE conversation. FIPA-SL is a human-readable language, which defines the syntactic rules needed to parse or encode an ontology-based content. In addition, the content manager registers the common ontology schemas. Ontology schemas define the abstract structure pattern and the semantics needed to construct or interpret an ontology-based ACL message. At the sender side, the content manager checks the semantics of the sent ACL message based on a common ontology schema and decodes that message into a stream of bytes via FIPA-SL. At the receiver side, the content manager parses the received ACL message into a human-readable content via FIPA-SL and then structures it on the basis of a common ontology schema.

V. Case Study

C. Case-Study Description

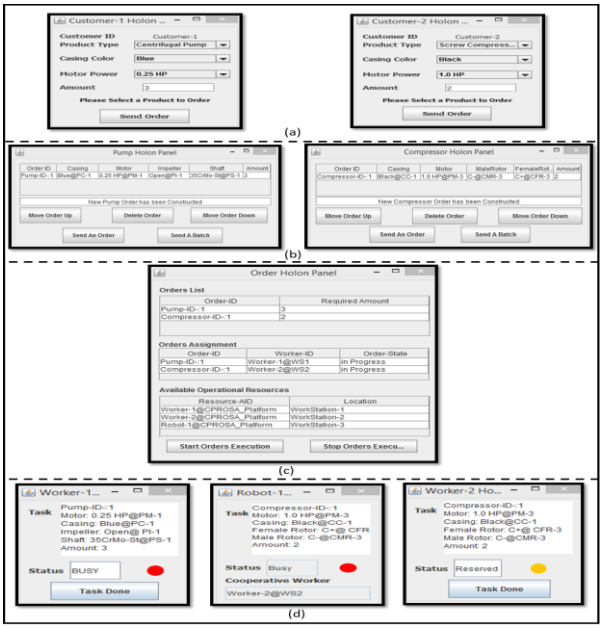

Figure 3. (a) CH UI, (b) PH UI, (c) OH UI, and (d) ORH UI.

During this research, we selected a specific case study

where two workers are in cooperation with one cobot. The goal of this case study is to implement the solution concept. The CPROSA-holarchy deployment contains four containers that present the previously described CPROSA holons. Two CHs can be found in this case study as seen in Fig. 3(a). Both CHs have a similar UI. The UI of the CH provides a tool for ordering a specific product with certain features (i.e., parts). The customer selects the basic features and defines the needed amount of the product and then sends the order to the PH.

Two products can be manufactured in this case study: a centrifugal pump and a screw compressor. The UIs of the pump and the compressor holons can be seen in Fig. 3(b). These two products share some features, such as the casing and the electrical motor. The pump has two unique features (the impeller and the shaft), whereas the compressor has another two unique features (the male rotor and the female rotor). When a PH receives a product order from the CH, it constructs the building plans for this product order, as will be discussed later in details. The PH also has the ability to rearrange the orders or modify them before sending them to the OH.

The OH is responsible for collecting the product orders from all the other PHs, as shown in Fig. 3(c). Simultaneously, the OH discovers the existence of the operation resources. Furthermore, it starts and stops the production process. Two WHs (WH1 and WH2) and one Robot Holon (RH) can be found as operation resources in this implementation, as shown in Fig. 3(d). The function of the workers within this case study is to perform an assembly operation of the customized product orders, whereas the function of the cobot is to pick and place the customized features of every production order to the worker workstation.

As we did not have robot hardware during this implementation, we assumed that the cobot always takes two seconds to pick and place one product. Therefore, the RH multiplies the number of products by two to obtain the overall time needed for the whole pick-and-place operations. Accordingly, the RH can have either of two statuses: busy or free. Another status is required for the WH, which is a reserve status. In the reserve status, the WH waits for the cobot to load at least one product to the worker; therefore, the worker can start the assembly operation and subsequently the WH status turns to busy. The WH stays in the busy status until the worker presses the task-done button; then, the WH status becomes free.

