Applications of Semantics in Agent-Based Manufacturing Systems

Marek Obitko, Pavel Vrba and Vladimír Mařík
Rockwell Automation Research Center
Pekařská 695/10a, Prague, Czech Republic
E-mail: {mobitko, pvrba, vmarik}@ra.rockwell.com

Miloslav Radakovič and Petr Kadera
Department of Cybernetics, Czech Technical University in Prague
Technická 2, 166 27 Praha 6, Czech Republic
E-mail: {radakm1@, kaderp1}@fel.cvut.cz

Keywords: semantics, ontologies, industrial systems, distributed systems, multi-agent systems

Received: October 30, 2009

Distributed intelligent control systems compared to traditional centralized manufacturing architectures provide much more powerful instruments for developing robust, flexible and reconfigurable factory automation systems. The basic characteristic of any distributed system is a communication between the system's components needed for information exchange and coordination of activities for accomplishing collective goals. To achieve effective knowledge exchange and integration in open, reconfigurable environments, an explicit definition of semantics is needed to capture the data and information being processed and communicated. The paper shows how semantics and ontologies can be employed in industrial systems, considering particularly distributed, agent-based solutions. A new manufacturing ontology providing semantic model of production planning and scheduling, material handling and customer order specification is presented. Its integration with an agent-based simulation and control system MAST is demonstrated.

Povzetek: S pomočjo ontologij in semantike je izdelan vmesnik za agentni sistem.

1 Introduction

It is not easy to avoid starting the paper about holonic and multi-agent control systems with the usual words about the growing requirements on flexibility, reconfigurability and robustness, which can be hardly met with traditional centralized control systems [26] . Although the classical architectures based on PLCs and IEC 61131-3 programming languages are still predominantly used in industry and there is a very little number of real deployments of these new approaches [39] , this paper tries to document another significant step that the research in the field of intelligent manufacturing systems is going to take.

This area becomes strongly influenced by the recent advances in semantic technologies, like semantic web and semantic web services. In the past a strong emphasis was put on creation of standards for interoperability in heterogeneous interacting systems. A significant standardization endeavor aimed at physical applications of multi-agent systems was undertaken within the FIPA consortium [http://www.fipa.org/]. The adoption of FIPA standards and usage of compliant agent platforms like JADE or FIPA-OS became very popular for the implementation of intelligent control systems. In particular, the acceptance of Agent Communication Language [13] was seen as a way of ensuring interoperability in heterogeneous agent systems developed by various bodies. However, the interoperability within the FIPA context is guaranteed only at the syntactical level. Well defined syntax of a message offering elements like sender, receiver, communication protocol and content provides a simple way for composing a message structure by the sender and its explicit comprehension by the receiver.

Nevertheless, a full interoperability can be attained only if the receiver understands also the content of the message. This can be achieved by introducing a common semantics providing explicit and machine processable description of the objects and their relationships appearing in the selected domain. Shifting the attention to semantic techniques has been a natural step in the evolution of multi-agent systems. Unfortunately, FIPA did not catch up to this trend what we assume was one of the factors of gradual declination of activities around this organization. It is not only messaging but also gathering, internal representation and processing of knowledge about own states and goals as well as about conditions of the surrounding environment where semantic technologies can be advantageously applied. As it is shown in this paper, there has been a remarkable increase in pilot applications of ontologies in agent-based industrial control systems over the past few years.

In addition, the obsolete FIPA standards for messaging and agent services registration and lookup are also to be replaced. Increasing number of researchers has become to realize many similarities between mutli-agent systems and service-oriented architectures (SoA) [32] and forecast a high potential of the synergy of these two approaches. Basically, agents provide capabilities to other agents in the same way as services are provided in service-oriented systems and also use messages to exchange data. Moreover, when considering increasing requirements for seamless integration of control system with enterprise business systems the SoA represent a suitable technological framework for implementing the agent-based rapidly reconfigurable factory automation systems [24] .

The intention of this paper is to provide an overview of current state-of-the-art in application of semantic technologies in industrial systems with the particular aim at the distributed agent-based solutions. Section 2 provides a general introduction into the world of semantics, ontologies and semantic web. Section 3 gives an overview of recently emerged applications of semantics and ontologies in the industrial domain and gives a summary of general aspects of such applications. Section 4 describes a new modular ontology developed for discrete manufacturing and provides guidelines how to utilize this ontology in multi-agent control systems. Section 5 presents a case study of deployment of the developed ontology for assembly line production control. Section 6 concludes the paper with the presentation of major advantages of using ontologies in multi-agent industrial systems.

2 Semantics and ontologies

The most common type of interaction in any distributed heterogeneous system is the exchange of messages. Obviously, there must be an agreement on both the syntax and the semantics of them. The syntax defines the structure of a language, i.e., a grammar typically in a form of rules that govern the structure of sentences. Semantics is dealing with the aspects of meaning as expressed in the language, i.e., the sense of language elements and their combination, including the relation of these elements to the real world.

The semantics is often captured by an ontology. The term ontology as clarified in [16] comes from philosophy, where it refers to the study of being or existence. It attempts to describe categories and relationships of all existing things. In engineering applications, this is reduced to a model of a part of a selected domain – a model that is processable by a machine and is appropriate for a specific application. The ontology in this context is often defined as a formal explicit specification of a shared conceptualization, where conceptualization is a shared view of how to represent the world. Ontology is then formal description of this view and generally consists of concepts representing classes of objects in the real world, their attributes, relations and constraints. In fact, the ontology provides a static vocabulary describing the general patterns occurring in given reality. Such a vocabulary is then used to describe the actual state of the observed real system and its dynamism – the model is composed of instances of classes defined in ontology and is usually called a knowledge base.

The utilization of ontologies in software engineering is mainly linked with the Semantic Web (SW) and Semantic Web Services architectures. SW aims to provide a common framework that would allow data to be effectively shared and reused. It is an extension of the World Wide Web that brings the semantic description of a content so that it can be found, processed and integrated by software agents more effectively [21] . The core semantic web technologies are Resource Description Framework (RDF) and Web Ontology Language (OWL). Although these technologies were primarily designed for the Web, they have been found suitable for other applications as well [38] .

The RDF is a standard for expressing structured data in a form of simple statements [30] . Each statement is expressed as a triple *subject-predicate-object* and each participant of the triple is a web resource identified by Uniform Resource Identifier (URI). In the place of object, a literal such as string or number can be used. The triples, which can be gathered from distributed sources on the web, are linked together to a searchable graph. For searching within an RDF graph there is the Simple Protocol and RDF Query Language (SPARQL) enabling specification of a required graph pattern for building a query [41] . The Web Ontology Language (OWL) is a widely recognized language for describing ontologies [12] . OWL is based on the description logic that allows sound and complete reasoning in a practically usable time. It defines a semantic description of concepts and roles in a particular domain in form of classes also called T-boxes (terminological boxes), their relations and constraints on the use. The ontology forms in fact a language that is applied for giving the meaning and semantic context to the information observed in a real world. The information is stored in a knowledge base in form of instances (called A-boxes or assertions) of the classes and relations defined in the ontology, which represent particular objects and structure of the real system.

The aim of the classical Web Services is to provide a coherent framework for publishing, discovery and remote execution of services over the Internet. To ensure syntactic interoperability of Web Services, there are core specifications defined by W3C, including message protocol, language for description of the service interface and protocol for registration and searching for services [45] . Semantic Web Services are semantic extension of the Web Services, like Semantic Web is a semantic enrichment of the Web. The main goal of the Semantic Web Services is to provide a semantic interoperability, which is needed for automatic, machine-orchestrated discovery, execution and composition of services. The typical scenario is a decomposition of a complex task into a serial/parallel call of number of various services, where resulting data from one service are used as an input to another service or where data from more service

have to be assembled to create a sophisticated reply to the original request.

Special ontologies were developed for these purposes, such as Web Service Modeling Ontology WSMO [25] or OWL-S [45] . The OWL-S ontology, which is built on top of the Web Ontology Language contains three main parts: the service profile used to describe what the service does, the service model describes how to interact with the service and the service grounding specifies the details of interaction with the service. Such a description is necessary for influencing the broader service-oriented architecture vision to allow truly open architectures that would enable integration of various heterogeneous services. In the internet environment such a vision is often referred to as Internet of Services [43] .

3 Semantics in distributed intelligent industrial systems

Holonic and multi-agent systems have been widely recognized as enabling technologies for designing and implementing next-generation of distributed and intelligent industrial automation systems [8] . These systems are characterized by high complexity and requirements for dynamic reconfiguration capabilities to fulfill demands for mass customization, yet low-volume orders with reduced time-to-market. Self-diagnostics and robustness that allow efficient continuing in operation even if the part of the system is down are other important properties.

