6 Applications of Agent Systems in Intelligent Manufacturing

Aleksey Bratukhin[1], Arndt Lüder[2] and Albert Treytl[3]

[1] Austrian Academy of Sciences, Research Unit for Integrated Sensor Systems, Viktor Kaplan-Strasse 2, 2700 Wiener Neustadt, Austria, e-mail: bratukhin@fiss-oeaw.at

[2] Otto-von-Guericke University of Magdeburg, Faculty for Mechanical Engineering, Institute for Ergonomics, Manufacturing Systems, and Automation, Universitätsplatz 2, 39106 Magdeburg, Germany, e-mail: arndt.lueder@ovgu.de

[3] Austrian Academy of Sciences, Research Unit for Integrated Sensor Systems, Viktor Kaplan-Strasse 2, 2700 Wiener Neustadt, Austria, e-mail: treytl@ict.tuwien.ac.at

Abstract Facing the fast growing needs regarding flexibility and adaptability of manufacturing systems, the decentralization of manufacturing execution control has attained high importance. Hence, in recent years different research and development activities have tackled the problem of decentralizing manufacturing execution control and implementing these decentralized systems within control architectures. One major result of these activities is a set of design patterns describing possibilities for decentralization including the description of major entities and interaction schemas.

These activities have also shown that agent systems are an appropriate means for the implementation of decentralized manufacturing execution control systems. They cope with decentralization by nature and (especially in the case of the Foundation for Intelligent Physical Agents (FIPA) - compliant agents) provide appropriate means for the implementation of the internal behavior of entities and entity interaction.

In this chapter some major design patterns for decentralized manufacturing execution control systems are described and mapped to three major approaches with Product-Resource-Order-Staff Architecture (PROSA) and MetaMorph (both based on Holonic Manufacturing Systems) and PABADIS (Plant Automation Based on Distributed Systems), and which finally are compared. Exploiting this comparison the PABADIS'PROMISE (PABADIS based Product Oriented Manufacturing Systems for Reconfigurable Enterprises) architecture is described as an architecture trying to incorporate the advantages of the different approaches and avoid its disadvantages.

6.1 Introduction

Conventional centralized control systems face challenges in adapting to the requirements of modern production systems [1]:

- Unpredictable order flow: customer and production orders are issued dynamically, often when production has already started.

- Dynamic shop floor: configuration of the resources on the field level changes during production, which makes it difficult to plan the execution of orders.
- Complexity of the shop floor and orders: modern production systems are characterized by increasing complexity of both the complexity and flexibility required by production orders as well as the variation in the shop floor configuration.

Due to their hierarchical nature, centralized systems are highly static and difficult to adapt to such changes as late modifications of customer orders or new/broken field level devices [2]. Additionally, the decision-making process is concentrated on the top layer of the automation pyramid [usually Enterprise Resource Planning (ERP) and related systems], and therefore the production planning is hardly able to react to changes or exceptions on the shop floor.

In environments where high flexibility is required such as highly customized small lot production, the absolute optimization of production can be partly neglected in favor of flexibility. The point is that in a permanently changing environment, the lack of flexibility would make it impossible to realize any optimization mechanisms.

Distributed control systems aim to solve such problems by providing two general mechanisms:

- Moving the decision-making process from ERP down to the Manufacturing Execution Systems (MES) layer, which has a shorter planning horizon and, hence, is able to react to the changes faster.
- Distributing control over a set of independently acting entities that take some responsibility for fulfilling the order. This provides concurrent processing and, therefore, minimizes the drawbacks of the hierarchical structures.

In order to provide higher flexibility of production control and planning, the paradigm of Distributed Control Systems (DCS) was defined and further developed into several concepts and architectures. Most notably is the Holonic Manufacturing Systems (HMS) [3] concept and its architectures such as Product-Resource-Order-Staff Architecture (PROSA) [4] and MetaMorph I and II [5].

The general idea of DCS is that the decision-making process as well as system functionalities are distributed among independently acting entities called “holon” in HMS. More commonly the concept of an “agent” can be used [6].

Multiagent Systems (MAS) became de facto a standard for DCS applications and has found slowly its way into the domain of industrial automation. Nevertheless, there are a few challenges the agent systems face in order to adapt to centralized and homogenous ERP systems. It seems unrealistic that in the near future the ERPs will also be able to adapt to the distributed paradigm. Therefore, the main burden for establishing a DCS lies with the MES layer.

6.2 Existing MES Solutions

The whole spectrum of solutions in the distributed automation system developed and researched ranges from partly centralized to totally distributed approaches. Some of them provide the complete architectures, integrating not only MES but also enterprise- and control-level systems. Others provide only supportive mechanisms that allow conventional systems to fit the requirements of the modern industry.

Most of the distributed MES architectures use multiagent technologies [7] and the term "agent" in particular, despite some principal differences in the origins of the MAS [6].

The most known architecture, PROSA, is a holonic reference architecture based on three types of basic holons: Product, Order and Resource [8, 9]. Additionally, typically to all distributed architectures, the need to observe the shop floor and provide an interface to enterprise-level systems evolved into the creation of a fourth special type of component. In PROSA, the Staff holon assist and supervise the basic holons. Another architecture based on the HMS concept – MetaMorph and the follow-up MetaMorph II – uses a mediatorcentric federation architecture for intelligent manufacturing [10]. It uses a term mediator that provides communication mechanisms to different systems and components, and takes over some of the functionalities of the MES. Yet it must be noted that the original Metamorph is more an integration tool for other systems rather than a complete solution.

Eventually, MetaMorph evolved into other multiagent architectures, that attempt to integrate the functionalities of a manufacturing enterprise within a distributed environment. For instance, Agent-Based Manufacturing Enterprise Infrastructure (ABMEI) is a hybrid agent-based architecture combining the mediator and the autonomous-agent approaches [11].

Similar agent-based architectures with slightly different implementation focuses are Autonomous Agents at Rock Island Arsenal (AARIA) [12] and Manufacturing Control Systems Capable of Managing Production Change and Disturbances (MASCADA) [13]. Such architectures as HOLOS/MASSIVE [32] and its followup Decentralized Decision-Making and Scheduling (DEDEMAS) [14] concentrate on decentralized decision making and scheduling.

Architectures like Advanced Fractal Companies Use Information Supply Chain (ADRENALIN) [15, 16] consider agent-based resource brokering functionality enabling a decentralized manufacturing order navigation based on local optimization strategies.

Opposite of that, Plant Automation Based on Distributed Systems (PABADIS) and its followup PABADIS'PROMISE (PABADIS based Product Oriented Manufacturing Systems for Reconfigurable Enterprises) provide agent-based architectures that cover the whole automation pyramid but with a primary goal of distributed MES [17].

6.3 A Generic Design Pattern for Manufacturing Execution Control

The organization of an MES and its interconnection to the ERP and field control layers has some general patterns that are followed by most of the solutions implementing MES in a distributed fashion. As a starting point these basic design pattern exploited in MES system design will be presented and exploited later on for the comparison of HMS and PABADIS approaches.

