A Layered Manufacturing System Architecture Supported with Semantic Agent Capabilities

Munir Merdan[1], Mathieu Vallée[2], Thomas Moser[3], Stefan Biffl[3]

[1]Automation and Control Institute (ACIN)
Vienna University of Technology, Vienna, Austria
merdan@acin.tuwien.ac.at

[2]Institute of Computer Technology
Vienna University of Technology, Vienna, Austria
vallee@ict.tuwien.ac.at

[3]Christian Doppler Lab Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology, Vienna, Austria
{thomas.moser, stefan.biffl}@tuwien.ac.at

Abstract Manufacturing control systems are a mission-critical application domain for semantic agents systems. While multi-agent systems have been explored in the manufacturing systems domain, there is very little work on semantically enabled agent systems. This chapter introduces a layered architecture for manufacturing systems based on agent systems and discusses relevant capabilities of the semantic agents based on real-world use cases.

1. Introduction

The manufacturing sector, faced with growth in the variety of products and at the same time with a decreasing product life cycle duration, is forced by global competition to produce customized products in a short time at low price. Manufacturers have to be capable to effectively react to sudden changes in customer demands, as well as to cope with unpredictable events such as failures and disruptions. The aspects of complexity and flexibility of mission-critical manufacturing systems make this domain interesting as test bed for semantic agents systems.

A manufacturing system is defined as "*a collection or arrangement of operations and processes […] to make (a) desired product(s) or component(s)*" (Black 1991). Such a system consists of interrelated elements (people, equipment, subsystems, etc.) introduced to cooperatively achieve the overall objective of transforming raw material into commercial products effectively, efficiently, and robust against failures.

The control systems, which are currently applied in practice, usually consist of heterogonous units, which use different types of data and data structures, and are not capable to ensure the uninterrupted flow of information between and sometimes through the controlled levels. The applied methodologies in these systems are based on disconnected ordering, scheduling and execution processes, and lack the agility needed for enterprise-wide integration. Process planning is usually separated from scheduling as well as control activities and unnecessary information gaps between implicated systems are created, even though the outputs and data from one application could be fluently used as inputs for another application.

Multi-agent system (MAS) technologies offer a convenient way to cope with dynamics in large complex systems, using distributed control of the system, thereby reducing the complexity, increasing flexibility and enhancing fault tolerance. This approach replaces a centralized database and control computer by a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to that environment.

Agents communicate and negotiate with each other in order to perform the operations based on the available local information or to solve possible conflicts. In order to ensure correct understanding of the exchanged messages, agents must have the same presentation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. Ontologies are of vital importance for enabling knowledge interoperations between agents and at the same time a fluent flow the different data from different entities. The ontology can be a description of the concepts and relationships that can exist within a multi-agent system.

In this chapter we report on the engineering of the semantic agent architecture in four layers (management, planning, scheduling, and execution) and the related ontology for each particular layer in the application domain, the manufacturing system, and provide lessons learned based on real-world use cases.

2. State of the art

This section summarizes related work on manufacturing system control, multi-agent systems and on semantic systems.

2. 1 Centralized Manufacturing System Control

The factory control is defined "*… as the actuation of a manufacturing plant to make products, using the present and past observed state of the manufacturing plant, and demand from the market*". It is the fundamental system of a factory, because "*it coordinates the use of the factory's resources, giving the system its purpose and meaning*" (Baker 1998). Manufacturing control can be divided into low-level and high-level control (Christensen 2003). The high-level control (HLC) of the factory is responsible for the coordination of the manufacturing resources and government of the production including the ERP (Enterprise Resource Planning) as well as the MES (Manufacturing Execution System) levels. The low-level control (LLC) is focused on the control of the individual manufacturing resources and their reliable function during the execution of operations organized by the HLC.

In current manufacturing systems, the centralized and hierarchical structures are the most commonly used control architectures. However, due to their rigid character and limited adaptation capabilities such systems respond weakly to frequently changing customer demands in terms of performing necessary changes in the manufacturing environment itself (Jones and McLean 1986; Parunak 1996; Shen and Norrie 1999). Additionally, the construction of a centralized system, due to large complexity and the necessity to centralize all logic for sensing, actuating and control into a single entity, usually requires a huge investment, long lead times, and in turn, results in generating a rigid control system (Colombo, Schoop et al. 2005). The central controller, as it needs to have the accurate information about each unit in the system in order to make right decisions, can be seen as a single point of failure and its breakdown could stop the whole system (Krothapalli and Deshmukh 1999). Scheduling, in centralized and hierarchical control structures, is established such that each level creates the scheduling for its subordinate levels having a weak feedback from lower levels and almost without any consultation and coordination with higher layers of neighboring units. Such an approach works well only if everything goes as expected; otherwise it could completely fail when unpredictable disturbances occur (Bussmann, Jennings et al. 2004).

2.2 Multi-Agent Systems as Foundation for Decentralized Control

The application of decentralized control architectures based on autonomous and co-operative units is considered as a promising approach for overcoming the weaknesses of centralized manufacturing control. The multi-agent systems (MAS) approach has been widely recognized as enabling technology for designing and implementing the next-generation of distributed and intelligent manufacturing systems (Bussmann, Jennings et al. 2004; Pěchouček and Mařík 2008). Making the control of the system decentralized, intelligent agents offer a convenient way of

modeling processes and systems that are distributed over space and time, thereby reducing the complexity, increasing flexibility and enhancing fault tolerance (Jennings and Bussmann 2003). MAS can be defined as a network of autonomous, intelligent entities – agents – where each agent has individual goals and capabilities as well as individual problem-solving behaviors. Due to their lack of a global system objective and overview, agents have to cooperate and communicate with each other in order to achieve common aims, which are beyond the individual capabilities and knowledge possessed by each agent. There are two architecture approaches for agent encapsulation in agent-based manufacturing systems: the functional decomposition approach and the physical decomposition approach (Shen and Norrie 1999). In the functional decomposition approach, agents are used to encapsulate modules assigned to functions such as an order, task, etc. In the physical decomposition approach, agents are used to represent entities in the physical world, such as a robot, conveyor, or pallet.

2.3 Agent Systems facilitated by Semantic Technologies

As agents are applied in a distributed and heterogeneous environment and have by themselves only partial representations of the environment, these agents have to communicate with each other to coordinate their activities. Semantic technologies, such as ontologies have been developed and investigated in the areas of artificial intelligence and natural language processing to facilitate knowledge sharing and reuse (Kulvatunyou, Cho et al. 2005). Semantic technologies are vital for enabling knowledge interoperations between agents and, at the same time, a fluent flow of heterogeneous data from a range of entities. Ontologies allow the explicit specification of a domain of discourse, increase the level of specification of knowledge by adding semantics to the data, and promote knowledge exchange in an explicitly understandable form.

An ontology is defined as an explicit specification of conceptualization (Gruber 1993), where conceptualization means the shared view of environment representation. From the viewpoint of inter-agent interactions, the explicitly defined and commonly accepted ontology is an indispensable tool for ensuring interoperability between agents in the sense of providing a formally defined specification of the meaning of those terms which are used during the inter-agent communication. Ontologies can also capture actions and events in a uniform and processable way so that they can be recorded in time and further analyzed. The usage of ontologies for knowledge representation, sharing and high-level reasoning could be seen as a major step ahead in the area of agent-based control solutions (Obitko and Mařík 2002).

Nevertheless, ontologies have been rarely used with software agents and most of the existing MAS are not aware of ontologies at all: the information processing and reasoning are hard coded in the agents' behaviors. Although important stan-

dardization work has been done by introducing the message transport service for sending FIPA-ACL Messages (Foundation 2003) by defining message types, protocols, etc., the agents are not able to semantically interpret the domain-specific content of the exchanged messages or the knowledge held by other agents (Qiu 2005).

3. Research Issues

Multi-agent system (MAS) technologies offer a convenient way to cope with dynamics in large complex systems, in our case manufacturing control systems. This approach replaces a centralized database and control computer by a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to that environment. Each agent is a representation of a manufacturing component and serves as an artificial "brain" of the real-world physical component. The agents are supposed to supervise the physical components of the system and to cooperate with other components. The MAS approach has proven to successfully handle complexity and dynamics in a number of comparable systems.