The trend of applications of multi-agent systems is apparent at all levels of the manufacturing business. At the lowest, real-time control level, so called holons or holonic agents are usually tightly linked with the real time control programs (implemented in IEC 61131-3 or IEC 61499 standards) through which they can directly observe and actuate the physical manufacturing equipment [7] . Intelligent agents are also used for production planning and scheduling tasks both on the workshop and factory levels [40] . More generic visions of intensive cooperation among enterprises connected via communication networks have led to the ideas of virtual enterprises [9] .

The common principles in industrial deployment of the agent technology is the distribution of decision-making and control processes among a community of autonomously acting and mutually cooperating units – agents. At the shop floor level, for instance, an agent represents and independently controls particular physical equipment, like a CNC machine, conveyor belt or docking station. The substantial characteristic is the cooperation among the agents as they pursue either their individual goals or the common goals of the overall control system. The inter-agent interactions vary from simple information exchanges, for example about the state of processing as the product moves from one machine to another one, through requests to perform a particular operation, for example requesting an automated guided vehicle to transport a product to a particular work station, to complex negotiations based on contract-net protocol or different auction mechanisms.

As the information representation and exchange is the essence of such systems, the need of explicitly defined and shared ontologies becomes apparent. From our experience, the exploitation of semantics and ontologies in the area of agent-based industrial systems seems to be very intensive these days, which was not the case even a few years ago. The researches apparently realized that the syntactical interoperability, predominantly ensured by the adoption of FIPA standards and XML-based messaging, will not be sufficient to keep the pace with the trend towards semantically interoperable knowledge based systems. Thus, the use of semantic web technologies has accelerated significantly in the agent research community over the past few years.

Generally speaking, the formal ontology brings unambiguousness in the sense of explicitly defined vocabulary for manufacturing domain that facilitates communication and cooperation in a distributed industrial system. Also, the formal ontology allows reasoning over the shared knowledge. The intelligent autonomous controllers can process, exchange, search and reason about the knowledge related to the manufacturing plant much more efficiently if the information and data is given a clear semantic context.

3.1 Domain-specific ontologies for agent-based manufacturing control systems

The number of reports about the deployments of ontologies in agent-based manufacturing systems increases. Usually, a domain-specific ontology covering a subset of the manufacturing area, for instance assembly, is developed and utilized only for the purposes of the particular agent-based control application.

In [10] the ontology for shop floor assembly is described. Two basic categories of concepts are proposed: *modules* and *skills*. Modules represent physical processing units or their aggregation. One of the modules is for instance a *workcell*, which is defined as a composition of *workstations*, where a workstation is a composition of *units*. The examples of units are *transforming unit*, *flow unit* and a *verification unit* where transforming unit can be further specialized as pick&place unit or milling machine. The two typical ontological constructs, `composedOf` and `isA`, are used to describe the composition and specialization (inheritance) relations between concepts in ontology. Skills represent abilities to perform manufacturing actions, as for instance *MoveLinear*, where complex skills are represented as a composition of basic skills. The basic element in the multi-agent system which uses ontology as a data model for reasoning about objects and their relations is the Manufacturing Resource Agent (MRA). This agent, representing for instance a robot, searches the ontology after its instantiation for skills it supports using its serial number and type of equipment. Then it registers its capabilities in the yellow pages provided in the multi-agent system by a special Directory

Facilitator (DF) agent, which manages and provides information about services provided by the agents. The particular MRA agents can form coalitions in order to provide combined skills. In such a case there is a Coalition Leader Agent, which registers in the DF all complex skills provided by the coalition and subsequently coordinates the execution of elementary actions by particular coalition members. The proposed solution has been deployed in the NovaFlex laboratory environment installed at the Intelligent Robotic Center at UNINOVA in Portugal. A simple assembly line is composed of two robots, each with four different types of grippers and tools, and an automated transportation system that connects the robots and a storage unit. According to the authors, the proposed solution proved enhanced reconfiguration capabilities. The components can be added to the system at runtime, and thus the line can be easily adapted to new types of products.

An OWL-based ontology developed for agent-based reconfiguration purposes is reported in [1] . The application of ontology is illustrated on a small laboratory manufacturing environment consisting of two machines equipped with different mechatronic devices such as a rotating indexing table, plunger, drill, picker, etc. The basic ontology concepts are *material resource* and *operation*. In fact, this abstraction is identical with the previous example, only with the difference in the used terminology (module vs. material resource and skill vs. operation, respectively). The resources are *machine* and *tool* with corresponding subclasses like *handling machine* and *processing machine* as well as *rotary indexing table*, *drill*, *kicker*, etc. The operations are subdivided into *manufacturing operation* and *logistic operation* with further classification on *sorting*, *hole testing*, *drilling* and *picking*, *kicking* and *rotating*, respectively. References between machine and operation concepts express the facts that machine *enables realization of* an operation. These general concepts from the ontology are then instantiated to capture the real environment, such as the particular machines and their relations. Such a dynamic part of the ontology is also expressed in OWL, thus allowing the agents to reason about the available machines and operations in the semantic context.

Magenta Technology company provides another example of exploration of ontologies in agent-based applications. The details of an Ontology Management Toolset are given in [42] . This set of multi-agent tools enables developers to create and edit the static aspects of ontology as well as the dynamic aspects, here called *scenes*. The ontology developed by this toolset for supply chain and logistic planning is then presented in [2] . The examples of concepts are for instance *factory*, *cross-dock*, *truck*, etc., and relations like *is booked for a demand*. Although it is not explicitly mentioned in the last two cited papers, the Magenta's multi-agent engine provides a mechanism of updating the agent's behavior (i.e., the program code) dynamically as the ontology is being extended. The corresponding piece of code providing an agent with an algorithmical description of its behavior associated with a particular new ontology concept is sent to the agent so that it can subsequently execute the code to react appropriately.

Merdan et al. report on the application of ontologies in a transportation domain [33] . The OWL ontology has been developed for supporting the interactions of agents controlling the palette transfer system, which is a part of the Vienna University of Technology's testbed for Distributed Holonic Control [18] . The agents represent basic components of the transport system such as conveyors, diverters, junctions, index stations and palettes. The agents use the ontology for representation of the real state of the environment, like mutual connections of components, actual position of palettes, failure states, etc. Such data are stored in agent's knowledge base, which is continually updated as the agent perceives the dynamic changes in the environment. Such changes in the knowledge base trigger the rule-based behavior of the agent to properly react to the new facts. More details on the proposed ontology in the broader context of production scheduling in the assembly domain are given in [34] . The *product* is described as a composition of *subassemblies* that further contain *parts* representing raw materials. Each subassembly is produced as a result of step, in which a particular *operation* has to be performed by a manufacturing *resource*. The relations *needsPredecessor* and *isFollowedBy* between steps is used by the Supply Agent that schedules the production in cooperation with resource agents. A related case study presented in [23] examines the usability of the proposed ontology-based agent architecture in the resource allocation tasks.

Hellingrath et al. present the FRISCO ontology designed to support organization of knowledge in automotive supply chains [19] . Five different models have been designed: the Sourcing Model gives information about the products sold to customers and the parts procured from suppliers; the Resource Model contains all manufacturing resources that are relevant for planning like machines, workers, etc., including their capacities; the Adjustment Measure Model provides structures to represent network adaptivity; the Demand Constraint Model describes relations between demand and capacity to allow real-time capable-to-promise processes; and the Time Model provides structures for different calendars in order to create common understanding of dates between customers and companies. A proof-of-concept has been designed to verify the applicability of the proposed knowledge models. The scenario encompasses an OEM producing cars and two suppliers providing parts. The partners in supply chain are modeled as agents in a multi-agent system. The ontology is used for negotiations between car producer agent and part supplier agents about planning of delivery of required parts.

The proposal of an ontology for organizational model of general holonic systems deployment is presented in [11] . The context of an organization is described in terms of project *management*, *manager*, *employee* and the roles such as *supervise* and *assigns*. Other general concepts for the agency and holonic

domain are defined like *agent*, *agent role*, *holon*, *holon role*, etc.

3.2 General-purpose ontologies for manufacturing domain

The previous section documented that even though there are many efforts towards designing ontologies for manufacturing domain, different developers use slightly different vocabularies for describing similar concepts like for example *module* versus *resource* or *skill* versus *operation*. Moreover, developed ontologies cover usually very narrow areas. More complex and consistent general ontologies for manufacturing domain together with a series of complementary, coherent, domain specific ontologies would be helpful for achieving better interoperability and reusability. The existing norms and standards like ANSI/ISA-95 a ANSI/ISA-88 [3] could serve as a good basis in this effort. The ISA-95 „Enterprise-Control System Integration“ standard describes hierarchical model of production organization and the event flow and provides basic concepts for the integration of control system with business systems of the enterprise. The ISA-88 “Batch Control” describes in more detail the batch process production environment. It defines hierarchical model of production system from the enterprise, through areas and units down to control modules. It also defines process model for description of sequenced production phases and actions.

