Multi-Agent-Systems - A Natural Trend in CIM

Klaus-Peter Keilmann

University of Essen, D-45117, Germany

Abstract. During the last few years Computer Integrated Manufacturing (CIM) became an important aspect in the manufacturing industry. The different components of CIM - as there are the CAx techniques on the engineering side and manufacturing planning and control systems (MPC-systems) on the operational side - now show their influence on organizational structures and the working environment. Limits and problems of the fully automated manufactory - often called the "factory of the future" - have been recognized. Two ongoing trends can be observed: the decentralization of MPC-systems, leading to distributed MPC-systems, and the integration of the different modules of CIM, resulting in new requirements for the underlying information systems as well as for the persons using these integrated systems. The emerging field of distributed artificial intelligence (DAI) seems to offer interesting solutions to these requirements. Especially multi-agent-systems (MAS) have shown their feasibility in the area of CIM. The principles of coordination and communication are central aspects of these systems. Different prototypes have been implemented, for example HBBS: a hierarchical blackboard system for concurrent engineering, also used in the area of distributed manufacturing planning and control.

1 Introduction

Current markets require products that fulfill the specific wishes and needs of the customers. Therefore it forces the manufacturer to strengthen the customer orientation and the flexibility of his manufacturing units to fulfill customer demands fast. This enforces more complex information and manufacturing systems. Current development of computers and sensors supports this direction, but as more complex these systems are the less is their technical reliability [1]. How to deal with these complex systems is one more problem. Humans - the people that have to use and to work with these systems - cannot deal with over complex systems. Work nowadays can be characterized by a change from physical to psychic demands. Cause of the black-box character of most of the computational units the worker does not know what is going on inside. He has to react to instructions of a systems and therefore his behavior is no longer determined by himself. Another point is the lack of qualification to deal with these new systems. On the one side are technical skills - for example the knowledge of the command language, which button to push under which circumstances- on the other side knowledge about how to use the system - for example the use of a pointer device in the area of CAD instead of using a pencil [1].

As an overall postulation there must be the aim that the human has to be the controlling one in such complex systems. This is only possible if he knows the programmed functional logic inside his information system and is able to control it [1]. Segmentation at the manufacturing level and the development of distributed systems for CIM seem to offer possibilities to achieve this goal.

The conceptual idea behind segmentation of production is to simplify or to minimize the connections and relationships between the different manufacturing units. The aim is to gain more flexibility and productivity in the area of manufacturing. Team orientation is one of the important basic ideas. In all manufacturing concepts that follow the idea of team orientation, separated operating units will be defined, which are responsible for the manufacturing of a specific group of parts or a specific service. These units operate, within a specific framework autonomous with respect to the planning of the different necessary working steps. Different kinds of team oriented organizational structures can be seen, as there are manufacturing islands, flexible manufacturing centers, flexible manufacturing cells and flexible manufacturing systems [2]. Distinctions between these different forms of organization can be made in terms of complexity, the amount of available operations, the internal kind of material flow and the flexibility of the underlying transport system [3]. Segmenting the manufacturing is not enough. Responsibilities (e.g. for resources, scheduling, money, human resources) and decision making have to be delegated to lower levels. Due to increasing integration and networking capabilities of computerized manufacturing systems functional, organizational, and social separated units and structures should come to closer relationships [4]. DAI in our opinion offers technological and organizational possibilities to implement this new paradigm of manufacturing.

An overview of DAI, a classification of systems and agents and references to basic literature will be given in chapter two. Chapter three deals with the basic aspects of cooperation and communication. Chapter four contains a description of the structure of the hierarchical blackboard-system HBBS and its usage in the area of distributed manufacturing planning and control systems.

2 Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) is a relative new field of research. DAI is aiming to develop methods and techniques for solving complex problems by means of intelligent behavior of a distributed system. The first workshops and conferences (DAI workshops) took place in the United States, later in Europe - the Modeling Autonomous Agents in a Multi-Agent World (MAAMAW) workshops. The first monograph on DAI appeared 1987: Distributed Artificial Intelligence edited by M. Huhns [5].