Agents communicate and negotiate with each other in order to perform the operations based on the available local information or to solve possible conflicts. Inter-agent communication capability provides the essential means for agent collaboration. In order to ensure correct understanding of the exchanged messages, agents must have the same presentation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. Ontologies are of vital importance for enabling knowledge interoperations between agents and at the same time a fluent flow of the different data between different entities. Ontology is defined as an explicit specification of conceptualization. Explicit specification of conceptualization means that ontology is a description of the concepts and relationships that can exist within a multi-agent system.

A key research question is how to reduce the complexity of a manufacturing system. Therefore, we propose to divide the control of a manufacturing system into "hierarchically" ordered layers. The layered system structure enables the functional decomposition into subsystems, which can then be easily further decomposed but also integrated and managed in bigger systems as well. In the layered structure specification, besides the fact that particular subsystems logically symbolize some layers (e.g., planning and planning control layer), we consider that particular decisions have to be made in a real time requiring observation of a limited environment in contrast to decisions that require a global view without a special time limitation. The introduction of layers limits their responsibility and planning perspective, improving system performances and simplifying the concept.

In the context of a research project platform based on a real-world industrial use case we describe the layers, their contributions, and report lessons learned from the research work.

4. A Layered Manufacturing System Architecture

The layers of control in a manufacturing system correspond to planning and decision tasks in the domain and can, in general be divided between high-level control and low-level control. In more detail we specified four layers: management, planning, scheduling and executive layer (see Fig. 1) (Merdan 2009). These layers structure agent-based systems control and provide the context for deriving the necessary semantics. The functions and responsibilities of the four layers are defined as follows:

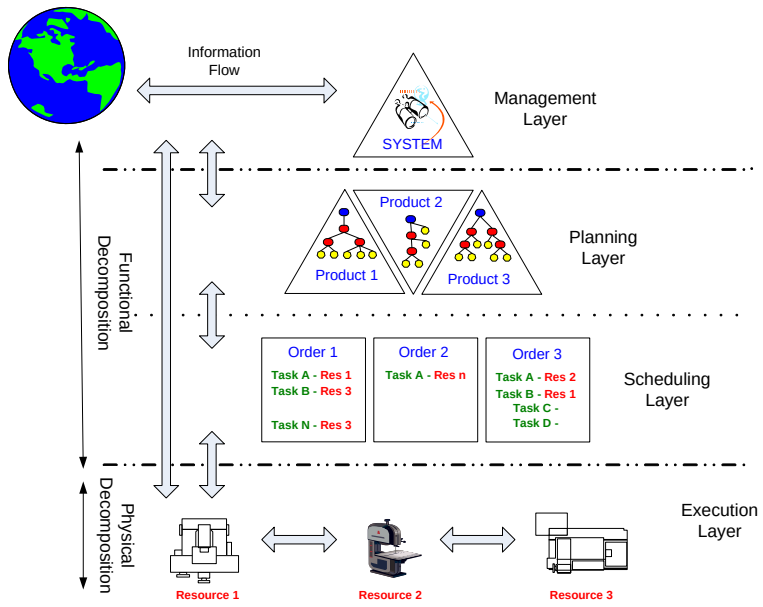

Fig. 1. The layers of manufacturing system control.

- The *Management Layer* is responsible for entire system stability and functionality. It supports production and resource initialization as well as their determination. It is also concerned with the communication with the external environment and provides solutions for complex problems related to the global environment. It accepts production orders on a routine basis.

- *The Planning Layer* links process planning with product design. It is basically concerned with the sequencing of process steps, identification of product types and quantities to be produced. It defines equipment and resources that could be used and ensures that the parts or components required for the production are available and the final product delivery dates not exceeded. The shop floor layout is also defined on this layer.
- *The Scheduling Layer* is concerned with the synchronization of production needs with available resource capacities. The goal is to reach the internal deadlines that are set on the planning level. This layer is responsible for negotiating with the resources, the tasks as well as parts, tools, and product allocation between resources.
- *The Execution Layer* is related to the physical job shop equipment. On this layer, the production tasks are executed considering the resources' constraints and abilities, their performances measured and if a failure or disruption is diagnosed, the scheduling as well as management layer is informed. Also specific activities related to the execution of particular actions (i.e., pallet routing, removing, fixing, etc.) are coordinated on this layer.

In this chapter we will present an approach where the manufacturing layers have been "agentified" to implement behaviors that represent the specific layer or the component of this layer as well as their objectives and functionalities, with each agent being responsible for carrying out different specific functionalities. We used the functional decomposition approach to create agents responsible for system support, process modeling and task scheduling (see Fig. 1), applying this approach to each of the three upper layers and generating particular agent types for each layer. We also applied the physical decomposition to create the related agents that have physical representation. In the rest of the chapter we will present the multi-agent system that governs specified manufacturing layers. Considering the role of the ontologies to enable data and information interoperability in extensive heterogeneous and content rich environment, we present the approach that integrates the related agent types at each manufacturing layer with ontologies. In the remainder of the chapter we will present the resulting multi-agent architecture.

5. The Management Layer

This section describes the management layer. First, a short introduction to Enterprise Resource Planning (ERP) and Virtual Enterprises is given. Then, the used layers and agents are introduced, followed by a description of the production process cycle used throughout this chapter.

5.1 Enterprise Resource Planning (ERP) and Virtual Enterprises

One of the most important trends in current IT development for companies is the trend to support and optimize the daily business in all areas of the business. Primary goal when deploying and using an ERP-System in a company is to increase the efficiency of established processes by supporting them through information technology. If the adaption of this system is a success and it supports the daily to-dos of the employees, it is over all a big opportunity to save time and money for the company. Of course, the first problem of establishing the new system and major investments has to be carefully planned (Hansen and Neumann 1982).

ERP-software involves the business task to plan the resources of a company like money, personal or equipment persistently, carefully and in an efficient form during the whole production process. Therefore companies need more and more IT infrastructure which gets even more complex and so harder to maintain. The simulation of the production flow by multi agent system tools also belongs to this package of planning software. Furthermore it can be an essential part of it and complete the functionality of standard ERP-software by providing further information about the production process and add important data into the central available data source. In this case, the ERP-software acts as media, which transports this information towards all roles that need the data.

A virtual enterprise (VE) (Kulvatunyou, Cho et al. 2005) is seen as an integrated network of regular companies that join their core services and resources in order to respond to unexpected business opportunities collaborating on an ad hoc basis. Such a network includes also suppliers, distributors, retailers and consumers requiring from involved companies to gather and share data and information about markets, customers and internal competences (Aerts, Szirbik et al. 2002). The capability of companies to form virtual enterprises and cooperate with partners is an important factor for keeping a competitive position on the market.

A new approach for virtual enterprise modeling as well as the fulfillment and consideration of several research challenges, such as improved knowledge exchanging and sharing, fast reaction to customer demand, re-organization capability, and integration of heterogeneous entities, are required (Roche, Fitouri et al. 1998). The introduction of tools, techniques and methodologies that will support interoperability, information search and selection, contract bidding and negotiation, process management and monitoring, etc., is also highly required (Camarinha-Matos 2002). In this context, the information and knowledge exchange between partners plays a critical role for the success of such networks. This is particularly due to the extreme heterogeneity of the VE environment, in which it is usually not transparent to the partners, which knowledge is available at whose partner's site or even if so, then in most cases the knowledge is not understandable due to the usage of different formats and tools.

Ontologies have been developed and investigated for quite a while in artificial intelligence and natural language processing to facilitate knowledge sharing and

reuse (Kulvatunyou, Cho et al. 2005). They are of vital importance for enabling knowledge interoperations between partners and, at the same time, a fluent flow of different data from diverse domains. Ontologies allow the explicit specification of a domain of discourse, increasing the level of specification of knowledge by incorporating semantics into the data, and promote its exchange in an explicitly understandable form (Silva and Rocha 2003). Semantic means in this context that all relevant concepts important for partners will be modeled in an ontology by capturing the associations between the domains ensuring at the same time the understanding of exchanged knowledge during the inter- as well as inside-company communication. This allows business partners to build open communities that define and share the semantics of the information exchanged in their domain.

5.2 Layers and Agents

As shown in Fig. 2, each of the layers introduced in the previous section involves a certain type of agent, which will be described in the following section. For the management layer, this agent type is the so-called *Contact Agent* (*CA*).

