ISeB (2006) 4: 159–185
DOI 10.1007/s10257-005-0030-5

ORIGINAL ARTICLE

Lars Mönch · Marcel Stehli

ManufAg: a multi-agent-system framework for production control of complex manufacturing systems

Published online: 7 December 2005

Abstract In this paper, we present a framework for the implementation of multi-agent-systems for production control of complex manufacturing systems. We present the results of a requirement analysis for production control systems for complex manufacturing systems; then we describe the framework design criteria. Our framework supports the inclusion of distributed hierarchical decision-making schemes into the production control. Furthermore, in order to increase the coordination abilities of multi-agent-systems, we follow the decision-making and staff agent architecture suggested in the PROSA reference architecture. We indicate the usage of the framework for designing and implementing an agent-based production control system for semiconductor manufacturing processes in a case study.

Keywords Agent-based analysis and design · Frameworks · Production control · Complex manufacturing systems · Software development

1 Introduction

Production control tasks for complex manufacturing processes are still challenging because of the complexity of the related decision tasks and information systems. A lot of work has done in this area over the last

L. Mönch (✉) · M. Stehli
Institute of Information Systems, Technical University of Ilmenau,
Helmholtzplatz 3, 98684 Ilmenau, Germany
E-mail: Lars.Moench@tu-ilmenau.de
E-mail: Marcel.Stehli@tu-ilmenau.de

20 years; however it seems that we are still far away from an ultimate solution.

Currently, it seems that the improvement of operational processes creates the best opportunity to realize the necessary cost reductions in manufacturing systems. Therefore, the development of efficient planning and control strategies is highly desirable in the manufacturing domain. In the course of the development of new planning and control algorithms, the researchers and developers have to take into account the new opportunities in advanced software technologies. However, because of the widely used legacy software systems in enterprises, it is hard to integrate more advanced production control strategies into real-world situations.

Software agents allows for the implementation of distributed planning and control algorithms. The agents are able to act autonomously; on the other hand their communication abilities ensure a cooperative behavior and the fulfillment of global system goals. Agents support concurrent execution of different production control algorithms and lead to robust and scalable production control applications. However, even if we can choose among a lot of agent-tools and agent frameworks (see for example JADE 2004; FIPA-OS 2004; Zeus 2004; Vrba 2003) it is still quite difficult to implement multi-agent-systems. This difficulty is mainly caused by the fact that agent-building tools are generic and domain independent. On the other hand, for the manufacturing domain, we found common features like spatial and temporal problem decomposition through the use of distributed hierarchies that should be supported by a large class of production control systems. Therefore, it makes sense to go a step beyond the existing tools and design and implement a framework for this class of production control systems.

The paper is organized as follows. In the next section we provide the results of a requirement analysis for production control issues in complex manufacturing systems. Then we discuss related work. We describe important features of the *Manuf*acturing *Ag*ency (ManufAg) framework. We complete the paper by indicating the use of the framework for implementing a production control application for the semiconductor manufacturing production control domain.

2 Requirement analysis

We perform a two-step requirement analysis. The objective of the domain analysis is the development of a domain model that contains the knowledge of a domain in a systematic way. The domain analysis includes a description of the domain, the acquisition of domain knowledge, and a structured description of constant and variable properties of the domain under consideration. Our approach for deriving a domain design is similar to the scope, commonality, and variability (SCV) analysis suggested by Coplien et al. (1998).

In a second step, a domain design is necessary. The key task of the domain design is the transformation of the results of the domain analysis into reusable, implementation oriented artifacts and the design of relations between these artifacts.

2.1 Domain analysis

A complex manufacturing system is defined as a system whose components are smaller production systems in their own right, i.e., a complex manufacturing system is a relation on interconnected manufacturing sub systems (cf. Mesarovic et al. 1970). We are interested in the domain of production control of complex manufacturing systems. Production control refers to dispatching and scheduling of jobs on scarce resources, i.e., machines. We are not interested in long-term and mid-term planning; however, we have to take into account that planning decisions influence the production control decisions. We denote the machinery of the manufacturing system as basis system. The basis process is given by the allocation of resources by jobs in a time dependent manner. Beside the basis system and basis process we consider the production control system and the production control process. The production control process is responsible for dispatching and scheduling decisions, i.e., to answer the question which job should proceeded in which time slot on a certain machine.

We use the terms basis system, basis process, production control system, and production control process in order to structure fix and variable properties of the domain.

We structure the domain knowledge according to the following dimensions (Bussmann 2003):

1. a set of production goals,
2. a decision space that includes especially various types of restrictions,
3. a set of decision rules ranging from myopic dispatching rules over negotiation-based schemes to more advanced scheduling approaches.

The production goals are usually tardiness related, i.e., our production control approach has to make sure that due dates are met by the jobs. Furthermore, a certain throughput has to be achieved. Note that opposite to mass production, tardiness related production objectives require that the jobs are tracked and controlled individually.

The decision space is formed by the following restrictions (cf. Uzsoy et al. 1994; Ovacik and Uzsoy 1997 amongst others):

– a diverse product mix,
– re-entrant process flows,
– parallel machines with complicated dedication practices,
– sequence-dependent set up times,
– a mix of different process types, including batch processes,
– an inclusion of preventive maintenance issues into production control,
– prescribed due-dates for the jobs,
– precedence constraints for the different operations of a single job according to the routes of the different products.

Note that the decision space is formed by restrictions from both the basis system and the basis process.

Decision rules are necessary to solve the problem which job should be processed next on a given machine. The decision rules try to fulfill the production goals subject to the restrictions of the manufacturing system. In some situations, very simple decision rules like dispatching rules or contract net type negotiation procedures are successful. In other situations, it seems to be more appropriate to use more centralized scheduling schemes.

We start with describing constant domain properties. Jobs are included in our domain as dynamic entities that have own goals (i.e., single jobs try to meet their own due date as much as possible). Machines are responsible for the processing of the jobs. Because machines used in complex manufacturing systems usually are expensive, the machines try to utilize their capacity as much as possible. The description of the constant properties of the basis system and basis process is completed by specifying the products given by routes and assignments from machines to single operations. Note that orders, resources, and products are also the main ingredients of the PROSA reference architecture for holonic production control systems (Van Brussel et al. 1998). From a production control point of view each control system of a complex manufacturing system contains a dispatch component that decides which job has to proceed next on a given available machine. Of course, this decision may be influenced by higher-level decisions.

Decisions that are based on automated material handling systems have to taken into account in more advanced manufacturing systems (Babiceanu et al. 2004).

Because we consider complex production systems, we have to face with a segmentation of the basis system into different sub systems. The production control of each sub system is performed using local data. In order to achieve a production control with respect to global goals, coordination has to take place between the different sub systems and the related sub processes. Interactions between different jobs and machines are also necessary in order to make optimal batch and set up decisions. Here, a batch is defined as a collection of jobs that are processed at the same time on the same machine.

The following flexible domain properties exist. We can use hierarchies in order to allow a distributed decision-making (Schneeweiss 2003). The distribution can be performed in a spatial and temporal manner. On the other hand, pure heterarchical approaches are possible. According to this structuring from an organizational point of view, flexibility is necessary in order to use various types of production control schemes. The applied production control scheme may also depend on a specific part of the basis system, i.e., a different scheme is eventually necessary for bottleneck machines. Flexibility has to be ensured with respect to batch production. In this situation, beside the assignment and sequencing task for jobs to machines, we have to solve the batch formation problem, i.e., we have to decide which jobs form a certain batch. Furthermore, we have to fix the concrete form of the routes.

The constant and flexible characteristics of complex manufacturing systems are represented in Fig. 1 by a property diagram (Czarnecki and Eisenecker 2000). We distinguish between permanent (indicated by bullets) and optional properties (denoted by circles) in this type of diagrams.