Research in DAI can be split in Distributed Problem Solving (DPS) and Multi-Agent-Systems (MAS) [6]. In contrast to distributed systems the different modules have to cooperate, therefore DPS and MAS have to be separated from Parallel AI [7].

Parallel AI: These are systems that use parallelism as a possibility for faster computation. The decomposition of the problem is done by defining separate subproblems where no cooperation is necessary to solve them. Consequently parallel AI is no field of research inside DAI [7].

Distributed Problem Solving: Research in the area of DPS is concerned with the question how a problem can be divided into subproblems such that distributed, cooperating agents can act together to solve it by solving the subproblems. The task of problem solving is collaborative in the sense that sharing information and knowledge is essential to solve the overall problem [8]. DPS-systems are "a kind of top-down designed system, since agents are designed to conform to the requirements specified at the top" [9].

Multi-Agent-Systems: In multi-agent-systems agents are designed first, the solution strategy is defined later - depending on the problem. Agents are therefore pre-existing, autonomous and often heterogeneous [9], working cooperatively towards individual goals that interact. MAS are able to develop their coordination structure by themselves and to rearrange their roles if the context has changed. "A MAS can also be viewed as a bottom-up designed system" [9].

DPS and MAS can be seen as two extremes of the same spectrum [9]. This point is also expressed in the definition of Dezentralized Artificial Intelligence (DzAi). DAI deals "with the cooperative solution of problems by a decentralized group of agents" [5], whereas DzAI is "concerned with the activity of an autonomous agent in a multi agent world" [8]. Following Demazeau and Müller DPS can be stated as the main issue of DAI, MAS as the one for DzAI. It has to be said that the term multi-agent-system has to be used carefully, cause this term is often also used for a collection of agents that do not fulfill the aspects of MAS [7].

2.1 Classification of Systems

Work in DAI can be classified by eight dimensions; first introduced in [10], as shown in Tab. 1:

Dimension	Spectrum of Values
System Model	Individual ..... Commitee ..... Society
Grain	Fine ..... Medium ..... Coarse
System Scale	Small ..... Medium ..... Large
Agent Dynamism	Fixed .... Programmable .... Teachable .... Autodidactic
Agent Autonomy	Controlled ..... Interdependent ..... Independent
Agent Resources	Restricted ..... Ample
Agent Interactions	Simple ..... Complex
Result Formation	By Synthesis ..... By Decomposition

Table 1. Dimensions for categorizing DAI Systems

The dimensions are [9]:

System Model: Is the system represented as one single agent, consisting of distributed components, or are there multiple agents, forming committees or societies?

Granularity: How detailed can the problem be decomposed?

System Scale: The number of processing units. Is there one large system or is the computation done by many smaller units?

Adaptiveness: Do the units of the system have the ability to learn? Simple units are often programmed and therefore static, whereas others may have the ability to learn.

Control Distribution: How independent are the elements of the system. Is there a master-slave relation or are they autonomous? Are there units that have control over others? To which extend do elements need others to solve their local problems?

Resource Availability: Who has access to which resources and which access restrictions exist? Are there limited resources?

Communication: How complex are the interactions between elements? Is there a simple or standardized communication or do complex interactions exist?

Problem solving: How is problem solving done? Are the problems decomposed (top-down) or are solutions found due to synthesizing results from different elements (bottom-up)?

2.2 Classification of Agents

There exist multiple points of view, what an agent is or what an agent should be. To avoid the definition of a universal agent usable for all kinds of problems, a hierarchy of agents as shown in Fig. 1 seems to be useful [11].

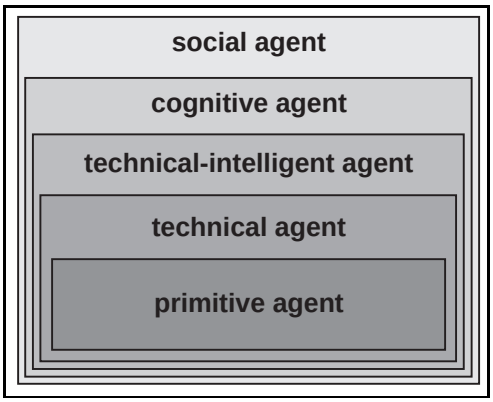

Fig. 1. Agent hierarchy

Primitive agent: Sensor-actor system that reacts on information kept by its sensors in a fixed manner.