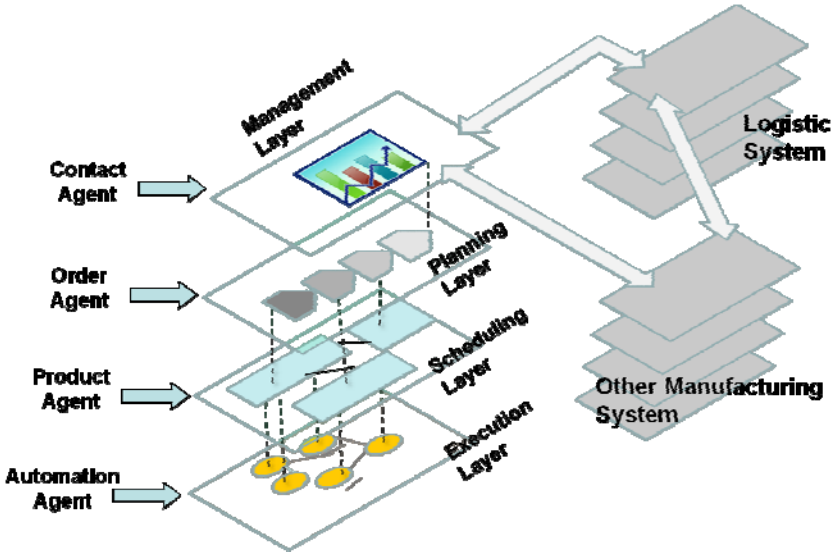

Fig. 2. Overview Agents and Layers.

The *Contact Agent* (CA) is related to the Management Layer and accordingly it has responsibilities that encompass organizational and supervisor functions. The CA is created at the start-up of the system and it is always active. It is concerned with the system stability and in the case that one part of the system collapses; this

agent considers its influence on the system performance and, if significant, undertakes particular steps in order to bring the system back into the optimal state. Its further responsibilities are to receive a customer order and create one Order and Supply agent for each related product order. This agent also creates an agent for each new resource introduced in the system. After the order was accomplished or particular resource removed from the system, the CA determinates the related agent. However, having only one instance of this agent for the whole system and considering it as a possible single point of failure, the replication technique (Mellouli 2005) and replication service provided with used agent platform (Bellifemine, Caire et al. 2008) can be applied to enhance agent's failure tolerance level.

5.3 Production Process Cycle

The following description shows a possible execution of incoming orders of customers for a company. This designed process is the economic background and core of the project focused in this chapter to reproduce a manufacturing assemble line to simulate actions on a flexible production unit. The optimization focus of the job-shop-scheduling is realized over simulation runs with different parameter settings to maximize the production system output. The explanation describes the involved roles on the one side and necessary information flows on the other side.

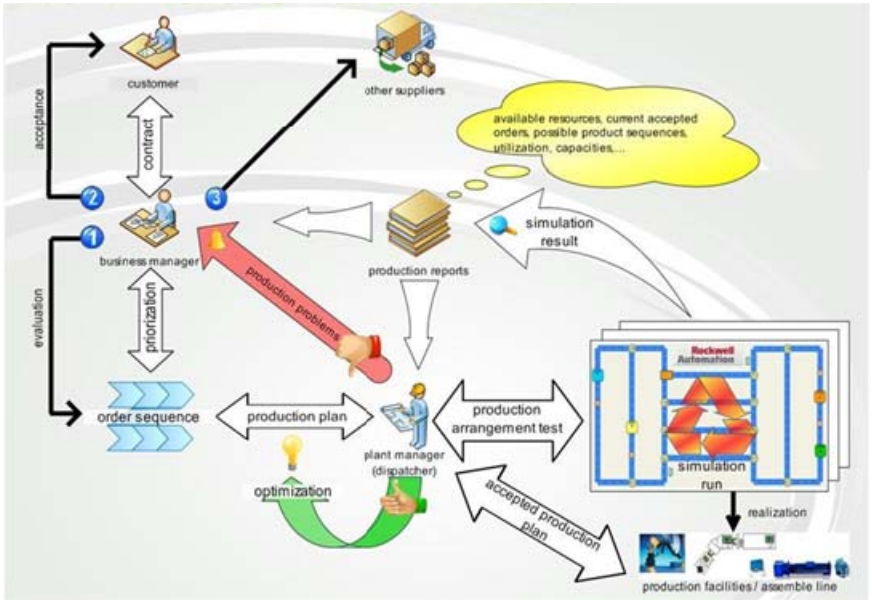

Fig. 3. Production Process Cycle (Merdan, Moser et al. 2008).

As outlined in Fig. 3, the normal treatment of incoming orders from customers involves more than one enterprise-internal role in a coordinated and balanced task sequence. The described draft of the process shows the close dependencies between all involved acting units/agents. In addition the draft expresses the possibility, how appearing production complications like resource bottlenecks, machine or conveyor failures or capacity overruns are handled within a production planning and control system. To use the available capacity optimally, the reactions to solve such unexpected problems on the first level are focused directly at the executive layer by the involved role. Only if this is not possible an agent on a higher level receives an exception ticket to avert the danger. The role/agent/execution unit on the higher level eventually can use other possibilities to solve the problem because of more available information or more granted authority (e.g., it is not possible to fulfil the order with the actual time and resource restrictions – they have to be extended or manufacturing of the product has to be outsourced).

Starting point for all production planning actions is the incoming detailed order of the customer. The assumption of the order can take place on several ways: personally at negotiations (completion of contracts at large orders) written by letters or forms, by telephone or on new media like the internet (online orders via E-commerce-trading, web-shops, etc.). In the optimal case, the orders arrive already in digital form on prepared web forms at the responsible person which acts in the role of a customer adviser. In this manner media breaks are eliminated as a source of error and the passing-on/takeover of the order into the system used in the company for the order management as well as the PPC is accelerated immensely.

Then the business manager takes over the order winding by making a first optimisation as a next step. Therefore he lines up the received orders to produce the requested quantity of goods in time according to certain criteria (such as sequence of the incoming orders, dates of delivery, urgency of the ordered goods for the customer, etc.) which are actually decisive for the enterprise success.

In this manner the first order sequence arises considering certain priority criteria and thus serves as an input for the production planning of the plant manager.

The job of the plant managers is to distribute the order list transmitted to him efficiently to the available capacities and resources. However, besides he should be anxious to keep the predefined sequence to avoid unnecessary difficulties. His concrete job is to compile an efficiently allocation plan for the available production lines under the current time, capacity and resources restrictions. Available simulation tools can be used by the plant manager to complete this job. By defining the available orders by configuration of input parameters (e.g., number and kind of products) as well as different settings of factors which influence the production process (e.g., speed of conveyor belts, available free pallets, shift duration, assemble line arrangement) the expected production process can be predicted to find out the best production sequence. This simulated and tested production order can be transmitted to the physically existent production units as an optimized production plan.

The assemble lines should preferably act automated to carry out the transmitted production instructions to fulfil their manufacturing task. During the whole production process data of the current state (order, product sequence, number of pieces, order status, utilisation, …) is collected and made available on the information system to the different roles of the working on unities and employees.

If failures occur during the production, the units/agents of the assemble line automatically react to the resulted failure to compensate (compensation at operation layer) without further intervention. Only if this is not possible any more, i.e., the entire processing by resource lack, machine failure, overload, etc. cannot be executed in the planned time, a suitable announcement about the production report is done.

6. The Planning and Scheduling Layers

The production planning and scheduling issues are of essential importance for the manufacturing domain today, especially due to the dynamic and competitive nature of the nowadays global market that needs enterprises to be adaptive, flexible, robust and collaborative. The process planning is usually separated from scheduling and unnecessary breaks between these systems are created, even though the outputs and data from one application could be fluently used as inputs for another one. In this section we present an approach, how to integrate these layers using agent technology and ontologies.

6.1 Planning

Process planning (PP) plays a very important role in the product life cycle by linking the product design with the manufacturing phase. Process planning resolves between what and how will be produced. The process planning phase has to consider the product requirements (price, quantity, geometry, tolerance, material, etc.) as well as production constraints (machine capacity, tool characteristics, etc.).

Due to the lack of intelligent capabilities, current process planning systems have difficulties to automatically adapt plans according to the availability of resources, or share knowledge among the various planning related functional modules (Bose 1999). Additionally, a low frequency of planning runs and difficulties in coping with new organizational forms of manufacturing such as product oriented or customer driven production, require new skills and new approaches capable to handle the shortcomings mentioned above (Azevedo and Sousa 2000).