Design patterns are a widely used design aid in information sciences. They date from the initial work of Alexander in 1979. Alexander, who was an architect, came to the conclusion that building design always follows the same basic rules with culture and geography dependent implementations [18].

With the emergence of object-oriented programming the idea of design patterns as application-independent basic design principles has been adopted to software design in information sciences. The main research activity initiating this trend was the work of Gamma et al. [19].

Within the last few years the use of design patterns has been integrated in several disciplines including the design of control applications. Within control design the application of design patterns as a description of basic design principles has been extended to the design of complete control applications including control software and hardware, as well as the plant itself. Within this field valuable progress has been made. Overviews of the reached results are given in [20, 21, 22].

Within the field of distributed control systems the application of agents has gained wide acceptance [6]. That is why design patterns have act in this field as well been recognized and described [23, 24]. This chapter intends to sketch the MES-related design pattern briefly and describe how it occurs and is applied within the different agent-based approaches.

Within MES systems there are two main general entities with three categories of knowledge that are relevant for the execution process of manufacturing orders. These entities are order and resource usually accompanied by the categories product knowledge, order knowledge, and resource knowledge.

Equally, Resource Agents (RAs) are associated with production units of different types. Each agent represents one resource with certain production capabilities it provides to the agent community containing all resource-related knowledge. It is responsible for all resource-related control decisions including resource scheduling (based on cooperative scheduling algorithms) and resource control on the field control layer but also supportive actions like maintenance or life cycle activities.

Facing these entities and knowledge distributions, the overall system behaves in the following way. All agents in the system form an agent community containing all Order Agents (OAs) and RAs. This agent community also contains a special registration function for RAs. At startup of an RA it registers its manufacturing capabilities, which are represented by usable production functions, within the

agent community. These production functions can then be searched, initialized, parameterized, started, and stopped by OA during the OA–RA interaction.

On the other hand, each OA covers a product order consisting of a set of production steps that are necessary to produce the ordered product. These production steps can be fulfilled by a set of production processes. The OA then has to negotiate for appropriate manufacturing capabilities offered by the RA to fulfill its order. Important and helpful with respect to flexibility, OAs and RAs both handle its own schedules and make its own scheduling decisions independently yet in a collaborative way. The resulting structure is given in the following class diagram and described in what follows (Fig. 6.1).

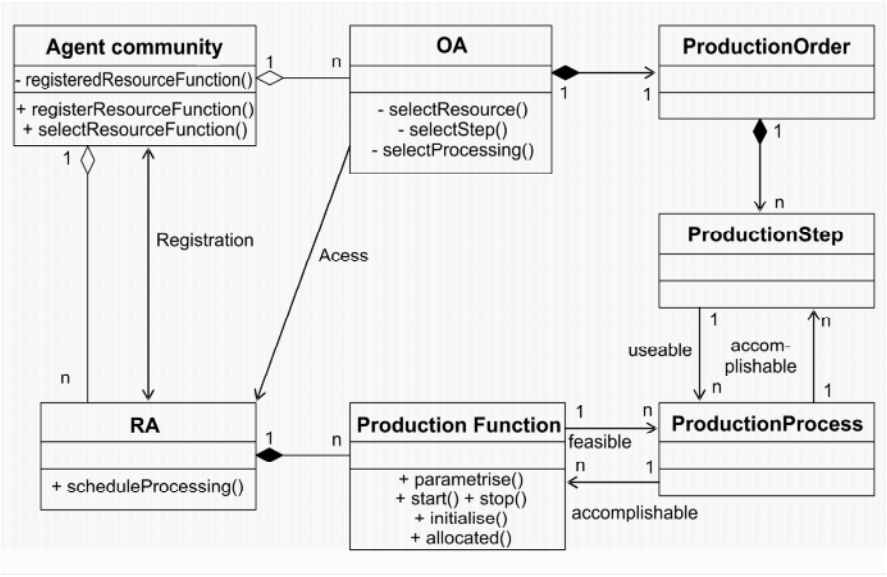

Fig. 6.1 Order-resource pattern

During the start of an RA, the RA makes itself known within the agent community. In this way, the agent community is informed about all production functions belonging to and being controllable by the RA. Each OA processes its production order step by step following the interaction scheme among OA and RA given in the next figure (Fig. 6.2).

1. The OA selects the production step to be executed next.
2. The OA determines within the knowledge of the agent community the set of RAs containing a production function that can be used for the next production step of the OA.
3. The OA negotiates with the determined RA set the next production function to be used and the usage schedule. Therefore, the OA asks all RAs for possible schedules, decides about the next production function to be used, and allocates

this function. The RA related to the selected production function reserves resources for this within its schedule.

4. At the agreed upon moment the OA accesses the RA to parameterize and start the expected production function.
5. If the processing is finished, then the processing function provides the processing result to the OA by using the RA. The OA can adjust its list of necessary production steps and proceed further with step 1.

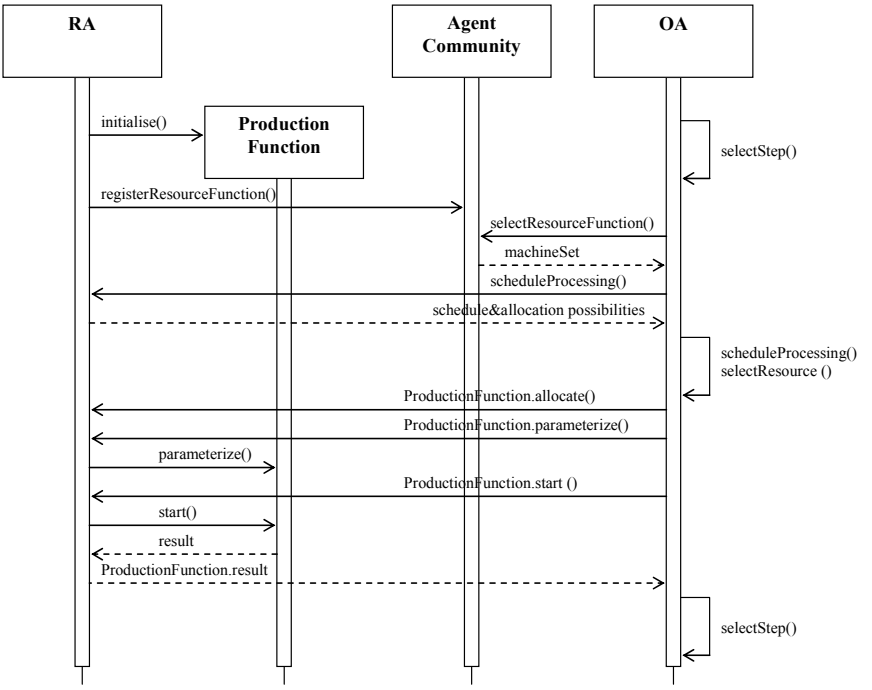

Fig. 6.2 Order and resource agent communication pattern

6.4 Distributed Approaches Analysis

Because different MES solutions usually focus on a specific area of implementation, e.g., scheduling, customer support, resource utilization, system integration, they often cannot be compared. For this chapter the focus is on systems that cover the complete manufacturing process. Hence HMS-based concepts, such as PROSA and METAMORPH, and MAS-based concepts, such as PABADIS and PABADIS'PROMISE, that cover wider fields of control systems and focus on the vertical integration of all layers of the automation pyramid from the ERP down to field control level are compared in the following chapter.