Very promising standardization effort seems to be concentrated around the O^{3}NEIDA (Open Object-Oriented kNowledge Economy for Intelligent inDustrial Automation) consortium that aims at creating the open technological infrastructure for automation components [50] . The goal is to create the architecture for hardware and software compatibility at all levels of automation components market, from device and machine vendors, through system integrators up to industrial enterprises. The basic element in this architecture is the a*utomation object* that is an abstraction of mechanical device with encapsulated intelligence, i.e., software components providing different functionality like control, visualization, simulation, diagnostics, etc. with well defined interfaces. Simple automation object such as sensors, drives and microprocessors can be then used as reusable modules for creating more complex objects (such as machines) that can be further used in the same modular way for building the whole industrial enterprises. The use of ontologies, mainly OWL, is promoted for the description of automation objects. This would allow automatic machine processing and reasoning as well as simplifies search of automation objects in repositories.

A complementary work to OOONEIDA initiative presented in [28] aims at semantic extension of automation objects by applying the semantic web technologies. Two separate ontologies for mechatronic devices reference model (covering both the hardware and the software features) and the IEC 61499 reference model respectively are proposed and merged into an ontology for Automation Objects reference model (proposed by IEC-TC65 group). The basic concepts designed for the lowest level include function blocks, events, I/Os, etc. The device/machine level part of ontology provides concepts like function block application, resource, etc. Two examples of semantic description of automation objects – Conveyor and Lifter – are sketched.

National Institute of Standards and Technology (NIST) devotes considerable standardization effort to manufacturing domain. For instance, shop data model is described using UML diagrams and XML serialization examples [31] . The model includes description of organization, bill of materials, process plans, resources, schedules, etc. Although it is not a formal ontology in the sense described earlier in this chapter, such standards are important as a base for ontologies that would be widely accepted. Another example of NIST activities is the Process Specification Language PSL [15] that is a logical theory that covers generic process representation, which is common to all manufacturing applications. The PSL ontology contains axioms grouped to theories describing aspects such as complex activities and can serve as a solid base or upper ontology for representing processes. The PSL ontology contains primitive concepts like *activity*, *object* or *timepoint*, functions such as *beginningOf* and *endOf* and relations like *between*, *is_occuring_at*, etc.

MASON (MAnufacturing’s Semantic ONtology) presented by Lemaignan et al. [27] represents another contribution in this area. The goal is to develop an upper ontology that would allow seamless integration of more specific ontologies using the common cognitive architecture. The ontology is based on OWL and describes the taxonomy of concepts such as entities, operations and resources and their relations like associating a tool with an operation (property *requiresTool* with the domain *ManufacturingOperation* and a range *Tool*). It is reported that currently the ontology, which is available on-line at http://sourceforge.net/projects/mason-onto, constitutes of more than 220 base concepts and 40 properties. Moreover, a mapper has been developed between OWL ontologies and the internal ontology model used by the popular Java Agent DEvelopment Framework (JADE) agent platform (http://jade.tilab.com/). Although some of the constructs in the ontology seem to be application specific, for example, restricting previous operation in the definition of operation concepts, this work can be seen as an important step towards formalizing the vocabularies used to describe manufacturing domain.

When building the general-purpose manufacturing ontologies, it is obviously necessary to have solid basis in form of well developed foundational (upper) ontologies incorporating for example spatial or time theories. Unfortunately, the direct utilization for manufacturing purposes is limited because these foundational ontologies are often created in very expressive languages without taking care of computability. The formalization of ADACOR ontology (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) using the DOLCE methodology

(Descriptive Ontology for Linguistic and Cognitive Engineering) is outlined by Borgo and Leitão [6] . ADACOR is originally described using Unified Modeling Language (UML) diagrams and natural language descriptions, while DOLCE uses first order modal logic and aims at capturing the ontological categories underlying natural language and human commonsense, such as physical or abstract objects, events and qualities. The alignment of ADACOR to DOLCE yields well formalized and well founded ontology. The ontology described in ISO 15926 "Industrial automation systems and integration" also uses well founded principles of temporal and spatial representation of objects in a form of four dimensional approach to simplify reasoning in the process engineering domain [4] .

3.3 Common properties of ontologies deployment in agent-based manufacturing systems

Within the frame of semantic extension of multi-agent industrial control applications certain characteristics could be identified that differ from the common usage of ontologies in pure software systems like web applications.

Static and dynamic aspects of ontology – the OWL language allows one to express two kinds of terms. First are static, unchanging concepts, so called T-boxes (terminological boxes) that represent vocabulary for description of selected domain in form of classes and their relations. The latter group are so called A-boxes (assertion boxes) that have the meanings of particular assertions about the described part of the real world and that are formulated using the vocabulary of T-boxes. Each A-box is an instance of corresponding T-box. A set of A-boxes form together a *knowledge base*. The first part that could be called an *ontology* is invariant or is changed rarely. In case of a knowledge base, which is sometimes called also a scene [2] , often changes are supposed as there are dynamic changes in the observed part of real world. An example from the automation area is the set of classes (T-boxes) *machine* and *operation* with relation *providesOperation*. A knowledge base that describes current state of the factory shop floor contains for example instances *machine_M25* and *drilling_D14* as instances of these classes connected together with the mentioned *providesOperation* relation. In case of deployment of agents the ontology or its part is shared by all agents while the knowledge base is created and maintained by each agent individually as the agent perceives the changes in its environment by means of sensors, communication with other agents (the agents share their knowledge bases) or by communication with a human.

Interaction with a real world – an important part of agent's knowledge base, which represents and controls physical manufacturing components, is the information related to actual state of the controlled equipment (readings from sensors) and the status of controlled production process (e.g., actual location of the product). Another important factor to be considered is also a possible physical interaction or collision avoidance with other equipment. The agent has to be aware of the physical effects of its decision making. Its actuation could in negative case lead to a failure or damage of the equipment or to increased number of defected products. The link between an agent and the real world in case of manufacturing control deployment is usually ensured through the interaction with the low-level control (LLC) layer, which might be implemented according to IEC 61131-3 or IEC 61499 standards. The ontology could be advantageously used for designing the semantic model of the low-level layer and corresponding interface. This ensures keeping the agent away from the details of software and hardware implementation of the LLC and thus makes the integration of multi-agent control system with current PLC-based architectures much easier. The first attempts to design the ontology-based interface between agents and LLC is presented in [18] . It is argued that the use of semantic model for these kind of interactions keeps the agent and LLC layers more loosely coupled, rather than tightly coupled as seen for instance in real-time interface described in [29] . Loose coupling is a desired property, because it enables that the LLC layer can be still operational even if the agent layer becomes unavailable or faulty. Also the technology of radio frequency identification (RFID) becomes to play increasingly important role in tracking and localization of parts, semi-products and products in manufacturing environment. The architecture integrating RFID with agents is described in [47] . The use of ontologies in this field is also expected to provide semantic interoperability between applications processing RFID data and events and the RFID infrastructure involving tags, readers, middleware, etc. [22] .

Reactivity – imagine a situation when an agent notes a particular event in the real world, for instance, detects a failure of the controlled machine. It creates a corresponding fact consistent with the ontology (describing relation between a *machine* and *failure* concepts) and stores this information into its knowledge base. The agent's inference engine can then possibly deduce other new facts, but this still does not directly lead to reaction. But often, particularly in agents acting in real world, some action or reaction needs to be taken by the agent – for instance, actuating (stop the drive) or informing other related agents. So the meaning of the particular concept from the ontology is sometimes not only knowledge-based but also "algorithm-based". The ontology should provide the agent also with the explicitly defined rules in a form of algorithms (or directly a program code) to be executed by the agent to react appropriately. This issue is not sufficiently discussed in literature. As mentioned earlier, Magenta agent runtime environment [42] provides such features – program code is sent to the agent at runtime to modify or extend its behavior.

Ontology-based service matchmaking – one of the basic concepts of multi-agent systems is the advertisement of agents' skills and services in the Directory Facilitator (DF), known also as yellow pages

services. Other agent can then search the DF to find out particular service providers. However, the information held in the DF in majority of agent platforms available today, such as JADE [5] , Cougaar [20] or A-globe [44] , can be registered only in a very simple form. It usually contains just type of the service (for instance Drilling) but it cannot be further parameterized (diameter: 10-100 mm, hole depth: 5-20 mm). Obviously, to fully explore the potential of semantics in agent-based systems, ontologies must be deployed for service registration and lookup through DF as well. Within the registration the agent sends the corresponding part of its ontology (services it offers) to the DF. DF can be then queried for finding particular service providers using more complex queries, like "find all machines that can drill a hole of 50 mm diameter and 15 mm depth". The result sent back by DF to the requester (also in form of ontology) might be with convenience supplemented by the message template and protocol to be used in the corresponding inter-agent negotiations, as discussed in [10] . Services provided by agents can be described using OWL-S in a similar way as semantic web services. Matchmaking of services can be then made using OWL reasoning.