A. Case-Study Ontology

As discussed earlier in the solution concept, JADE uses three different types of schemas to construct its ontology. Figure 4 shows all the required schemas used to build the case-study ontology. The first set of schemas, which can be seen in Fig. 4 is the terms (i.e., concepts and primitives):

- Compressor-Customer-Order: a schema that encapsulates some attributes such as the required color, the needed hydraulic power, and the required amount. In addition, it contains an AID as every customer order is an agent that needs an ID.
- Pump-Customer-Order: a schema that encapsulates some attributes such as the required color, the needed hydraulic power, and the required amount. In addition, it contains an AID as every customer order is an agent that needs an ID.
- Casing: a shared feature between the pump and the compressor. The casing schema contains two attributes, which are the casing color and the position at the features storage space.
- Electrical motor: a shared feature between the pump and the compressor. The motor schema contains two attributes, which are the motor electrical power and the position at the features storage space.
- Shaft: a unique feature of the pump. The shaft schema contains two attributes, which are the shaft material and the position at the features storage space.
- Impeller: a unique feature of the pump. The impeller schema contains two attributes, which are the impeller type and the position at the features storage space.
- Female rotor: a unique feature of the compressor. The female-rotor schema contains two attributes, which are the rotor size and the position at the features storage space.
- Male rotor: a unique feature of the compressor. The male-rotor schema contains two attributes, which are the rotor size and the position at the features storage space.
- Compressor: a concept schema that encapsulates many other schemas under it; those schemas are the casing, electrical motor, female rotor, and male rotor. Every compressor is an agent; therefore, it must contain an AID.
- Pump: a concept schema that encapsulates many other schemas under it; those schemas are the casing, electrical motor, shaft, and impeller. Every pump is an agent; therefore, it must contain an AID attribute.
- Compressor-Order: a schema that extends the compressor schema by adding the required amount of units.
- Pump-Order: a schema that extends the pump schema by adding the required amount of units.
- Operations-List: a schema that includes a list of operations that can be used to manufacture either a pump or a compressor. This schema can be used to manufacture a product that needs three operations or less.
- Compressor-Manufacturing-Order: a schema that combines the Compressor-Order schema and Operations-List schema. In addition, it has an AID attribute as it acts as an agent.