Manufacturing systems change over time. For example, based on different customer behavior the product mix is a subject of changes. Furthermore,

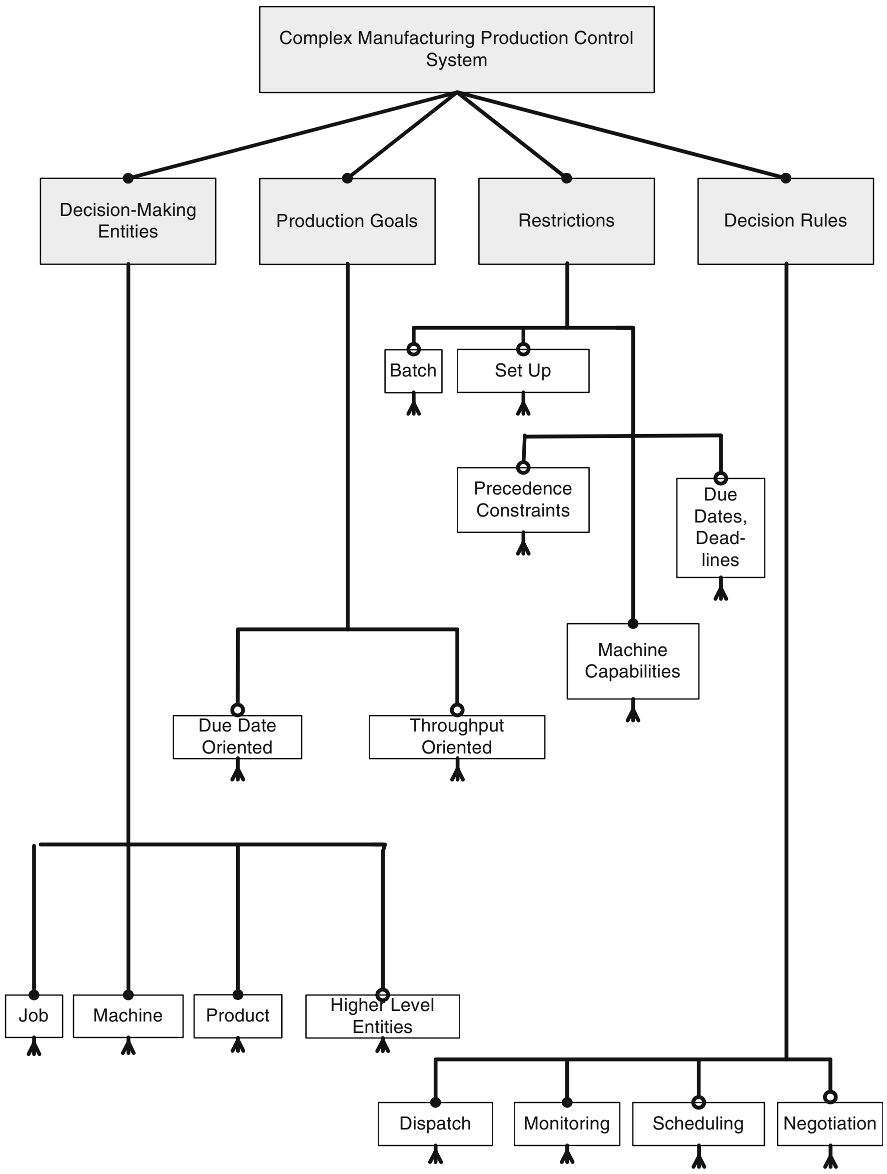

Fig. 1 Property diagram for the complex manufacturing domain

dynamic bottlenecks occur from time to time as a result of machine break-downs and product mix changes. Therefore, the used production control strategies should be adaptable to specific situations on the shop-floor in order to allow for a situation-dependent decision-making.

Complex manufacturing systems are dynamic and stochastic, i.e., new orders arrive in manufacturing systems over time, machine break-downs happen, or the priority of customer orders may change. This is the reason for problems related to performance assessment of manufacturing systems. Discrete event simulation offers a way to model the appropriate dynamic and stochastic behavior of manufacturing processes. Hence, a performance assessment of novel production control schemes is possible by means of emulation and should be supported by the production control system (Brennan and O 2000).

2.2 Domain design

From the domain analysis it follows that we need decision-making units that could be arranged into hierarchies. Furthermore, the decision-making is based on local data and therefore coordination schemes are necessary. Beside the decision-making entities, different types of decision rules are necessary.

Because of the distributed decision-making, agents are appropriate software artifacts in order to implement the resulting production control system (Schneeweiss 2003). Agents have by definition communication capabilities; hence, coordination mechanism can be implemented quite easily (cf. the literature in the book edited by Weiss (1999)).

Following the PROSA reference architecture, we distinguish between decision-making agents that are the software representation of the decision-making entities in Fig. 1 and staff agents that encapsulate the decision rules.

The decision-making agents have to allow the representation of hierarchies. So far, there exist several prototype multi-agent-systems that are based on the PROSA reference architecture (see, for example, Heikkilä et al. 2001; Indrayadi et al. 2002). However, so far no framework exists that supports the development of such type of multi-agent-systems in a generic way.

3 Literature review and analysis of existing frameworks

The analysis and design step for agent-based production control systems is a highly non-trivial task that should be based on an analysis of the decision problems to be solved (Bussmann et al. 2001; Bussmann 2003). Recent surveys of existing multi-agent-systems in the manufacturing domain are provided by Shen and Norrie (1999) and Caridi and Cavalieri (2004). Although a couple of systems exists (for example the system by Ünver and Anlagan 2002), it seems to be still difficult to implement an agent-based production control system starting from scratch.

Frameworks for developing multi-agent-systems are widely accepted and provide support for easy and fast agent development. However, most of the existing frameworks are generic with a focus on infrastructure issues and only rarely dedicated to a specific domain.

The widely used frameworks JADE (2004); FIPA-OS (2004); Zeus (2004) have been assessed with respect to the capabilities to easily adapt the agent model and the underlying system to a specific field of interest; especially to the manufacturing control domain. We identified four key features that are crucial for agent systems in manufacturing control in order to decide whether the frameworks are suitable or not:

– representation of agent hierarchies,
– representation of decision rules,
– modeling of process restrictions,
– support for discrete event simulation.

Agent hierarchies are necessary to model multi-agent societies for production control of complex production systems.

The capability to represent decision rules is essential to enable the agent to decide what is the next action based on different criteria, its own knowledge, and the environmental circumstances.

As explained in sect. 2, manufacturing control has to take into account a lot of restrictions of the controlled processes. Therefore, we look for an easy way to model these restrictions in our framework. However, appropriate modeling of process restrictions is a challenging and time-consuming task, which is essential for the domain of complex manufacturing systems. The framework should provide support for the modeling efforts.

Discrete event simulation is necessary to assess the performance of the production control schemes developed by using the existing framework. A simulation tool is required for the emulation of the dynamic and stochastic behavior of the manufacturing system to be controlled.

In Table 1, we investigate whether the frameworks JADE, FIPA-OS, and Zeus support the key features or not.

After an analysis of some generic frameworks, it turns out that major efforts are required in order to implement multi-agent-systems for production control using the generic frameworks.

Especially agent hierarchies have to be modeled by the system designer. It is not clear whether such a change of the agent organization could be easily handled by the runtime environments of the corresponding agent-based systems. The runtime environment provides an infrastructure for multi-agent-systems. It is necessary to change internals within the runtime environment regarding the communication and agent service handling in case of agent hierarchies.

Only the Zeus-Framework provides several types of organizational relationships within agent societies. However, hierarchical structures are not supported. Another feature of Zeus is an agent-to-legacy system interface to facilitate inter-operability with existing software systems. This feature could

Table 1 Requirements vs. existing frameworks

Framework/Tool	JADE	FIPA-OS	Zeus
Evaluation criteria			
Agent hierarchies	No	No	No
Capabilities of representing different decision rules	Integration of JESS (an expert system)	Integration of JESS (an expert system)	Deliberative and goal-directed agents
Support of discrete event simulation	No	No	Agent-to-legacy system interface
Modeling of process restrictions	No	No	Agent tasks with constraints

be used to implement a coupling interface between multi-agent-systems based on Zeus and an appropriate discrete event simulation tool.

Furthermore, only Zeus provides a generic technique to describe constraints for agent tasks that could be used to model typical restrictions of the complex manufacturing system domain.

Another source of problems is the poor (run time) performance of Java programming language-based multi-agent-systems. This is also true for BDI-type agent toolkits like dMARS (D'Inverno et al. 2004) and JACK (Howden et al. 2001). Among the Java programming language-based frameworks and tools only the JADE Framework is able to treat a large amount of operating agents with common computational resources as discussed by Vrba (2003). FIPA-OS and Zeus failed the performance evaluation with a large number of agents. JADE is also the only tested framework still supported.

It seems to be possible to enrich existing frameworks by more domain specific knowledge and concepts. This approach is suggested, for example, in the business process domain by Shepherdson et al. (2003). Here, the authors customize the JADE-Leap framework in order to develop multi-agent-system applications for their domain. An application framework closely related to PROSA was discussed by Heragu et al. (2002). However, mainly control decisions on the machine level are considered. No software development support is discussed in this paper. Another example for an application framework for intranet document management was suggested by Ginsburg (1999).

However, because we have to model agent hierarchies, have to assess the impact of new production control schemes on the manufacturing system characteristics, and because our domain requires performance critical production control schemes, we decided to design our own framework in order to develop multi-agent-systems for production control purposes.

