INT. J. PROD. RES., 2004, VOL. 42, NO. 2, 303–320

Taylor & Francis
Taylor & Francis Group

MANPro: mobile agent-based negotiation process for distributed intelligent manufacturing

MOONSOO SHIN† and MOOYOUNG JUNG†*

This paper examines negotiation procedures in an agent-based distributed shop floor control system (SFCS). A distributed SFCS is under a heterogeneous environment, which is controlled through negotiations between autonomous agents. The negotiation-based control can be considered as the core of a distributed control paradigm. An efficient information exchanging mechanism and an information model with reasonable structure are indispensable for effective negotiations. This paper proposes a novel negotiation mechanism, called a mobile agent-based negotiation process (MANPro), which applies a mobile agent system to the process of information exchange. Since using mobile agents allows each component to execute asynchronously and autonomously and to adapt dynamically to the execution environment, MANPro may guarantee autonomy of agents. Moreover, it is possible to build a fully distributed and autonomous SFCS by using MANPro. MANPro is based on the agent-based control architecture, which includes a communication architecture and an information architecture. The communication architecture provides the exchanging mechanism of information, defining functional modules to support the mechanism while the information architecture provides the framework for information modelling on negotiation, proposing information models required for introducing the ontology concept.

1. Introduction

The rapidly changing environment with high product variety and short life cycles renders a dynamic adaptive control of manufacturing systems indispensable. The shop floor control system (SFCS) must be able simultaneously to respond to changes of the internal and external shop status. Moreover, it must not only be able to continue to operate even though one or more of its components are out of control but also it must have the capability of adding new components to the system without major changes in the existing components. However, no conventional SFCS based on centralized or hierarchical control architecture can handle such situations. Therefore, the control architecture of an SFCS is gradually being shifted to the distributed control architecture. Since the distributed SFCS may have complete local autonomy, governing the re-configurability, scalability, as well as fault-tolerance, it is suitable for a dynamically changing environment. Next generation manufacturing systems such as the Bionic manufacturing system (Okino 1993, Ueda 1993), the fractal manufacturing system (Warnecke 1993), and the holonic

Revision received May 2003.

†Advanced Product & Production Technology Centre, Department of Industrial Engineering, Pohang University of Science and Technology (POSTECH), Hyoja San 31, Pohang 790-784, Korea.

*To whom correspondence should be addressed. e-mail: myjung@postech.ac.kr

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online © 2004 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/0020754032000123597

manufacturing system (Valckenaers *et al.* 1994) are based on the distributed SFCS. Over the past few years much research in agent-based approaches to these systems has been conducted (Park and Lee 2000, Ryu *et al.* 2001). Agents are considered to provide a suitable paradigm for designing intelligent manufacturing systems to enhance flexibility and re-configurability.

In the agent-based distributed SFCS, each agent negotiates with others to pursue the system goals. Since each agent has only a local view, it should take a global view of the system by exchanging information with others. Therefore, an efficient negotiation mechanism is indispensable in building agent-based distributed SFCSs. However, the current state is not regarded as a satisfying settlement yet because there is no apparent standard for communications among agents to support intelligent and efficient network communication. Even though many researchers have been interested in this problem and have proposed several negotiation mechanisms and protocols, more sophisticated negotiation mechanisms and protocols are necessary (Shen 2002).

This paper focuses on negotiation among fully distributed agents of distributed agent-based SFCSs and proposes a novel negotiation mechanism; namely, the mobile agent-based negotiation process (MANPro). MANPro is based on the agent-based control architecture and includes a communication architecture and an information architecture. The communication architecture provides the mechanism for information exchange that applies a mobile agent system to the process of information exchange. Moreover, the communication architecture defines functional modules to support the mechanism. The information architecture provides the framework for information modelling on negotiation, introducing the ontology-based information model.

2. Negotiations in agent-based distributed SFCSs

In an agent-based distributed SFCS, manufacturing resources such as machines, transportation systems, and buffering systems are treated and represented as autonomous agents, and a negotiation process is initialized for cooperation among these agents. Figure 1 shows the framework of an agent-based distributed SFCS. In this framework, the shop works through autonomous negotiation among resource control agents, and the process plan and the production plan are generated from the external systems such as a CAPP system and a production planning system. Each resource control agent has a local database system, which controls a corresponding device. The shop coordinating agent mediates conflicts between resource control agents to enhance the interoperability of the shop.

2.1. *Contract Net Protocol (CNP)*

A negotiation process can be considered as a serialized process composed of announcing, bidding and awarding phases. The contract net protocol (CNP) (Smith 1980, Smith and Davis 1981) defines the basic messages according to these phases and describes the processing of each message. Once an order is released to the shop floor, the manager agent of the order becomes the initiator of negotiation, and a negotiation process is initialized. Conventionally, in the announcing phase, the manager broadcasts a task into the network. Then, every controller bids for the task in the bidding phase. After the manager collects and evaluates all the bids, the task is awarded to the best bidder in the awarding phase.

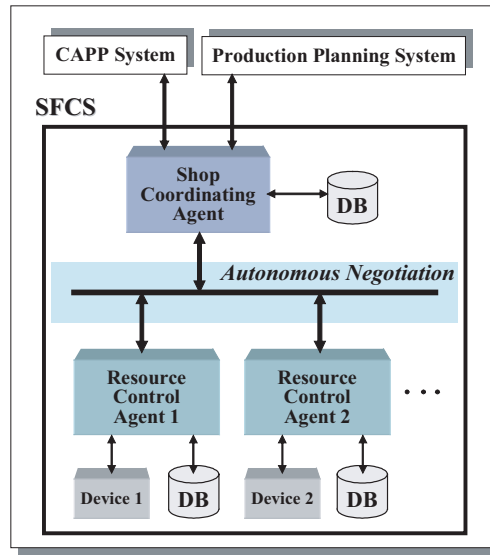

Figure 1. The framework of an agent-based distributed SFCS.

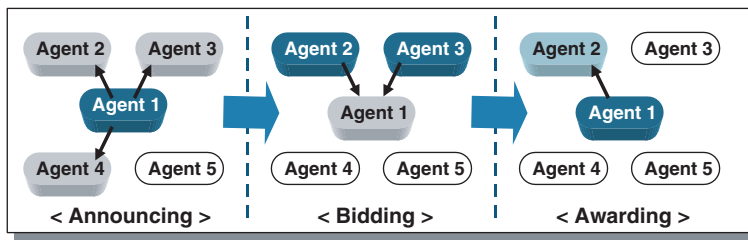

Figure 2. Information flow in bidding-based negotiation.