Orchestration of manufacturing processes – service integration and composition becomes very attractive topic in the Service-Oriented Architecture (SOA) domain. Authors in [17] describe a solution based on multi-agent and holonic techniques. Community of interacting holons, representing service providers and requestors, can be nested so that requested complex service is automatically orchestrated as a composition of basic services. An important function of a reconfigurable distributed manufacturing system is the distribution of tasks over multiple agents or holons. This goes beyond the simple service matchmaking – the whole process must be decomposed, executed and problems occurring during the runtime must be resolved. For that, manufacturing processes should be also specified in ontologies [24] . We envision the ontology-based recipes compiled as a sequence of elementary operations described in a suitable ontology to allow automatic discovery of equipment that can perform requested operations (see next Section for more details).

Interoperability – represents property required within a manufacturing system as well as between other systems on MES (Manufacturing Execution System) or ERP (Enterprise Resource Planning) levels. Translation between ontologies is a way of integrating systems that use different ontologies [36] . One agent prepares a message in its ontology, and the message is then translated into the ontology used by the receiving agent while preserving the meaning of the message. The details on the translation using semantic web technologies and OWL reasoning are described using transportation domain examples in [38] . The architecture of integrating systems has to be considered as well – the low level control devices would be hardly able to do such translation themselves, and so they need to ask a special service to provide translation for them or the translation has to be made automatically in the message transportation layer [37] .

3.4 Semantic search

One of the core applications of the semantic web is a semantic search, i.e., search within semantically enriched data. The design, operation and maintenance of a manufacturing system is very knowledge intensive task and involves handling of information stored in different forms – for example, function blocks or ladder diagrams describing the real-time control system, SCADA/HMI (Supervisory Control And Data Acquisition/ Human Machine Interface) views, collected historical data, etc.

It is often not easy to search within such information space even using plain text search. However, when the information is accessible in the semantic web form, it is possible to make queries beyond the classical keyword search. We have investigated the use of semantic web technologies for the semantic search within various information sources of an assembly line. Our conclusion is that the RDF/OWL form of data is appropriate for storing the information and for querying [38] . The SPARQL language is capable to express structural queries that are of practical interest for the control system designers as well as maintenance personnel. Example of such a query is to find all projects where specific ladder code instructions (e.g., XIO – eXamine If Open) with specific variable (e.g., Valve13) occur in a rung. In the prototype prepared by Rockwell Automation the data extracted from the control system, such as ladder logic programs and HMI views, are annotated automatically depending on their context, so the process of indexing, which is necessary for structural queries, is fully automatic.

In addition, the implicit information (such as the `part-of` relation) can be made explicit in the ontology describing the manufacturing system. The query engine can employ an OWL reasoner to include this information into query results, so that for example query results containing the part-of relation contain transitive closure of this relation. An important advantage of using RDF is that all the distributed information can be merged into a single RDF graph. This allows asking for connections of information from different sources, such as in the query to find all HMI views that have push buttons connected to a ladder code project that is used in a specified area.

4 Generic manufacturing ontology and related multi-agent architecture proposal

In this section we describe the utilization of ontologies for a multi-agent industrial production system. The system consists of agents that handle various aspects of a typical flexible discrete production system. The goal is to isolate knowledge and semantics so that typical system deployment would mean only extending ontologies without having to reprogram the multi-agent system.

4.1 Usage of ontologies in multi-agent system

The reason for integrating ontologies into multi-agent system is that we would like to express semantics explicitly for agents so that they are able to operate differently when ontology or knowledge base is changed and also so that integration of new agents can really proceed without reprogramming existing agents. Our general testing scenario is flexible and reconfigurable distributed production system that consists of workstations able to provide different manufacturing operations and of transportation between these workstations. The system accepts different highly customized orders, is able to create customized production plans and is able to execute these plans. The plan is executed by selecting workstations that provide required manufacturing operations needed in steps of the plan. The manufactured product is transported between workstation together with material until the plan is finished, i.e., until the final product is produced.

The orders, production plans, capabilities of workstations, transportation system and other components are described in ontologies and knowledge bases and not in agents' code. This is very important factor that makes future extension of the system very easy. It is possible to introduce new product type by adding its production plan or it is possible to introduce a new operation by extending the ontology. Only the update of the ontologies and/or knowledge bases of appropriate agents is required for introducing completely new functionality. For example, when a new product type is added, it is enough to extend the order ontology to cover new product type and its parameters, to add general production plan for that product type, and to add rules for conversion from product order to specific production plan. When a machine with new kind of operation is added to multi-agent system, it is again enough to extend ontologies to cover this new operation.

4.2 Description of ontologies

From the state of the art description we could see that there are some interesting features in existing ontologies. We were inspired by ontologies described earlier and decided to design a new ontology based on existing standards and ontologies such as ANSI/ISA 88, OOONEIDA or MASON. It has been found that none of these ontologies provide a suitable semantic model that would fulfill our requirements for generality, extensibility and deploy-ability in distributed control applications devoted for discrete manufacturing – i.e., handling customized orders, handling customized production plans, handling material and semi-product transportation between production machines and describing these machines. ANSI/ISA 88 is exclusively designed for batch processing industry, OOONEIDA does not provide what we needed for this purpose and MASON seemed to be too limiting for us.

When designing a new ontology, the attention has been paid to two complementary aspects that influence feasibility of a real deployment at a wide spectrum of tasks. First, the ontology has to be as much general as possible in order to provide a common, consistent model of base concepts, on which application specific extension ontologies could be designed. Second, the ontology has to be relatively simple in order to be applicable in real control systems. The common issue of many foundational ontologies that are defined in very expressive languages is that the processing of these ontologies is very computationally demanding. It does not meet constraints imposed by PLC-based architectures and real-time or near-to-real-time fashion of control programs that should work with the ontology.

Our new manufacturing ontology includes three different aspects of automation systems, as illustrated in the upper part of the Fig. 1: (i) specification of customer order, (ii) definition of production plan and (iii) transportation and material handling. The ontology is implemented in OWL as three separate ontology modules that describe these aspects in general. All of them reuse classes and properties from the Common Ontology that for example separates physical and information resources. There are also other ontologies, such as ontology for the configuration of the system. Reusing particular classes or properties from one module in another one is implemented using OWL imports.

These ontologies are intended as a base for the multi-agent system operations. Agents should understand these ontologies and should be able to handle their extensions. As we can see in the Fig. 1, it is possible to extend the ontologies with application specific description of product order and product plan. The advantage of this approach is that for particular product and product plan we only need to extend existing ontologies (i.e., subclass existing classes) and the system is able to handle new product without any other changes.

The lower part of the Fig. 1 shows operation of the multi-agent system together with illustrating which parts of the whole ontology are used. The workflow of the system is as follows. First, the Order Agent receives a customer order. Based on that order, corresponding Product Agent is created that receives production plan created by Production Plan Agent individually for the customer order. The Product Agent then executes the production plan by contacting Workstation and Transportation Agents. Note that multiple agents can process multiple customer orders at the same time.

As we can see, different agents use different parts (modules) of the whole ontology. We present more detailed overview of the ontology modules together with their intended usage in the next sections.

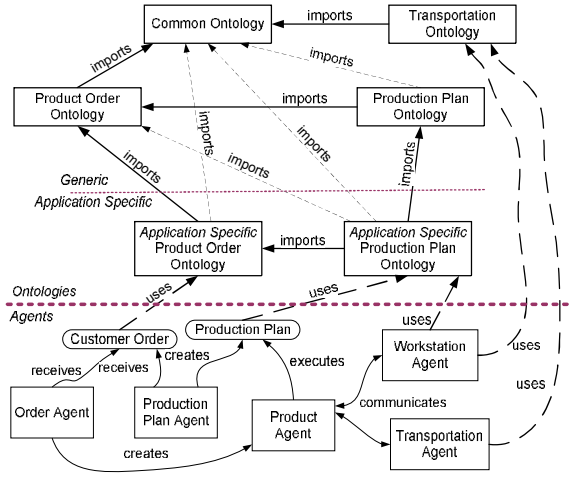

Figure 1: The upper part shows modularized manufacturing ontology with generic and application specific parts including *import* relation between individual ontology modules; the dashed *import* lines show transitive closure of the *import* relation. The lower part shows agents that use different modules of the ontology.