In this paper, we describe an application framework related to production control of complex manufacturing system including all relevant software development issues.

4 Framework design criteria

An object-oriented framework is defined as a pre-fabricated extensible set of classes or components with pre-defined collaboration between them and extension interfaces (Fayad et al. 1999). Thus, a framework forms a pre-defined architecture that models the interaction between its components and potential extensions (Fontura et al. 2002).

We differentiate between general design criteria and domain dependent requirements that have to be considered in order to develop a generic, flexible, scalable, and reusable architecture that fits the needs of a given domain and takes current standards into account. Furthermore, a software framework for multi-agent-systems additionally has to meet the requirements that arise from the agent point of view (Weiss 1999).

The following five major design criteria had been specified.

(Domain) generic requirements The focus of the suggested framework is the manufacturing control domain. Therefore, a universal architecture is not necessary. We have to design a framework that allows treating different organizational types of manufacturing systems in combination with a set of different production control paradigms.

Flexibility Flexibility means that a system should be able to adapt to various circumstances. A certain degree of flexibility (caused by actions) is a feature that a multi-agent-system has by definition (Weiss 1999). In the case of our framework, we incorporate flexibility in the context of production control of complex manufacturing systems.

Scalability We define scalability of production control systems developed by the suggested framework as the possibility to increase the number of machines and jobs without a significant loss of run time performance. This design criterion is very important because most of the decision problems related to production control are NP hard. Hence efficient distributed heuristics have to be applied to solve these decision problems.

Reusability Software reusability is one of the major goals of developing frameworks. In our case, software reusability is ensured by developing a (generic) agent infrastructure, a set of basic agents, roles, and interaction protocols.

Compatibility with existing standards The suggested framework has to comply with FIPA standards (FIPA 2004; Mařík et al. 2003) for agent-based systems as close as possible. Furthermore, the framework has to follow state of the art middleware standards.

5 Framework architecture

In this section, we will discuss the architecture and technical details of the ManufAg framework concerning the design criteria and requirements from the former sections. Starting with technical details on the used middleware we will look closer at the underlying software architecture for both infrastructure and agents. Furthermore, we discuss organizational issues with respect to the domain and the suggested agent architecture.

5.1 Infrastructure issues and technical details

An Agent Runtime Environment had been developed that extends the FIPA Abstract Architecture (Mařík et al. 2003) to ensure compatibility with existing standards. This runtime environment provides the infrastructure for agent execution and allows the agents a concurrent execution on the same host. The FIPA Abstract Architecture specifies an agent directory, a message transport system, an agent communication language (ACL), and a service directory as mandatory parts shown in Fig. 2.

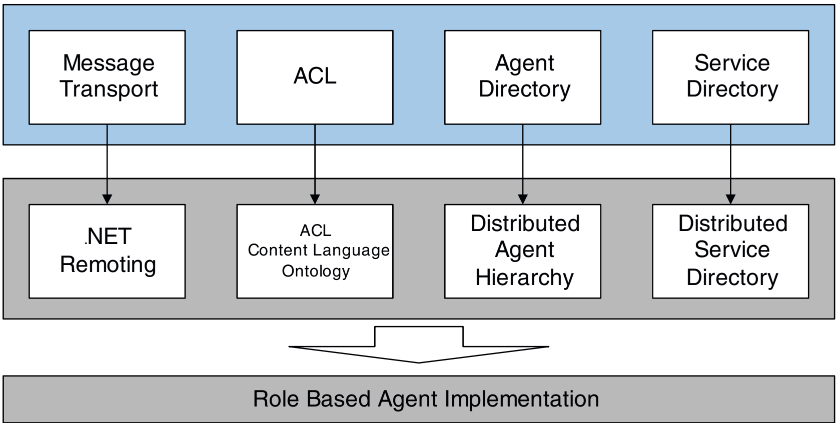
Message Transport
ACL
Agent Directory
Service Directory
.NET Remoting
ACL Content Language Ontology
Distributed Agent Hierarchy
Distributed Service Directory
Role Based Agent Implementation

Fig. 2 FIPA abstract architecture and its usage within the framework

The agent directory is extended by our framework to enable the implementation of a distributed agent hierarchy. The service directory is modified in a way that fits the requirements for service distribution as a consequence of the different organizational structures existing in the manufacturing domain. We chose Microsoft .NET as middleware for the implementation of the Agent Runtime Environment. .NET provides by the .NET Remoting Framework a powerful and scalable technology for implementing a message transport system for inter-agent communication using FIPA ACL.

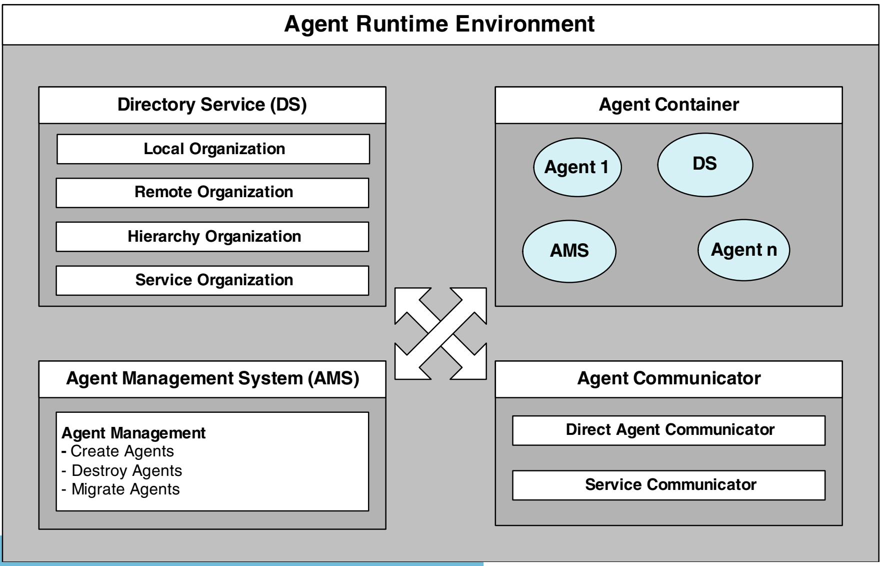

Fig. 3 Agent runtime environment

Figure 3 gives an overview of our developed Agent Runtime Environment. Note that the FIPA agent directory and service directory are encapsulated by the directory service agent (DS). Furthermore, the message transport and ACL processing is encapsulated by the agent communicator. The two additional parts implemented in the runtime environment are the agent container that holds all local agents hosted within the environment and the agent management system (AMS) responsible for creating, destroying, or migrating the agents hosted on that platform.

The service directory agent is important for the environment. It manages the hierarchical as well as the local organization of the agents within a multi-agent-system. It is also responsible for the organizational communication with other known runtime environments. To optimize the overall system performance, several runtime environments can run on local and remote computers to distribute a multi-agent-system.

5.2 Communicational architecture

Communication among autonomous agents is a key feature of multi-agent-systems to enable cooperation and coordination features. The communicational part of the framework is organized in a way that makes it possible to compose and restrict these features depending on the chosen organizational structure of the multi-agent-system. We have to introduce communicational rights to the multi-agent-system framework because we implement the basic system using some kind of peer-to-peer service announcement system instead of the FIPA suggested global directory facilitator as a single point of failure. The suggested communicational rights are summarized in the Table 2.

The communicational rights form a relation among the agents. This relation is symmetric, i.e., when agent A has the right to communicate with agent B then agent B can also communicate with agent A. However, due to the hierarchical structuring of the control system the relation is not transitive.

We use the administrative part of FIPA ACL messages and also the basic speech act types suggested by FIPA.

The DS is responsible for establishing connections between different remote runtimes. If the runtimes are known to each other, the agents on these runtimes can communicate and cooperate with respect to their communicational rights. These rights are also used for publishing the agent's services that are provided within the multi-agent-system. As mentioned before, a peer-to-peer service communication is utilized to distribute the different agent services among the known runtime environments. To store the remote service information, a local service-blackboard at each runtime is updated. Each agent can explore this blackboard in order to find an agent offering a required service.

5.3 Single agent architecture

We define an agent as an autonomous entity that can carry out some actions on behalf of another agent or a user. An agent can also perceive information from its environment and tries to achieve a set of explicit or implicit goals.