The CNP, which is a well-known protocol and a pioneering work in this research area, is quite simple. However, since the increase in the number of agents makes the number of messages on the network increase, agents may, in the worst case, spend more time processing messages than doing the actual task (Shen 2002). Figure 2 shows the information flow in a modified CNP-based negotiation where the initiator, Agent 1, sends announcing messages to selected agents instead of broadcasting them.

2.2. *Bidding mechanisms*

Negotiations can be classified into the following three groups in terms of a bidding mechanism (Shen 2002): (1) part-oriented (Lin and Solberg 1992); (2) resource-oriented (Baker 1991, Butler and Ohtsubo 1992); and (3) bi-directional. The part-oriented bidding mechanism and the resource-oriented bidding mechanism are one-side bidding approaches. In the part-oriented bidding, a part agent generates bids with reference to part information such as process plan, production plan, etc., and submits the bids to the resources. On the other hand, in the resource-oriented bidding, machine control agents generate a bid based on resource capability to a part order. The initiator agent selects a contractor by comparing bids that participant

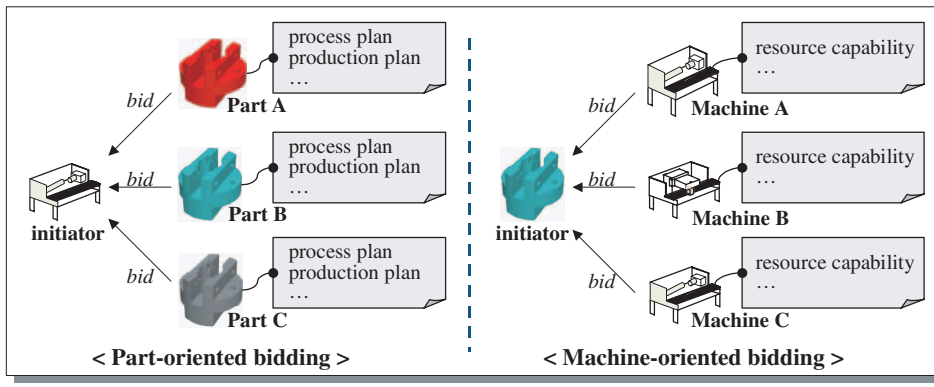

Figure 3. The bidding mechanism of part-oriented and machine-oriented negotiation.

agents generate with the given policy. Figure 3 illustrates the part-oriented bidding mechanism and the resource-oriented bidding mechanism.

The bi-directional bidding mechanism makes it possible to generate bids from the viewpoints of both the parts and the resources. While the bi-directional bidding mechanism is more similar to human-based negotiation and more reasonable than one-sided bidding approaches, it is more complicated to design than one-sided bidding approaches, and it requires extremely high overheads because all the participants must have the ability of generate bids. In order to induce reasonable and effective results, each bidder must have the intelligence to generate or analyse bids. Therefore, it is necessary to develop a novel mechanism not only to support bi-directional bidding but also to reduce the overheads. Moreover, all the participants, including the initiator, are required to have intelligence to resolve potential problems and to cope flexibly with the rapidly changing environment.

2.3. *Communication architecture*

The communication architecture defines a mechanism by which each agent collects information. Since a well-designed negotiation mechanism requires an efficient flow of information, the ability of the communication architecture to support the efficient flow of information has a critical effect on the negotiating performance. In the open literature, the following three types of communication architectures have been proposed (Kádár *et al.* 1998): (1) blackboard-based architectures, (2) message passing-based architectures, and (3) hybrid architectures.

In a blackboard-based architecture, each agent interacts and shares information through a blackboard (Ito and Salleh 2000). A blackboard is a common database on which all associated agents write messages, post partial results, and obtain information. However, the centralized control of communicating agents disturbs the autonomy of each agent, and a common database system conflicts with the philosophy of a distributed manufacturing system. Thus, conventional message-passing based architectures using a local database system and a hybrid architecture that combines the message-passing based architecture with the blackboard-based architecture, have been generally employed in distributed manufacturing systems.

In a message passing-based architecture, autonomous agents with an individual knowledge source converse with other agents by exchanging messages. This is comprehensible because of its similarity to actual negotiations in the real world.

However, information exchange by message-passing has limitations in its ability to express information. Since an agent is just a type of programmed machine, it has no ability to understand natural language. For agents to understand exchanged messages, the structure and content of messages must be predefined. Therefore, each negotiating system needs a protocol specific to the application domain. However, since no negotiation protocol can actually cover all types of messages, the negotiation system is restricted in its ability to express the content of information. Moreover, integration of the systems using variant protocols needs to unify protocols, and even the revision or extension of a single negotiating system needs to modify the existing protocol. These factors have a negative influence on the reconfiguration of manufacturing systems. The current rapidly changing environment necessitates building a negotiating system that can dynamically adapt to the changed environment and can combine with other systems without any difficulty. Applying a mobile agent to the communication architecture makes it possible to build a negotiating system with these properties.

3. Mobile-agent based negotiation

A mobile agent is not bound to the system where it begins execution but is free to travel through a network. It is able to transport itself freely from one system to another. There are several distinctive reasons for using a mobile agent (Lange and Oshima 1998): (1) it reduces network load, (2) it overcomes network latency, (3) it encapsulates protocols, (4) it executes asynchronously and autonomously, (5) it adapts dynamically, (6) it is naturally heterogeneous, and (7) it is robust and fault-tolerant.

These features are useful to the negotiation process. Problems related to communication load, which is a critical deficiency of the classic message passing-based negotiation, can be overcome by using the mobile-agent based negotiation. When many messages have to be exchanged, it is more efficient and reliable to send an agent to a target system and perform the job locally than to communicate with the system by message passing. Since a mobile agent can move by itself to a system that contains an object with which the agent wants to interact, it can obtain information on a local knowledge source without network communication. Figure 4 represents the difference in the information flow between the classical message-passing based negotiation and the mobile-agent based negotiation.