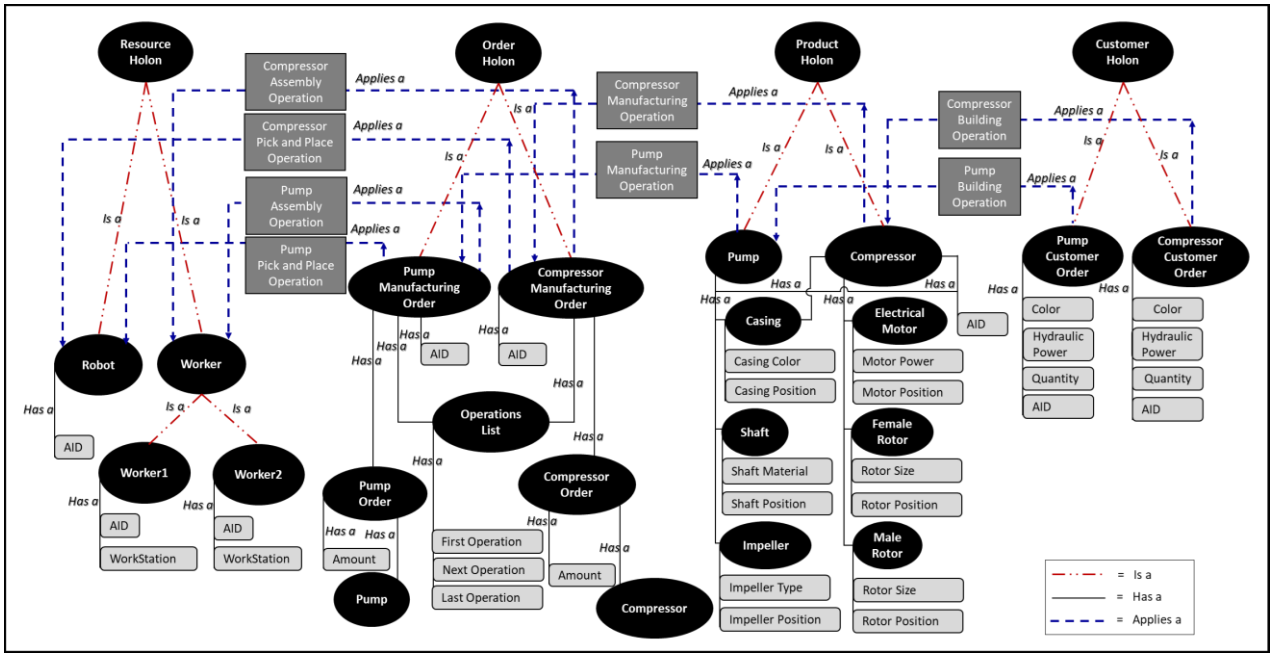

Figure 4. Case-Study Ontology.

- Pump-Manufacturing-Order: a schema that combines the Pump-Order schema and Operations-List schema. In addition, it has an AID attribute as it acts as an agent.
- Worker: a schema that contains two attributes: the first one is the worker AID as it acts as a life agent and the second one is the worker location within the workcell (i.e., workstation). The worker agent provides the worker with a UI in order to provide the assigned task and inquire about the task-done event [see Fig. 3(d)]. Two instances of the worker agent exist in this case-study scenario. As has been mentioned before in Section V.A, the worker can have three statuses: (1) a free status when there are no product orders or the production is not started, (2) a reserve status when the worker is waiting for the first product unit to be placed by the cobot, and (3) a busy status when the cobot is still handling the orders and until the worker triggers the task-done button.
- Robot: a schema that contains one attribute, which is the robot AID, as it acts as a life agent. The robot schema does not have a workstation attribute because, in this specific case study, we have one cobot that is responsible for the pick-and-place operations. Therefore, the location of the cobot is not necessarily required; however, in case there is more than one cobot, this attribute could be important. The robot agent provides a UI to show the assigned task and the status of the cobot [see Fig. 3(d)]. As has been mentioned before in Section V.A, the cobot can have two statuses: (1) a free status when there are no product orders or the production is not started and (2) a busy status when the cobot is picking and placing the product orders. A timer of two seconds has been assigned to every pick-and-place operation.

The second set of schemas that can be seen in Fig. 4 is the predicate schemas that are addressed as follows:

- (concept-x) <Is-a> (concept-y): usually a relation between two concept schemas. This relation is similar to the object oriented abstraction. Thus, this predicate expression has been used to express the parent–child relationship between the concepts.
- (concept-x) <Has-a> (attribute-x): usually a relation between a concept and an attribute. An attribute can be a concept schema or a primitive. This relation is similar to object oriented inheritance. Thus, this expression is used to form sophisticated objects from simpler ones.
- (agent-x) <Applies-a> (action-x): usually a relation between a concept and an action schema. A concept uses this predicate expression to trigger one or more actions at the same time. The action schemas will be discussed below in details.

The third set of schemas that can be seen in Fig. 4 is the action schemas that are addressed as follows:

- Pump-Building-Operation: this action schema expects a Pump-Customer-Order concept as an input, and it can be deployed by either customer-1 or customer-2 agents. An example of this operation can be seen in the ACL-message content in Fig. 5(a).