Table 2 Communicational rights of the agents

Communicational rights	Description
Public	The agent can communicate with other public agents and provide its services to all connected agent runtimes
Private	The agent can communicate with other agents only at its home runtime
Hierarchical	The agent is organized within a hierarchy. It can only communicate with its parent and child agents
Hierarchical echelon	The agent can communicate with other agents from same echelon additionally to the hierarchical communication rights
Cluster	A multi-agent-system can be separated into several clusters and the communication within these clusters can be decoupled from each other
Combined rights	Several communicational rights can be combined, i.e., the cluster right can be combined with the hierarchical and the hierarchical echelon right

An agent mainly consists of a set of behaviors and actions defining the agent reaction to different circumstances like incoming messages or events. An agent observes events generated by the environment and can act within that environment. Agent behaviors form the basis for reactivity and pro-activity. Mainly the behaviors are used to implement agent interactions. Therefore, the behaviors handle the message exchange between one or more agents depending on the purpose of interaction. Furthermore, an agent provides a set of services that can be requested by another agent. A service is a logical envelope for a set of actions owned by an agent. As we mentioned earlier, an agent has goals and if they are not implicitly given, explicit goals can be defined. An agent uses its actions to fulfill its goals. We use this goal-oriented approach to model some sort of pro-activity for each agent. Pro-activity means that an agent itself can decide to perform an action at a certain point of time or under certain circumstances depending on its goals. A goal is a list of (current and future) actions, called action item list (AIL), which is restricted to special environmental events or a given time frame. An agent can build and modify its own AIL at runtime, i.e., to monitor some critical actions within a given horizon or to initiate an action as a result of a specific time or environmental event. The proposed pro-active architecture makes it possible to implement agents that can react to specific events from the environment and handle them. In addition, the agents are able to interpret events and initiate itself some special actions from the AIL.

For example, a machine group agent reacts to a machine break-down of a single machine always with a maintenance action and later starting a new scheduling action for the jobs waiting in its queue.

Finally, an agent is part of some kind of environment, which it is connected with. It observes events raised from the environment and it can act within that environment. Figure 4 shows a UML class diagram illustrating the basic agent architecture.

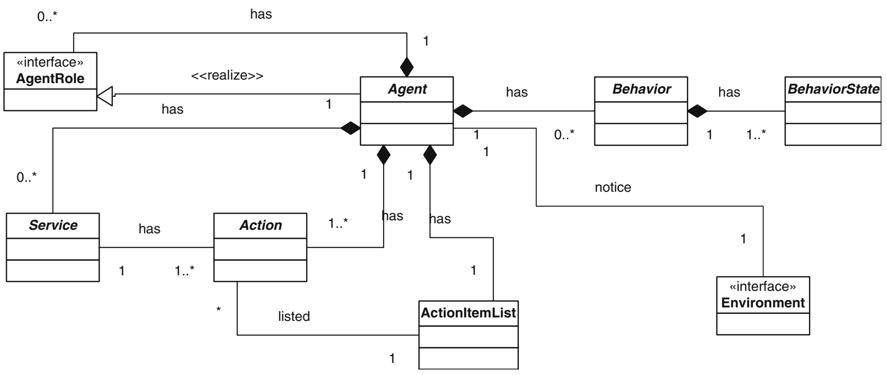

Fig. 4 UML class diagram for the agent architecture

In order to introduce flexibility, we assign agent roles to each agent. The role handling ability provides the key tool to introduce flexibility into the multi-agent-system developed by using the framework. The software architecture of a single agent is designed using a role-based approach (Odell et al. 2003). An agent itself has its primary role that cannot be changed during lifetime. Nevertheless, the agent can adopt and discard roles during runtime if necessary. Therefore, the characteristic of an agent can differ on demand. A new role can provide the agent with new features like services, goals, and actions; it may assign new behaviors to the agent. These behaviors define when and why an agent acts on special events like incoming messages, environmental or agent state changes.

The flexibility provided by the role concept and the role handling gives a multi-agent-system the ability to adapt to several circumstances with respect to the manufacturing domain. A multi-agent-system for the control of flexible manufacturing systems could be modeled using different roles for the different capabilities of such a system and allows the reconfiguration during runtime. Furthermore, adaptability and reconfigurability can be modeled by the role-based implementation to enhance or expand the functionalities of an agent-based manufacturing system to meet changes in technology without reconfiguring the entire system.

To ensure online availability of active agents a multi-threading approach of implementing the agent and the agent's behavior had been developed that allows concurrent execution. An agent can handle multiple requests from different agents within the same context or react to different events occurring at the same time. For example, an agent may negotiate with two agents while it requests a service from a third agent. During that conversation, an environmental event could occur that needs immediate treatment.

5.4 Multi-agent-system architecture

The agent architecture is the basis for the further development of a generic multi-agent-system architecture for the manufacturing domain applying the

PROSA reference architecture. As mentioned before, we have to treat different organizational types of manufacturing systems, i.e., job shops, flexible job shops, and flow shops combined with several types of production control paradigms like simple dispatching based approaches (Indrayadi et al. 2002), distributed negotiation-based resource allocation schemes like the contract net or several types of auctions (Srinivas et al. 2004), and more centralized scheduling approaches like the shifting bottleneck heuristic (Mönch and Driessel 2005).

We apply the PROSA reference architecture for holonic manufacturing systems (Van Brussel 1998) to achieve these goals. The result of the design process is a generic architecture describing agent hierarchies. The hierarchies can be used for appropriately modeling of different types of manufacturing systems (i.e., basis systems). PROSA suggests the use of decision-making and staff agents as a starting point for further adaptation to specific production control problems. Staff agents can be used to implement different types of production control schemes.

Combining the single agent architecture with the multi-agent-system architecture based on PROSA we suggest a hybrid agent architecture containing reactive and deliberative approaches. The well-known hybrid architecture InteRRaP (Müller 1996) utilizes three layers within each single agent. The first layer treats reactive situations. The second layer is a planning layer, whereas the third layer is a cooperation layer for agent interactions. The decision-making and staff agent approach of PROSA implemented in the frameworks allows a hybrid layered architecture offering a separate planning layer by the staff agents and a reactive and pro-active layer using the decision-making agents.

PROSA and the suggested layered approach is also the key to ensure scalability for a multi-agent based control system. Scheduling algorithms that are computationally expensive can be assigned to staff agents that can be distributed over several powerful computers. Using this idea, it makes it easier to adapt a production control system to different work load situations. Hence, a good overall runtime performance can be ensured.

5.5 Pattern for decision-making and staff agents

Based on the PROSA reference architecture, we distinguish between decision-making agents and staff agents. Decision-making and staff agents are independent of the organizational form of the multi-agent-system, i.e., we have to consider these two types of agents both in heterarchically and in hierarchically organized multi-agent-systems.

Decision-making agents provide the following functionality:

1. prepare decisions,
2. make decisions,
3. information support for other decision-making agents,
4. activation of other decision-making agents,
5. time driven request of results of staff agents,
6. request of services of other decision-making agents,

7. treatment of exceptions during the preparation of a decision and during the decision-making process,
8. check the list of future or current agent activities (AIL) for relevant entries.

The functionality of decision-making agents is represented by a set of behaviors. We describe the resulting behavior of decision-making agents in more detail in Table 3.

The following generic functionality is provided by staff agents:

1. prepare the problem solution,
2. calculation of internal parameters of the problem solution method,
3. feed the problem solution method with data and parameters,
4. solution of the problem,
5. interruption of the solution process in an event or time driven manner,
6. providing the results of the problem to other agents,
7. treatment of exceptions during the preparation of a decision and during the decision-making process.

Based on the previous described functionality of decision-making, we can derive the following behavior of staff agents shown in Table 4.

Table 3 Behaviors of decision-making agents

Behavior	Description
Prepare_decision_making	Modeling of the preparation phase of a certain decision
Make_decision	Modeling of the decision-making phase
Inform_DM_agent	Information of other decision-making agents on certain decisions
Start_staff_agent	Activation of a decision-making agent
Get_staff_agent_result	Decision-making agent asks for the transfer of results of a certain staff agent
Request_DM_agent_service	Request for services from other decision-making agents

Table 4 Behaviors of staff agents

Behavior	Description
Prepare_solution	Modeling of the preparation phase of a certain solution activity
Parameterize_algorithm	Parameterization of a certain solution algorithm with external or internally determined parameters
Solve_or_interrupt	Determination of (feasible) solution of the problem under consideration. The decision-making agent can terminate the solution process
Communicate_solution	The decision-making agent that requests the solution of a certain problem will be informed on the obtained solution

Interaction between decision-making agents and staff agents can be described in a rather generic way. The actions of a certain agent can be described by behavior states. Transitions from one state to another are required in order to perform an action. Transitions are triggered by certain events or other pre-conditions. Due to space limitations, we do not describe the states of each behavior. But we explain the interaction between decision-making and staff agents by means of UML sequence diagrams. In Figs. 5 and 6, we present the interaction between decision-making agents and staff agents from a decision-making agent point of view. We represent each single behavior by a separate class. We describe the state transitions of each single behavior by using the self-delegation mechanism from UML sequence diagrams. We also show the used speech act types according to FIPA ACL.