Moreover, mobile-agent based negotiation makes it possible to build a fully distributed SFCS. Since the mobile-agent based negotiation allows participant agents to execute asynchronously and autonomously and to adapt dynamically to the execution environment, subordinates of an SFCS can take out a guarantee on autonomy and self-reliance. Each subordinate need not depend on other systems, and nor is it affected by the status change of other systems. These properties make an SFCS not only insensitive to a change of the shop layout but also make it easy to reconfigure the layout.

When data are exchanged in a distributed system, a definite protocol that defines grammar is required properly to encode outgoing information and interpret incoming information. However, it is impossible for a protocol to comprehend overall situations that may happen on the shop floor because a protocol is dedicated to a specific scenario of communication. Therefore, adjustment of a predefined protocol to new requirements is indispensable. However, since mobile agents have intelligence and internally encapsulate protocols, they can not only adapt to dynamically

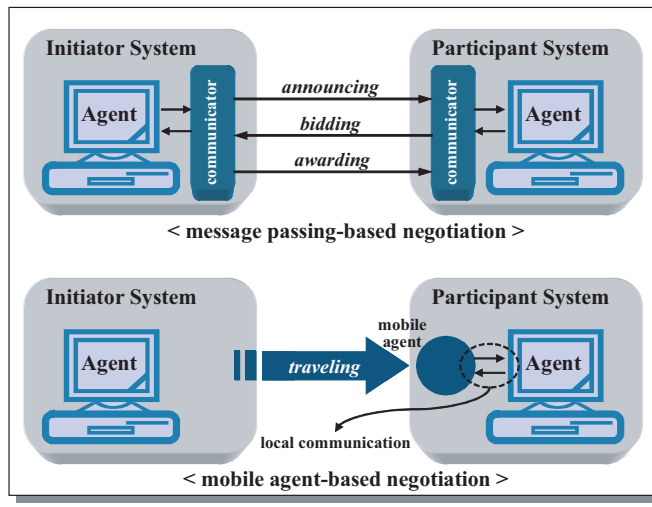

Figure 4. Information flow in the message passing-based negotiation and the mobile-agent based negotiation.

changing environments but also accommodate to new requirements of a predefined protocol without difficulty. In other words, adjustment of a predefined protocol in a mobile-agent based negotiation is easier than that in a message-passing based negotiation. Moreover, a mobile agent sequentially inspects information and generates bids, so that it can use prior knowledge to generate a new bid. The agent need not inspect unnecessary information judged from prior knowledge. Therefore, the overall search space is reduced.

4. Mobile-agent based negotiation process (MANPro)

This paper proposes a novel negotiation mechanism based on the mobile agent for agent-based distributed SFCSs, namely, the mobile agent-based negotiation process (MANPro). MANPro defines a communication architecture and an information architecture for autonomous negotiation among agents. The communication architecture provides the mechanism for information exchange using a mobile agent. The mobile agent encapsulates several functions required for streamlining negotiations. Moreover, the communication architecture defines functional modules to take charge of actual processes for negotiation. The information architecture provides the framework for information modelling on mobile-agent based negotiation, and it introduces an ontology-based information model. Moreover, the information model employs a data format compatible with the extensible mark-up language (XML) to enhance interoperability between subordinates.

4.1. *MANPro: communication architecture*

The communication architecture in the MANPro involves a mobile agent encapsulating functions such as information acquisition and bid generation in the negotiation process. The mobile agent travels the network of participant agents, collects information, and generates bids. Moreover, this architecture introduces agents representing tasks and resources and a coordinating agent for distributed

problem solving. Decomposition into tasks and resources makes it easy to implement the bidding mechanism so that each agent bound to a task or a resource may generate bids. However, a coordinating agent is indispensable, because distributed problem solving may have conflicts between distributed modules. The coordinating agent systematically supervises the overall negotiation process to coordinate interactions between agents.

The communication architecture in the MANPro defines the following four types of functional modules: (1) bid manager (BM); (2) task agent (T-agent); (3) negotiation agent (N-agent); and (4) resource agent (R-agent). The BM is a kind of coordinating agent, and it supervises the overall negotiation processes on the shop floor by managing initialization of the negotiation process. The T-agent is bound to a task, and it manages one negotiation process. Each task is an atomic operation that is executed by only one resource, and it is the issue of one negotiation process. The BM and the T-agent are located in a virtual controller, which is a virtual environment helping agents to perform their own operation. The N-agent, which executes an actual negotiation process, is a kind of mobile agent, and it travels the network of resource controllers to collect information and generate bids. The R-agent is bound to one resource of a shop floor and essentially plays the role of a resource controller. Moreover, it controls the access of N-agents and gives N-agents the interface with a resource database. Figure 5 illustrates the functional modules and information flow between them.

This paper assumes a situation of resource-oriented bidding. Each T-agent manages a negotiation process from the viewpoint of a part, and R-agents call for bids as latent contractors. That is, a T-agent negotiates with several R-agents to solve given problems related to a part order. However, it is also possible to perform bi-directional bidding as well as part-oriented bidding by involving a T-agent in a resource.

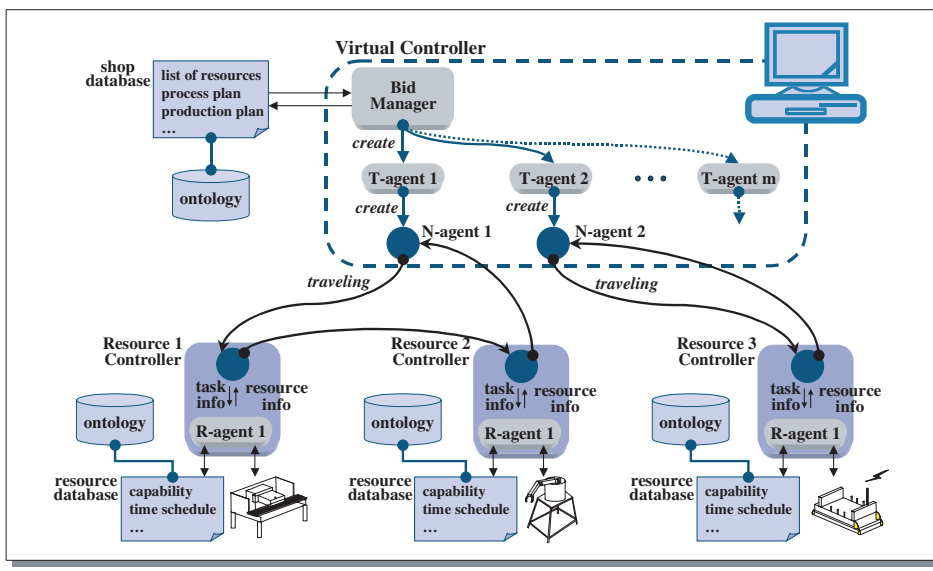

Figure 5. Framework of MANPro.