In order to have a unique name for the analysis the terms used by the HMS have been chosen because they are widely known. Furthermore, if HMS, which does not provide an architecture, lacks the required terms, PROSA terms (Order, Resource, and Stuff holons as well as MetaMorph mediators) are used to compare with PABADIS architecture with its respective Product, Residential and Plant Management Agents (PA, RA, and PMA).

6.4.1 Resource Holon and Residential Agent

Generally speaking, there are two main elements of distributed plant automation systems: resources and customers represented by orders as given in the above design pattern.

Resources in HMSs are represented by Resource Holons, which are responsible for machine-level representation. The Resource Holon consists of a physical part, namely, a production resource in HMS, and of an information processing part that controls the resource. This second part holds the methods to allocate the production resources, and the knowledge and procedures to organize, use, and control the physical production resources to drive production.

In terms of PABADIS, Resource Holon is a Co-Operative Manufacturing Unit (CMU) which is a building unit of the shop floor. The information processing part of the Resource Holon perfectly fits the concept of the Resource Agent, or Residential Agent as it is called in PABADIS, which represented the CMU and partly controls the function.

The main differences are that in PABADIS the Residential Agent is more or less an interface between the agent community and the CMU (machine, function unit), and the Residential Agent has a generic interface enabling the integration of an arbitrary machine hardware. In HMS the Resource Holon is more than this interface. It is rather a CMU-RA analog, where the whole functionality of the CMU (function, control level, agent communication level) is implemented. PABADIS distinguishes the standard (logical) part of the control level, which can be used for each CMU, from the "machine"-specific part, which has to be implemented by the customer of the system. This makes the system more flexible and encapsulates the MAS from the control level. That makes the process of adding new functionality easier, because the customer (industrial company) has to add a specific plant-dependent resource to the system and does not care about the interoperability of a new component with the existing control system. The customer just follows the interface, and incorporation into the system is provided by the generic Residential Agent, which is able to communicate within the system.

Additionally to this, Resource Holons do not just provide resources, but also manage the whole production facility. This means that they are able to communicate between each other in order to find the best use of machines. PABADIS does not allow RAs to communicate with each other. This is done in order to make the

system product oriented and not machine oriented, which may be the biggest difference from the HMS concept. The product and its performance are the main goal of the system in PABADIS. HMS is more focused on machine utilization, where an Order Holon is simplified to a set of product parameters that has to be produced.

6.4.2 Order Holon and Product Agent

Compared to Resource Holons and Residential Agents the difference between definitions of the Product Agent in PABADIS and the Order Holon in HMS is stronger:

- A Product Agent is an instance that manages the whole production of a single workpiece. It bases its decisions, actions, and knowledge on the so-called Work Order (WO), which gives a full specification of the production activities regarding the tasks that have to be fulfilled in order to complete the product. At the same time, the WO does not assign exact machines; instead it describes the function that has to be used to do that. This principle gives the Product Agent the freedom to decide what machine to use and introduces the possibility of changing the machine during execution.
- An Order Holon is much simpler. It is more or less the customer request, where the requirements to the product are defined. It has an advantage in mass production, because an Order Holon is not attached to a single workpiece. An Order Holon does not do scheduling or resource allocation, because it simply does not have knowledge about the physical layout of the plant and the product specification.

6.4.3 Product Holon and Product Agent

The necessity of the Product Holon rides on the technology dependence of the product. That means that the Order Holon, representing the workpiece, does not have knowledge about the technological properties of the plant. This means that it has no information on how to produce the product. Order Holons contact the Product Holon in order to get information: how to produce this product. Based on the Product Holon's response, an Order Holon performs the product execution.

The standardization of the functions provided by the CMU's concept gives a generic description of the functionality that does not depend on the specific technology.

In PABADIS no analog to the Product Holon is required. Product Agents incorporate both Product Holons and Order Holons. The advantage of the Product

Holon is that the new product specification can be added to the system at any time, but on the other hand Order Holons cannot adapt their behavior to the unplanned changes in the system because they have no knowledge about the product performance. Product Agents have a complete specification of the product execution and knowledge how to analyze these data.

6.4.4 Stuff Holon and Plant Management Agent

The Stuff Holon fits the definition of the PMA in PABADIS. The main purpose of the Stuff Holon is to give the other components an overview of the system. Therefore, it is often used for supervision or support functionality implementation.

One of the typical main MES functions represented by this concept is scheduling. For example, PROSA uses a centralized unit called the Scheduler Stuff Holon that performs scheduling for the whole plant.

In PABADIS the scheduling is simplified and distributed in Product Agents. In contrast, this approach makes the system more flexible, yet it must be noted that it does not provide the optimal solution. In PABADIS, scheduling is done from the point of view of a single product, which makes it impossible to guarantee the optimum. PA negotiations and the benevolent behavior of Product Agents can increase scheduling performance, but this still does not guarantee the global optimum.

PABADIS minimizes the number of new instances by using existing basic architecture. That makes the components, basically Product Agents, more complex but keeps the architecture simple. In contrast, PROSA introduces the new instances and keeps the basic holons simple.

What is common to both architectures is that they try to avoid centralized units, which PMA and Stuff Holon are by definition.

6.4.5 Aggregation

Aggregation is a key point in HMS. Aggregation is structuring agents in a hierarchy. This is the appropriate solution to tackling complexity of independent Holons. This solution avoids complex communication and heavy network load in the system, but reverts back to the centralized systems, causing a loss of system flexibility and scalability. Aggregation introduces a new layer(s) in the control pyramid and makes the logic of holons more complex. In practical implementation even communication is not simpler because holons have to communicate on different layers, which brings complexity to the process in general yet simplifies one-to-one communications. A tradeoff between simpler single communication mechanisms and additional overhead due to aggregation has to be found.

PABADIS does not use aggregation of agents and tries to avoid the complexity of communication via giving the single instances more independence, which decreases the necessity of communication. This leads to a reduction in optimization yet efficiently increases system flexibility and scalability, which is the main goal of PABADIS.

6.4.6 Mediator

It is difficult to compare PROSA or PABADIS architectures with the MetaMorph approach. Mediators in MetaMorph are powerful tools to connect different systems together, but can hardly be considered as a manufacturing system architecture. To a certain extent, mediators behave as Staff Holons in HMSs or as PMAs in PABADIS. They provide centralized functionality to the system by coordinating the actions of other agents.