We present the interaction between staff agents and decision-making agents from a staff agent point of view in a similar way in Fig. 7.

5.6 Organization of multi-agent-systems

5.6.1 Federated agent organization

Federated structures of multi-agent-systems for production control are usually used to generate agent systems without any central control. Negotiations using contract-nets as well as auction-based solutions are widely

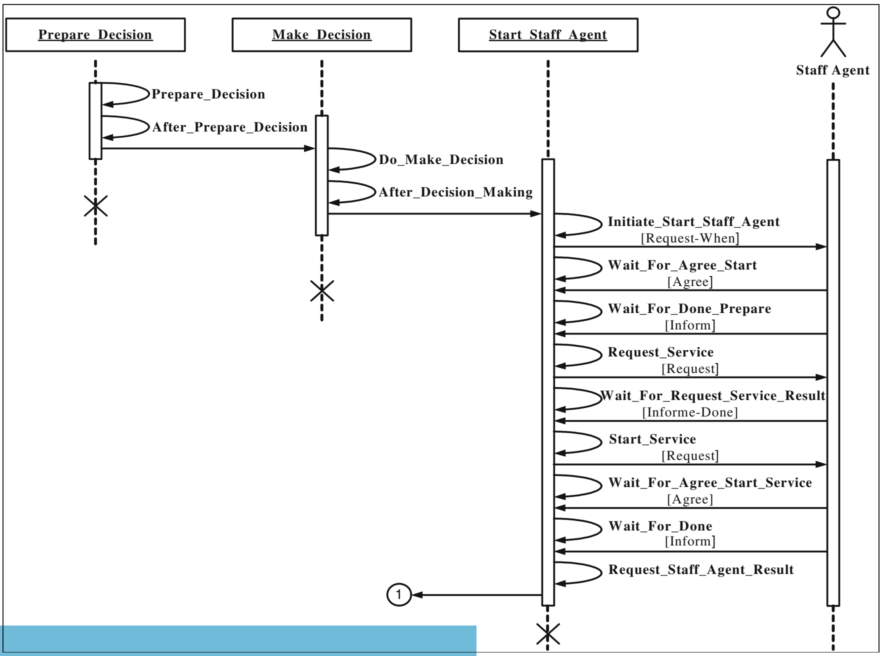

Fig. 5 Interaction of decision-making and staff agent I

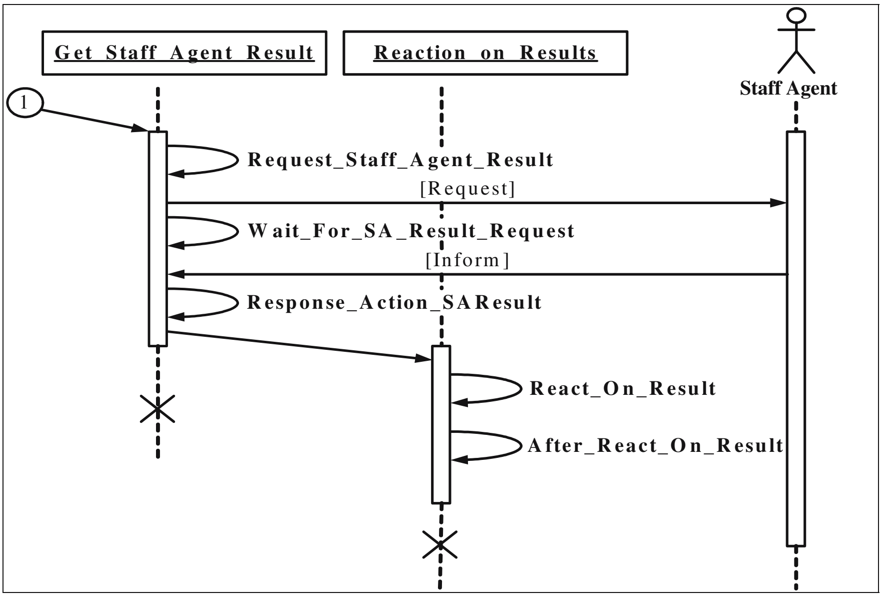

Fig. 6 Interaction of decision-making and staff agent II

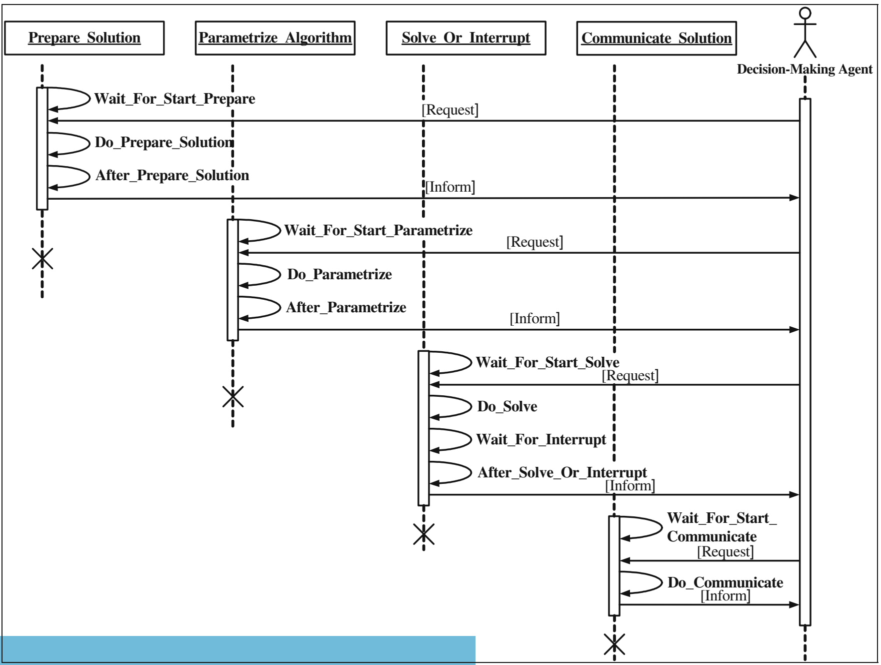

Fig. 7 Interaction between staff agent and decision-making agent

accepted problem solvers within multi-agent-systems. Supporting federated structures is essential for multi-agent frameworks as well as providing templates for the coordination processes like contract nets. Based on the architecture for staff and decision-making agents another point of view can be added to such flat structured organizations by allowing algorithmic approaches to be added to the cooperation schemes.

Usually, a large number of low-level agents form a complex system with difficult and hard to predict system behavior. However, for smaller problems like a single work center, federated organization for dispatching and scheduling jobs on the machines could be an appropriate way.

5.6.2 Hierarchical agent organization

To tackle the problem of complexity, hierarchical agent structures are introduced to achieve a predictable and under global aspects favorable system behavior (Mesarovic et al. 1970; Schneeweiss 2003).

Based on the PROSA reference architecture for holonic manufacturing systems, a top-down tree-like agent hierarchy can be build. Figure 8 shows a general agent hierarchy. From the control point of view the hierarchical decomposition means that the upper layers provides control guidelines for the next level below by means of feed forward instructions. The lower level uses these guidelines for its decision-making process. On the other hand, feedback cycles are possible that influences the upper decision-making entities.

Another important feature of this approach is that the different control strategies achieved by the implemented layers are distributed in spatial and temporal manner and allows, in principle, a concurrent decision-making by agents at the same or different echelons (cf. Mesarovic et al. 1970 for the notation of echelons in hierarchical systems).

