1

Multi-Agent Based Distributed Manufacturing

J. Li, J.Y H. Fuh, Y.F. Zhang and A.Y.C. Nee

1. Introduction

Agent theory is developed from distributed artificial intelligence, which is regarded as a prospective methodology suitable for solving distributed complex problems, and it has been applied in many areas including manufacturing engineering. In this chapter, some basic issues for agent theory are described and an example of one agent-based distributed manufacturing system is presented.

1.1 Agent and multi-agent system

Jennings and Wooldridge (Jennings and Wooldridge 1998) have defined an agent as "a computer system situated in some environment and capable of autonomous action in this environment, in order to meet its design objectives". Some of the main properties of agents are autonomy, socialability, reactivity, and proactiveness (Wooldridge and Jennings 1995):

- Autonomy: Autonomy characterizes the ability of an agent to act on its own behalf. Agents can operate without direct intervention of humans or other agents, and have a some kind of control over their actions and internal states (Castelfranchi 1995).
- Socialability: Agents can interact with other agents via agent communication languages (Gensereth and Ketchpel 1994)
- Reactivity: Agents can perceive the changes of their environment, which may be the physical world, a collection of other agents, the Inter net, and other fields, and respond to make the related decision accordingly in real time.
- Proactiveness: Agents do not only act in response to their environment, but also exhibit goal-directed behavior by taking the initiative

Source: Manufacturing the Future, Concepts - Technologies - Visions , ISBN 3-86611-198-3, pp. 908, ARS/plV, Germany, July 2006, Edited by: Kordic, V.; Lazinica, A. & Merdan, M.

All of these properties are necessary for agents to act as autonomous, loosely coupled and self coordinating entities in an open distributed system; which forms a multi-agent based system (MAS). A MAS consists of a group of agents that play individual roles in an organizational structure (Weiss 1999). The most important characteristic of MAS is the agents' capabilities of communication and cooperation, which make them to interact with other agents to achieve their individual objectives, as well as the common goals of the system (Wooldridge and Jennings 1995). Other important characteristics of the agent-based systems include scalability, modularity and re-configurability.
In an MAS model, every agent is a representative of a functional cell. For instance, in order to agentify a complex system, it will be divided into some subsystems, each of which is further encapsulated into an agent. Each agent conquers its individual problem, and cooperates with other related agents to solve the whole problem. In the distributed system modeling, an agent is the representative of a distributed cell which solves its own problems and can cooperate with other agents to fulfill a task if necessary. A comprehensive book on multi-agent theory can be found in (Weiss 1999).

1.2 The architecture of MAS

The architectures of multi-agent based systems provide the frameworks within which agents are designed and constructed (Shen 2002). Similar with the organization of the distributed manufacturing system, there are three types of architecture for multi-agent based systems, which are hierarchical (A), heterarchical (B) and hybrid structures (C), as shown in the figure 1.

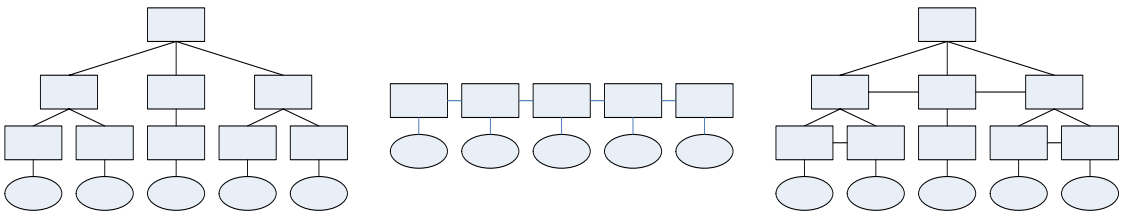

Figure 1. Architecture of MAS

In the hierarchical architecture (A), the agents are connected with layered relationship and all of the control modules are organized into a hierarchical man-

ner. Each agent will have only one direct supervisor at its directly upper layer and several subordinate agents at its directly lower layer. The agent executes the commands and plans only from its supervisor agent and gathers the feedback information from its subordinate agents. The main advantage for this structure is that global optimization can be achieved possibly as the complete information and status of the system can be collected by the agent at the highest layer; while the main disadvantages resides in less adaptability and reliability because the system may be malfunction once the central controller agent breaks down.

Heterarchical architecture (B) is another different style compared with the previous one because there is no central controller in this kind of structure and the relationship of the agents is peer to peer. Each of the agents is autonomous and has its own decision-making mechanism. The cooperation work among agents is to be realized by negotiation: the related agents will negotiate and make tradeoff for a variety of factors. The advantage for this type architecture is its high robustness because breakdown of one agent will not influence others and the rest can still work. The main problem for this architecture lies in the difficulty to achieve a global optimization as no single agent can collect the full information and status from others. Furthermore, another shortcoming is that the execution efficiency is relative low in such framework because the negotiation process may be inefficient and less effectiveness, especially for those tasks need to be completed by several cooperative agents.