Mediators can successfully be used in cooperation with other concepts, such as PABADIS or PROSA, for the interconnection of different elements and for providing solutions for problems that require a temporal overview of the system. But individually, mediators cannot be a manufacturing-system-oriented architecture because they do not have dedicated MES functions (such as scheduling) or do not represent different actors in the plant (such as products and resources).

An attempt to create an architecture (MetaMorph II) is more or less a centralized approach with dedicated mediators for each function of the plant automation system and a strong hierarchy among them. Perhaps the most important achievement of the mediator concept is its ability to dynamically group entities into virtual groups according to the needs of the system. This ability can greatly improve the flexibility of a system, not only for MES applications.

6.4.7 Flexibility Versus Optimization

In conclusion of the previous sections, it can be said that PABADIS architecture is particularly usable in a highly turbulent environment where flexibility is crucial to system performance; PROSA is more suitable for systems were optimization is more important and flexibility is secondary.

Due to the fully distributed MES layer, PABADIS is more scalable than HMS since the latter has the remains of the centralized structure where each function is dedicated to a single entity. The same argument can be applied to the MetaMorph approach where, instead of distribution of intelligence, the intelligence is focused on the function-representation mediators.

Both PABADIS and PROSA provide comprehensive solutions to the problems in existing MES systems face, trying to adapt to modern trends in manufacturing,

in particular mass customization. These trends require more flexible production management and planning to react to the turbulence in customer demand and shop floor configuration. They shift decision making from the level of business systems (such as ERP) to the middle layer of the MES. Therefore, they shorten the reaction time to changes in the plant and provide vertical integration of the automation pyramid, making it possible to bring the strategy made at the top level to the field level of control devices.

Both architectures provide distribution of the MES system but approach this problem from different sides. PROSA tries to imitate conventional centralized systems by providing entities for different functions. In contrast, PABADIS dissolves the functions of MES in the community of agents. The actual difference between the approaches lies in the balance between optimization and flexibility. It is clear that PROSA provides a higher level of optimization compared to PABADIS. It is also clear that PABADIS is more adaptive, scalable, and flexible than PROSA. PABADIS'PROMISE architecture developed later, attempts to find a balance between the above-mentioned concepts.

6.5 PABADIS'PROMISE Hybrid Approach

PABADIS'PROMISE is the followup of the PABADIS architecture that nevertheless combined the key features of all three afore mentioned concepts:

- General architecture PROSA and PABADIS - Order and Resource concept;
- Distributed functionality approach of PABADIS;
- Aggregation of resources of PROSA;
- Clustering and mediation of resources for scheduling of MetaMorph.

Being a followup of the PABADIS architecture, PABADIS'PROMISE comprises the general notions of Order and Resource agents used by PABADIS but improves its functionality by introducing concepts developed by the holonic architectures of PROSA and MetaMorph.

6.5.1 Resource Handling

Hierarchy of resources is a key distinction point of distributed concepts. On the one hand, such architectures as PROSA provide strict control of resources over each other. It improves resource utilization but makes it complex to implement an actual system and has problems in adapting to a changing environment. On the other hand, systems such as PABADIS make resources totally independent of each other (for instance, PABADIS Residential Agents do not even communicate with each other).

The PABADIS'PROMISE approach to Resource Agents is similar to the MetaMorph approach in terms of scheduling by the usage of clustering of related resources and makes it possible for Resource Agents to allocate other resources for certain tasks. This approach improves the flexibility of the shop floor compared to PROSA, because there is no direct static connection between two resources, and optimizes system performance by including Resource Agents in the decision-making process during scheduling.

6.5.2 Order Management

Another important aspect is order management. As with the other topics, there is always a balance between optimization and flexibility and the hierarchy of orders and management entities related to them. Where PROSA orders are organized in a rigid hierarchy and PABADIS has its minimization of interdependencies between orders, PABADIS'PROMISE provides what can be called "implicit hierarchy." There is no strict decision making, the organization is implemented via the structure of the Production Order that possesses interdependencies between different production steps (called Process Segments) via so-called node operators. The mechanism of order decomposition developed in PABADIS'PROMISE allows for the on sign autonomously acting Order Agents to each Process Segment. It is similar to the PABADIS approach with the difference that higher-level Order Agents can influence the agents responsible for subtask execution. For instance, the Order Agent that is responsible for the assembly of a car needs an engine to be produced before performing its tasks. Therefore, it assigns the deadlines for the engine agent, so the engine is delivered on time to assembly the complete car.

Another issue is the behavior of agents involved in production. PROSA with its hierarchy does not need to pay attention to this topic, but in more distributed systems such as PABADIS it has greater importance because it can increase performance efficiency. Product Agents in PABADIS are selfish in their "nature." As a result the system has low optimization of order execution compared to PROSA but much higher flexibility. PABADIS'PROMISE again tries to combine both benefits by introducing benevolent behavior of agents that is, agents have to obey a certain set of rules that benefit not only their local goals (finishing assigned Process Segments on time), but also to sacrifice their goals to help agents struggling to meet their deadlines. This approach considerably improves optimization without dramatically reducing flexibility, which guarantees higher overall system performance and stability.

6.5.3 Supervisory and Supporting Functionalities

As was mentioned above, both PROSA and PABADIS define a special entity (Stuff Holon and PMA, respectively) to cover functionality not reflected by order- and resource-related agents or holons.

PABADIS'PROMISE also provides supervision management entities, namely, Order and Resource Agent Supervisors (OASs and RASs) to implement functions that are centralized by nature such as supervision or reporting. But in contrast to the earlier architectures, PABADIS'PROMISE does not define a specific entity that deals with such functions as tooling or maintenance. Instead of creating an extra agent, the concept incorporates support functions via ontology and organization of agent communities that allow for implementing tooling or maintenance using the general mechanism of an order-resource relationship.

6.5.4 PABADIS'PROMISE Scheduling

The advantages of PABADIS'PROMISE as a hybrid approach compared to PROSA and PABADIS architectures (MetaMorph is more a concept than an architecture) can be summarized in the way it approaches scheduling. Instead of the single centralized concept of PROSA and the selfish order-oriented approach of PABADIS, PABADIS'PROMISE comprises resource-oriented scheduling on the shop floor with dynamic clustering for solution finding, and order-oriented benevolent rescheduling on the MES layer.

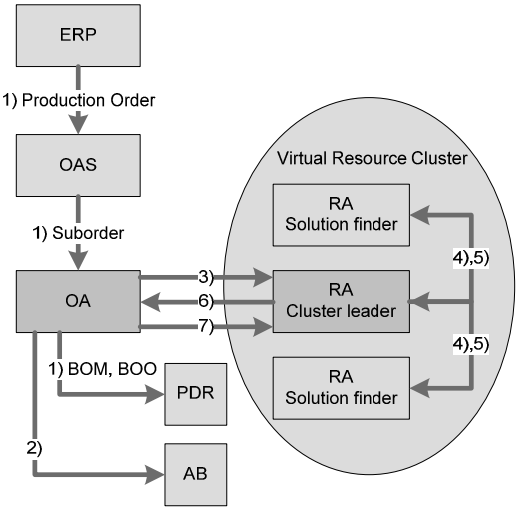

Fig. 6.3 Resource-oriented scheduling

Scheduling in PABADIS'PROMISE can be divided into two mechanisms: resource-oriented initial scheduling and order-oriented rescheduling [25].