4.1.1 *BM*

The core role of the BM is to coordinate the overall negotiation among agents by supervising the initialization of the negotiation processes for systematic problem solving. The BM controls the creation of T-agents to manage the initialization of a negotiation process. Moreover, for efficient and systematic negotiation, the BM schedules the initialization time of the negotiation processes between distributed and autonomous agents and constructs a T-agent to initialize the negotiation process.

In an event that requires a negotiation among agents, the BM gets ready to create a T-agent. However, before creating a T-agent, it must consider whether any R-agent related to the negotiation participates in an ongoing negotiation process because if one R-agent negotiates with several T-agents at the same time, a conflict may arise. If one R-agent participates in several negotiation processes at the same time, the same or the overlapped bid can be proposed to several T-agents at the same time. If several T-agents select the bid, a conflict occurs. For example, in a task allocation problem, several tasks can be dispatched to the same time interval of one resource. Although it can be considered as an alternative policy to prevent one from proposing the same bid to several T-agents at the same time, it is not efficient. The previously proposed bid cannot be proposed to another T-agent until the corresponding negotiation process is terminated, although it is not accepted. Therefore, multiple participations in negotiation make the performance system to go down, and each R-agent must not participate in more than one negotiation process at the same time. R-agents participating in an ongoing negotiation process are registered at the shop database, and the BM refers to the information before creating a T-agent.

In terms of the supervising role, it seems that the BM is similar to the central controller of a centralized control system. However, since the BM engages only in the initialization of the negotiation processes, it has an obvious distinction from the central controller that supervises all operations of an SFCS and solves all the problems. The BM is internally composed of the following three types of functional modules, as shown in figure 6: negotiation scheduler, T-agent creator, and database manager.

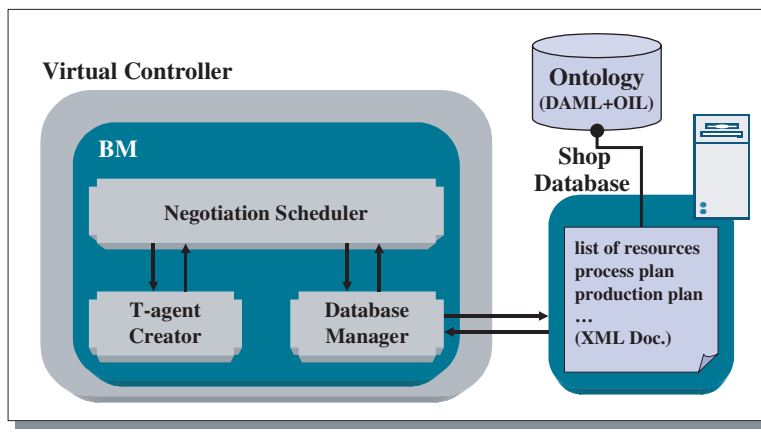

Figure 6. Functional architecture of the BM.

Negotiation scheduler. The function of the negotiation scheduler is to determine which negotiation process is initialized. It is the core function of the BM, which coordinates the negotiation process by supervising the initialization of the negotiation process. Therefore, the negotiation scheduler can be considered the core module of the BM. The negotiation scheduler makes a schedule for the initialization of the negotiation processes. Negotiation processes are initialized and progressed according to the schedule that is generated by the negotiation scheduler. Generally, there are several concurrent events that need negotiation processes between distributed systems in a shop floor. Without a systematically arranged schedule, the overall negotiation processes can become complicated. Moreover, the timing when a negotiation process is initialized can be considered as a principal factor that determines the overall performance of the system. In the real-time system, such as a real-time scheduling system, its effect is critical. If a negotiation process is not initialized at the appropriate time, the solution to a given problem that needs the negotiation process can be delayed, and a task cannot be performed at the requested time. To supervise these negotiation processes, a negotiation scheduler must systematically make a decision on the time for initialization of negotiation processes.

T-agent Creator. The T-agent creator generates a T-agent that is bound to a task when a negotiation process is initialized according to the negotiation schedule. It encapsulates task information into the T-agent.

Database Manager. The database manager manages the shop database that registers all resources in the shop floor and stores information about part orders such as process plan and production plan. When the negotiation scheduler wants to examine whether to initialize a negotiation process, the database manager scans the shop database and reports the information to the negotiation scheduler. Moreover, it supplies sub-modules with required information, such as part information.

The database systems in the MANPro use an ontology-based information model, and the data format is compatible with XML. For example, the shop database preserves information on an operation sequence converted into an XML document based on ontology, and the database manager interprets the information through an XML-parsing system. Since XML can represent the web-based domain ontology, it not only reinforces interoperability between various systems but also makes it possible to implement the web-based distributed SFCS. Since the database does not store information to cover the overall shop floor, and since the overall information disperses in physically distributed systems, it does not have an adverse effect on the autonomy of each subordinate. The information model of the database system is detailed in the information architecture.