4.3 The order ontology

The ontology module for customer order specification is schematically shown in Figure 2. The dashed line separates the general order ontology (on the left), domain specific extension of the ontology for hypothetical production of a filled bottle (on the bottom) and a knowledge base corresponding to a sample order of three filled bottles (on the right). The ontology as well as knowledge base is expressed in OWL. The ontology is shown schematically without all OWL relations, but in general blue rectangles in ontology correspond to classes and green rounded rectangles correspond to properties. Object properties use normal font, data type properties use italics. In the instance part, black rectangles correspond to instances and black ellipses to RDF literals.

The general concepts of the order ontology are as follows. Order is the top most representation of the customer order; it contains one or more `ProductOrders` to enable ordering of more products of different types in a single order (for example three bottles with water and five cans of coca-cola). `ProductOrder` represents part of the order corresponding to products of the same type (e.g., three bottles of water). The `hasProductOrder` property is used to express the fact that an `Order` is composed of `ProductOrders`. The quantity property represents number of ordered products of the same type; the domain of this property is the `ProductOrder` class and its range is an integer literal. `ProductSpecification` specifies for a particular `ProductOrder` the type of the product (e.g., filled bottle) and parameters or attributes that all the products of the same type should have (e.g., water as a liquid in the bottle, 0.5 volume in liters, etc.). The `hasProductSpecification` relation represents that a `ProductOrder` has assigned `ProductSpecification`. The Product class represents the type of the product being ordered in particular `ProductOrder` (e.g., filled bottle). The control system is supposed to find appropriate production plan (a recipe how to make a product) based on this parameter. Thus this class provides a link between this ontology module and production plan ontology.

To represent different parameters, a class Parameter was introduced to represent a general parameter that can be further specialized. It is supposed that subclasses of this class are defined to represent different, domain specific parameters. The parameters are connected using the `hasParameter` property. An example of parameter is `ParameterBoolean`, which represents a general boolean parameter, and uses booleanValue relation to hold a boolean literal (true or false).

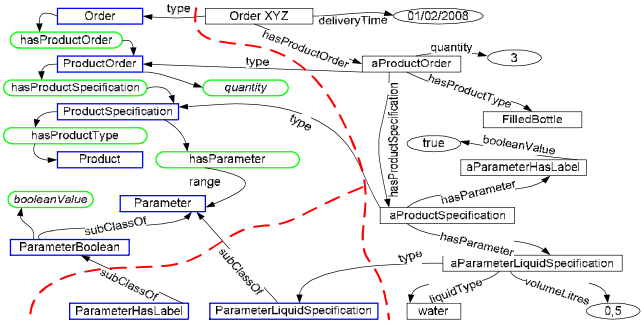

Figure 2: Schema of the ontology for customer order specification (on the left); domain specific extension of the ontology for hypothetical production of a filled bottle (on the bottom); knowledge base corresponding to sample order of three filled bottles (on the right).

The domain specific extension of the general order ontology regarding a hypothetical production of filled bottles includes the following subclasses of the `Parameter` class. `ParameterHasLabel` is a subclass of a `BooleanParameter` for specifying if a label is required on the bottle. `ParameterLiquidSpecification` is a parameter specifying type and amount of a liquid to be filled in the bottle (see the instance part for the intended usage).

A sample instance of order (knowledge base) depicted on right part of Fig. 2 represents a customer order for three bottles filled with 0.5 liters of water and attached label. All the rectangles are instances of classes from the ontology. In this case the order instance (Order XYZ) contains only one `ProductOrder` class instance (`aProductOrder`). It says that the type of the ordered product is filled bottle (`FilledBottle` as instance of Bottle class that is subclass of `Product` class). The specification of the product (`aProductSpecification`) includes a parameter specifying type and amount of liquid (`aParameterLiquidSpecification` instance with relations to water and 0.5 literals). The other parameter is

aParameterHasLabel with relation to true literal specifying that a label should be attached on the bottle.

To summarize, the Product Order ontology serves primarily for description of product parameters in customer orders. These parameters are then used to create production plan, as described in the next section.

4.4 The production plan ontology

The second ontology module shown in Figure 3 provides concepts for description of a discrete production process. It is supposed that a production process involves consecutive execution of steps with defined order, where parallel branches are also supported. In each step there is a particular manufacturing operation associated (like filling) that is performed upon the semi-product, which passes through the process. Performing the operation could require an additional material, like for instance a liquid to be filled to a bottle within the filling process. The ontology includes following concepts.

Product is a class representing a general product of particular type which the production system is able to make. Subclasses of this class are supposed to be introduced (like Bottle) in domain specific extensions of the ontology. The same Product class appears in the previously described order ontology as a type of product that can be ordered. ProductionPlan represents general production plan that describes a consecutive execution of steps that transform raw materials and semi-products into the final product. ProductionStep represents a single production step in a discrete manufacturing process; basically, in each step a particular operation upon the semi-product is executed; a raw material(s) could be required by the operation. The relation hasProductionStep represents that a production plan is composed of several production steps. The precedes relation between two steps A and B is intended for representing the fact that the step A has to be executed prior to execution of the step B (but not necessarily immediately before). It is a transitive relation what means that if step A precedes a step B and step B precedes step C than also step A precedes step C. The inverse relation to precedes is follows. For expressing the fact that no other step can be executed between steps B and A there is relation mustImmediatelyFollow. An example is shown on right part between CapStep and FillStep. The requires relation between two steps A and B expresses the fact that that step A requires a presence of step B in the plan. It is expected that there is a special component in the control system that modifies templates of plans according to user orders (see further). Usually if some feature of product is not required, some steps are deleted from the plan. The requires relation means that if step A is to be preserved in the plan then also step B has to be preserved and cannot be deleted (it does not say anything about order of executing the steps).

Operation represents a manufacturing operation to be executed within a step. It is supposed that an operation is provided in a manufacturing system by a workstation. The operation is usually a complex task that involves execution of a series of sub-operations executed by different machines and tools that are part of the workstation. The relation hasOperation serves to represent that a production step has a single operation associated with it. The hasParameter relation is again intended for expressing various parameters of the operation. The parameters are usually copied or transformed from the order at the time of creating the plan for a particular order from the general product plan.

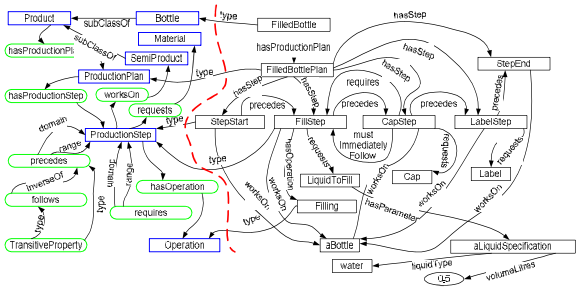

Figure 3: Ontology for description of a discrete production process as a series of consecutive/parallel production steps in which particular manufacturing operations are executed upon the semi-product; right hand side shows sample plan describing production of a filled bottle.

SemiProduct is a class representing a semi-product – an object that is being transformed by operations associated with production steps into a final product. A particular piece of raw material(s) as an input to production step is transformed by associated operation into semi-product(s). For example, the material input into assembly step can be pieces X and Y and the output is a new semi-product XY. Semi-product can then enter other steps as an input, and other steps can further transform it using other operations and additional material. The worksOn relation represents those semi-products that are both input and output ones for a particular production step. Material class represents of raw material that is required as an input for a particular operation in production step or is generated as an output of a production step (for example, in a disassembly process).

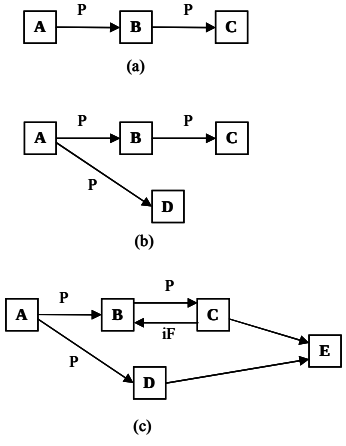

Figure 4: Illustration of various alternatives of precedes (P) and immediatelyFollows (iF) relations between production steps.

The relations between production steps (`precedes`, `follows` and `immediatelyFollows`) allow for expressing either serial or parallel branches or various combinations of those in a production plan. Some of the alternatives are illustrated in the Figure 4. In the case (a) there is a serial plan where the order of executing the steps is A-B-C (step B requires step A to be done before as well as step C requires step B to be done before). In the case (b), A is the first executed, and then B and D can be executed in any order or in parallel. Step C can be executed after step B is finished regardless of the status of D. In other words, the order of execution without considering the possibility of parallel execution of steps B and D can be: A-B-D-C, A-D-B-C or A-B-C-D. In the case (c) there is the `immediatelyFollows` relation between steps C and B, so the step C has to be executed immediately after B. Step D can be executed in parallel with steps B or C or before them or after them. In other words, the order of execution can be thus following: A-B-C-D-E or A-D-B-C-E.