The first mechanism (Fig. 6.3) consists of the following steps:

1. OA retrieves its first unscheduled Process Segment (PS);
2. OA asks the Ability Broker (AB) for an RA with the capability required by the PS;
3. OA requests an RA for allocation;
4. RA forms a cluster for scheduling;
5. RAs perform scheduling and find a solution;
6. RA sends a proposal to the OA;
7. OA accepts or rejects the proposal.

In steps 1 and 2, the initial processes of the OA parsing an order and discovering resources are executed. Steps 4 and 7 are negotiations between an OA and an RA. And the actual scheduling algorithm is applied in steps 4-6. Although the RAs implement scheduling algorithms for finding a solution to reserving resources, the OA is responsible for decisionmaking. That is in contrast to the MetaMorph original concept, where the cluster leader's decision is final. The reason to shift the decisionmaking to the OA is to provide optimization of the order execution and not just of resource utilization. In what followings the seven steps of the scheduling procedure are described in more detail.

6.5.4.1 Order Agent Receives the Production Order and Parses It

Depending on the "depth" scheduling, meaning the number of PSs scheduled in advance, the OA fills in the time schedule to create the framework for the execution of the Production Order (PO). The reason behind introducing "depth" is to avoid the snowball effect of the network overload caused by the rescheduling of the previously allocated task due to the requests from the higher-priority OAs.

6.5.4.2 Order Agent Asks the Ability Broker for Resources

In the next step, the OA sends a request to the AB that maintains the actual list of abilities and resources required by the OA to carry out the order. In PABADIS'PROMISE, ability is a certain function that a resource provides. It can be a physical operation, computation function, or an action of a human. It reflects the definition of a resource in the concept that varies from a robot to the entire production line or a plant and can even be a human being.

6.5.4.3 Order Agent Asks a Resource Agent for Allocation

After receiving the RA address that can perform the requested ability, the OA sends a scheduling request that contains the Process Segment and the time slot the OA desires for the ability execution.

6.5.4.4 Resource Agent Forms a Cluster for Scheduling

Upon receiving the request, it is the task of the RA to communicate to other identical resources and form a resource cluster. This communication can follow the sequence of actions as proposed in MetaMorph that is, the leader can first broadcast a message to all similar RAs, then in reply the RAs can join in the cluster, and this cluster can then participate in the process of scheduling under the leadership of this leader. However, the difference from the classical MetaMorph concept is that in PABADIS'PROMISE the leader does not have the pointers through which other identical resources can be accessed, therefore it needs to find out those pointers first. In order to find the similar abilities and respective resources, the cluster leader RA contacts the AB and receives the pointers. Then the leader broadcasts the request for clustering to the RAs with the same ability. In response, all the RAs to which a request was send evaluate the request message and reply to the leader about their decision; the request is then either accepted or dropped. The decision is based on the availability of a resource, meaning that if a resource is already allocated for a certain period, then it will not participate in the cluster. After that leader receives the responses, it forms a virtual cluster from the resources that responded positively. The important feature of the virtual clusters is their dynamism, meaning they are created on demand and are not permanent. A cluster is created for an order and then can be broken after completion of the scheduling activity of the order. Moreover, it is also possible that an agent that is participating as a leader in one cluster will also act as a participant in another cluster.

6.5.4.5 Resource Agents Perform Scheduling and Find a Solution

When cluster is formed, the mechanism of finding a quasioptimal solution starts. Within this mechanism a task leader asks the agents in the cluster for their proposals regarding the requested task execution and evaluates the results. There is also a local internal evaluation process at the RAs using the so-called evaluation function that considers the availability and costs of resources, as well as tooling and waiting time due to the gaps in the RA schedule [25]. Generally speaking, the evaluation function provides an optimal solution at the particular time for a single ability with respect to resource utilization optimization.

6.5.4.6 Resource Agent Sends a Proposal to the Order Agent

After the cluster leader RA receives all the bids from the cluster members, it verifies the proposals, and based on additional parameters given by the OA or general for the shop floor, chooses the best solution.

The general parameters for the shop floor are those that concern the optimization of the whole field layer and not focused on the local optimization of a single resource. Finally, the RA sends a proposal to the OA that had send the scheduling request.

6.5.4.7 Order Agent Accepts or Rejects the Proposal

Then it is up to the OA to evaluate the proposal and to accept or reject it. If the OA accepts the proposal than it allocates the resource on the proposed conditions. If the proposal is not suitable, then the OA can proceed in two possible ways:

- Starting the process from the beginning, meaning asking the AB for the given ability, asking the RA to form a new cluster, and so on. Due to the dynamism of the shop floor and of the order flow, the result of the new evaluation can be different from the old one.
- Asking the cluster for other solutions that fit the OA in a better way, but are less optimal for the resources. This mechanism depends on the configuration of the system that has to evaluate the importance of the shop floor optimization compared to the order flow optimization.

Eventually, if a solution cannot be found within the resource-oriented scheduling then the OA has to negotiate with the OAs or OASs to find a solution. This mechanism is based on the benevolence of the OA behavior, assuming that the OAs respect the needs of the others. An exact criterion for evaluation depends on a particular application, but for the actual implementation of the demonstrator the criterion for rescheduling is based on the due deadlines and due dates of the orders.

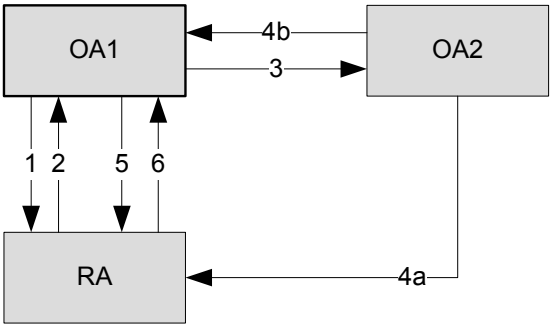

Fig. 6.4 Order-oriented rescheduling

Each order has a deadline and due date that are given to the OA by the ERP system and expected to be fulfilled. Each OA estimates the execution time for the order in general and for each activity in particular. Therefore, if the activity execution proposed by the RAs does not fit the deadline/due date, an OA contacts OAs that use the resources the RA is interested in. The information about such OAs is sent by the RA together with the refusal of allocation within the initial scheduling request protocol. Figure 6.4 shows the general communication mechanism of rescheduling and consists of the following steps:

1. OA1 sends a scheduling request with the possible execution time;
2. If the RA cannot find a solution that fits the required parameters, it responds with a message of refusal and informs the OA1 about the reasons for the refusal. In particular, it is the list of OAs (with the resource allocation identifications) that allocates the requested time slots of the RAs for its own activity execution;
3. OA1 sends a request to the OAs that reserved the resources. In the example request, OA1 asks OA2 to cancel a particular resource allocation and provides the deadline/due date of the order of OA1;
4. Depending on the applied constraints for the plant, OA2 evaluates the possibility for rescheduling. For instance, if OA1's deadline is in 1h and deadline OA2's is in 1 day, than the OA2 agrees to cancel a requested reservation. Before informing OA1, OA2 sends a reservation cancelation request to the RA and informs the RA that the cancellation is being done for another OA, namely, OA1. Therefore, the RA only allows OA1 to use this time slot;
5. OA sends a scheduling request to the RA again;
6. The RA confirms the reservation.