4.1.2. *T-agent*

Each T-agent encapsulates information on one task, which is the issue for negotiation. It has the information needed in solving problems such as the definition of the problems and the list of a subject for negotiation, etc. That is, the T-agent knows 'what the goal of the negotiation is' and 'who the partner of the negotiation is'. The T-agent manages one negotiation process at a time according to such information. Moreover, the initialization of one negotiation process is executed by the creation of one T-agent, and the negotiation process is completed with the destruction of the T-agent. The

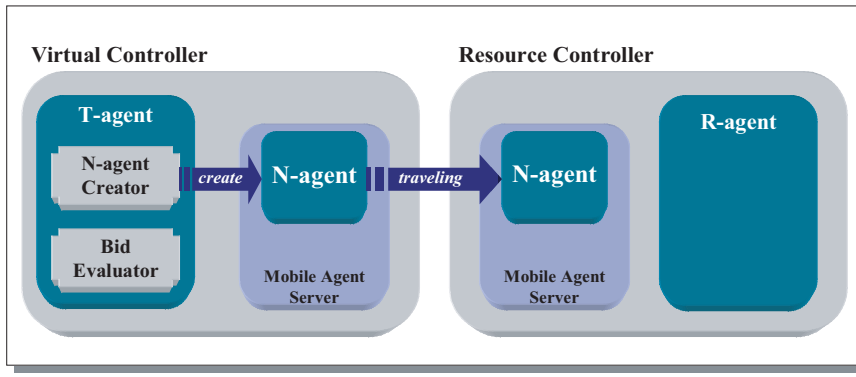

Figure 7. Functional architecture of the T-agent.

main function of the T-agent is to create an N-agent and to evaluate bids, and it has two functional modules, as shown in figure 7: N-agent creator and bid evaluator.

N-agent creator. The N-agent creator generates an N-agent that has a routing schedule, and it initializes travel-to-target resources that participate in the negotiation in order to collect bids for resolving problems. An actual negotiation process begins with the creation of an N-agent.

Bid evaluator. The bid evaluator plays the role of evaluating collected bids that are reported by the N-agent, which is the core function of the T-agent, and selects the best bid according to the given evaluation policy. Moreover, it makes the task offer and sends the awarding message to the R-agent that proposes the selected bid. When the T-agent receives the acceptance message from the R-agent that receives the awarding message, the T-agent destroys itself.

4.1.3. *N-agent*

The N-agent travels the network of resource controllers that participate in the negotiation with a routing schedule, and it collects bids of the resource controllers. Since the N-agent is a kind of mobile agent, it is capable of travelling between remote systems. Moreover, the virtual controller and resource controllers must have the mobile agent server, which is the environment in which agents operate and a kind of operating system for mobile agents, as shown in figure 7. For problem solving, the N-agent inspects the status of resources, and it generates bids while it travels from a mobile agent server of one resource controller to the mobile agent server of another resource controller. Moreover, when it completes travelling according to the routing schedule, it reports the bid list to the T-agent and destroys itself. A negotiation process proceeds with the travelling of the N-agent. In MANPro, since a bid of each resource is generated by the N-agent created by the issuer, the N-agent has the ability to generate a specific bid according to the objective of the negotiation process, and the participants in the negotiation only offer necessary information. The participants need not take care of the negotiation issue. The N-agent has the following three types of functional modules, as shown in figure 8: bid generator, communicator, and pre-evaluator.

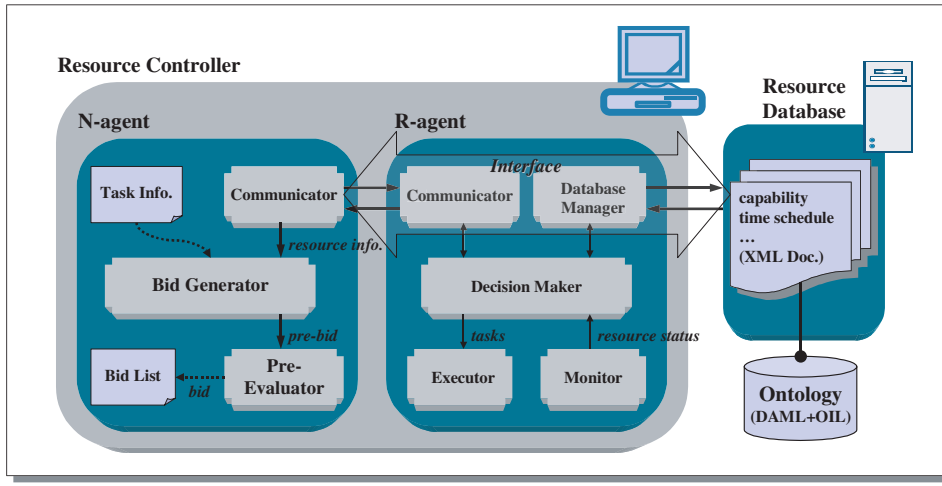

Figure 8. Functional architecture of the N-agent and the R-agent.

Bid generator. The bid generator module generates a bid for a given task according to the given bidding policy. In the case of a scheduling problem, it makes a schedule based on the status of a resource. When an N-agent arrives at a resource controller, the N-agent takes information on the status of corresponding resource through the communicator module, and the bid generator module makes a schedule for a given task. For example, the bid generator module finds the time interval that satisfies the timing constraints of a given task and a given scheduling policy out of an unassigned time interval. The time interval becomes the bid of the resource. To take the result with high quality through the negotiation process, the bid generator module needs intelligence and adaptability. The performance of the bid generator module can be considered as the factor that determines the performance of the overall negotiation process.

Communicator. The communicator module starts a conversation with an R-agent to inspect the status of a resource in the local area of a resource controller. Since the N-agent is transferred from the external system, it should not directly access the resource database from the point of view of security. The N-agent must pass the access control of the R-agent, and it takes information through the interface supplied by the R-agent. Moreover, since the N-agent visits resource controllers using a diverse information model, it must be able to translate diverse information models into a target model.

Pre-evaluator. The pre-evaluator module filters out generated bids in order that all participants do not have to wait for the response of the T-agent. It is not efficient that although a generated bid has low performance in comparison with other bids, the resource corresponding to the bid could not participate in another negotiation while waiting for a response from the T-agent. Therefore, the N-agent adds only the bid that passes pre-evaluation in the bid list. When the pre-evaluator module completes pre-evaluation of a generated bid, the N-agent travels to another resource controller according to the routing schedule.

4.1.4. *R-agent*

The R-agent is bound to one resource of a shop floor, and it essentially plays the role of a resource controller. Moreover, it controls the access of N-agents, and it offers the N-agent interface with the resource database. If it receives the awarding message for the generated bid, it sends the task acceptance message to the T-agent and records corresponding information in a resource database. Since the N-agent generates a bid for ongoing negotiation, the R-agent need not be concerned with the negotiation itself. In the proposed negotiation scheme, the R-agent only gives necessary information to the N-agent under access control. It is internally composed of the following five types of functional modules, as shown in figure 8: communicator, database manager, decision maker, executor, and monitor.