6.2 Application of Agents in Process Planning

As a possible way to overcome the shortcomings of the decentralized architecture that spreads the planning process between several entities/agents, each capable to create, control, and observe the execution of its own plans, is suggested. The agents cooperate and coordinate their actions in order to effectively accomplish their plans as well as to reach the commons system goals. Such organization brings time improvements, since the complex problems are partitioned between the entities by giving each entity a part of the problem to solve instead of dispatching the whole problem to the central unit, what could cause difficulties especially when the data is voluminous and changes frequently. The distributed agent-based approach allows proactive data processing at the place of its origin and data exchanges are only those necessary for effective system functioning (Pěchouček, Vokrinek et al. 2005). Moreover, large and complex problem structures become more simple and the possible failures easier to track (Seilonen, Pirttioja et al. 2003). They conducted an extensive literature reviews related to agent-based collaborative process planning done by Zhang et al. (Zhang and Xie 2007).

6.3 Production Scheduling

The task of scheduling is the allocation of jobs and activities to available resources over time considering relevant constraints and requirements (Rajpathak, Motta et al. 2006). Its main objectives are the minimization of the production time of jobs, production costs, increased resource utilization, etc. Most of the developed scheduling systems are based on centralized structures, which make manufacturing systems scheduling even more complicated (Shen, Wang et al. 2006). The application of agent technology can significantly improve the efficiency and performance of the entire system (Merdan, Moser et al. 2008). Extensive surveys of dynamic scheduling in the manufacturing environment considering also agent-based systems were done by Babiceanu and Chen (Babiceanu and Chen 2006) as well as by Ouelhadj and Petrovic (Ouelhadj and Petrovic 2009).

6.4 Integration of Process Planning and Scheduling

Process planning and scheduling are highly related, because when the planning ends the scheduling phase starts. The process plan restrictions and shop floor constraints have to be considered in the scheduling phase that could become a very complicated and time consuming process, if applied in a dynamic environment. In order to make more realistic and applicable plans, the integration of the planning

and scheduling phase is necessary. Nevertheless, the traditional approaches execute these processes separately, mostly ignoring the condition of resources on the shop floor (e.g., machine workloads, etc). That leads to the under- or over-utilization of certain resources or even that some of the process plans perhaps cannot be executed requiring alterations or replanning (Kumar and Rajotia 2006). A lot of work has been done in the past to optimize and integrate process planning and scheduling in the area of manufacturing (Shen, Wang et al. 2006).

However, one of the main shortcomings of the presented architectures is the lack of interoperability, since the applied methodologies separate planning activities (e.g., process planning) from executing activities (e.g., production control and scheduling), creating a gap between the involved systems. The problem in current distributed systems is that they are still tightly coupled from the point of view of automated gathering and integration of data, information and knowledge, being programmed with the focus on performing particular tasks rather than on interoperability and openness (Obitko, Vrba et al. 2008). Shen et al. defined the integration of process planning, manufacturing scheduling, and control as one challenging research topic where much more attention has to be set on the complexity analysis and formal modeling of such integration (Shen, Wang et al. 2006). The assimilation of different knowledge sources is considered as an important problem that has to be solved being marked as not easy task due to different representations, foundations, and levels of abstraction of various knowledge sources (Bose 1999). Being mostly applied in heterogeneous environment, agent has to understand it as well as the knowledge of related agents in order to reason about it, prior to making decisions. Moreover, considering that future distributed manufacturing systems will need to handle a great diversity of autonomous agents and mechatronic devices interacting intensively, there is as strong need that all components understand the exchanged information and know how to communicate (Christo and Cardeira 2007). According to Finin et al. (Finin, Labrou et al. 1997) for software agents to interact and interoperate effectively three fundamental and distinct components: (i) a common language; (ii) a common understanding of the exchanged knowledge; and (iii) the ability to exchange whatever is included in the previous two; are required. The usage of machine-interpretable semantics (ontologies) to describe the components of manufacturing systems enables other intelligent components (agents) to perform reasoning and infer sufficient knowledge to interact as well as to overcome current interoperability barriers (Lastra and Delamer 2006).

6.5 Planning and Scheduling in the Assembly Domain

In order to automate the process planning generation, the product model representation has to be made in a way that enables understanding the designer's intention, offer information about specific features (connections, definitions, constraints, etc.) that could be used for the selection of appropriate equipment as well as tool

set-up definitions. At the same time, knowledge about the capabilities of the equipment could facilitate the product design and ensure its manufacturability. The ability to present the product model in a same way as production process and production equipment can support easier mapping between these three key manufacturing elements and enable easier optimization of both planning and scheduling process.

Assembly is much more than a process where two or more parts are connected, since the whole process is accompanied with preceding as well as following actions (supply, transportation, inspection, handling, delivery, etc.). "Assembly model" or models must be capable of capturing a diverse set of information needed to describe the entities and activities associated with assemblies and assembling so that designers of products, assembly systems, logistic systems, supplier relations, field support, and finally disassembly and recycling, can have access to the information they need (Whitney 1996). However, the lack of ways to standardize and describe assembly domain knowledge is an obstacle to achieve an easy flow of information. This is the reason why we select the assembly domain and its automation as test case for our knowledge-based multi-agent concept.

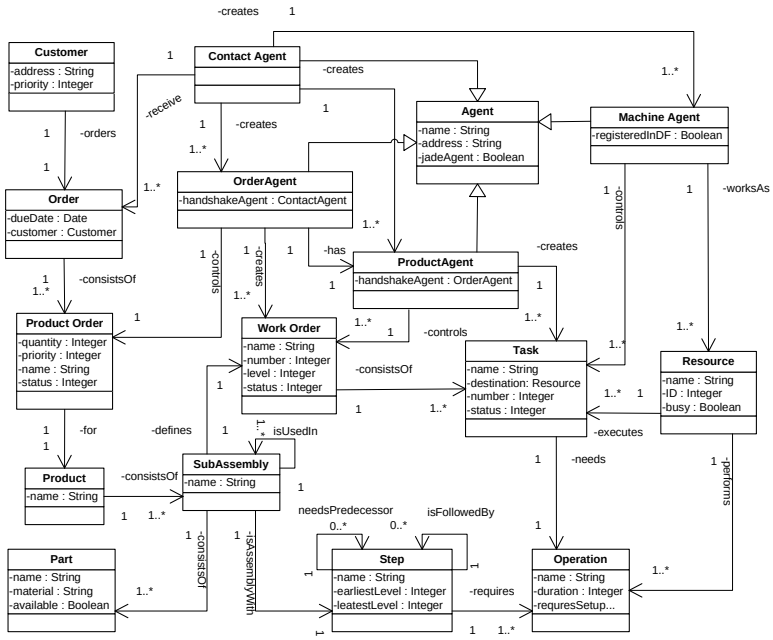

Fig. 4. System Ontology.

Traditionally, the product model is geometric based and provides incomplete product definitions because besides the assembly geometry, the understanding of its physical effects as well as the design intentions (e.g., joint type) is required.

The meaningful representation of product data is necessary to enable semantic interoperability across different application domains (Patil, Dutta et al. 2005). The ontology-based assembly model was presented in (Kim, Manley et al. 2006) and serves as a formal, explicit specification of the assembly design so that it makes assembly knowledge both machine-interpretable and shareable at the same time. We use the same concept to link product designs, assembly planning processes and required assembly equipment together. The ontology based product model is used to extract the production/assembly operations from the product design and link particular tasks, which have to be performed for the production/assembly of a *product*, to particular *resources*. Each *task* consists of a series of actions that can be executed by one or more *resources*. The connection to the *product order* is made through the type of ordered product, quantity, which defines the number of parts that have to be available to start the assembly process, and due date that defines the priority of the order.

In this context, a *product* is presented as a hierarchy of subassemblies and parts together with all their properties and relationship between them. *Parts* are defined as components, described by a set of attributes, properties, constraints and relations to other parts. A *subassembly* is a non-empty subset of parts that either has only one component or is such that every part has at least one surface contact with another part in the subset (Rabemanantsoa 1993). The relationship between parts within a subassembly defines operations that have to be done to connect these parts and represents how these subassemblies should be put together to complete the product. An *operation* is defined as a discrete set of actions which leads to a certain change of state in or on the part.