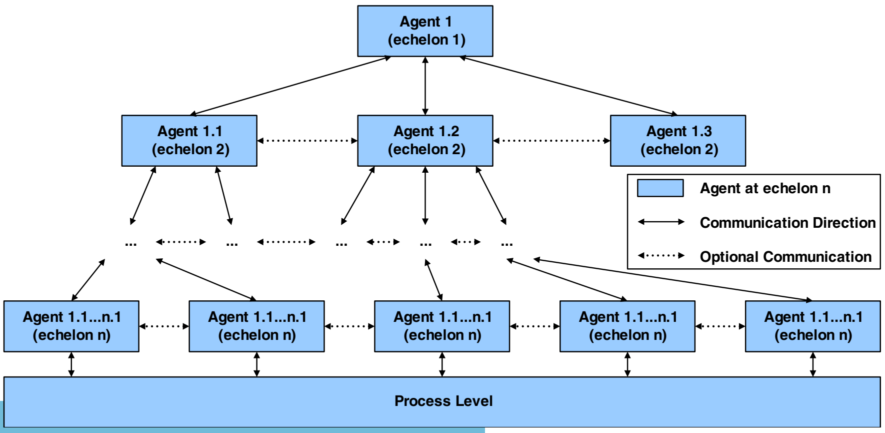

Fig. 8 Hierarchical organization of a multi-agent-system

A hierarchical architecture is a powerful way to map the organizational structure of production control systems to an agent-based system with different levels of competence (Schneeweiss 2003). A chain of responsibility can be build through the whole hierarchy to decompose a production control problem into different sub problems and use the solutions obtained from agents above and below in the hierarchy to get a better overall solution with respect to the global design goals of the production control systems. Usually, the granularity of the decision problem decreases with the hierarchy level of the agent. Agents at the upper levels of the hierarchy modify goals or the decision space of decision problems by the lower level decision-making agents. Furthermore, direct instructions expressed through actions to be performed are also possible. The later approach is called primal coordination in coordination theory, whereas the first approach is called image coordination or dual coordination (Mesarovic et al. 1970).

Despite the hierarchical structure and the communication restrictions among the levels, there is no rigid master slave relationship between an agent and its parent or child agent. Feedback cycles are established where the parent agent modify its solutions tacking into account the more detailed solution of its child agents.

When a hierarchical organization of an agent-based manufacturing control system implemented by using the ManufAg framework is the preferred solution, then this choice is not necessarily absolute for all agents. A hierarchically organized production control system is usually divided into two classes of agents. Hierarchy agents serve within a hierarchy of agents either as a parent agent or child or both and can only communicate and cooperate with agents in adjacent levels. Hierarchy free agents are not part of any hierarchy; they are able to interact with all agents at an arbitrary hierarchy level. This is important for agents that represent entities that usually move through a production systems like jobs or provide over time changing process related information.

In order to build a distributed agent hierarchy across different platforms, a unique identifier, called hierarchy identifier (HI), is introduced. This identifier is the organizational knowledge base for each agent. It provides the agents with a system wide unique hierarchy name that does not depend on the runtime where the agent is hosted. Furthermore, the HI stores the potential parent agent of a hierarchically organized agent and keeps track of its subsidiary child agents. The hierarchical organization of the agents is performed automatically as soon as an agent gets alive. The hierarchical organization system owned by each single runtime helps the agent to find its place within the hierarchy by looking for the appropriate parent and child agents.

The following announcement protocol shown in Fig. 9 makes them known each other. The child agent is responsible for the hierarchy announcement. It sends a request for registration to its parent agent. If the parent agent agrees to the request it organizes its new child agent within its local child repository. Both agents are now able to communicate with each other.

Another aspect of the hierarchical organization is the access to the services provided by the agents. The services have to be organized similar to the formed hierarchy with restricted access depending on the type of the agent and its level in the hierarchy.

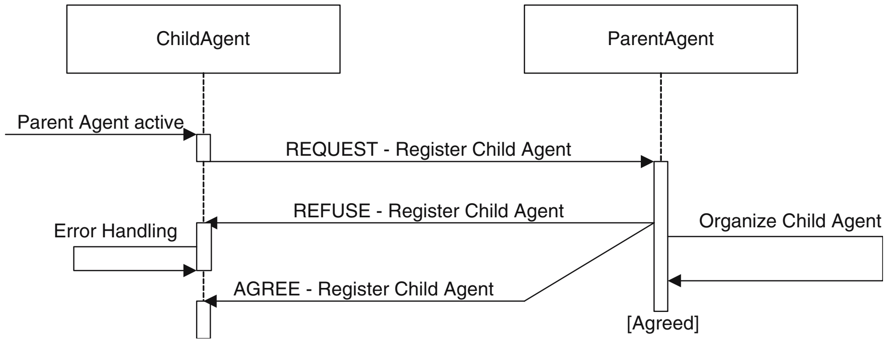
ChildAgent
ParentAgent
Parent Agent active
REQUEST - Register Child Agent
REFUSE - Register Child Agent
Organize Child Agent
Error Handling
AGREE - Register Child Agent
[Agreed]

Fig. 9 Hierarchical announcement

5.6.3 Cluster organization

A clustered organization can be established by using the ability of connecting multiple runtimes and the clustering of communication rights. This form of organization can be utilized to structure a large number of agents into several sub organizations related to a given context and with a proxy agent as interface for such a sub organization. The clustering of agents within several contexts is useful to define areas or scopes that cannot be organized within a hierarchy, for example a supply chain network. This approach is similar to the concepts of holons (Mařík et al. 2002; Fischer et al. 2003). Hence, it is possible to integrate such a cluster into a hierarchy of agents by adding the proxy agent to the hierarchical structure. Figure 10 shows an example for a cluster organization of a hierarchically organized multi-agent-system and a federated agent system.

5.7 Ontological and content language support

Ontology and content language are important features to allow for a meaningful communication of different agents. We refer to ontologies as a conceptualization of a domain (Obitko and Mařík 2002), i.e., we use mainly the terminological component of an ontology. Due to the wide-spread range of problems within the manufacturing domain and the required

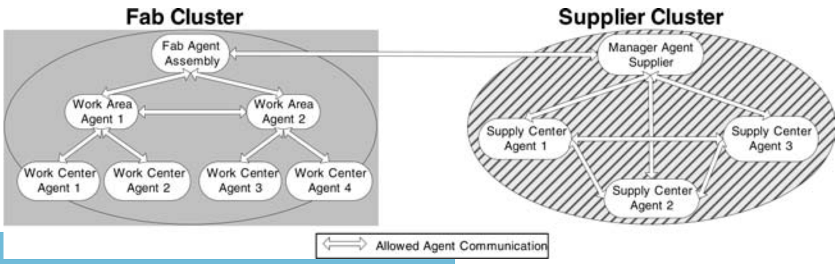

Fig. 10 Example for cluster organization

problem-specific ontological expressions, we did not integrate a domain dependent ontology into the ManufAg framework. Nevertheless, we provide a flexible and customizable approach for ontology support. There is no prescription of how an ontology should look like. As a minimum requirement, it must be possible to convert the elements of the used ontology into .NET compatible class structures that can be interpreted by an agent. An Ontology description element has to be defined additionally in order to label the ontology, integrate the ontology into the system, and publish it to the agents. A similar approach is chosen for providing content language support. A Codec element has to be implemented that encapsulates the decoding and encoding of a message using the implemented content language. Note that each agent can handle more than one ontology and content language by using its content handler. Each conversation has to refer to the used ontology and content language.

Ontologies for the complex manufacturing system domain can be derived from the broader ontologies Enterprise Ontology (Uschold et al. 1998) and OZONe (Smith and Becker 1997). Furthermore, it is possible to use derivates from the general purpose FIPA content languages SL0 or SL1 (see FIPA 2004 for more details).

5.8 Framework issues

We discuss design decisions for our framework from a software engineering point of view. Note that these decisions are strongly influenced by the suggested multi-agent-system architecture. We can distinguish between white-box, black-box, and grey-box frameworks (Fayad et al. 1999). A white-box framework provides a set of abstract classes and interfaces without a meaningful default implementation, while a black-box framework consists of classes that could be customized by composition techniques and parameter settings. A grey-box framework can be considered as a mixture of both types (Fayad et al. 1999).

The ManufAg framework is designed as a grey-box framework mainly influenced by the manufacturing control domain. As pointed out in the domain analysis section, we can differentiate between constant and flexible domain properties. Usually, we use black-box constructs for constant domain properties while white-box techniques are exploited for flexible domain properties.

A set of agent roles and agent interactions are provided by using black-box components because of the constant character of the relationship between the different elements of a manufacturing system and the related production control system, i.e., there will be always jobs and resources for processing the jobs as well as schedules and decision-making entities that determine these schedules. The relationship between these elements is prescribed by PROSA. Therefore, these entities can be modeled using black-box techniques.

It is necessary to provide a certain degree of flexibility in order to meet the characteristics of specific production control applications. White-box classes are used for the purpose of developing new roles for agents and the interaction relationships between them to meet requirements that could not be modeled by using the black-box components because they are too specific for certain applications. These elements are composed to a software-architecture that allows a smooth operation for all agent and non-agent parts of the production control system that has to be developed. Table 5 gives an overview of the main elements of the framework and its relationship with respect to the type of the framework.