Technical agent: Robots and flexible transport-systems, as they are available nowadays. They often include some sort of programmable control.

Technical-intelligent agent: Autonomous systems that are capable of solving a class of defined problems on their own under circumstances that do not have to be known a priori.

Cognitive agent: Agents that are able to learn.

Social agents: The discussion which capabilities are necessary to characterize a social agent is still open. Therefore no definite description could be given.

This hierarchy does not imply, that the capabilities of an included agent must be available for all upper agents. Every agent has features that are not of interest for others - for example an agent responsible for some kind of optical control in manufacturing has the capability to decide quickly if the current piece of work is in a regular condition. This ability is not mandatory for social or even cognitive agents [11].

2.3 Bibliography

Literature concerning Distributed Artificial Intelligence as a new field of research is not as easy to find as for other research directions (for example theoretical computer science, artificial intelligence, ...), therefore a bibliography of good introductory texts and overview of DAI will be given. The work of Huhns, Bond and Gasser can be seen as the basic literature in the area of DAI (Huhns published 1987 the first book in DAI) [5, 6, 12]. The series "Dezentralized Artificial Intelligence" edited by Yves Demazeau et al. contains the proceedings of the European Workshop on Modeling Autonomous Agents in a Multi-Agent World [1] held every year in a different city in Europe. Therefore these publications give a good overview of the ongoing research in Europe in the field of DAI.

Surveys of research can also be found in the following journals:

(Special Issue on Intelligent and Cooperative Problem Solving) International Journal of Intelligent & Cooperative Information Systems, vol.1, no. 2, June 1992

(Special Issue on Distributed AI) Group Decision and Negotiation, vol.2, no. 3, 1993

(Special Issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behaviour) International Journal of Intelligent Systems in Accounting, Finance, and Management, vol. 2, no. 4, 1993

An overview of current work in DAI can also be found in [13, 14]. The book of Müller [14] concerns on the german map of research. An annotated bibliography

[1] The title of the wokshops differs slightly over the years. For example in 1991: Modeling Autonomous Agents and Multi-Agent Worlds

is also given in [15], a bibliography indexed by application domains and authors could be found in chapter two of [6].

3 Cooperation and Communication

Cooperation and communication are basic aspects of DAI (among others like representation and reasoning about plans, recognizing and reconciling conflicting intentions) [6]. Bond and Gasser define coordination as "a property of interaction among some set of agents performing some collective activity" [6] whereas cooperation is stated as "a special case of coordination among nonantagonistic agents". In our case we will make use of a definition given by Demazeau and Müller: "Cooperation: in order to perform a personal task, an agent will have to cooperate with others either because it is not able to accomplish it itself (restricted possible solutions), or because others successfully accomplish it more efficiently (e.g. within a shorter interval of time)." [8] The field of communication deals with the specification of protocols, messages, and the kind of information exchange between agents. Different languages have been defined for DAI purposes so far.

3.1 Cooperation

Agents or distributed problemsolvers can coordinate in many different ways - they often have to cooperate to solve complex problems. One question now is why should they cooperate or what are the goals of cooperation? Durfee, Lesser and Corkill [16] state the following goals:

Performance: Working in parallel could improve the performance of the problem solving task. The grade of improvement depends on the structure of the problem to be solved.

Alternatives: Allowing agents to form local solutions without being influenced by others increases the variety of solutions.

Confidence: Agents can verify results of other agents, possibly using own data and problem solving strategies.

Fault tolerance: Assigning important tasks to multiple agents increases the probability of finding a solution, even if some agents fail.

Less effort: Letting agents recognize and avoid useless redundant activities reduces the amount of unnecessary tasks (computations).

Improvement: Permitting agents to exchange predictive information improves the general problem solution.