6.6 Summary

From a general point of view PABADIS'PROMISE, PROSA, and PABADIS can be summarized with respect to the following criteria:

Autonomy and aggregation: on the one hand, PROSA lacks flexibility due to the direct control over aggregated entities, and on the other hand, PABADIS lacks optimization because of total distribution. PABADIS'PROMISE defines Production Order decomposition as something that establishes rules of controlling autonomous entities without dramatically reducing flexibility.

Cooperation and hierarchy: where PROSA has an explicit hierarchy that causes rigidness of order changes, and PABADIS provides no hierarchy that implies overhead in the case of managing complex products, PABADIS'PROMISE offers implicit hierarchy (Production Order decomposition; flexible structure of orders; dynamic control of resources by resources) that finds a balance between two approaches.

Decisionmaking: while decision making in PROSA is centralized, meaning there is one control entity per functionality, PABADIS supports a completely distributed decision-making mechanism. PABADIS'PROMISE provides a semi-distributed approach based on clustering of resources at the shop floor and PO decomposition at the MES layer.

Data interoperability: with PROSA having implementation-specific data that cause difficulties for the system installation and PABADIS data that virtually do not have an established connection to the ERP system, PABADIS'PROMISE ontology covers the entire automation pyramid linking all three layers together.

Control flow: on the one hand, PROSA has a strict vertical control flow that lacks feedback to the upper layer of the ERP. PABADIS, on the other hand, has a limited feedback to the ERP, providing it only at the end of the production cycle. Therefore, PABADIS'PROMISE supports a permanent connection with the ERP via planned periodic or event-based reports during the Production Order life cycle.

Table 6.1 Distributed approaches comparison

	PROSA	PABADIS	PABADIS'PROMISE
Autonomy	Low	High	High
Aggregation	High	Low	Production order decomposition
Cooperation	Low	High, selfish	High, benevolent, dynamic resource control
Hierarchy	Explicit	No hierarchy	Implicit, flexible order structure
Decision-making	Centralized	Distributed	Distributed, shop floor clustering
Data interoperability	Implementation-specific data	Limited ERP connection	Common ontology throughout the system
Control flow	Vertical	Horizontal	Bidirectional
ERP feedback	No feedback	Limited feedback	Periodic and event-based reports

In conclusion PABADIS'PROMISE has achieved a balanced combination of the rigid PROSA architecture and the chaotic PABADIS approach with vital contributions from of MetaMorph dynamic optimization.

All the above-mentioned architectures approach manufacturing automation from the conceptual points of view and often overlook the application aspects that are brought by the distributed nature of the concepts. In particular, security, product identification, and data interoperability are vital aspects for practical implementation.

Distributed systems lack single point of control with decision making spread over the multiple entities that communicate with each other. This causes higher

security risks, due to the intensive communication that such architectures require and the fact that there is no single entity that controls the system.

Another challenge of distributed systems is the lack of general overview of the processes and components on the shop floor. It is difficult to keep track of the products, work in progress, and materials in the plant, and it is often impossible to say where exactly a specific piece is located. Therefore, more advanced identification of the workpieces is required to guarantee the efficient operation of the system.

Last but not least, distribution of decision-making functionality requires interoperability of information flow over all three layers of the automation pyramid as well as mechanisms of distributed databases. Therefore, a common ontology with mechanisms of data abstraction for different control entities is required to guarantee the coordination within the system.

6.7 Practical Implementation Aspects

In order to address the practical implementation issues raised in the previous section, PABADIS'PROMISE defines features of data security, material identification, and consistent data management. The following sections describe the chosen solutions, namely, a three-zone security architecture, Radio Frequency Identification (RFID)-based product tracking, and XML-based automation ontology.

6.7.1 MES Security Architecture

Often overlooked by developers as a less important issue, a missing security concept always strikes back when it comes to actual implementation. It is especially vital in a distributed environment when the use of standard IT technologies, a collaborative agent system, and low-resource RFID devices introduces new threats to the usually closed automation environments. Security threats from the agent system point of view are:

- Modification of agent data and code during transmission;
- Abuse of a platform by a malicious or strayed agents including authentication theft;
- Misuse of resources (unauthorized access) or wrong pairing of entities, i.e., loss of origin or untraceable unitary (unauthenticated communication).

The PABADIS'PROMISE architecture integrates the security needs of the two loosely coupled agents with the main focus at the control and MES layers but also the interfaces to the surrounding layers.

The problem of securing a distributed MES is the use of devices with low resources in the MES layer and the field layer that are not capable of carrying the additional load of (strong) security measures. Hence, a hierarchical security concept is applied that is organized in three zones of different mutual trust. This is a common approach in industrial automation [26, 27].

For the purpose of the PABADIS'PROMISE system a security system with three main zones is a suitable solution (Fig. 6.5). The three zones match the three functional areas of MES: high-layer components of ERP; manufacturing execution with order and Resource Agents, and interfacing between the enterprise layer and the MES. Further subzones, called local and functional domains, are introduced that encapsulate operations such as real-time communication that conflict with the usual security measures.

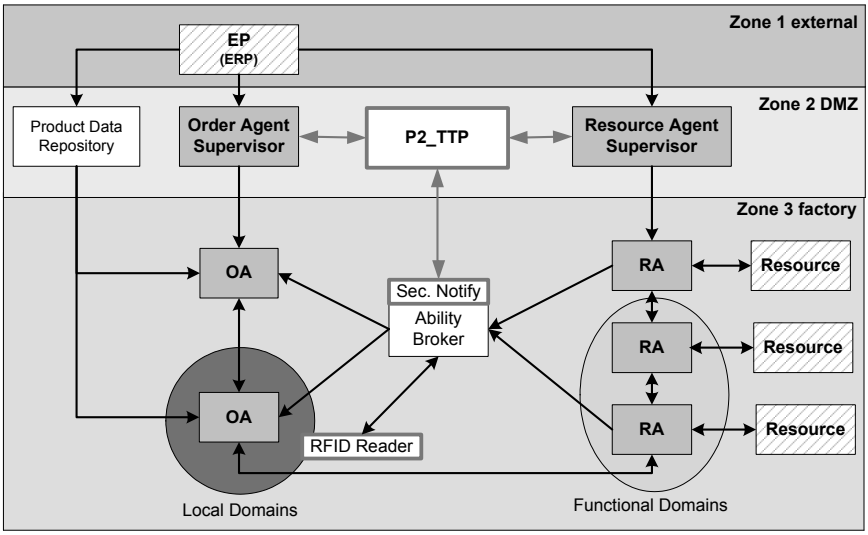

Fig. 6.5 Three-zone security model

The topmost is Zone 1: external meaning outside of the MES layer, including some ERP components of the enterprise layer.