Each product type is described through its own process plan. A *process plan* (assembly tree) specifies the sequence of manufacturing or assembly operations which have to be performed in order to make a product. Each operation could be performed by different resources and consequently can be accompanied by related transport operations, making all together *work orders*. The ontology-based concept of the product production/assembly described with *Steps* ensures the exact decomposition of the product orders to related work orders and further associated tasks and their correct indexing. This is particularly supported with the integrated planning relationships *needsPredecessor* (Step) and *isFollowedBy* (Step) that enables an agent to reason when and why to start particular task allocations, which are known as scheduling activities. A *resource* is a physical component able to perform a certain action. However, since this component embodies agent as its control part, we consider also the agent concept as integral resource element.

In order to reach the goal of modern assembly planning systems to create activity sequences that are not only feasible but also optimized according to one or more parameters, such as makespan, machine or tool utilization, the agents that are supervising a particular resource or process planning system can use the accurate information stored in the ontology to reason about available resources and utilize appropriate optimization heuristics (Lastra and Delamer 2006). We are using this ontology to provide semantic understanding among software agents. The

agent interaction with the ontology in the background ensures that when an agent extracts relevant information from a message it understands the meaning of the terms in the message and the way this terms are combined in the statement. The presented concept distribution and ontological representation of a production process improves the way components communicate and exchange information in the manufacturing environment. Our ontology covers the environment structure, characteristics, states and components interrelationships enabling the related agents to interpret their environment, reason about it and make right decisions.

We used the functional decomposition approach to create agents responsible for process modeling and task scheduling (see Fig. 1), applying this approach to each planning and scheduling layer and generating particular agent types for each layer.

The ***Order Agent*** (OA) captures the goals and tasks of the Planning Layer. The OA is responsible for accomplishment of one product order, respecting due dates and the like; and handling customer requests for modifying or cancelling their orders. The essential information for an order agent is: type of product, the production deadline, quantity, and the priority of the client. Having the knowledge about all products corresponding to a single order, this agent combines the ontology-based model for a particular product together with other information, sequences this into work orders and sends it to the supply agent. Based on this knowledge and contacting the storage, the OA checks if all parts and materials required for execution of a single order are available. During the production, this agent collects also information concerning the status of current product orders or the system's performance. The OA is responsible for loading products into the system when a product order reached the system and for unloading products from the system when all of their processes are finished.

The ***Product agent*** (PA) maps the functions of the scheduling layer and has such name since it "supplies" the job shop with tasks. The PA is in charge for coordinating the production execution in order to achieve the best possible production results, including on-time delivery, cost minimization, and so forth. It also manages the movement of related product order's subassemblies and materials across the job shop. After the OA decomposes the product order into work orders, they are forwarded to the PA. Using the ontology and taxonomic relations specified in the product definition (see Fig. 4), the PA extracts tasks from work orders and schedules the ones that have to be completed at first. After that, the PA initially sends requests for bids to all machine agents that have the capability to process the first task. The interested machine agents respond with their bids. Each bid contains an estimated queuing time and finishing time for the requested operation. After collecting the bids, the PA evaluates the bids and selects the best one. When the related machine is identified, the agent negotiates with transport agents to route the task there. Whenever a current task is completed, this agent sends bid requests for the next operation. This bidding procedure continues until all the requested features of a job are finished. When the last task in the production process is finished, the agent sends the notification to the OA.

The important advantage of the introduced ontology-based approach is the achievement of the preconditions for easy assembly and disassembly of the product. Our knowledge-based system does not need to be told, how a problem has to be resolved (i.e., which and when particular tasks have to be done), but the concept and the goal is described instead. The system decides on its own how to achieve the goal.

7. The Execution Layer

Within the Layered Manufacturing System, the execution layer is responsible for the execution of production tasks. More specifically, the main role of the execution layer is to integrate the basic functionalities provided by physical resources (e.g., drilling, transporting pallets) into the Layered Manufacturing System.

In this section, we first recall the general requirements on the execution layer. We then detail the design of the execution layer as a system of semantic agents. Finally, we discuss some lessons learned in using semantic agent technologies in the execution layer.

7.1 Requirements of the Execution Layer

In order to correctly play its roles in the Layered Manufacturing System, the execution layer needs to satisfy a number of requirements. We could categorize these requirements into three groups: requirements regarding the *execution of production tasks*, requirements regarding the *robustness of machine control*, and requirements regarding the *diagnosis of disturbances and failures*.

Providing the Layered Manufacturing System with means of executing production tasks is the main requirement of the execution layer. More specifically, this can be decomposed into two aspects: identifying the appropriate resources for executing a production task and controlling the execution of a particular task. To do so, the execution layer relies on a set of machines (mechatronic components), each capable of performing specific tasks. Because of the heterogeneity of these machines, the identification of appropriate resources is not straightforward. Therefore, the execution layer must be able to match the requested tasks with available resources. In addition, the execution layer should consider the runtime constraints of the machines (load, temporary unavailability, etc…), in order to obtain the most suitable solutions.

Providing a robust control of production machines is a major requirement for the execution layer. To do so, the execution layer should keep the various control algorithms and components as independent and loosely-coupled as possible, in order to avoid local failures to hinder the whole system. Decomposing control into

smaller, well-defined control units also makes it easier to design and to test, thus enabling a better quality.

Being able to diagnose disturbances and failures is an important requirement. In such as physical system, it is not possible to assume that all machines will function seamlessly and always respond as expected. On the contrary, disturbances and failures can happen at any time, and their occurrence is likely to impact the production process negatively if they are not tackled properly. Therefore, the execution layer must cope with unexpected disturbances and failures. In particular, it should be able to detect disturbances and failures, to determine their impact of the production tasks it is in charge of, to report problems to the scheduling layer, and possibly to elaborate recovery solutions when possible.

7.2 Semantic Agents for the Execution Layer

This section presents an overview on semantic agents used in the execution layer. We call them automation agents (Vallée, Kaindl et al. 2009), as these semantic agents are more specifically dedicated to the control of mechatronic components.

The execution layer consists in various types of automation agents, corresponding to the various mechatronic components and machines (resources) available in the factory. Fig. 5 presents an overview of some automation agents of the execution layer. Each agent is responsible for the control of its dedicated resource, and can only influence the behavior of another resource by sending a request to the corresponding automation agent.

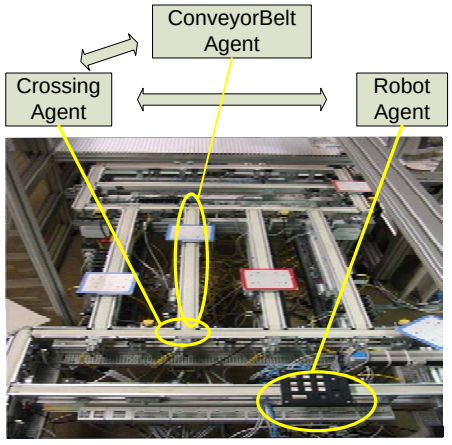

Fig. 5. Overview of some automation agents in the execution layer.

The Automation Agent Architecture

In order to facilitate the design of agents for specific mechatronic components, we define a generic architecture for automation agents (Vallée, Kaindl et al. 2009). This architecture distinguishes between a high-level control (HLC), a low-level control part (LLC) and the mechatronic component itself (see Fig. 6).

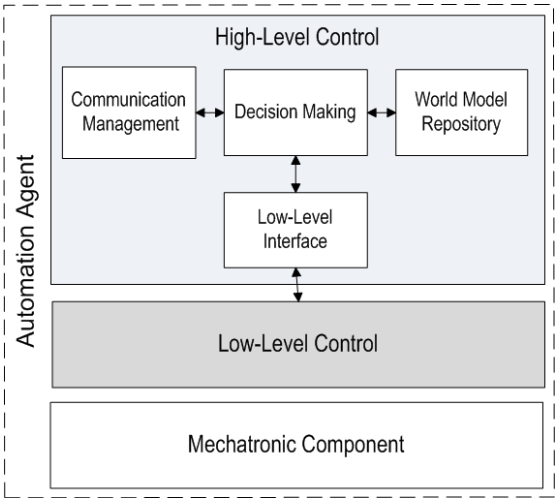

Fig. 6. Internal architecture of an automation agent.