Less communication: The selection of types of messages to be exchanged reduces the amount of communication.

Balance computation: Agents that are allowed to exchange tasks between computational units can make better use of these resources through balancing the load.

Select agents: The individual agent expertise can be used better if the agents are able to exchange tasks with the purpose that the task is performed by the most capable agent.

Time: Coordinating activity with respect to the time agents are waiting for results from each other.

Cause these goals conflict with each other, agents could not achieve the goals simultaneously [16]. Depending on the task to be performed and the structure of the problem to be solved the concrete form of cooperation has to be determined for each system uniquely.

Cooperation can be done with or without communication. There exist different opinions whether communication is necessary or not - depending on the underlying definition of agents. Following Martial [17] communication is a necessity for coordination in the multi-agent planning area. Genesereth, Ginsberg and Rosenschein claim that "intelligent agents must be able to interact even without the benefit of communication" [18].

Cooperation without communication. Cooperation is no problem if there are common, non-conflicting goals valid for all agents. This is an optimistic point of view. In a world of limited resources conflicts have to be considered. The basic question is therefore how can two or more agents work together (achieve their goals) such that both (all) are able to reach their goals? Two approaches can be seen here: intuitive and the theory of games.

Intuitive: Rosenschein [19] states, that often people choose the same solution even if there are a lot of equivalent ones available. The assumption is that if there are multiple solutions every person chooses the one that seems to differ a little bit from the others. For example [11]: Two persons are sent in different rooms. Both have to split a staple of 100 pieces of paper (banknotes) in two staples. They know that, if they choose the same solution, every one will get a mercedes. Most people split the staple in two staples with 50 pieces each. The 50-50 solution is the only one where the two staples are equal. This seems to be the slightly different one from all other solutions.

Game theory: To solve problems together agents have to know that there are other ones, have to assume what these agents know and how they behave. They have to have common knowledge: knowledge of the payoff-functions and some sort of common knowledge about the behavior of the other agents (behavior in the context of selection of possible actions). Payoff functions represent the evaluation of the result of each course of action an agent can follow, this includes the assumption that every agent knows enough to evaluate the results of his possible actions [5].

Cooperation with communication. Cooperation using communication can also be seen as a way to reduce the amount of data being exchanged between agents. If agents cooperate they do not have to exchange information about their current state, how they solve a problem or doing a computational task.

Communication to provide cooperation is used as well to transmit some kind of meta-level information. For example: If someone agrees with another that he will phone Mr. X, he will not have to tell him, that he takes some coins out of the pocket, throw them into the slot, etc. The term "phone to" includes implicitly all these actions: meta-level information or knowledge. Three kinds of message transfer to support cooperation can be seen: actors, contract-nets and blackboards. The underlying communication has great influence how coordination can be realized in the system.

Actors. Actors have been defined first from Hewitt and Agha [22]. An actor is "a computational agent that carries out its actions in response to processing a communication" [22]. Therefore message-passing is used as the base for concurrent computation. Every actor has its own mail address and they work (carry out their actions) in parallel. The current state of an actor can be described by his internal variables, which can change over time, and their values. A protocol defines to which types of messages the actor reacts and in which manner. Possible actions of an actor are:

- send messages to other actors or to itself
- create actors
- change state or protocol of actor

The behavior of each mail-address (actor) can be described by specifying a local behavior function. The combination of a communication and the target (the mail-address) is called a task. An actor processes each message sent accordingly to his protocol. A message which type is not known by the protocol of the recipient (no method exists in the protocol for this type) will be rejected. To response to a message an actor may send different messages to other actors it knows. Reconfiguring the system of actors dynamically is possible through changing the mail-address of a communication.

Contract-net. The contract-net protocol was first introduced by Reid G. Smith [20]. A contract-net consists of a set of nodes (agents) which negotiate with one another using asynchronous message passing. Contracts exist between two nodes. Each agent can play different roles during the act of problem solving. These roles can be changed dynamically over time. Roles - also mentioned as classes of nodes - are:

Manager: He identifies tasks to be done, decomposes these tasks and distributes the subtask within the net. Monitoring the execution of the tasks and processing their results is also in his responsibility.