- Compressor-Building-Operation: this action schema expects a Compressor-Customer-Order concept as an input, and it can be deployed by either customer-1 or customer-2 agents. An example of this operation can be seen in the ACL-message content in Fig. 5(b).
- Pump-Manufacturing-Operation: this action schema expects a Pump-Order and a Pump-Operations-List concept schema as an input, and it is deployed by the pump agent. A detailed example of this operation can be seen in the ACL-message content in Fig. 5(c).
- Compressor-Manufacturing-Operation: this action schema expects a Compressor-Order and a Compressor-Operations-List concept schema as an input, and it is deployed by the compressor agent. A detailed example of this operation can be seen in the ACL-message content in Fig. 5(d).
- Pump-Pick-and-Place-Operation: this action schema expects two concept schema inputs; the first concept schema input is the Pump-Order, which contains the detailed specifications of the pump. Therefore, the cobot can use this information, especially the pump features' positions, to perform the pick operation. The second concept schema input is the target worker. Therefore, the cobot can use the worker workstation location to place the pump features at this location. This action schema is deployed by the orders agent to assign a task to the robot agent. A detailed example of this operation can be seen in the ACL-message content in Fig. 5(e).
- Compressor-Pick-and-Place-Operation: this action schema expects two concept schema inputs; the first concept schema input is the Compressor-Order, which contains the detailed specifications of the compressor. Therefore, the cobot can use this information, especially the compressor features' positions, to perform the pick operation. The second concept schema input is the target worker. Therefore, the cobot can use the worker workstation location to place the compressor features at this location. This action schema is deployed by the orders agent to interact with the robot agent. A detailed example of this operation can be seen in the ACL-message content in Fig. 5(f).
- Pump-Assembly-Operation: this action schema expects one concept schema input, which is the Pump-Order. This operation is beneficial for the worker to provide him with the required features to build a customized pump. Moreover, it provides the amount of the required units. This action schema is deployed by the orders agent to assign a task to any of the worker agents on the basis of their status. A detailed example of this operation can be seen in the ACL-message content in Fig. 5(g).
- Compressor-Assembly-Operation: this action schema expects one concept schema input, which is Compressor-Order. This operation is beneficial for the worker to provide the know-how of building a customized compressor. Moreover, it provides the amount of the required units. This action schema is deployed by the orders agent to assign a task to any of the worker agents on the basis of their status. A detailed example of this operation can be seen in Fig. 5(h).