In the middle of the three-zone model is Zone 2: Demilitarized Zone (DMZ). In the DMZ are located the supervising entities RAS, OAS, and PDR but also security management entities such as a Trusted Third Party (P2_TTP). These entities translate the semantics and syntax of the ERP payload to the factory layer. They are also responsible for establishing secure connections to the ERP, which is mainly based on standard Internet technologies such as SSL/TLS and XML encryption for Web services security to the limited security operations of the factory floor.

At the bottom of the three-zone model is the factory zone is placed. Due to hardware limitations of the embedded systems running OAs and RAs they are nei-

ther capable of doing strong computations nor can deal with high communication loads. For example, OAs are usually run on embedded devices that move around with their associated products and are often connected only through RFID communications. Hence, within the third zone only less resource consumption and, therefore, usually rather week security measures exist for authentication and access control. Nevertheless, overall security is maintained since the entrance to the zone is protected by strong security measures in the zone(s) above. Each data or request for operation must pass all zones on the way to its destination to allow weak authentication and encryption inside the inner zone, e.g., a request from the ERP first has to pass the Web-service security, then the checks at the OAS, the firewall to the factory zone, and the authentication inside the factory zone. If a security check fails, than the requested operation is not permitted and an exception handling takes place that is part of the P2 protocols.

The PABADIS'PROMISE security model allows one to apply a defense-in-depth concept that enables the system engineer to integrate weak components such as RFID tags and low resource Programmable Logic Controls that are not capable of implementing heavy security functions.

6.7.2 Radio Frequency Information Technology (RFIT)

The general problem with distributed system architectures is their abstraction from the real world, and that comes with a price when applying them to applications. One of the main challenges is tracking products, materials, and work-in-process pieces that are managed by purely software-based entities, namely, holons or agents.

PABADIS'PROMISE uses RFID tags to track the physical pieces in the plant and provides two types of RFID tags (PIT and PAHT): Product Identification Tag (PIT) – simple passive RFID tags used for identification of materials – and Product and Agent Host Tag (PAHT) – active RFID tag equipped with a Foundation for Intelligent Physical Agents (FIPA)-compliant agent platform that is used to identify products and host OAs that manage their production.

In the latter approach, RFID tags not only contain the product identification and product data, but also run an agent host providing an agent platform for the OA it hosts. It provides an environment for an agent to perform its action on the tag. This confers a considerable advantage in performance, because the OA is running constantly but also requires more tag resources (memory, processor), due to the fact that it has to run an agent host. Because standard agent platforms such as JADE [28] require a Java Virtual Machine (JVM) that consumes a lot of memory and processor capacity, PABAIDE'PROMISE developed a C-based agent environment called CARE (C-Agent Runtime Environment) that provides sufficient agent functionalities and is FIPA-compliant [29].

The main advantage of PAHT is that RAs do not need to retrieve product identification or product data at all, because the product and the OA are the same entity. There is no need for product identification at all because OA identification for PAHT is sufficient [30].

Both the physical and the logical connection of a product and an agent are archived, which gives a higher level of flexibility to the agent and product. Products can be freely moved from one environment to another, without regard to the actual location of the associated agent.

Due to the fact that products and their agents are the same entity, there is no need for additional intelligence of a tag other than an agent. That means that the tag does not need to analyze data or provide communication mechanisms, but rather serves as a database and as an agent host.

6.7.3 Data Interoperability

Last but not least is the challenge of data interoperability that many concepts (i.e., PROSA) shift to the realm of applications rather than including them in the conceptual core of systems.

This often leads to integration problems, especially when trying to combine ERP systems with shop floor solutions. PABADIS'PROMISE provides an ontology that incorporates standard structures that can be filled with specific data and can be decomposed in order to optimize performance and utilizes the resources efficiently. The best example of the PABADIS'PROMISE approach to data handling is the PO that is used by all three layers of the automation pyramid (Fig. 6.6).

The structure of the PO is designed to provide the possibility for its further decomposition as well as possibilities of the concurrent execution. The general description of the PO contains information about the type of product, its quantity, deadlines and due date and additional information specific for a particular instance of a product. The rest of the PO is a combination of PS and Node Operators (NOs).

The Process Segment (PS) is a basic construction element of the PO describing a single task or operation that the system has to fulfill in order to go further to complete the product. Each PS is specific to a PO but can serve as a reusable core. An Ability is a recipe of a single operation that is predefined and a set of parameters that are unique for each product or PO. In addition to the Ability description, material data are specified in a way that the consumed and produced materials are defined for each PS. That makes it possible to decompose the order into a set of suborders that can be executed concurrently, based on the principle that there is one OA that is responsible for a single piece of material/work in progress/product [31].

Finally, NOs are the logical links between different PS and combine them into a single hierarchical structure of the PO also facilitating order decomposition. There are several types of NOs declaring the type of the operator (Sequence, BranchOr, BranchAnd, JoinOr, JoinAnd) as well as input and output. There can be multiple inputs and outputs making it possible for different ways to execute the PO, which therefore gives extra flexibility to the production system by adaptation to the loss of a single type of a machine or transport line.

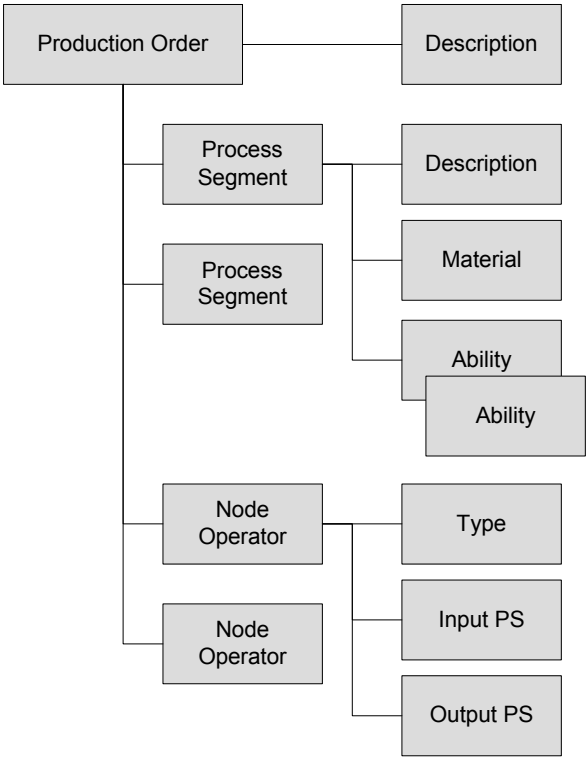

Fig. 6.6 Production order structure

6.8 Conclusion

Distributed Control Systems are a state-of-the-art approach. Based on agent technology, there are several implementations especially for the Manufacturing Execution Control layer.