6 Case study: FABMAS

FABMAS is a hierarchically organized multi-agent-system for production control of semiconductor manufacturing processes. It was developed based on the ManufAg framework. Semiconductor manufacturing is an example for a complex manufacturing process (cf. Atherton and Atherton 1995; Uzsoy et al. 1994). It includes a large number of machines (up to 1,000), an over time changing product mix, sequence-dependent setup times, a mix of different process types, for example batch processes, customer related due

Table 5 Features and framework class categorization

Feature	Type	Comment
Agent	Grey-box	Each agent needs a main role and can be customized by additional roles. Almost ready to run agents for common purposes that can be parameterized
Agent role	Grey-box	Either a black-box approach using pre-defined roles or a white box approach if new roles for special purposes are needed is supported
Behavior	Grey-box	Some behaviors are provided with the framework. Other behaviors must be derived from base classes
Agent goals	White-box	Goals are given implicitly or explicitly. The framework specifies a structure for goal description using the AIL
Agent tasks	White-box	Implementation depends on agent goals and its specialization. Only a rough structure is provided
Scenarios	Black-box	Scenarios encapsulate a specific interaction between different agents and consist of pre-defined behaviors, agent goals, and tasks that can be parameterized
Ontology	Grey-box	A basic ontology for management purpose is provided and can be extended. Other ontologies can be added on demand
Communication language	Black-box	The communication language serves as a technical container for agent communication. It is standardized by using FIPA ACL
Content language	White-box	A content language can be added by implementing a specific codec that supports the language

dates of the jobs, preventive maintenance tasks because of complicated machinery, and frequent machine break-downs.

A three-layer hierarchical production control approach is suggested for the FABMAS system due to the physical decomposition of the shop-floor of a semiconductor wafer fabrication facility (wafer fab) into work areas and on the next level into machine groups that contain parallel machines. Hierarchical communicational rights are applied due to the structure of the feedback enabled scheduling process. Production control guidelines and schedules are communicated top-down while process information is provided bottom-up.

On the top layer we use a beam-search-type algorithm (Habenicht and Mönch 2002) in order to assign planned start and completion dates to so called macro operations. A single macro operation consists of a certain number of consecutive process steps that are assigned to process in a fixed work area. Then, we use the macro operation related start and completion dates to determine detailed schedules for the jobs in each single work area. Here, we use a distributed variant of the shifting bottleneck heuristic in order to carry out the detailed scheduling of the jobs (Mönch and Driessel 2005). We solve scheduling problems for the jobs in each single work area via the shifting bottleneck heuristic (Pinedo 2002). Therefore, we assign a scheduling agent for the shifting bottleneck heuristic to each single work area agent. We exchange the planned start and end dates of the jobs between the work areas in order to improve the work area schedules with respect to the overall total weighted tardiness on the entire wafer fab. The (rigid) hierarchical communication rights are extended to allow interactions between the different work area decision-making agents at the work area echelon.

We use the following decision-making agents in the FABMAS system due to the hierarchical decomposition of the manufacturing system:

– single production system agent,
– multiple work area agents,
– multiple machine group agents.

These agents are implemented by using agents with hierarchical organization abilities while the organizational structure is given by the underlying semiconductor wafer fabrication facility model.

The following agents represent dynamic entities of the manufacturing system and are not part of a hierarchy and free of any communicational restrictions:

– each job agent represents a single job,
– batch agents are used to model batches, i.e., a temporary set of jobs with the goal to process the jobs simultaneously on the same machine.

We avoid the modeling of products by product agents.

The appropriate staff agents are represented as child agents for the decision-making agents. They are described in Table 6.

A specific ontology (Mönch and Stehli 2003) and content language (Mönch and Stehli 2004) for production control in the semiconductor

Table 6 Decision-making agents and staff agents within the FABMAS system

Decision-making agent	Staff agent	Description
Production system agent	Job planning agent	Determines job plans
	Production system monitoring agent	Determines certain performance measures on the shop floor level
Work area agent	Work area scheduling agent	Determines detailed schedules for the jobs of one single work area
	Work area monitoring agent	Determines certain performance measures on the work area level
Machine group agent	Machine group monitoring agent	Determines certain performance measures on the machine group level
	Machine group mediator agent	Mediator in contract net type resource allocation schemes
Job (batch) agent	Machine group mediator agent	Mediator in contract net type resource allocation scheme

manufacturing domain are developed. The ontology contains more than 100 domain concepts and takes into account the hierarchical production control approach. The content language relies on a context free grammar and uses appropriate XML objects that are derived from the domain concepts described in the ontology. The FABMAS system is completed by a blackboard that is between the agent-based production control system and the discrete event simulation tool AutoSched AP. We summarize the adaptation and extension of ManufAg that leads to the FABMAS prototype in Fig. 11.

The FABMAS system was successfully applied to certain reference simulation models of semiconductor wafer fabrication facilities to improve on-time-delivery performance. The MIMAC I model (Fowler and Robinson 1995) consists of over 200 machines that are organized in about 80 different machine groups. The machine groups form five work areas. According to this structure of the manufacturing system, we run the FABMAS system distributed on five PC's. We required computation time for one scheduling decision of the entire manufacturing system requires two up to 6 min on 2.4 GHz Pentium-IV PC's with 256 MB RAM and an Ethernet network with 100 Mbit/s. The FABMAS scheduling approach clearly outperforms pure dispatching schemes like first in first out (FIFO) or earliest due date (EDD) with respect to the performance measure total weighted tardiness in case of a heavy load of the wafer fab and of tight due dates of the jobs. For a more detailed description of the FABMAS prototype and also for computational experiments and results we refer to (Mönch et al. 2005).

7 Conclusions

In this paper, we outline a framework for multi-agent-systems for production control in the complex manufacturing domain. Based on a requirement analysis, we describe framework design criteria. We present an approach for the hierarchical organization of the multi-agent-systems using our framework. Furthermore, we describe a generic architecture for the implementation

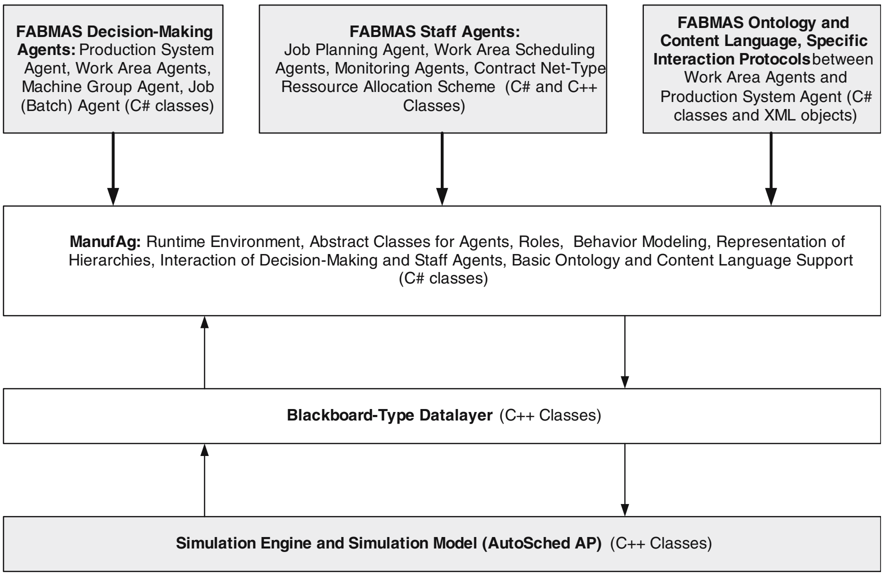
FABMAS Decision-Making Agents: Production System Agent, Work Area Agents, Machine Group Agent, Job (Batch) Agent (C# classes)
FABMAS Staff Agents: Job Planning Agent, Work Area Scheduling Agents, Monitoring Agents, Contract Net-Type Ressource Allocation Scheme (C# and C++ Classes)
FABMAS Ontology and Content Language, Specific Interaction Protocols between Work Area Agents and Production System Agent (C# classes and XML objects)
ManufAg: Runtime Environment, Abstract Classes for Agents, Roles, Behavior Modeling, Representation of Hierarchies, Interaction of Decision-Making and Staff Agents, Basic Ontology and Content Language Support (C# classes)
Blackboard-Type Datalayer (C++ Classes)
Simulation Engine and Simulation Model (AutoSched AP) (C++ Classes)

Fig. 11 Customization of the ManufAg framework that leads to FABMAS

of decision-making and staff agents that allows for a flexible integration of different control strategies. We finish the paper by indicating the use of the ManufAg framework for the implementation of an agent-based production control application for semiconductor manufacturing processes.