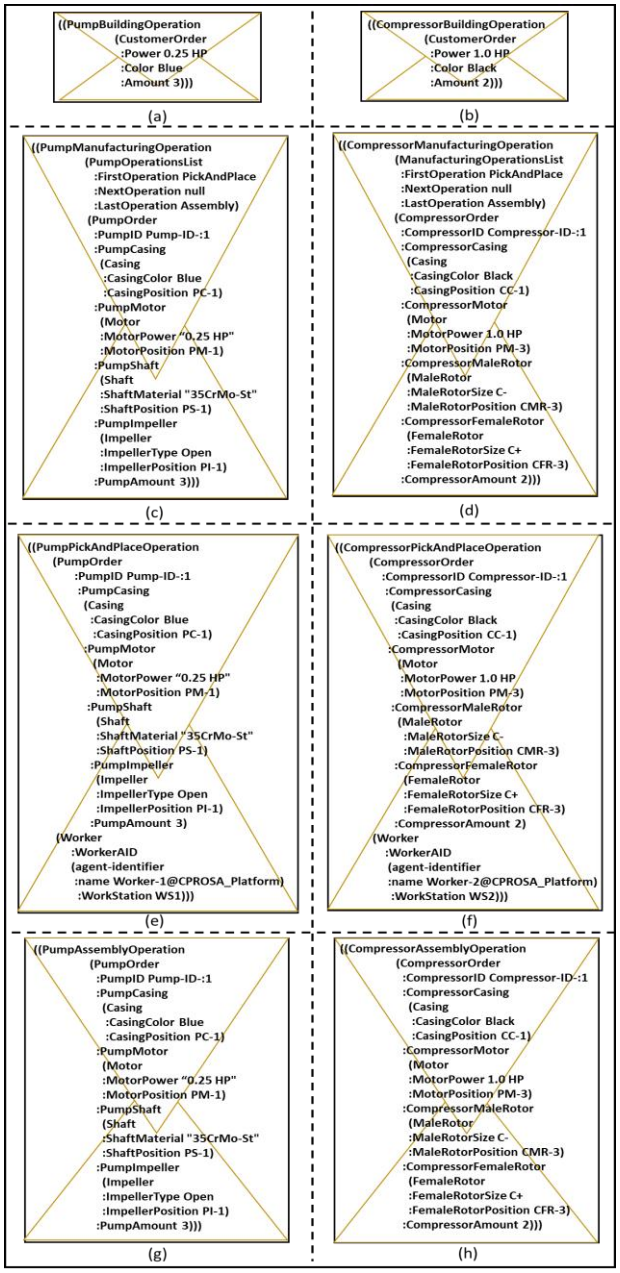

Figure 5. Ontology-based ACL messages used during the case study.

B. Case-Study Interaction Scenario

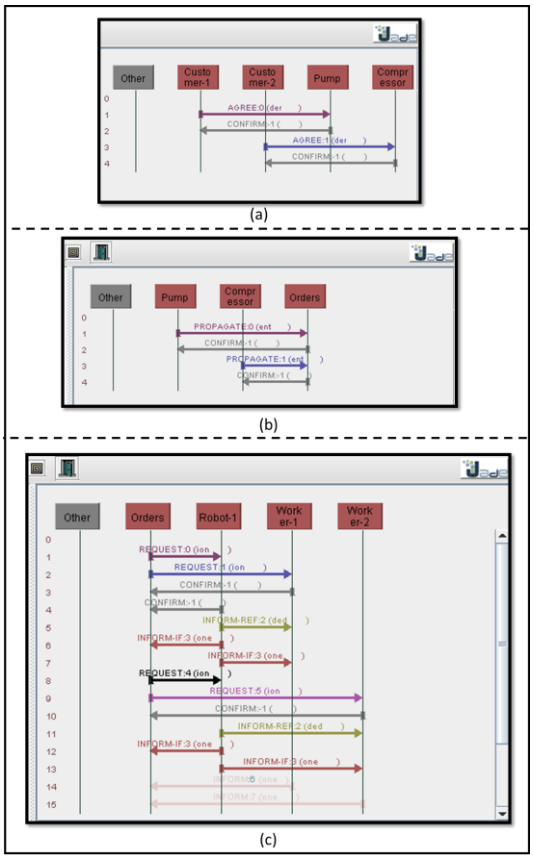

Figure 6. Case-study interaction scenario.

Figure 6(a) shows a JADE interaction scenario among the CHs (i.e., customer-1 agent and customer-2 agent) and the PHs (i.e., pump agent and compressor agent). In this scenario, customer-1 agent sends an ACL message with an AGREE communicative act. The AGREE message contains a Pump-Building-Operation and a Pump-Customer-Order. This AGREE message is received by the pump agent. Therefore, the pump agent confirms the receipt by sending back a CONFIRM message to customer-1 agent. Simultaneously, the pump agent constructs a pump instance based on the incoming customer order. The same mechanism is used between customer-2 agent and the compressor agent to construct a new instance of a compressor associated with a customer-2 order. Figure 6(b) shows a JADE interaction scenario between the PHs (i.e.,., pump agent and compressor agent) and the OH. This interaction follows the same mechanism used before in Fig. 6(a), except that it replaces the AGREE messages with PROPAGATE messages.