Bidder: This is a node that offers to perform a task.

Contractor: A successful Bidder. He is responsible for the execution of the task for which his bid has been accepted. He is allowed to split his ask and award contracts to others.

Communication between agents consists of different types of messages. The messages and the processing of each message are as follows:

Task announcement: A manager announces a task to be processed. The message consists of a description of the task, a bid specification and expiration time.

Bid: Bidders send this message to show their willingness and availability to process the announced task. The message contains a node abstraction showing the relevant capabilities of the node with respect to the announced task. The manager receives these bids and ranks them relatively. A contract is awarded to the bidder that has sent the most satisfactory bid.

Award: Manager sends the award message to the node with the successful bid. This node is now called contractor.

Acknowledgment: The contractor acknowledges the award to the manager. This acknowledgement can be either rejecting or accepting. Every node can bid to several announcements, therefore it is possible, that a bid sent is not longer valid if the award for this bid arrives.

Report: This message is used by a contractor to inform the manager about the current status of task execution (interim report) or the termination of a task (final report). The result of the execution is part of this message.

Termination: A manager makes use of this message type if the contractor should interrupt or terminate task processing prematurely. A contractor receiving this message stops task execution and all outstanding subcontracts.

Availability Announcement: A node that is idle and searching the tasks broadcasts this message giving specifications of tasks and own capabilities.

This protocol, defined through the type and processing of each message, shows the dynamic character of contract-nets. Cause of the direct agent-to-agent communication there is no need for an overall control structure allowing a finer degree of control than possible with traditional mechanisms [20].

Blackboards. The basic idea of blackboards was first published by Newell [21]. "Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it. This conception is just that of Selfridges Pandemonium: a set of demons, each independently looking at the total situation and shrieking in proportion to what they see that fits their nature." [21].Blackboards are a special kind of data structure, often separated in different levels or regions. Knowledge Sources (KS), as the independent processes are called, use this data structure as shared memory. KS write messages and partial results of their computations to the blackboard where other KS's are able to read them. The levels allow different representations or different levels of abstraction of a problem. Agents which work on different levels see the corresponding blackboard-level and their neighbor levels. Therefor generated data could be given to upper levels whereas goals can be written to lower levels of

the blackboard [7]. Often a control system is used to supervise the blackboard systems, for example to synchronize access or to avoid or recover deadlocks.

The different experts (agents in our case) monitor the blackboard that contains the current state of problem solving. Each agent recognizes if it is able to support the problem solving task, solves the specific subproblem if possible and writes back the result - which can be a solution of the problem or some decomposed subproblems. The following assumptions have to be made:

Independence of expertise: Every agent solves its problem without help of others. It does not even know that there exist other agents. Communication is only done using the blackboard. As a consequence it is easy to add or remove agents from the system.

Different problem solving techniques: Cause each agent behaves independently it could choose its own problem solving technique. Therefor it could use different methods for every subproblem.

Flexible representation: The form of problem representation can be defined different for every application.

3.2 Communication

After the decomposition and distribution of a problem and instantiating different tasks, one has to make considerations how agents interact and communicate to solve their subproblems. Following Gasser [23] DAI designers have to consider:

Unit of interaction: Problems can be decomposed in different levels of granularity. Talking about strategic goals needs other forms of communication than interchanging pure facts, for example machine dependent data or construction details.

Structures and processes of interaction: There exist different modes of interaction as there are marketplace transactions, forum-based discussion or master-slave relationship. Organizational structures influence interaction too.

Protocols and languages: How communication takes place is highly dependent on the protocols and languages used. Protocols describe the semantics and rules of the communication between agents. The language used for communication can have a small amount of message types and small vocabulary -therefor needing a highly structured syntax- or can be more flexible allowing dialogues with rich semantic. Another point we have to mention is the kind of information we want to communicate - the content of a message.

Demazeau and Müller distinguish among three types of information [8]:

Knowledge: Agents interchange knowledge to come to a consistent and complete (with respect to the problem to be solved) description of the current situation. Cause of their different points of view there could exist contradictory descriptions of the environment. In this case a decision has to be made which description matches better.