The LLC is built on top of the physical mechatronic component and is directly responsible for managing the underlying physical system. It consists in "pure" control software and comprises a limited set of reactive behaviors to directly control the physical component, collect and process the information from sensors and based on the result perform particular actions. The LLC is particularly designed to perform in real-time. It can also diagnose certain types of failures (e.g. a conveyor stock) and informs the upper layer about it.

The HLC is composed of four main modules:

- The world model repository contains a world model, i.e., a symbolic representation of the world of the agent. The world model repository provides facilities for updating and querying the world model.
- The low-level interface enables the agent to use functionalities provided by the LLC. It especially provides facilities for receiving event notifications about the current operations of the LLC and for requesting particular operations from the LLC. Because various kinds of LLC may exist in a system, we use a unified low-level interface as described in (Lepuschitz, Vallée et al. 2009).
- The communication manager provides facilities for managing communication with other agents. The communication between agents depends on the knowledge they have to exchange and on the tasks they have to achieve collectively.

- The decision-making component is in charge of reasoning about the states of the world and deciding what to do (e.g., communicate with other machines, request an operation from the LLC, issue notifications to an operator). Event notifications generated by the low-level, by communication with other agents or by the world model trigger the decision-making procedures. These procedures then update the world model, request operations from the LLC and communicate with other agents.

Semantic Technologies for Automation Agents

Within an automation agent, semantic technologies are mainly used for dealing with the representation of activities. In our context, an activity can be defined as "a process occurring in the world in which the agent is participating". This definition not only covers actions directly performed by the agent, but also the occurrence of events that the agent only observes. In the first case, the agent is an actor in the activity, while in the second case it is only an observer. In the automation domain, both aspects are useful, as a change in the environment often happen without an agent being directly responsible for the change. Using this general notion of activity enables representing several important concepts related to an automation agent, such its capabilities (what activities it can perform), its goals (which activities it intends to perform), its actions (what activities it is actually performing) or its interdictions (what activities it does not have the right to perform).

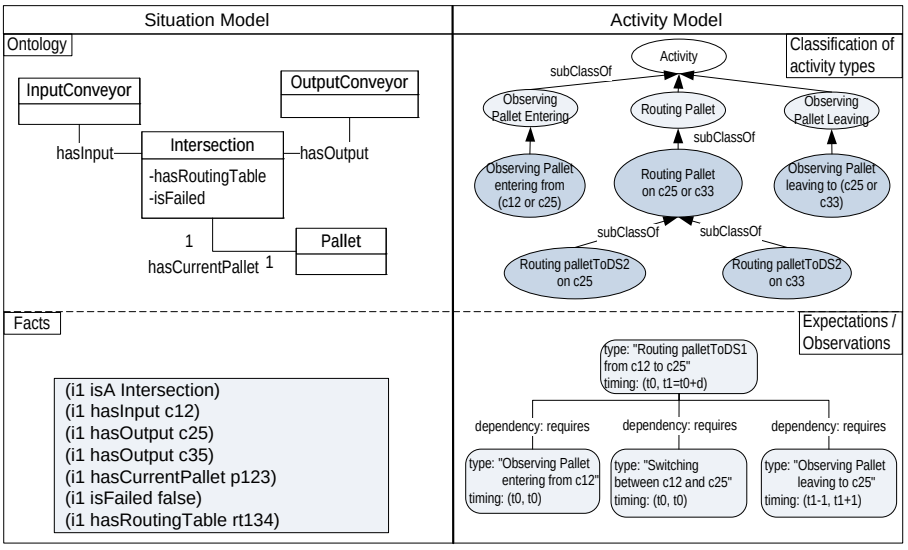

Fig. 7. World Model of an Automation Agent controlling an intersection.

This representation is contained in the world model of the automation agent. Fig. 7 depicts the world model of an automation agent for an intersection. It consists of two parts: the situation model and the activity model. The situation model holds knowledge about the agent's situation. The situation of an agent consists both of its own characteristics and its relations to other entities in the world. The activity model holds knowledge about activities of the agent. The situation model is composed of a domain ontology and a set of facts:

- The *domain ontology* (top) is a model of the type of entities in the domain of the agent (Chandrasekaran, Josephson et al.). It defines relevant classes of entities as well as relations between entities. It also serves as a vocabulary for referencing these classes and relations, thus ensuring the consistency of the world model as well as the interoperability between different agents. For our example, the ontology defines that an intersection can have input conveyors, output conveyors, and one current pallet inside it. Such concepts and relations can be extracted from existing ontologies, such as (Merdan, Koppensteiner et al. 2008).
- The *facts* (bottom) express the current knowledge about the world. Facts are expressed using the vocabulary defined by the ontology. For our example, facts express that i1 is an intersection, which has one input conveyor (c12) and two output conveyors (c25 and c33). It is important to note that facts expressed in the situation model do not intend to represent completely the world of the agent. They rather represent an abstraction of some meaningful aspects of the world, which can be used for realizing high-level control tasks. Although more complete and dynamic information about the world may be available at the low-level control, it is only processed in this layer, usually under real-time constraints.

The activity model is composed of a classification of activity types and a model of expectations and observations:

- The *classification of activity types* (top) models the types of activities in which the agent can be involved. Types are defined formally using description logics formulas and they are organized hierarchically based on the subsumption relationship (noted subClassOf) (Baader, Horrock et al.). Primitive types are defined as direct subclasses of Activity. Derived types are defined by restriction of the primitive types to take into account the actual world of the agent. For instance, the generic type “Routing Pallet” is refined to the more specific type “Routing pallet on c25 or c33” corresponding to the present situation.
- The *expectations and observations* (bottom) is a model of the activities that are expected and observed by the agent. Expectations and observations are defined by the specification of a type (based on the classification of activity types) and timing. Timing is expressed using time intervals (Allen 1983). While observations can express a precise timing, expectations rather express constraints on their timing. Expectations are linked by dependencies, indicating how observations on one expectation can have consequences on other expectations. For in-

stance, Fig. 8 depicts that the expectation that ``Routing palletToDS2 on c25'' should occur starting at time t0 and ending at time t1=t0+d (where d is the time for going through the intersection). This in turns implies that the activity ``Observing palletToDS2 entering from c12'' should occur at t0, the activity ``Switching between c12 and c25'' should also occur at t0, and the activity ``Observing palletToDS2 leaving to c25'' should occur at t1.

The world model using semantic technologies plays a key role in three mechanisms of automation agents: the *initial configuration* (and eventual reconfiguration), the *response to production task requests* and the *interaction with the LLC* for executing and monitoring production tasks.

Fig. 8 illustrates the mechanism for initial configuration of an automation agent. On the right is a declarative description of the configuration, indicating the input and output conveyors of the intersection. Based on this description, the generic activity type "Routing Pallet" is refined for this particular intersection. A more specific class describing the capability of this intersection is defined as "Routing Pallet on c25 or c33". Even more specific classes can be defined based on the routing table contained in the configuration. They indicate that pallets with destination DS2 are routed on c25 and pallets with destination DS3 are routed on c35.

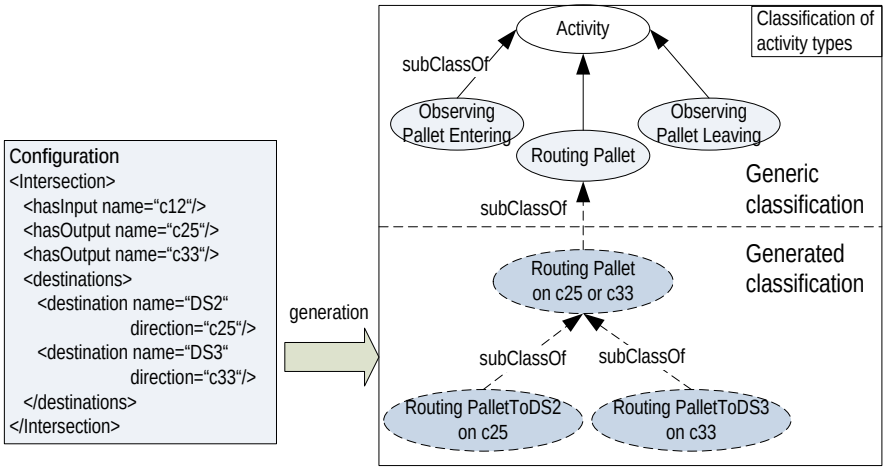

Fig. 8. Mechanism for initial configuration.