Figure 6(c) shows a JADE interaction scenario between the OH and the ORHs (i.e., worker1 agent, worker2 agent, and robot agent). During this interaction, the manufacturing operations are assigned to the operational resources on the basis of their status. As it can be seen in lines 1, 2, 3, and 4 of Fig. 6(c), the orders agent sends two REQUEST messages that are replied to by two CONFIRM messages. The first REQUEST message assigns a Pump-Pick-and-Place-Operation to the robot agent. The second REQUEST message assigns a Pump-Assembly-Operation to worker1 agent. The reason why the Pump-Order has been processed first by the orders agent is that it is the first product order in the order list [refer to Fig. 3(c)]. In line 5 of Fig. 6(c), the robot agent sends an INFORM-REF message to worker1 agent to confirm that it placed the first pump unit. Then, the robot agent sends two INFORM-IF messages to the orders agent and worker1 agent to confirm that it finished handling all the required pump amounts [i.e., three pump units by referring to Figs. 3(a), 3(b), and 3(d)]. The two INFORM-IF messages can be seen in lines 6 and 7 of Fig. 6(c). The same interaction mechanism can be seen in lines 9, 10, 11, 12, and 13 to assign the Compressor-Order manufacturing operations to the worker2 agent and the robot agent. Lines 14 and 15 in Fig. 6(c) show the INFORM messages to express task-done signals that are generated by worker1 and worker2 agents.

VI. Conclusion

During this research, we focused on developing the proper information control and communication solution to enable worker–cobot agile manufacturing. At the beginning, we investigated the existing solutions for the agile manufacturing system, because the problem of agility in manufacturing is relatively older than the worker–cobot collaboration problem. Accordingly, we found that the PROSA holonic control model is the most common solution for agile manufacturing. However, the PROSA model needed to be modified to fit the context of collaborative manufacturing.

Therefore, we added a new basic holon to the PROSA model; this new holon is a CH, which grantees the continuous adaptability to the variation in the production customization and rate. The autonomous agent technology has been used to implement our CPROSA model. Autonomous agent communication via ontologies has been selected to represent the shared work environment for the worker and the cobot. This method was very successful as it is naturally human-readable, yet it can be understood by the machine software. In addition, we defined an interaction model among the holons. One of the benefits of this interaction model is dealing with the stochastic time variation that occurs in the system because of the existence of the human worker.

In order to complete the understanding of our CPROSA model, we demonstrated a case study. The case study has two customers who order two different products with different customization features and amounts. The operational resources in the case study were two workers in cooperation with one cobot. The aim of the demonstration was to show the capability of the proposed CPROSA-holarchy to adapt to the customers' needs. This adaption has been achieved by distributing the responsibilities over the CPROSA holons, which ultimately led to a collective decision to assign the manufacturing operations over the present operational resources. On the contrary to the PROSA model, which is very abstract, the functions and the interaction pattern of the CPROSA holons were very clear in this case study.

In future work, we intend to take into account more operational resources and customers. Other technologies such as XML and industrial web services could also be used to implement the CPROSA-holarchy instead of the autonomous agent technology. Most importantly, in future research, we will use real hardware for the robot instead of a UI only.

REFERENCES

[1] M. Goodrich and A. Schultz, "Human-robot interaction: A survey," *Foundations and Trends® in Human-Computer Interaction*, vol. 1, no. 3, 2007, pp. 203-275.

[2] S. Pinker, *How the Mind Works*, 1st ed. New York: Norton, 1997, pp. 119-127.

[3] G. B. Avanzini, N. Ceriani, A. Zanchettin, P. Rocco, and L. Bascetta, "Safety control of industrial robots based on a distributed distance sensor," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2127-2140, 2014.

[4] A. Lasota, F. Rossano, and A. Shah, "Safe close-proximity human-robot interaction with standard INDUSTRIAL Robots," in *Proc. IEEE International Conference on Automation Science and Engineering (CASE)*, 2014.

[5] A. Koestler, *The Ghost in the Machine*, 1st ed. New York: Macmillan, 1968.

[6] V. Botti and A. Giret, Holonic Manufacturing Systems, ANEMONA-*A Multi-agent Methodology for Holonic Manufacturing Systems*, 1st ed. London: Springer, 2008, pp. 7-20.

[7] R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: A survey," *Journal of Intelligent Manufacturing*, vol. 17, no. 1, pp. 111-131, 2006.

[8] H. V. Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: PROSA," *Computers in Industry*, vol. 37, no. 3, pp. 255-274, 1998.

[9] N. Jennings and M. Wooldridge, *Agent Technology*, 1st ed. Berlin: Springer, 1998, pp. 3-28.