Communicator. The communicator offers the N-agent interface with resource database under access control. If the N-agent requests information on the status of resource, the communicator gives the N-agent accessibility to the corresponding information after a security inspection.

Database manager. The database manager manages the local database of the resource controller that stores XML-based information about the status of resource and pre-assigned process plans, etc. If an R-agent receives the awarding message with a task offer, the database manager adjusts the corresponding XML documents in the local database.

Decision maker, executor and monitor. Fundamentally, the R-agent is the controller of resources on the shop floor. The decision maker module receives a perceived situation from the monitor module, determines what to do with it and transmits it to the executor module. The executor module sends a control message to the resource according to the schedule stored in a resource database, and the monitor module observes the status of the resource. If a resource cannot take the tasks reserved because of a breakdown, the R-agent sends the resource fault message to the BM to reschedule the cancelled tasks.

4.2. *MANPro: information architecture*

A distributed SFCS is composed of diverse systems. Subordinate systems are based on the vendor specific control architecture and information architecture, and they can be different from each other in terminology. However, integration of diverse terminologies into a standard terminology with mutual agreement needs hard work, and it is actually not feasible to establish a standard model to cover all vendor-specific information models. Therefore, it is necessary for an interchanging mechanism to bridge various information models in the heterogeneous environment. Since the interchanging mechanism can dynamically translate an information model to a target model, and since it can allow an information model to be freely designed, it is more efficient to build an interchanging mechanism rather than establishing a standard model.

The information architecture in the MANPro proposes the ontology-based information model, in which each terminology is translated on the basis of the ontology. Ontology defines a reusable and extendable concept library (Li *et al.* 2001). It can be used to declare explicitly the knowledge embedded in applications, and it can help agents to extract information from diverse systems. Ontologies built

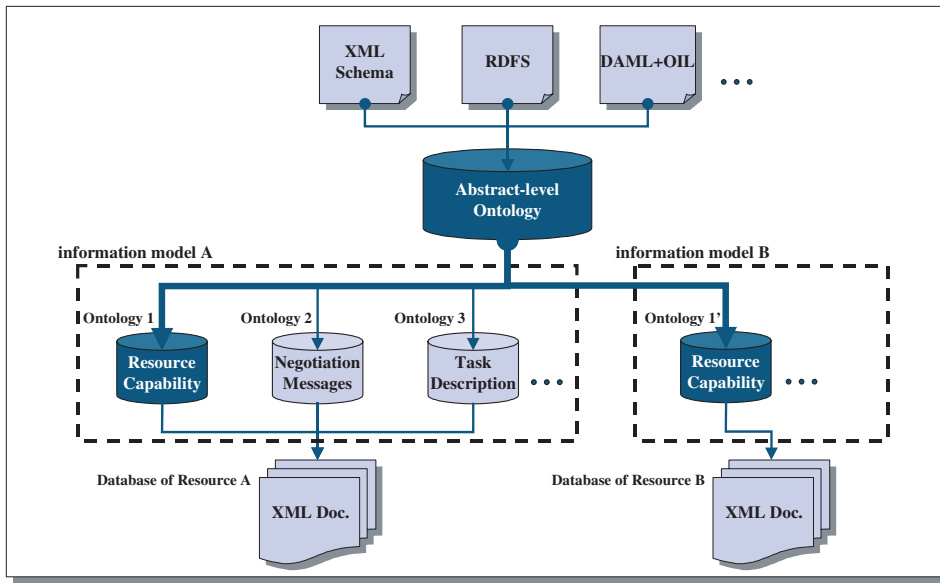

Figure 9. The ontology-based information architecture using DAML + OIL.

with a reasonable structure make it possible to represent semantic information, which is useful for the heterogeneous data exchange (Gómez-Pérez and Corcho 2002). Figure 9 shows the proposed ontology-based information architecture. One information model includes several ontologies to represent specific knowledge, such as resource capability ontology, negotiation message ontology, task description ontology, etc. Several ontologies can be inherited from the same ontology, and they can be translated into others by using a semantic relationship. Therefore, each resource controller uses diverse information models but, if the information model is based on a predefined abstract-level ontology, it is possible to represent the semantic information of the model, and the model can be translated into other information models based on the same abstract-level ontology.

Over the past few years, several ontology languages have been developed for building ontologies, such as Resource Description Framework (RDF) (Lassila and Swick 1999) and RDF Schema (RDFS) (Brickley and Guha 2002). Moreover, DAML + OIL (Harmelen *et al.* 2001) – the union of DARPA Agent Mark-up Language and Ontology Inference Layer – is built on RDF and RDFS. They are based on XML syntax. DAML + OIL provides not only richer expressiveness but also richer semantics for an information model than do RDF and RDFS because it has underlining model-theoretic semantics. Furthermore, it is able to express classifications by inference rather than by explicit listing (Ouellet and Ogbuji 2002). The ontology-based information modelling using DAML + OIL can supply the bridge between diverse information models by semantic-based inference. The information model of the proposed information architecture employs a data format compatible with XML based on DAML + OIL.

4.3. *Negotiation process*

The negotiation process proposed in this paper applies a mobile agent system to exchanging information. A negotiation process has a life cycle according to creation,

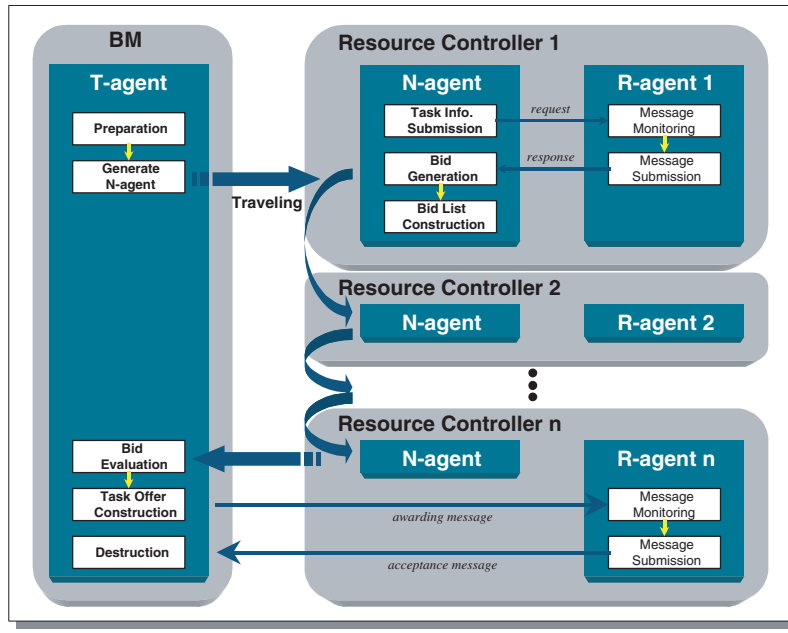

Figure 10. MANPro-based negotiation process.