Fig. 9 illustrates the mechanism for responding to production requests. When a message containing a request for routing a pallet arrives, it is translated in terms of an activity type to be performed. This request is matched with known types from the activity classification, in order to evaluate if the request can be honored. A request can be honored only if it some activity type can be both more specific than the request and than one capability of the agent. This is the case here for "Routing Pallet124 on c25".

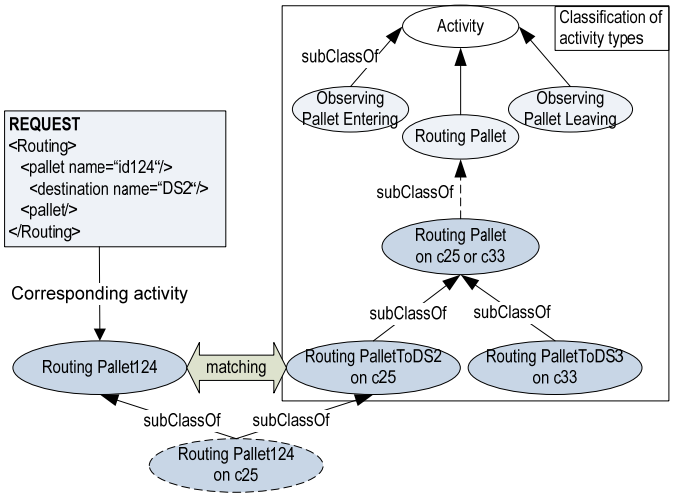

Fig. 9. Mechanism for responding to production requests.

Fig. 10 illustrates the mechanism for interaction with the LLC. This mechanism is described more precisely in (Lepuschitz, Vallée et al. 2009) and enables the integration of heterogeneous types of LLC. The HLC expresses a request using a description of a type of activity, which is independent of a particular LLC. Depending on the technology employed for the LLC, this request is translated into a specific message. To do so, we use a semantic description of the LLC functionality, which relates the LLC-independent concepts used by the LLC and the LLC specific types of message.

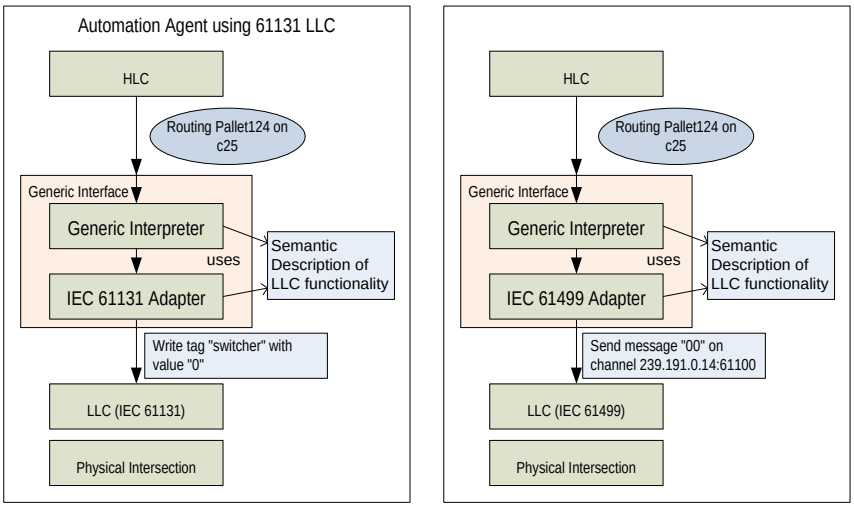

Fig. 10. Mechanism for interaction with the LLC.

7.3 Lessons learned - practical use of semantic agent technologies

Designing and using the automation agent architecture described above enables us to discuss more precisely the practical use of semantic agent technologies in a Layered Manufacturing System.

Automatic reasoning and interoperability in an open environment are often claimed a major advantage of semantic technologies. This approach provides the most flexibility, as the agent can operate in an open environment and cope with unexpected cases. However, this requires a completely consistent, formal definition of the ontologies in use. In particular, various ontologies used within the system have to be properly aligned in order to provide the expected results. This is often unpractical, mainly due to the complexity of the domains involved and the lack of experts in ontological modeling. Moreover, solutions based on runtime reasoning from basic principles (e.g., the axioms of Description Logics underlying OWL) can be computationally expensive for a large ontology.

Using a rule-based engine with domain-specific rules operating on a knowledge base can be a more tractable solution to enable interoperability among agents through translation between the ontologies in use. This facilitates the partitioning of the domain ontologies into several parts, reconciled with appropriate rules. In our example, reasoning about activity type for the intersection can be performed in a restricted domain, without considering a general ontology of all types of activities likely to appear at the level of the whole factory.

Model-based generation of procedural code for managing interactions is another practical approach. In particular, an efficient handling of ontologies can be obtained by taking into account that many entities in the domain of a production system are fixed, at least for some reasonable time scale. Although changes of the production process, of the types of products or even of the types of machines may occur, this only happens at a reduced rate. Therefore, it is practical to transform the knowledge expressed in ontologies into a fixed set of useful data structures and constants, easier to manipulate at runtime. This can be done at deployment time, when the system starts to operate under a fixed configuration (with a fixed set of machines and products). Parts of the code can be re-generated when new conditions appear, thus providing sufficient flexibility. In that sense, ontologies are used as a tool for humans to design and understand the system, rather than as a runtime artifact.

In the execution layer of the Layered Manufacturing System, we employ these three ways of dealing with ontologies. In particular, we found model-based generation of procedural code to be particularly suitable for dealing with ontologies when interacting with the LLC. As only few changes in the LLC can be expected after deployment, it is possible to obtain a specific set of adapters for transforming all possible requests of the HLC into relevant LLC messages. A rule-based engine is useful for interpreting declarative configuration and messages and translating them into the corresponding activity types. Especially, implicit references to vari-

ous entities are resolved by the use of a knowledge base which keeps track of the successively available pieces of information. Automated reasoning tools are best used for matching requests with capabilities, as they provide the most flexibility. Complexity can be tackled by considering only a small ontology of activity types, formally defined only in the context of a particular agent.

8. Conclusion and Further Work

Manufacturing control systems are a mission-critical application domain for semantic agents systems. While multi-agent systems have been explored in the manufacturing systems domain, there is very little work on semantically enabled agent systems.

In this chapter we reported on the engineering of the semantic agent architecture in four layers (management, planning, scheduling, and execution) and the related ontology for each particular layer in the application domain, the manufacturing system. Lessons learned based on real-world use cases in the context of the Layered Manufacturing System are:

- The semantic agent architecture enables related layers to communicate directly avoiding unnecessary "stage by stage" procedures, since the agents from each layer are able to communicate and understand agents from any other layer.
- The presented system assures clear definitions of each layer role in the system as well as associated tasks that have to be done in order to achieve common goals. This further enables the smooth creation of related agent classes and mapping of ultimate system goals to these agents.
- Within each layer, agents have the ability to maintain an accurate internal representation of pertinent information about the environment in which they operate. The world model of an automation agent contains a symbolic representation of the manufacturing domain, more precisely in terms of situations and activities. For this representation, we make use of ontologies.
- The application of ontology based representation enables solving the interoperability problem in all the layers (including control, planning and scheduling) which are managed by agents.

We can summarize that in order to make agent-based system more reliable and more attractive the importance of symbiosis of this technology with other emergent technologies such as semantic web is essential, since it brings advances in a way of treating and dealing with information which is the most important factor in present information age.

Future Work. As one can see major steps for introduction of semantic technologies in manufacturing domain have been already done and the pioneer work for its fusion with multi-agent technology exists. Nevertheless, the related works

that will be concerned with ontology life cycle in terms of maintaining and engineering of ontology models in distributed manufacturing environment is still missing and could be seen as next logical step.

Acknowledgments This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria, as well as by the FIT–IT: Semantic Systems program, an initiative of the Austrian federal ministry of transport, innovation, and technology (bm:vit) under contract FFG 815132. In addition, this work has been partially funded by the Vienna University of Technology, in the Complex Systems Design & Engineering Lab.

References

Aerts, A. T. M., N. B. Szirbik and J. B. M. Goossenaerts (2002). "A flexible, agent-based ICT architecture for virtual enterprises." Computers in Industry **49**(3): 311-327.