Back to Fig. 3, on its right part there is a sample instance showing a production plan for making a filled bottle. It can be seen that such a plan is developed as a knowledge base composed of instances of classes from the ontology. The plan starts by a step `StepStart` in which a suitable empty bottle is retrieved – it is represented by an instance `aBottle` linked by `worksOn` relation. This bottle instance then enters all other steps. The `FillStep` represents filling of the bottle. It requires a liquid as an input material represented by `LiquidToFill` instance. Specification of the required material is attached as parameter `aLiquidSpecification` instance with subsequent relations `liquidType` to value water and `volumeLiters` to value 0.5. These values are copied from the customer order (see Fig. 2 where the same `aLiquidSpecification` instance is attached). Immediately after filling operation the capping operation (`CapStep`) is required to be executed. The last step is labeling (`LabelStep`) in which a label is attached in the bottle. A `true` value of the `aParameterHasLabel` parameter is again copied from the order. In the case when a label is not required, the label step is removed from the plan.

The plan instance is created by a component of the control system (i.e., agent) by modifying general production plan for a given product type. This agent has access to knowledge base containing general production plans for various types of products. On the basis of an order for particular product piece this component is able to modify the general production plan by rules or other kind of procedural knowledge. The result of the modification is a specific production plan instance tailored to the particular ordered product. This modification is done by copying the parameters from order to corresponding parameters of operations and material in the production plan. Possibly, some steps that are not required because of the specific product feature is not listed in the order are removed from the plan.

4.5 Workstation concept and material handling ontology

Material and operations associated with production steps are provided in the distributed control system by workstations. In other ontologies this concept is also known as work cells, work places, manufacturing cell or just cells. Each workstation is a logical composition of physical manufacturing equipment or devices. It represents an individual, stand-alone manufacturing entity providing different processing capabilities and/or material resources. The workstation is represented by an autonomous control component (workstation agent) that can negotiate about the allocation of its advertised resources with the agents that control the execution of production steps. The single operation provided by a workstation is usually internally decomposed into the execution of several sub-operations carried out by the particular equipment. Such an execution is supervised and controlled by the workstation agent by negotiations with the subordinate equipment agents.

Figure 5 shows a part of the Cambridge's DIAL manufacturing testbed [47] with two workstations – VS1 and VS2 (see also Sect. 5). Each of it contains the following equipment: raw material storage, manipulation robot and docking station(s) connected to a conveyor-based transportation system. One of the operations provided by this workstation is box packing – the robot picks the raw item from the storage and inserts it into a slot in the box. Boxes are carried out on top of palettes moving in the transportation system.

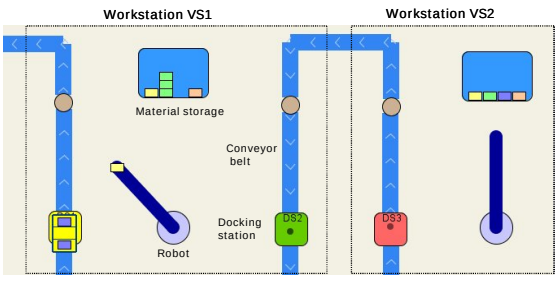

Figure 5: Part of the manufacturing system (Cambridge's DIAL testbed) with two workstations, each composed of material storage, robot and docking station(s).

As can be seen in Fig. 6 showing the third ontology module, the `contains` relation is used to express the fact that an equipment (`Equipment` class) is part of the workstation (`Workstation` class). The `providesOperation` relation is then used to associate all operations (`Operation` class) provided by the workstation (not shown in Fig. 6).

The internal functionality of a workstation can be advantageously described using the same concepts of this third ontology module. In the knowledge base of the workstation there is a production plan (`ProductionPlan`) associated with each operation (`Operation`) externally advertised by the workstation. This plan is again composed of a sequence of steps (`ProductionStep`), while the operations associated

with those steps are performed by the equipment in the workstation. Similar concept of hierarchical classification can be applied also to the production steps themselves. The proposed ontology provides an effective tool that allows one to replace a single operation associated with production step (`ProductionStep`) with the whole production plan (`ProductionPlan`). This plan is again described with the same ontology concepts as a sequence of steps, where each step can again have another production plan associated (see Fig. 7). Using such a simple mechanism enables to handle production plans with different level of granularity at different levels of enterprise. At the roughest resolution it is for example possible to decompose the car production into basic steps like production of engine, gear box, chassis, body, and so on. For each of these steps there is a more detailed plan; engine production is composed for instance from steps for making cylinder head, cylinder body, valves, camshaft, etc. These steps can be further refined using the same ontology concepts down to most fine details at the level of basic operations performed by equipment

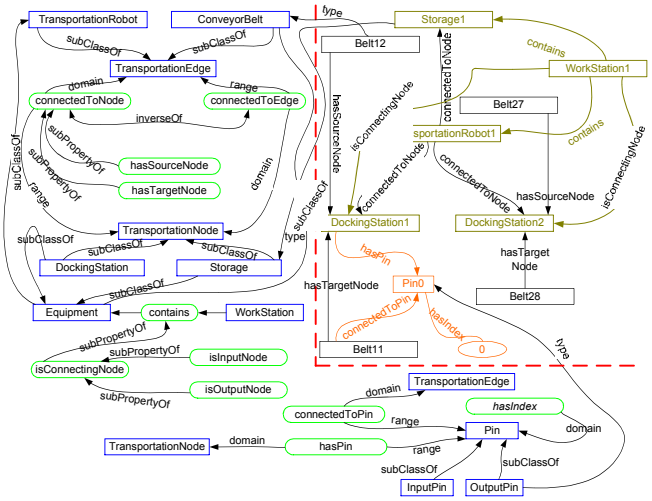

Figure 6: Ontology for description of material handling and transportation aspects including definition of WorkStation concept is shown in the left part of the figure. Sample knowledge-base related to the workstation VS1 of Fig. 5 is shown in the right part of the figure.

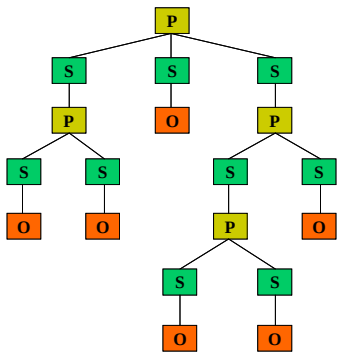

Figure 7: Example of a production plan (P) defined by ontology, where some production steps (S) have an operation (O) associated and some have another production plan associated.

inside workstations.

The third ontology module (Fig. 6) also includes aspects of material handling and transportation. Using the graph theory, the ontology defines a transportation node (`TransportationNode` class), a transportation edge (`TransportationEdge`) and their connection using two different relations, `connectedToNode` and `connectedToEdge`. From the viewpoint of product transportation between workstations, the ontology is used to define which equipment that is part of the workstation is also a transportation node, i.e. an input (`isInputNode`) or output (`isOutputNode`) point of the workstation through which a product or material can be delivered to or out of the workstation respectively. In Fig. 5, docking stations DS1 and DS2 are connecting nodes of the workstation VS1 (connecting node is such a transportation node that is both an input and output node), while docking station DS3 is a connecting node of the workstation VS3. Such an information is used by the components that ensure transportation of material (e.g., transportation agents controlling Automated Guided Vehicles) for planning of optimal transportation paths between the workstations.

5 Case study: MAST system and DIAL scenario

This section demonstrates the integration of developed ontologies in the agent-based manufacturing control system MAST. The functionality is illustrated on a real-world manufacturing scenario – the packing and assembly environment of the DIAL laboratory.

5.1 Ontological extension of MAST system

The Manufacturing Agent Simulation Tool (MAST), developed by Rockwell Automation, was designed with the intention to provide a simulation tool transparently demonstrating the advantages of application of multi-agent system technologies in the industrial control domain [46] . The MAST system consists of agent classes that represent various manufacturing components such as a conveyor belt, diverter, storage, docking station, sensor, etc. A typical task for agents is the transportation of products between work cells through a complex and redundant network of conveyor lanes. The agents use messages to exchange information about optimal routes through the system, about failures or transportation jams and about currently transported products. From the first simulation prototype MAST system matured into a comprehensive simulation and control tool which can interact with the real Programmable Logic Controllers to actually control the real physical equipment [48] .