travel, and disposal of the mobile agent. The proposed negotiation process has the following three phases: (1) preparation, (2) travelling, (3) awarding. Figure 10 illustrates the MANPro-based negotiation process, and figure 11 shows an activity diagram of the negotiation process.

Negotiation phase 1 – Preparation

In this phase, the BM prepares a new negotiation process. First, the BM confirms whether the R-agents of resources that are candidates for taking a given task participate in an ongoing negotiation process, before the BM creates a new T-agent bound to a given task. If all R-agents can participate in the new negotiation process, the BM creates a T-agent to initiate the negotiation process for a given task. Otherwise, the BM prepares the negotiation process for another task, of which the priority is equal to the one of the given task. When the BM creates a new T-agent, the BM registers the relevant resources at the shop database. The T-agent created by the BM makes the N-agent initiate a new negotiation process.

Negotiation phase 2 – Travelling

The N-agent made in the previous phase travels across resource controllers and collects bids to make a list of bids. When the N-agent arrives at a resource controller, it receives the status of the resource through the R-agent. Next, the N-agent generates a bid according to the given policy of negotiation and filters out the bid by pre-evaluation. The completed list of bids is reported to the T-agent.

Negotiation phase 3 – Awarding

This is the final phase of the negotiation process. First, the T-agent evaluates the bids reported by the N-agent and selects the best bid. Next, the T-agent makes a task

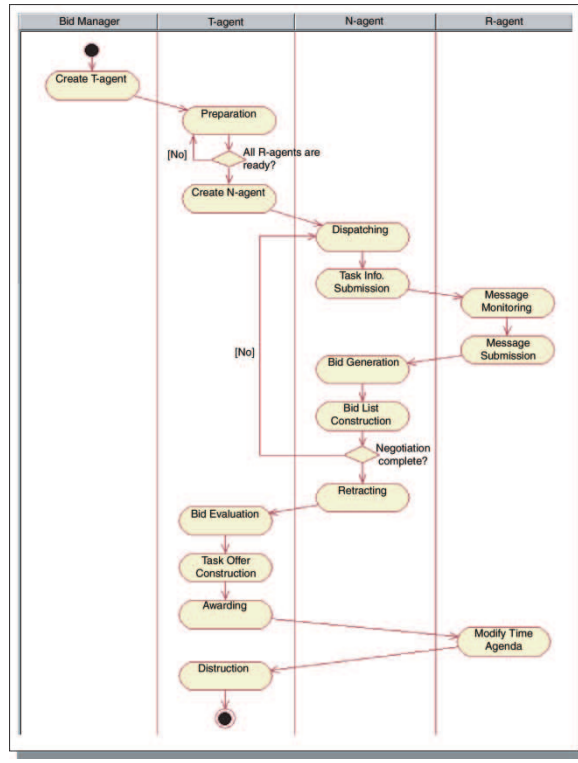

Figure 11. The activity diagram of mobile agent-based negotiation.

offer and sends the awarding message with the task offer to the R-agent that proposes the selected bid. Then, the T-agent revises information related to resource of the shop database in order that the resource may take part in a new negotiation process. The selected R-agent that receives the awarding message sends the task acceptance message to the T-agent. When the T-agent receives this message, it destroys itself, and the negotiation process is completed.

5. Prototype example

In this paper, Java and Aglet are used to build a prototype system for real-time scheduling by using MANPro. Aglets are a mobile agent system developed by the IBM Tokyo Research Laboratory. Aglets are based entirely on the Java language and have features and limitations based on the use of Java. Java presents a complete system built for a mobile code, based on the concept of a virtual machine. Software developed in Java must be able to run on any machine that is running a Java virtual machine. Aglets allow users to create mobile platform-independent agents, based on the Java programming language and use the agent transfer protocol (ATP) to transfer agents over the network. ATP is an application-level standard protocol for distributed agent-based information systems (Lange and Oshima 1998).

Figure 12 shows a mobile agent server, Tahiti, which was presented by Aglet, as well as a developed prototype of N-agent and R-agent. When an N-agent arrives at a resource controller, it is registered in Tahiti and begins to execute operations, such as inspecting the status of the resource and generating bids. The captured situation

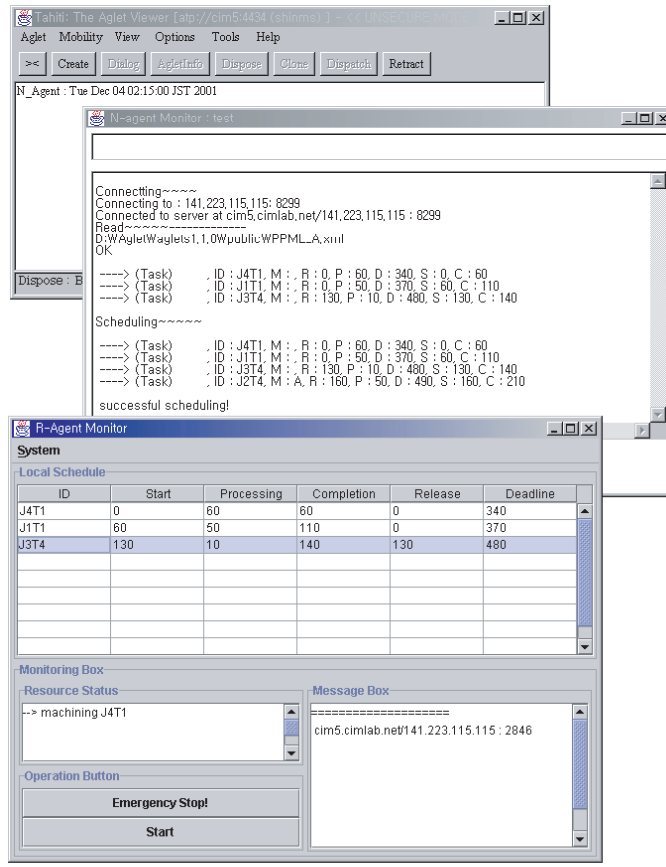

Figure 12. Agent server, N-agent, and R-agent monitor.

represents the case where a resource has three allocated tasks, J4T1, J1T1 and J3T4, and an N-agent travels the network of resource controllers to allocate a new task, J2T4. The N-agent generates a bid to allocate J2T4 to the resource. The bid is compared with other bids by the T-agent. However, the prototype system is incomplete because it has no complete bid generation mechanism and bid evaluation mechanism. It merely represents the negotiation mechanism using MANPro.