The third type (C) is the hybrid architecture, which can be regarded as a compromise of the above two kinds. The hierarchy of the system enhances its efficiency and effectiveness on a global basis while achieving some advantages of the heterarchical architecture to keep the good adaptability and autonomy. In this architecture, the agents at the lower level are also intelligent and have some degree of autonomy, which can be viewed as a heterarchical structure. But the agents also have their upper layered supervisor agent, which can collect the information and distribute tasks to some capable subordinate agents. As the upper level supervisor agent can get a global view for its subordinate agents, some global optimal decision can be achieved. At the same time, as the lower level agents are autonomous, some decisions can be made locally and will not impact other agents, which can improve the robustness and adaptability of the whole system.

1.3 The coordination methodology for MAS

The methodology of negotiation and coordination is one of the bases for effective management and control in a distributed system. Presently, the well-known Contract Net Protocol (CNP) (FIPA 1997; FIPA 2000(1)) is adopted as the coordination and negotiation protocol in most of multi-agent systems. CNP method was proposed by smith (Smith 1980; Davis and Smith 1983; Smith 1988) and recommended by FIPA(The Foundation for Intelligent Physical Agents)(FIPA 2000(1); FIPA 2000(2)), an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. A standard process for the CNP involves four basic steps as shown in Figure 2:

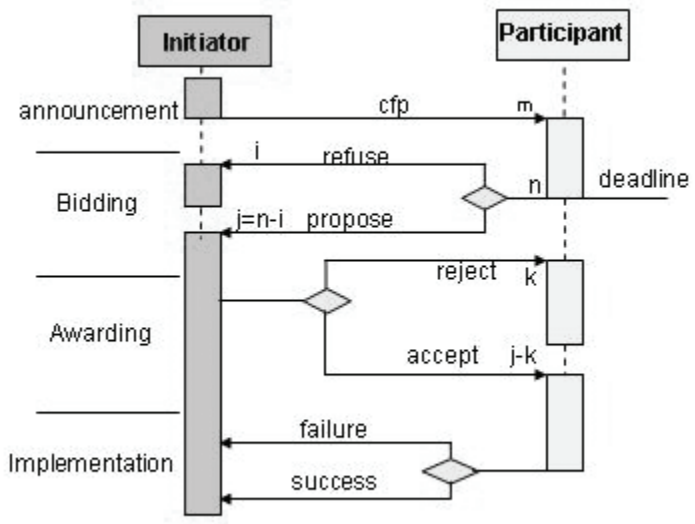

Figure 2. FIPA Contract Net Protocol (FIPA 2000(1))

- Task announcement: The initiator agent broadcasts an announcement to the participant agents to call for proposal (cfp).
- Bidding: Those participants that receive the announcement and have the appropriate capability to make the evaluation on the task, and then reply their bids to the initiator agent.
- Awarding: The initiator agent awards the task to the most appropriate agents according to the proposals they have submitted.

- Implementation: The awarded participant agent performs the task, and receives the benefits predefined.

Currently, the CNP method is widely used for negotiation and coordination among agent systems, and has been proved to be effective to solve distributed problems.

1.4 The development platform for agent-based systems

Most of the intelligent agent and multi-agent systems are working under distributed and heterogeneous environments, and C++ and Java are the two most-adopted programming languages. At the early stage, some works were developed from scratch, which were rather difficult to deal with. Recently, useful platforms and templates have been provided by some institutes, which can provide some basic and necessary modules such as communication, interface design, agent kernel template, etc. The adoption of these platforms facilitates the development and let the designers focus on the functional modules programming, thus to reduce the workload and difficulty of agent applications development. Among these development platforms, JADE (F. Bellifemine, Caire et al. 2006) and Jatlite (JATLite) are two typical and widely applied systems.

JADE (Java Agent DEvelopment Framework) is a software framework developed in Java language by TILAB (JADE 2005). It is composed of two parts, one is the libraries (Java classes) required to develop the agent applications and functions and the other is a run-time environment providing some necessary services for the agents' execution. The platform can be executed in a distributed, multi-party application with peer-to-peer communication, which include both wired and wireless environment. The platform supports execution with cross operation system and the configuration can be controlled via a remote GUI; furthermore, the platform also supports hot exchange, moving agents from one machine to another at run-time.