From the viewpoint of knowledge handling and exchange, the original agent architecture could be characterized as implicit and rigid without the notion of semantics being applied. The agent's representation of the surrounding world was held in local variables, and the content of exchanged messages was encoded in XML. The major deficiency was that the interpretation of

meaning of received messages is not described explicitly but is tightly embedded in the agent program code. This complicates further extension of the system as well as integration of new agents, because it requires reprogramming the existing agents to embrace the new conditions.

To overcome these problems and to allow automated processing of knowledge including reasoning, we have started to extend the MAST system to use explicit semantics. The agents use RDF and OWL ontologies described in Sect. 4 to handle, store and exchange the semantic information about customer orders, production plans and actual schedules [49] .

5.2 Extending ontologies for the DIAL scenario

The holonic packing cell installed in DIAL (Distributed Information and Automation Laboratory), formerly known as CDAC, at Cambridge University's Institute for Manufacturing [14] was used in past as a real-world platform for verification of the MAST agent-based control principles. The system receives customer orders in a form of selection from two box types and from selection of different cosmetic items (gel, razor, deodorant and foam) that should be placed to specific slots in the selected box. The system layout is illustrated in the Fig. 8. There are two independent production stations (marked as Workstation1 and Workstation2) containing a vertical, four-slot storage unit for holding the raw items and a Fanuc M6i robot that picks the items and places them into box. Empty as well as finished boxes are stored in an Automated Storage and Retrieval System (Workstation 3).

To be able to execute this scenario in MAST system, the DIAL-specific extensions has been created on top of the general ontology concepts described earlier. The order ontology has been extended with the description of types of boxes, with the description of different cosmetic items. The production plan ontology has extended by the pack operation that simply puts cosmetic items to a specified slot in a box. The orders are placed through the OrderManager component in the graphical user interface. Its task is to compose a semantic description of the order based on parameters specified by the user in the form of an RDF description compatible with the order ontology. The order agent created for the order then instantiates a new product agent giving it the specification of the order. The product agent then asks a special Production Plan Agent for an instance of production plan for box packing. This agent modifies the general production plan according to the user order. This modification includes association of parameters from the order to particular production steps (type of box, types of inserted items) and optionally deletion of some of the three steps for item inserting in case particular slot is requested to be empty. According to this production description, the product agent plans execution of these steps, i.e. searches the Directory Facilitator for agents providing operations associated with step. The cost-based model, extended contract-net protocol and RDF content language are used for negotiation about resource allocation between the product agent and the workstation agents. In bidding on cost of providing operation the workstation agent considers the availability of raw material (boxes and cosmetic items), capacity and load of workstation and product priority. The product agent selects the lowest cost and subsequently negotiates with the transporter agents representing shuttles about the delivery of product to the selected destination. Once the operation is finished, the product agent starts the planning of the next step defined in the plan ontology.

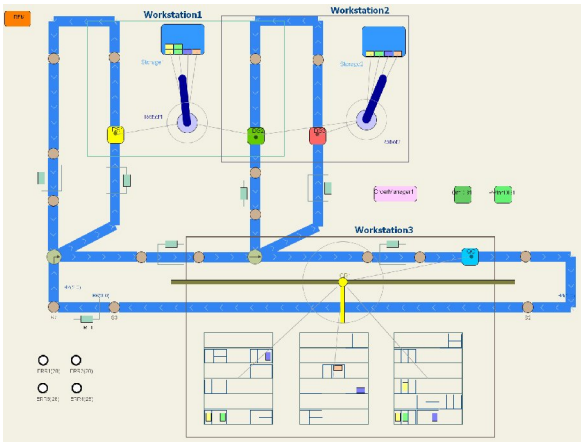

Figure 8: MAST System – DIAL scenario layout.

6 Conclusion

The main added features of the multi-agent system that uses ontologies as a base for information processing are extensibility, modularity and flexibility.

We have shown how the multi-agent production system can be modified for new types of products only by extending ontologies, without any need to reprogram or even shutdown and restart agents. The *extensibility* means that new kinds of physical machines with previously unknown manufacturing operations or new kinds of transportation means can be added to the system, and the system is able to cope with these changes without having to interrupt the system run. Moreover, new products with specific production plans, which are unknown in the design phase of the control system, can be added online without any further modifications needed. Thus it can be stated that *extending the system is only a matter of extending the ontology*.

The *modularity* of ontologies is inspired by software modularity. There are modules that serve to describe classes and properties common to the whole base system; there are modules specific for operation of part of the system; and there are modules that are application specific. Each of the modules can be extended separately – for example, we can extend parameters for customer orders without affecting the rest of the ontology modules. The customer order description extensions can be directly reused for production plan extensions thanks to the modularity and relations between ontology modules. In other words, the extensions are made only where necessary and do not affect the rest of the whole ontology. This is also possible, because an agent that is

designed for a particular task does not need to be equipped with all the ontology modules. The agent needs only selected module or modules.

Even when we do not consider updating agents during runtime, the use of ontologies increases the *flexibility* of the agent-based control system. In the DIAL scenario described in the previous section, the original version of the product agent was hard coded for a particular product type. Although the agent was able to discover alternative routing in the case of conveyor failures, the agent could not be used to control production of other product type without reprogramming. In the described semantically enriched solution, the production process is not described in the program code of the agent, but it is specified explicitly based on the shared ontology. The agent is then able to process such a recipe automatically, in such a way it schedules the execution of steps by negotiating with workstation agents about providing the operations associated with the steps. This means that the agent is much more flexible. This applies also to other agents when comparing the original implementation of MAST system with the semantically enriched version of the system.

The paper shows that the industrial automation domain, and especially intelligent control system, are being more and more influenced by semantic web technologies. The survey of existing ontologies applied in various applications like assembly or supply chain declares a need of standardized upper ontologies that would provide seamless information integration throughout the different levels of manufacturing enterprises. We have presented a compact, yet general ontology for description of discrete manufacturing processes and outlined guidelines for integration of this ontology in a multi-agent control system.

We have described a real-world scenario application of semantic technologies in the MAST system and have shown how semantic technologies can be employed to describe the operation and capabilities of the system not in the code of agents but rather in ontologies and knowledge bases. The advantage is that introducing new product type, introducing new or updated product plan, introducing new kind of machine or operation means only extension of one of the ontologies or updating knowledge base, and the agents in the MAST system do not have to be changed to be able to use such update.

7 Acknowledgements

This paper is extended version of our paper “Semantics in Industrial Distributed Systems” presented at the 17th IFAC World Congress.

This work has been supported by Rockwell Automation, by the Ministry of Education of the Czech Republic within the Research Program No.MSM6840770038: Decision Making and Control for Manufacturing III and by the FIT-IT: Semantic Systems Program, an initiative of the Austrian Federal Ministry of Transport, Innovation, and Technology (bm:vit) under the contract FFG 815132.

8 References

[1] Y. Al-Safi, and V. Vyatkin (2007). An Ontology-Based Reconfiguration Agent for Intelligent Mechatronic Systems. In: *HoloMAS 2007*, LNAI 4659, Springer, Berlin - Heidelberg, pp. 114-126.

[2] V. Andreev, G. Rzevski, P. Skobelev, and P. Shveykin (2007). Adaptive Planning for Supply Chain Networks. In: *HoloMAS 2007*, LNAI 4659, Springer, Berlin - Heidelberg, pp. 215-224.

[3] ANSI/ISA-88.01.1995 (1995). *Batch Control Part 1: Models and Terminology*. The Instrumentation, Systems and Automation Society.

[4] R. Batres, M. West, D. Leal, D. Priced, and Y. Nakaa (2007). An upper ontology based on ISO 15926. *Computers & Chemical Engineering*, Vol. 31, No. 5-6, pp. 519-534.

[5] F. Bellifemine, G. Caire, and D. Greenwood (2007). *Developing multi-agent systems with JADE*. Wiley, Chichester.

[6] S. Borgo, and P. Leitão (2004). The role of foundational ontologies in manufacturing domain applications. In: *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, LNCS 3290, Springer, Berlin - Heidelberg, pp. 670–688.

[7] R. Brennan, P. Vrba, P. Tichy, A. Zoitl, C. Sünder, T. Strasser, and V. Mařík (2008). Developments in Dynamic and Intelligent Reconfiguration of Industrial Automation. *Computers in Industry*, Vol. 59/6, Elsevier B.V, pp. 533-547.

[8] S. Bussmann, N.R. Jennings, and M. Wooldridge (2004). *Multiagent Systems for Manufacturing Control: A Design Methodology*. Springer, Berlin - Heidelberg.

[9] L. M. Camarinha-Matos (2002). Multi-Agent Systems In Virtual Enterprises. In: *Proceedings of International Conference on AI, Simulation and Planning in High Autonomy Systems,* SCS, Lisbon, Portugal, pp. 27-36.