[10] W. Shen, Q. Hao, H. Yoon, and D. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415-431, 2006.

[11] "Jade Site | Java Agent DEvelopment Framework," Jade.tilab.com, 2017. [Online]. Available: http://jade.tilab.com/.

[12] W. Teahan, *Artificial Intelligence–Agent Behaviour*, 1st ed. BookBoon, 2010.

[13] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*, 1st ed. Chichester: Wiley, 2008.

[14] Welcome to the foundation for intelligent physical agents. *Fipa.org*, 2017. [Online]. Available: http://www.fipa.org/.

[15] A. Sadik and B. Urban, "A novel implementation approach for resource holons in reconfigurable product manufacturing cell," in *Proc. the 13th International Conference on Informatics in Control, Automation and Robotics-ICINCO 2016*, 2016, pp. 130-139.

[16] G. Black and V. Vyatkin, "Intelligent component-based automation of baggage handling systems with IEC 61499," *IEEE Transactions on Automation Science and Engineering*, vol. 7, pp. 337-351, 2010.

[17] S. Balakirsky, "Ontology based action planning and verification for agile manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 21-28, 2015.

[18] S. Fiorini, J. Carbonera, P. Gonçalves, V. Jorge, V. Rey, T. Haidegger, M. Abel, S. Redfield, S. Balakirsky, V. Ragavan, H. Li, C. Schlenoff, and E. Prestes, "Extensions to the core ontology for robotics and automation," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 3-11, 2015.

[19] N. Rodrigues, "Development of an ontology for a multi-agent system controlling a production line," MSc Thesis, Instituto Politécnico de Bragança, 2012.

[20] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout, "Enabling technology for knowledge sharing," *Al Mag.*, vol. 12, pp. 36-56, 1991.

[21] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 5, pp. 907-928, 1995.

[22] H. Wang, N. Gibbins, T. Payne, and D. Redavid, "A formal model of the Semantic Web Service Ontology (WSMO)," *Information Systems*, vol. 37, no. 1, pp. 33-60, 2012.

[23] D. Fensel, "Ontology-based knowledge management," *Computer*, vol. 35, no. 11, pp. 56-59, 2002.

[24] L. Lai, "A knowledge engineering approach to knowledge management," *Information Sciences*, vol. 177, 2007.

[25] A. Sadik and B. Urban, "A holonic control system design for a human & industrial robot cooperative workcell," in *Proc. 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2016, pp. 118-123.

Ahmed R. Sadik received his Bachelor of Mechatronics engineering from (Ain-Shams University-ASU) in Egypt–Cairo in 2006. His Master degree with the subject of Factory Automation has been obtained in 2013 from (Tampere University of Technology-TUT) in Finland-Tampere. He is currently pursuing the Ph.D. degree from University of Rostock in Germany and he is entitled as a researcher in Fraunhofer- IGD institute – Rostock.

He has experience in distributed control solutions design, autonomous artificial agents, and in embedded software development. His focus in his PhD research is to develop a control solution to enable the close proximity interaction of the human-worker and the industrial-robots in the same work environment.

Bodo Urban received an MSc (Dipl.-Math.) in Mathematics and a PhD (Dr.-Ing.) in Computer Science from the University of Rostock in Germany in 1978 and 1983, respectively. Until 1990 he was research associate at the Computer Science department of the University of Rostock. In 1991 he moved to the new founded Rostock Division of the Computer Graphics Center, and 1992 he became responsible for the new founded Rostock division of the Fraunhofer IGD.

In 1998 he was appointed to Professor of Multimedia Communication at the Computer Science Department of the University of Rostock. Since 2008 he is the head of the Competence Center-Interactive Document Engineering-at Fraunhofer IGD.

Prof. Urban is a member of GI, ACM, IEEE and Eurographics. He is a reviewer and a member of many program committees of international conferences and workshops. Prof. Urban is also member of several executive committees and advisory boards.