The suggested ManufAg framework fulfills the five major design criteria from sect. 4. We designed a framework that can be used for the implementation of both hierarchical and heterarchical organized production control systems. Therefore, different organizational types of manufacturing systems and related production control paradigms are supported by ManufAg. Because of the used differentiation in decision-making and staff agents and the suggested role-based approach the right degree of flexibility is obtained by the framework. The staff agents allow, in principle, for an implementation of distributed production control approaches. Hence, the suggested framework leads to scalable production control applications. The suggested infrastructure, the basic agents and roles, and the interaction protocols can be reused. ManufAg complies with state of the art middleware standards and with FIPA standards by using the .NET remoting framework and parts of the FIPA abstract architecture for ManufAg.

In future research, we try to use the framework for the implementation of an agent-based production control approach for flexible manufacturing systems. Furthermore, it is challenging to develop agent-based production control systems that combine scheduling and automated material handling issues based on the ManufAg framework.

Extending the framework by adaptive features is a logical next step for future research. Therefore, it seems to be necessary to consider specific staff agents that have machine learning capabilities and can be used to support a situation specific parameterization of other staff agents. Neural networks and decision trees seem to be appropriate.

Acknowledgment This research was supported in parts by the German National Science Foundation (DFG) under grant GM 18/1-2. The following people helped us during the development of the FABMAS prototype: Jens Zimmermann, Ilka Habenicht, Rene Drießel, Patrick Rempel, and Mario Angrabeit.

References

Atherton LF, Atherton RW (1995) Wafer fabrication: Factory performance and analysis. Kluwer Academic Publishers, Boston, Dordrecht, London

Babiceanu RF, Chen FF, Sturges RH (2004) Framework for the control of automated material-handling systems using the holonic manufacturing approach. Int J Prod Res 42(17):3551–3564

Brennan RW, O W (2000) A simulation test-bed to evaluate multi-agent control of manufacturing systems. Proceedings of the 2000 winter simulation conference, Orlando, USA, pp 747–1756

Bussmann S, Jennings NR, Wooldridge M (2001) On the identification of agents in the design of production control systems. Agent-oriented software engineering, Springer, pp 141–162

Bussmann S (2003) An agent-oriented design methodology for production control. Ph.D. Thesis, University of Southampton

Caridi M, Cavalieri S (2004) Multi-agent systems in production planning and control: an overview. Prod Plann Control 15(2):106–118

Coplien JO, Hoffman D, Weiss DJ (1998) Commonality and variability in software engineering. IEEE Softw 15(6):37–45

Czarnecki C, Eisenecker UW (2000) Generative programming: Methods, tools, and applications. Addison-Wesley, Reading, MA

D'Inverno M, Luck M, Georgeff M, Kinny D (2004) The dMARS Architecture: a specification of the distributed multi-agent reasoning system. Auton Agent Multi Agent Syst 8:1–49

Fayad ME, Schmidt DC, Johnson RE (eds) (1999) Building application frameworks: Object-oriented foundation of framework design. Wiley, NY

FIPA-OS (2004) http://www.emorphia.com/research/about.htm, 2004-11-18

Fischer K, Schillo M, Siekmann J (2003) Holonic multiagent systems: A foundation for the organisation of multiagent systems. Proceedings first international conference on industrial application of holonic and multi-agent-systems (HoloMas 2003), LNAI 2744, Springer, Prague, Czech Republic, pp 71–80

Fontura M, Pree W, Rumpe B (2002) The UML profile for framework architectures. Addison-Wesley, Pearson Education, London

Foundation for Intelligent Physical Agents (FIPA) (2004) http://www.fipa.org, 2004-11-18

Fowler JW, Robinson J (1995) Measurement and improvement of manufacturing capacity (MIMAC): final report. Technical report 95062861A-TR, SEMATECH, Austin, TX

Ginsburg M (1999) An agent framework for intranet document management. Auton Agent Multi Agent Syst 2:271–286

Habenicht I, Mönch L (2002) A finite-capacity beam-search algorithm for production scheduling in semiconductor manufacturing. Proceedings of the 2002 Winter Simulation Conference, San Diego, USA, pp 1406–1413

Heikkilä T, Kollingbaum M, Valckenaers P, Bluemink G-J (2001) An agent architecture for manufacturing control: manage. Comput Ind 46:315–331

Heragu SS, Graves RJ, Kim B-I, St. Onge A (2002) Intelligent agent based framework for manufacturing systems control. IEEE Trans Syst Man Cybern A Syst Hum 32(5):560–573

Howden N, Ronnquist R, Hodgson R, Lucas A (2001) JACK Intelligent Agents: summary of an agent infrastructure. In: Proceedings of the 5th international conference on autonomous agents

Indrayadi Y, Hadeli K, Valckenaers P, Van Brussel H (2002) Dynamic multi-agent dispatching control for flexible manufacturing systems. Proceedings 13th international workshop on database and expert systems applications (DEXA'02), pp 578–582

Java Agent Development Framework (JADE) (2004) http://jade.tilab.com/, 2004-11-18

Mařík V, Fletcher M, Pěchouček M (2002) Holons and agents: Recent development and mutual aspects. Proceedings MASA 2001, LNAI 2322. Springer, Berlin Heidelberg NY, pp 233–267

Mařík V, Pechoucek M, Vrba P, Hrdonka V (2003) FIPA standards and holonic manufacturing. Agent-based manufacturing. In: Deen SM (ed) Advances of the holonic approach, pp 89–121

Mesarovic MD, Macko D, Takahara Y (1970) Theory of hierarchical, multilevel, systems. Academic Press, NY, London

Mönch L, Driessel R (2005) A distributed shifting bottleneck heuristic for complex job shops. Comput Ind Eng 49:363–380

Mönch L, Habenicht I, Stehli M, Zimmermann J (2005) The FABMAS multi-agent-system prototype for production control of wafer fabs: design, implementation, and performance assessment, submitted to production planning and control

Mönch L, Stehli M (2003) An ontology for production control of semiconductor manufacturing processes. Proceedings first german conference on multiagent system technologies (MATES 2003), LNAI 2831, Springer, Erfurt, Germany, pp 156–167

Mönch L, Stehli M (2004) A content language for a hierarchically organized multi-agent system for production control. Proceedings ''coordination and agent technology in value network'', Essen, Germany, GITO-Verlag, pp 197–212

Müller JP (1996) The design of intelligent agents: A layered approach Lecture notes in computer science, 1177. Springer, Heidelberg

Obitko M, Mařík V (2002) Ontologies for multi-agent systems in manufacturing domain. Proceedings of the 13th international workshop on database and expert systems applications (DEXA'02), IEEE, pp 597–602

Odell J, Parunak HVD, Fleischer M (2003) Modeling agent organisations using roles. Softw Syst Model 2:76–81

Ovacik IM, Uzsoy R (1997) Decomposition methods for complex factory scheduling problems. Kluwer Academic Publishers, Boston

Pinedo M (2002) Scheduling: Theory, algorithms, and systems. 2nd edn, Prentice Hall, Upper Saddle River

Schneeweiss C (2003) Distributed decision making. Springer, New York Heidelberg Berlin

Shen W, Norrie DH (1999) Agent-based systems for intelligent manufacturing: A state-of-the-art survey. Knowl Inf Syst 1(2):129–156

Shepherdson JW, Lee H, Mihailescu P (2003) mPower—Component-based framework for multi-agent systems to support business processes. BT Technol J 21(4):92–103

Smith S, Becker M (1997) An ontology for constructing scheduling systems. Working notes of the 1997 Symposium on ontological engineering, AAAI Press

Srinivas M, Tiwari K, Allada V (2004) Solving the machine-loading problem in a flexible manufacturing system using a combinatorial auction-based approach. Int J Prod Res 42(9):1879–1893

Ünver HO, Anlagan O (2002) Design and implementation of an agent-based shop floor control system using windows-DNA. Int J Comput Integr Manuf 15(5):427–439

Uschold M, King M, Moralee S, Zorgios Y (1998) The enterprise ontology. Knowl Eng Rev 13(1):31–89

Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P (1998) Reference architecture for holonic manufacturing systems: PROSA. Computers in industry. Spec Issue Intell Manuf Syst 37(3):225–276

Vrba P (2003) Java-based agent platform evaluation. Proceedings first international conference on industrial application of holonic and multi-agent-systems (HoloMas 2003), LNAI 2744, Springer, Prague, Czech Republic, pp 47–58

Uzsoy R, Lee C-Y, Martin-Vega LA (1994) A review of production planning and scheduling models in the semiconductor industry, Part II: Shop-floor control. IIE Trans Scheduling Logistics 26(5):44–55

Weiss G (ed) (1999) Multiagent systems: A modern approach to distributed artificial intelligence. MIT Press, Cambridge, MA

ZEUS (2004) http://more.btexact.com/projects/agents/zeus/, 2004-11-18