6. Concluding remarks

In this paper, a mobile agent-based negotiation mechanism, MANPro, is presented. MANPro defines the communication architecture applying a mobile agent, N-agent, to the mechanism for information exchange and the information architecture introducing an ontology-based information model. The mobile agent executes an actual negotiation process, such as information acquisition and bid generation. The process is performed in the local area without network communication. Moreover, the mobile agent has the ability to translate from an information model to another model by semantic inference.

MANPro makes it possible to build a fully distributed SFCS. Since tasks, such as atomic operations and issues of negotiations, are embedded into agents, and since the agents are dispatched into the network, the participant systems can execute

asynchronous negotiation processes. Moreover, since each task is an atomic operation that is executed by only one resource, it is not necessary to consider dependency between resources. Therefore, the subordinate systems of the SFCS can take out a guarantee on autonomy, so that it need not synchronize with any other systems. These properties enable the SFCS to be fault-tolerant. Moreover, since the mobile agent has intelligence, and since it internally encapsulates protocols, it can not only adapt to dynamically changing environments but can also accommodate to new requirements of a predefined information model without difficulty. Furthermore, since an ontology-based information model supports an interchanging mechanism to bridge diverse information models, MANPro is appropriate in the heterogeneous environment of distributed SFCSs.

For further study, detailed internal strategic policies of the single agent, such as a bid generation mechanism and bid evaluation mechanism, will be developed in order to implement a real manufacturing system using MANPro. Moreover, the performance of MANPro will be tested, verified, and eventually compared with other methodologies.

Acknowledgement

This research was supported in part by grant No. 2001-1-31500-005-1 from the Basic Research Program of the Korea Science & Engineering Foundation and the BK 21 Project in 2003. The authors would like to express their gratitude for the support.

References

Baker, A., 1991, Manufacturing control with a market-driven contract net. PhD thesis, Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Inst., Troy, NY.

Brickley, D. and Guha, R.V., 2002, RDF vocabulary description language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema.

Butler, J. and Ohtsubo, H., 1992, ADDYMS: architecture for distributed dynamic manufacturing scheduling. In A. Famili, D. S. Nau and S. H. Kim (Eds) *Artificial Intelligence Applications in Manufacturing* (Menlo Park, CA: AAAI Press), pp. 199–214.

Gómez-Pérez, A. and Corcho, O., 2002, Ontology languages for the semantic web. *IEEE Intelligent Systems*, **17**, 54–60.

Harmelen, F., Patel-Schneider, P. and Horrocks, I., 2001, Reference description of the DAML + OIL ontology markup language. http://www.daml.org/2001/03/reference.

Ito, T. and Salleh, M. R., 2000, A blackboard-based negotiation for collaborative supply chain system. *Journal of Materials Processing Technology*, **107**, 398–403.

Kádár, B., Monostori, L. and Szelke, E., 1998, An object-oriented framework for developing distributed manufacturing architecture. *Journal of Intelligent Manufacturing*, **9**, 173–179.

Lange, D. B. and Oshima, M., 1998, *Programming and Developing Java Mobile Agents with Aglets* (Addison-Wesley).

Lassila, O. and Swick, R., 1999, Resource Description Framework (RDF) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax.

Li, Y., Huang, B., Liu, W., Gou, H. and Wu, C., 2001, Ontology for modeling and analyzing of enterprise competence. *Proceedings of 2001 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2941–2946.

Lin, G. and Solberg, J., 1992, Integrated shop floor control using autonomous agents. *IIE Transactions: Design and Manufacturing*, **24**, 57–71.

Okino, N., 1993, Bionic manufacturing systems. J. Peklenik (ed.) *Proceedings of the CIRP Seminar on Flexible Manufacturing Systems Past-Present-Future*, Bled, Slovenia, pp. 73–95.

Ouellet, R. and Ogbuji, U., 2002, Introduction to DAML: part I. http://www.xml.com/pub/a/2002/01/30/daml1.html.

Park, H. and Lee, W., 2000, Agent-based shop control system under holonic manufacturing concept. *Science and Tech. Proceedings of the 4th Korea-Russia International Symposium*, pp. 116–121.

Ryu, K., Shin, M. and Jung, M., 2001, A methodology for implementing agent-based controllers in the fractal manufacturing system. *Proceedings of Fifth Conference on Engineering Design & Automation*, Las Vegas, pp. 91–96.

Shen, W., 2002, Distributed manufacturing scheduling using intelligent agents. *IEEE Intelligent Systems*, **17**, 88–94.

Smith, R., 1980, The Contract Net Protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, **29**(12), 1104–1113.

Smith, R. and Davis, R., 1981, Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, **11**(1), 61–70.

Ueda, N., 1993, A generic approach toward future manufacturing system. In J. Peklenik (ed.) *Proceedings of the CIRP Seminar on Flexible Manufacturing Systems Past-Present-Future*, Bled, Slovenia, pp. 211–228.

Valckenaers, P., Bonneville, F., Brussel, H. V., Bongaerts, L. and Wyns, J., 1994, Results of the holonic control system benchmark at the K.U. Leuven. *Proceedings of the Computer Integrated Manufacturing and Automation Conference*, Rensselaer Polytechnic Institute, Troy, NY, pp. 128–133.

Warnecke, H. J., 1993, *The Fractal Company: a Revolution in Corporate Culture* (Berlin: Springer-Verlag).