Possible solution: If agents have to agree to a common solution or plan they have to exchange their solutions. Cause of different problem solving capabilities and strategies there may be no common solution acceptable for all agents. In this case the agents have to exchange knowledge to find other solutions.

Choice and result: If there are multiple possible solutions agents have to negotiate about the "common" solution. Choosing the first avoids this problem, but is sometimes not optimal. Every agent makes his own ranking of solutions, where the high ranking of one agent may be rejected by others that do not accept this solution. In the case where agents offer tasks to others choice has to be communicated too. If there exists a common choice the selected result has to be transmitted.

We also have to think about how communication can take place in multi-agent-systems. Two basic principles have to be considered:

Shared memory: Every agent is able to write the information it wants to share with others to a specific place to which every agent in the system has access to. Blackboards, as mentioned above, make use of this principle. The agents do not know who uses this information and therefore there is no need for them to know the other agents (or their addresses) in the system. To avoid inconsistency some kind of access coordination is needed - like in operating or database systems. This control can be done explicitly by one separate agent, acting like a gate keeper, or implicit through the system using shared memory (some kind of access rule). One problem of blackboard systems is the dependency of the system from one specific node - the blackboard. If the processing unit running the blackboard goes down, the whole system is unable to work. One great advantage can be seen in the anonymous character of the information exchange. No agent needs to know the addresses of the others, simplifying the management of the communication. To remove or to add other agents to those systems is therefor easy.

Message passing: Communication can also take place using message passing. Agents communicate through receiving and sending messages to each other. To do this they have to know either the addresses of the other agent or the address of a port the designated agent is listening to. One port can be used from multiple agents. Contract nets and actor systems make use of this principle. The theory of speech acts [24] deals with the semantic of messages. It distinguishes between sending a message, the intention of the sender and the effects of the sending of a message to the environment. The last point is only visible through the behavior of the receiving agent that can be observed from outside. One problem of systems using message passing for communication is the reconfiguration of these systems (add or remove agents) cause new agents (their addresses) have to be inserted in the list of addresses of each agent which has to communicate with them (or one of them) resulting in lots of update activities. The same has to be done if an agent has been removed. On the other side the system is more fault tolerant. Communication is not dependant on one specific agent.

Clearly these two basic principles can be mixed. One possibility are teams of agents that use shared memory internally and message passing for communication between teams. Information exchange can also be done using mailboxes. Messages can be send to agents and roles. These roles can be played by different agent. A mailbox stores the messages addressed to roles. Agents performing the addressed role could read these messages later [25].

4 The Hierarchical Blackboard-System HBBS

Weiß introduces in his PhD-thesis a hierarchical, distributed blackboard system for concurrent engineering [7]. In the area of concurrent engineering experts of different disciplines often geographically distributed and using different tools have to work together. Standardization and unification of schemes for data and processes would lead to an overwhelming amount of work. Main issues of concurrent engineering are therefore communication and coordination of engineering agents. Two directions can be identified: DAI that deals among other things with the integration of human and artificial (computational) agents, and Computer Supported Cooperative Work (CSCW) which deals with interactions between humans. Table 2 illustrates the embedding of DAI.

Concurrent Engineering		
CAD	DAI	CSCW
Artificial agents	Human agents	

Table 2. Embedding of DAI

HBBS is designed as an infrastructure for the development of multi-agent-systems. The infrastructure should simplify the development and implementation of agents [26]. The knowledge and details of communication are separated from the agents and therefore no longer part of the agent definition task. Elements of HBBS are meta-agents that do not have, in contrast to agents that are connected to HBBS, any application knowledge. They only provide coordination knowledge. Each meta-agent consists of a blackboard, an agenda, a scheduler and designated "ambassadors", which themselves are represented as knowledge sources [27].