[10] G. Cândido, and J. Barata (2007). A Multiagent Control System for Shop Floor Assembly. In: *HoloMAS 2007*, LNAI 4659, Springer, Berlin - Heidelberg, pp. 293-302.

[11] M. Cossentino, N. Gaud, S. Galland, V. Hilaire, and A. Koukam (2007). A Holonic Metamodel for Agent-Oriented Analysis and Design. In: *HoloMAS 2007*, LNAI 4659, Springer, Berlin - Heidelberg, pp. 237-246.

[12] M. Dean, and G. Schreiber (2004). *OWL Web Ontology Language reference*. http://www.w3.org/TR/owl-ref/ (Accessed 25 September 2007)

[13] FIPA. *ACL Message Structure Specification*. Available at: http://www.fipa.org/specs/fipa00061/index.html.

[14] M. Fletcher, and J. Brusey (2003). The Story of the Holonic Packing Cell. In: *Proceedings of 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, ACM Press, Melbourne, Australia.

[15] M. Grüninger, and J. B. Kopena (2005). Planning and the Process Specification Language, In: *Proceedings of WS2 ICAPS 2005*, pp. 22-29.

[16] N. Guarino, and P. Giaretta (1995). Ontologies and Knowledge Bases – Towards a Terminological Clarification. In: *Towards Very Large Knowledge Bases*, IOS Press, Amsterdam.

[17] Ch. Hahn, and K. Fischer (2007). Service Composition in Holonic Multiagent Systems: Model-Driven Choreography and Orchestration. In: *HoloMAS 2007*, LNAI 4659, Springer, Berlin - Heidelberg, pp. 47-58.

[18] I. Hegny, O. Hummer, A. Zoitl, G. Koppensteiner, and M. Merdan (2008). Integrating Software Agents and IEC 61499 Realtime Control for Reconfigurable Distributed Manufacturing Systems. *International Symposium on Industrial Embedded Systems*, June 2008, pp. 249-252.

[19] B. Hellingrath, M. Witthaut, C. Böhle, and S. Brügger (2009). An Organizational Knowledge for Automotive Supply Chains. In: *HoloMAS 2009*, LNAI 5696, Springer-Verlag Berlin Heidelberg, pp. 37-46.

[20] A. Helsinger, M. Thome, T. Wright, and T. Cougaar (2004). A Scalable, Distributed Multi-agent Architecture. In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, pp. 1910-1917.

[21] I. Herman, (2009). W3C Semantic Web Homepage. http://www.w3.org/2001/sw/ (accessed 11 April 2009)

[22] K. Jang-won, J. Keunhwan, J. Dongwon, K. Jinhyung, and B. Doo-Kwon (2006). An Ontology-based RFID System Model for Supporting Semantic Consistency in Ubiquitous Environment. In: *Proceedings of International Conference on Computational Intelligence and Security*, pp. 357-366.

[23] G. Koppensteiner, M. Merdan, A. Zoitl, and B. Favre-Bulle (2008). Ontology-Based Resource Allocation in Distributed Systems Using Director Facilitator Agents. *IEEE International Symposium on Industrial Electronics*, pp. 1721-1726.

[24] J. L. M. Lastra, and I. M. Delamer (2006). Semantic Web Services in Factory Automation: Fundamental Insights and Research Roadmap. *IEEE Transactions on Industrial Informatics*, Vol. 2, No. 1, pp. 1–11.

[25] H. Lausen, A. Polleres, and D. Roman (2005). Web Service Modeling Ontology WSMO. http://www.w3.org/Submission/WSMO/ (Accessed 26 July 2010).

[26] P. Leitao (2009). Holonic Rationale and Self-organization on Design of Complex Evolvable Systems. In: *HoloMAS 2009*, LNAI 5696, Springer-Verlag, Berlin - Heidelberg, pp. 13-24.

[27] S. Lemaignan, A. Siadat, J.-Y. Dantan, and A. Semenenko (2006). MASON: A Proposal for an Ontology of Manufacturing Domain. In: *IEEE Workshop on Distributed Intelligent Systems*, IEEE Computer Society Press, pp. 195–200.

[28] O. Lopez, and J. L. M. Lastra (2006). Using Semantic Web Technologies to Describe Automation Objects. *Int. J. Manufacturing Research*, Vo. 1, No. 4, pp. 482-503.

[29] O. Lopez, and J. L. M. Lastra (2007). A Real-Time Interface for Agent-Based Control. In *Proceeding of International Symposium on Industrial Embedded Systems SIES'07*, pp. 49-54.

[30] F. Manola, and E. Miller (2004). RDF primer. http://www.w3.org/TR/rdf-primer/ (Accessed 25 September 2007).

[31] C. McLean, Y. T. Lee, G. Shao, and F. Riddick (2005). Shop Data Model and Interface Specification, NISTIR 7198.

[32] M. Mendes, P. Leitao, F. Restivo, A. Colombo (2009). Service-Oriented Agents for Collaborative Industrial Automation and Production Systems. In: *HoloMAS 2009*, LNAI 5696, Springer-Verlag, Berlin, Heidelberg 2009, pp. 13-24.

[33] M. Merdan, G. Koppensteiner, I. Hegny, B. Favre-Bulle (2008). Application of an Ontology in a Transport Domain. *IEEE International Conference on Industrial Technology*, Sichuan University, Chengdu, China, pp. 1-6.

[34] M. Merdan (2009). *Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain*. Dissertation thesis, Vienna University of Technology.

[35] M. Obitko, and V. Mařík (2003). Adding OWL Semantics to Ontologies Used in Multi-agent Systems for Manufacturing. In: *HoloMAS 2003*, LNAI 2744, Springer, Berlin - Heidelberg, pp. 189–200.

[36] M. Obitko, and V. Mařík (2005). Integrating Transportation Ontologies Using Semantic Web Languages. In: *HoloMAS 2005*, LNAI 3593, Springer, Berlin -Heidelberg, pp. 189–200.

[37] M. Obitko, and V. Mařík (2006). Transparent Ontological Integration of Multi-Agent Systems. In: *IEEE International Conference on Systems, Man, and Cybernetics*, IEEE SMC, pp. 2488-2492.

[38] Obitko, M. (2007). *Translations between Ontologies in Multi-Agent Systems*. PhD thesis, Czech Technical University, Prague.

[39] M. Pěchouček, and V. Mařík (2008). Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. *Autonomous Agents and Multi-Agent Systems*, Vol. 17, Issue 3, p. 397-431.

[40] M. Pěchouček, M. Rehák, P. Charvát and T. Vlček (2007). Agent-based Approach to Mass-Oriented Production Planning: Case Study. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, Vol. 37, No. 3, pp. 386-395.

[41] E. Prud'hommeaux, and A. Searborne (2008). SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/ (accessed 11 April 2009).

[42] G. Rzevski, P. Skobelev, and V. Andreev (2007). MagentaToolkit: A Set of Multi-agent Tools for Developing Adaptive Real-Time Applications. In:

HoloMAS 2007, LNAI 4659, Springer, Berlin - Heidelberg, pp. 303-313.

[43] C. Schroth, and T. Janner (2007). Web 2.0 and SOA: Converging concepts enabling the internet of services. *IEEE IT Professional*, Vol. 9, No. 3, pp. 36–41.

[44] D. Šišlák, M. Rehák, M. Pěchouček, and D. Pavlíček (2006). Deployment of A-globe Multi-Agent Platform. In: *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1447-1448.

[45] The OWL Services Coalition (2004). *OWL-S: Semantic markup for web services*. http://www.daml.org/services/ owl-s/1.0/owl-s.html (Accessed 25 September 2007).

[46] P. Vrba (2006). Simulation in agent-based control systems: MAST case study. *Int. J. Manufacturing Technology and Management*, Vol. 8, No. 1/2/3, pp. 175–187.

[47] P. Vrba, F. Macůrek, and V. Mařík (2008). Using Radio Frequency Identification In Agent-Based Control Systems For Industrial Applications. *Engineering Applications of Artificial Intelligence*, Vol. 21/3, pp. 331-342.

[48] P. Vrba, V. Mařík, and M. Merdan (2008). Physical Deployment of Agent-based Industrial Control Solutions: MAST Story. In: *Proceedings of IEEE International Conference on Distributed Human-Machine Systems*, Athens, pp. 133-139.

[49] P. Vrba, M. Radakovič, M. Obitko, and V. Mařík (2009). Semantic Extension of Agent-Based Control: The Packing Cell Case Study. In: *HoloMAS 2009*, LNAI 5696, Springer-Verlag, Berlin - Heidelberg, pp. 47-60.

[50] V. Vyatkin, J. Christensen, J. L. M. Lastra, and F. Auinger (2005). OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Industrial Automation. *IEEE Transactions on Industrial Informatics*, Vol. 1, No. 1, pp. 4-17.