Allen, J. F. (1983). "Maintaining Knowledge about Temporal Intervals." Communications of the ACM **26**(11): 832-843.

Azevedo, A. L. and J. P. Sousa (2000). "A component-based approach to support order planning in a distributed manufacturing enterprise." Journal of Materials Processing Technology **107**(1-3): 431--438.

Baader, F., I. Horrock and U. Sattler (2004). Description Logics. Handbook on ontologies, Springer**:** 3--28.

Babiceanu, R. and F. Chen (2006). "Development and Applications of Holonic Manufacturing Systems: A Survey." Journal of Intelligent Manufacturing **17**(1): 131, 111.

Baker, A. D. (1998). "A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull." Journal of Manufacturing Systems **17**(4): 297--320.

Bellifemine, F., G. Caire and D. Greenwood (2008). Developing multi-agent systems with JADE, Springer.

Black, J. T. (1991). The Design of the Factory With a Future, {McGraw-Hill} Companies.

Bose, U. (1999). A Cooperative Problem Solving Framework for Computer-Aided Process Planning. Proceedings of the Thirty-second Annual Hawaii International Conference on System {Sciences-Volume} 8 - Volume 8, {IEEE} Computer Society.

Bussmann, S., N. R. Jennings and M. Wooldridge (2004). Multiagent systems for manufacturing control: a design methodology, Springer {Berlin-Heidelberg.}.

Camarinha-Matos, L. M. (2002). Virtual organizations in manufacturing: trends and challenges. 2th International Conference on Flexible Automation and Intelligent Manufacturing.

Chandrasekaran, B., J. R. Josephson and V. R. Benjamins (1999). "What are ontologies, and why do we need them?" IEEE Intelligent Systems and Their Applications **14**(1): 020--26.

Christensen, J. H. (2003). HMS/FB architecture and its implementation. Agent-based manufacturing. V. S. M. Deen**:** 53--87.

Christo, C. and C. Cardeira (2007). Trends in Intelligent Manufacturing Systems. Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on.

Colombo, A. W., R. Schoop and R. Neubert (2005). "An agent-based intelligent control platform for industrial holonic manufacturing systems." Industrial Electronics, IEEE Transactions on **53**(1): 322--337.

Finin, T., Y. Labrou and J. Mayfield (1997). KQML as an agent communication language. Software agents, MIT Press**:** 291--316.

Foundation (2003). Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification http://www.fipa.org/specs/fipa00037/ and FIPA ACL Message Structure Specification http://www.fipa.org/specs/fipa00061.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications." Knowl. Acquis. **5**(2): 199--220.

Hansen, H. R. and G. Neumann (1982). Wirtschaftsinformatik, Fischer.

Jennings, N. R. and S. Bussmann (2003). "Agent-based control systems: Why are they suited to engineering complex systems?" Control Systems Magazine, IEEE **23**(3): 61--73.

Jones, A. T. and C. R. McLean (1986). "A proposed hierarchical control model for automated manufacturing systems." Journal of Manufacturing Systems **5**(1): 15--25.

Kim, K.-Y., D. G. Manley and H. Yang (2006). "Ontology-based assembly design and information sharing for collaborative product development." Computer-Aided Design **38**(12): 1233--1250.

Krothapalli, N. K. C. and A. V. Deshmukh (1999). "Design of Negotiation Protocols for Multi-Agent Manufacturing Systems." Int. J. of Production Research **37**: 1601---1624.

Kulvatunyou, B., H. Cho and Y. J. Son (2005). "A semantic web service framework to support intelligent distributed manufacturing." Int. J. {Know.-Based} Intell. Eng. Syst. **9**(2): 107--127.

Kumar, M. and S. Rajotia (2006). "Integration of process planning and scheduling in a job shop environment." The International Journal of Advanced Manufacturing Technology **28**(1): 109--116.

Lastra, J. L. M. and M. Delamer (2006). "Semantic web services in factory automation: fundamental insights and research roadmap." Industrial Informatics, IEEE Trans. on **2**(1): 1--11.

Lepuschitz, W., M. Vallée, M. Merdan, P. Vrba and J. Resch (2009). Integration of a Heterogeneous Low Level Control in a Multi-Agent System for the Manufacturing Domain. 14th IEEE International Conference on Emerging Technologies and Factory Automation.

Mellouli, S. (2005). FATMAS: A Methodology to Design Fault-tolerant Multi-agent Systems, Université Laval. **PhD**.

Merdan, M. (2009). Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain, Vienna University of Technology.

Merdan, M., G. Koppensteiner, I. Hegny and B. Favre-Bulle (2008). Application of an Ontology in a Transport Domain. IEEE International Conference on Industrial Technology (IEEE ICIT2008), Sichuan University, Chengdu, China.

Merdan, M., T. Moser, D. Wahyudin and S. Biffl (2008). Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures. Industrial Engineering and Engineering Management, 2008.IEEE International Conference on.

Obitko, M. and V. Mařík (2002). Ontologies for Multi-Agent Systems in Manufacturing Domain. Proceedings of the 13th International Workshop on Database and Expert Systems Applications, IEEE Computer Society.

Obitko, M., P. Vrba, V. Mařík and M. Radakovic (2008). Semantics in Industrial Distributed Systems. The 17th {IFAC} World Congress, Seoul, Korea.

Ouelhadj, D. and S. Petrovic (2009). "A survey of dynamic scheduling in manufacturing systems." Journal of Scheduling **12**(4): 417--431.

Parunak, H. V. D. (1996). Applications of distributed artificial intelligence in industry. Foundations of distributed artificial intelligence, John Wiley & Sons, Inc.**:** 139--164.

Patil, L., D. Dutta and R. Sriram (2005). "Ontology-based exchange of product data semantics." Automation Science and Engineering, IEEE Transactions on **2**(3): 213--225.

Pěchouček, M. and V. Mařík (2008). "Industrial deployment of multi-agent technologies: review and selected case studies." Autonomous Agents and Multi-Agent Systems **17**(3): 397--431.

Pěchouček, M., J. Vokrinek and P. Becvar (2005). "ExPlanTech: multiagent support for manufacturing decision making." Intelligent Systems, IEEE **20**(1): 67--74.

Qiu, X. (2005). "Agent interaction in a Semantic Web environment: A state-of-the-art survey and prospects in knowledge mobilization." Information Systems Research in Scandinavia **IRISCA28, Kristiansand, Norway**.

Rabemanantsoa, M. (1993). Knowledge-based system for assembly process-planning. Software Engineering Standards Symposium, 1993. Proceedings., 1993.

Rajpathak, D. G., E. Motta, Z. Zdrahal and R. Roy (2006). "A generic library of problem solving methods for scheduling applications." Knowledge and Data Engineering, {IEEE} Transactions on **18**(6): 815--828.

Roche, C., S. Fitouri, R. Glardon and M. Pouly (1998). The Potential of Multi-Agent Systems in Virtual Manufacturing Enterprises. Proceedings of the 9th International Workshop on Database and Expert Systems Applications, IEEE Computer Society.

Seilonen, I., T. Pirttioja, P. Appelqvist, A. Halme and K. Koskinen (2003). Distributed planning agents for intelligent process automation. Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 {IEEE} International Symposium on.

Shen, W. and D. H. Norrie (1999). "Agent-based Systems for Intelligent Manufacturing : A State of-the-Art Survey." Knowledge and Information Systems, an International Journal **1(2)**: 129--156.

Shen, W., L. Wang and Q. Hao (2006). "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey." Systems, Man, and Cybernetics, Part C: Applications and Reviews, {IEEE} Transactions on **36**(4): 563--577.

Silva, N. and J. Rocha (2003). Ontology mapping for interoperability in semantic web. IADIS International Conference WWW/Internet.

Vallée, M., H. Kaindl, M. Merdan, W. Lepuschitz, E. Arnautovic and P. Vrba (2009). An Automation Agent Architecture with A Reflective World Model in Manufacturing Systems. IEEE International Conference on Systems, Man, and Cybernetics (SMC'09), USA.

Whitney, D. E. (1996). "The potential for assembly modeling in product development and manufacturing." MIT Press.

Zhang, W. and S. Xie (2007). "Agent technology for collaborative process planning: a review." The International Journal of Advanced Manufacturing Technology **32**(3): 315--325.