HBBS basically consists of four layers:

Application layer: This layer contains the whole application knowledge but no knowledge about control. Agents in this layer are all computational units

or humans, which communicate with meta-agents of the underlying control layer. From the point of view of the agents they work together via a common virtual working area; communication and HBBS are transparent for them. Information exchange between agents is defined by an interaction language, providing rules and syntax of messages. Agents that do not know the interaction language make use of front- end systems. These translate the internal knowledge representation of the agents to the format of the interaction language.

Control layer: This layer contains the meta-agents. Each knowledge source is defined by a condition part and an event action part. The condition part consists of a trigger that defines the goal the ambassador reacts to, a precondition, which has to hold after the trigger was successful, and a post-condition which holds after performing the designated action. The event part defines the action to take place if the condition part holds. There are two kinds of information exchange between agents and blackboards: the transmission of solutions to other blackboards and the transmission of goals to start some problem solving process on another blackboard. We can also distinguish between local and non-local actions, which effect remote blackboards.

Cooperation layer: This layer consists of mechanisms for the coordination of agents and meta-agents, and the construction of hierarchical multi-blackboard structures. If the agenda contains goals, the one with the highest priority will be selected and the condition parts of all knowledge sources will be tested. If one holds the corresponding action part will be executed. As the result of an action local goals could be generated which will be inserted into the agenda. If the agenda is empty, the meta-agent waits for incoming messages that may contain a goal. To avoid deadlocks meta-agents are allowed to wait for determined messages or events and to store events, which should not be handled in a buffer, where they have to wait until some lock is raised.

Hierarchies of meta-agents can be reached by exchanging agents with meta-agents. Knowledge sources make no distinction between the different kind of agents. Therefore agents can be replaced by whole blackboard- systems easily [7].

Communication layer: The communication layer is responsible for the physical communication. To support naming and location transparency every agent is known only by his logical name. The mapping of logical names to physical addresses is done using a central name database (also called configuration database). Physical communication is based on TCP/IP.

Phillip and Weiß [26] implemented a prototype for distributed MPC-systems. Manufacturing planning is done in units of manufacturing orders. Machine orders are the units of manufacturing control. In their model similar manufacturing machines, those which are able to perform the same machine orders, form machine groups. Planning and control is done using well known, traditional algorithms.

For each machine there exists one machine agent. It gets orders from the higher level machine group blackboard. This blackboard receives orders for the

machine group and organizes the cooperative work of the machine agents to perform the tasks. Every machine group blackboard is supported by a scheduling agent that schedules the different orders. The next higher level is build by the machine order blackboards and blackboards responsible for transport, human resource planning and inventory. To have access to these resources the machine order blackboard creates corresponding orders. Manufacturing planning is done by material requirements planning and capacity planning agents. The resulting orders are given to the manufacturing orders blackboard. There the orders are decomposed by another agent and scheduled, resulting in machine orders which are forwarded to machine order agents.

Cooperation and competition take place in this prototype. Machine, transport, and scheduling agents have to cooperate to fulfill their tasks (the right material at the right time on the right place). Material requirement planning and capacity management agents cooperate to formulate valid manufacturing orders and schedules. Competition exists between machine order blackboards cause of limited resources. Every blackboard wants to solve its own problems, leading probably to deadlocks in the manufacturing system. The prototype uses queues to avoid deadlocks.