In this chapter the basic structures used within the majority of approaches have been described. It has been shown using a comparative approach how they are re-

flected within the PROSA, the PABADIS, and the MetaMorph architectures. The benefits and drawbacks of each approach were discussed.

The PABADIS'PROMISE architecture has all the advantages of the above-named approaches, resulting in a new architecture most fitting for recent problems in factory automation on the MES layer.

It has been shown how PABADIS'PROMISE is incorporating the features and design decisions of PROSA, PABADIS, and MetaMorph. In addition, approaches addressing special requirements are addressed using the most recent technologies were presented.

References

[1] Cox, W.M and R. Alm. 1998. The right stuff; America's move to mass customization. *Annual Report Federal Reserve Bank of Dallas,* 3-26.

[2] Mönch, L. 2006. Agentenbasierte Produktionssteuerung komplexer Produktionssysteme. Wiesbaden: Deutscher Universitätsverlag.

[3] Wyns, J. 1999. Reference Architecture for Holonic Manufacturing Systems. PhD thesis, Katholieke Universiteit Leuven, Belgium.

[4] VanBrussel, H., J. Wyns, P. Valckenaers, L. Bongaerts and P. Peeters. 1998. Reference architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry,* 37: 255-274.

[5] Maturana, F. and D. Norrie. 1996. Multi-agent mediator architecture for Distributed Manufacturing. *Journal of Intelligent Manufacturing,* 7: 257-270.

[6] Shen, W., Q. Hao, H. Yoon and D.H. Norrie. 2006. Applications of agent systems in intelligent manufacturing: an update review. *International Journal of Advanced Engineering Informatics,* 20 (4): 415-431.

[7] Radjou, N. 2003. Software agents in business: still an experiment. Agent Link Magazine, issue 13, June.

[8] Bongaerts, L. 1998. Integration of Scheduling and Control in Holonic Manufacturing Systems, Ph.D. thesis, Katholieke Universiteit Leuven, Belgium.

[9] Deen, S.M. 2003. *Agent Based Manufacturing - Advances in the Holonic Approach, Advanced Information Processing*. Berlin: Springer.

[10] Maturana, F. and D. Norrie. 1996. Multi-agent mediator architecture for Distributed Manufacturing. *Journal of Intelligent Manufacturing*, 7.

[11] Shen W., D. Xue and D. Norrie. 1997. An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems. In: *Proceedings of the International Conference on the Practical Application of Intelligent Agents and Multi-Agents*.

[12] Van Dyke Parunak, H., A. Baker and S. Clark. 1998. The AARIA agent architecture: from manufacturing requirements to agent-based system design. In: *Proceedings of Workshop on Agent-based Manufacturing ICAA*.

[13] Heikkila, T., M. Kollingbaum, P. Valckenaers and G.J. Bluemink. 1999. manAge: an agent architecture for manufacturing control. In: *Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*.

[14] Toenshoff, H., I. Seilonen, G. Teunis and P. Leitão. 2000. A mediator-based approach for decentralised production planning, scheduling and monitoring. In: *Proceedings of International Seminar on Intelligent Computation in Manufacturing Engineering*.

[15] Klostermeyer, A. 2002. *Agentengestützte Navigation wandlungsfähiger Produktionssysteme*. Thesis (PhD). Otto-von-Guericke-Universität Magdeburg.

[16] Walsh, P., A. Vontas, P. Koutsakas, A. Koumpis, J. Martinetz and A. Klostermeyer. *Building enterprise-wide information supply chains based on the fractal company concept – lessons learnt, Challenges and Achievements E-Business and E-Work, Part 1*. Leipzig: IOS Press.

[17] Bratoukhine, A., T. Sauter, J. Peschke, A. Lueder, A. Klostermeyer. 2003. Distributed automation - PABADIS vs. HMS. *IEEE International Conference on Industrial Informatics INDIN'03*.

[18] Alexander, C. 1979. *The Timeless Way of Building.* New York: Oxford University Press.

[19] Gemma, E., R. Helm, R. Johnson and J. Villisides. 1995. *Design Patterns, Elements of Reusable Object Oriented Software.* Reading: Addison-Wesley.

[20] Sanz, R. and J. Zalewski. 2003. Pattern-based control systems engineering – using design patterns to document, tranfer, and exploit design knowledge. *IEEE Control Systems Magazine,* June: 42-60.

[21] Perronne, J.-M., L. Thiry and B. Thirion. 2006. Architectural concepts and design patterns for behavior modeling and integration. *Mathematics and Computers in Simulation*, 70: 314-329.

[22] Thramboulidis, K. 2008. Challenges in the development of mechatronic systems – the mechatronic component. In: *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation,* September, Hamburg.

[23] Lüder, A., J. Peschke, A. Klostermeyer and H. Kühnle. 2004. Design pattern for distributed agent based factory automation. In: *IMS International Forum 2004 – Global Challenges in Manufacturing.* Cernobbio. Proceedings: 783-791.

[24] Lüder, A. 2006. *Strukturen zur verteilten Steuerung von Produktionssystemen.* Thesis (Habilitation). Otto-von-Guericke-University Magdeburg.

[25] Bratukhin, A., B.A. Khan and A. Treytl. 2007. Resource-oriented scheduling in the distributed production. In: *Proceedings of the 5th International Conference on Industrial Informatics*, pp 1091-1096.

[26] Dzung, D., M. Naedele, T.P. von Hoff and M. Crevatin. 2005. Security for industrial communication systems. In: *Proceedings of the IEEE*, 93: 1152–1177.

[27] Schwaiger, C. and T. Sauter. 2002. Security strategies for field area networks. In: *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*. November Sevilla. pp 2915-2920.

[28] Bellifemine, F., G. Caire and D. Greenwood. 2007. *Developing Multi-Agent Systems with JADE*. Chichester: Wiley.

[29] FIPA consortium: FIPA web page. Available from: www.fipa.org. [Accessed: 19.03.2008].

[30] Heinze, M., A. Lüder and W. Gantner. 2008. Structure and Functionality of a PABADIS'PROMISE Agent System. In: *Proceedings of the 14th International Conference on Concurrent Enterprising*, June, Lisbon.

[31] Wuensch, D. and A. Bratukhin. 2007. Multilevel order decomposition in distributed production. In: *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation*, pp 872-879.

[32] Rabelo, R. and L. M. Camarinha-Matos. 1994. Negotiation in Multi-agent based dynamic scheduling. *International Journal on Robotics and Computer Integrated Manufacturing*, 2 (4): 303-310.