References

1. Martin, Hans: Auswirkungen auf die Arbeitssituation, in: Geitner, Uwe W. (ed.): CIM-Handbuch, Braunschweig, Vieweg 1991, pp. 645-652
2. Kurbel, Karl: Produktionsplanung und -steuerung, München ..., Oldenburg 1993
3. Lenke, Christian: Zukunftsorientierte Konzepte zur Produktionsplanung und -steuerung auf der Basis moderner Informationstechnologien und flexibler Fertigungsstrukturen, Diplomarbeit, Universität Essen 1994
4. Beckenbach, Niels; van Treeck, Werner: Betriebliche und soziale Auswirkungen, in: Geitner, Uwe W. (ed.): CIM-Handbuch, Braunschweig, Vieweg 1991, pp. 653-661
5. Huhns, Michael N. (ed.): Distributed Artificial Intelligence, Los Altos, Morgan Kaufmann 1987
6. Bond, Alan H.; Gasser, Les (eds.): Readings in Distributed Artificial Intelligence, San Mateo, Morgan Kaufmann 1988
7. Weiß, Michael: HBBS: Ein hierarchisches Blackboard-System für den Verteilten Entwurf, Dissertation, Universität Mannheim 1993
8. Demazeau, Yves; Müller, Jean-Pierre: Decentralized Artificial Intelligence, in: Demazeau, Yves; Müller, Jean-Pierre (eds.): Decentralized A.I.: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Cambridge, England, August 16-18, 1989, Amsterdam, North-Holland 1990, pp. 3-17
9. von Martial, Frank: Coordinating Plans of Autonomous Agents, Berlin ..., Springer 1992
10. Sridharan, N. S.: Workshop on Distributed AI, AI Magazine, Fall 1987, pp. 75-85
11. Fischer, Klaus: Verteiltes und kooperatives Planen in einer flexiblen Fertigungsumgebung, Sankt Augustin, Infix 1993
12. Gasser, Les; Huhn, Michael N.(eds.): Distributed Artificial Intelligence II, Los Altos, Morgan Kaufmann 1989

13. Avouris, Nicholas M.; Gasser, Les (eds.): Distributed Artificial Intelligence: Theory and Praxis, Dordrecht ..., Kluwer Academic 1992, pp. 9-30
14. Müller, Jürgen (ed): Verteilte Künstliche Intelligenz - Methoden und Anwendungen, Mannheim ..., BI-Wissenschaftsverlag 1993
15. Jagannathan, V.; Dodhiawala, Rajendra: Distributed Artificial Intelligence: An Annotated Bibliography, in: [5], pp. 341-390
16. Durfee, Edmund H.; Lesser, Victor R.; Corkill, Daniel D.: Cooperation Through Communication in a Distributed Problem Solving Network, in [5], pp. 29-58
17. von Martial, Frank: Planen in Multi-Agenten Systemen, in: [14], pp. 92-121
18. Genesereth, Michael R.; Ginsberg, Matthew L.; Rosenschein, Jeffrey S.: Cooperation without Communication, in [6], pp. 220-226
19. Rosenschein, Jeffrey S.; Kraus, Sarit: The Role of Representation in Interaction: Discovering Focal Points among Alternative Solutions, in: Steiner, Donald D.; Müller, Jürgen (eds.): MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on "Modeling Autonomous Agents and Multi-Agent Worlds", Kaiserslautern, August 5-7, 1991, pp.
20. Smith, Reid G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, in [6], pp. 357-366
21. Newell, A.:Some Problems of Basic Organization in Problem Solving Programs, in: Yovits, Jacobi, Goldstein (eds.): Proc. Conference on Self Organizing Systems, Washington, Spartan Books 1962, pp. 393-423
22. Agha, Gul; Hewitt, Carl: Concurrent Programming Using Actors: Exploiting Large-Scale Parallelism, in: [6], pp. 398-407
23. Gasser, Les: An Overview of DAI, in: [13], pp. 9-30
in: Müller, Jürgen: Beiträge zum Gründungsworkshop der Fachgruppe verteilte Künstliche Intelligenz, Saarbrücken, 1993, DFKI Document D-93-06, pp. 77-89
24. Austin, J. L.: How to do things with words, Oxford, University Press 1962
25. Tomalla, Alf: Verteilte Künstliche Intelligenz in der Produktion, Diplomarbeit, Universität Essen 1994
26. Philipp, Mathias; Weiß, Michael: Ein hierarchischer Blackboard-Ansatz für Verteilte PPS-Systeme, in: Müller, Jürgen: Beiträge zum Gründungsworkshop der Fachgruppe verteilte Künstliche Intelligenz, Saarbrücken, 1993, DFKI Document D-93-06, pp. 77-89
27. Weiß, Michael: HBBS: Object-Oriented Design of a Distributed Blackboard Kernel, in: Engineering SEKE-93, San Francisco, IEEE Press, 1993, pp. 285-287

This article was processed using the LaTeX macro package with LLNCS style