In JADE, middleware acts as the interface of low layer and applications. Each agent is identified by a unique name and provides a set of services. The agent can search for other agents to provide given services according to the middleware if necessary. With the role of middleware, the agents can dynamically discover other agents and to communicate with them by a peer-to-peer paradigm. The structure of a message complies with the ACL language defined by

FIPA and includes fields, such as variables indicating the context a message refers-to and timeout that can be waited before an answer is received, aiming at supporting complex interactions and multiple parallel conversations (F. Bellifemine, Caire et al. 2006). Furthermore, in order to support the implementation of complex conversations, JADE provides a set of skeletons of typical interaction patterns to perform specific tasks, such as negotiations, auctions and task delegation.

Compared with JADE, JATLite is a lighter and easier to use as a platform for agent-based applications. As it provides only some basic and necessary functions for agent applications, JATLite is more suitable for prototype development in agent-based research work. JATLite is composed of some java packages which help to build agent-based applications with Java language. In the package, four different layers: abstract, base, KQML and router layer, covering from the lowest layer with an operation system to the router function. The package is developed according to TCP/IP protocols, which ensures the system can be running in the Internet. In JATLite, the router acts as the key role in the message communication among the agents.

Although the functions of JATLite may not be as powerful as those in JADE, it is still widely used. The platform is simple and provides some reliable basic services for the agent execution. Furthermore, it sill provides some templates for agent execution; thus, the designers can implement their applications easily.

1.5 Application of MAS in manufacturing system integration

With manufacturing systems become distributed and decentralized in different geographical sites, it is necessary to study the solution of specific problems which arise in a distributed environment. As the MAS system shows the promising capability to solve distributed problems, a great amount of efforts have been made to apply the multi-agent theory to the manufacturing system integration, aiming to study the problems of the distributed manufacturing system. In this part, some typical agent-based manufacturing systems are introduced.

MetaMorph

MetaMorph and MetaMorph II are two consecutive projects developed in the University of Calgary (Shen, Maturana et al. 1998; Shen, Xue et al. 1998; Maturana, Shen et al. 1999; Shen 2002). MetaMorph is an adaptive multi-agent

manufacturing system aimed to provide an agent-based approach for dynamically creating and managing agent communities in distributed manufacturing environments (Maturana, Shen et al. 1999). There are two main types of agents in MetaMorph: *resource agents* and *mediator agents*. Resource agents are used to represent manufacturing devices and operations, and mediator agents are used to coordinate the interactions among agents (resource agents and also mediator agents).

Mediator-centric federation architecture is one of the system characteristics, by which the intelligent agents can link with mediator agents to find other agents in the environment. The activity for mediators is interpreting messages, decomposing tasks, and providing processing times for every new task. Additionally, mediators assume the role of system coordinators by promoting cooperation among the intelligent agents. Both brokering and recruiting communication are adopted to find the related agents for specific tasks. Once appropriate agents have been found, these agents can be directly linked and communicate directly without the aid of mediator.

The object of MetaMorph II project is to integrate the manufacturing enterprise's activities such as design, planning, scheduling, and simulation, execution, with those of its suppliers, customers and partners into a distributed intelligent open environment. In this Infrastructure, the manufacturing system is primarily organized at the highest level through 'subsystem' mediators. Each subsystem is connected (integrated) to the system through a special mediator. Each subsystem itself can be an agent-based system (e.g., agent-based manufacturing scheduling system), or any other type of system like the feature-based design system, knowledge-based material management system, and so on. Agents in a subsystem may also be autonomous agents at the subsystem level. Some of these agents may also be able to communicate directly with other subsystems or the agents in other subsystems. Mediators are also agents, called mediator agents. The main difference between a mediator and a facilitator is that a facilitator provides the message services in general, but a mediator assumes an additional role of system coordinators by promoting cooperation among intelligent agents and learning from the agents' behavior.

CIIMPLEX

CIIMPLEX (Consortium for Intelligent Integrated Manufacturing Planning-Execution) (Peng, Finin et al. 1998) was developed by UMBC and some other institutes, which presents an agent-based framework of enterprise integration

for manufacturing planning and execution. The system is composed of name server, facilitator agent and gateway agent and some executive agents. The different functions of the manufacturing process are encapsulated into individual agents. In the system, a set of agents with specialized expertise can be quickly assembled to gather the relevant information and knowledge, and to cooperate with other agents to arrive at timely decisions to deal with various enterprise scenarios.

Different executive agents are designed to perform special functions such as data collection, analysis of plans and schedules, resolving the conflicts; furthermore, some agents are created to integrate the function of the legacy system. With this architecture, the raw transaction data of the low level, such as shop floors activities, can be collected, aggregated, interpolated and extrapolated by agents and made available for other interested agents. Manufacturing planning and execution can thus be integrated through the collaboration of these agents.

The AARIA project

The AARIA project (Autonomous Agents at Rock Island Arsenal) (Parunak, Baker et al. 1998; Parunak, Savit et al. 1998) is an agent-based prototype system based on the Internet-related technologies. In the system, Internet is used as the platform, and distributed scheduling and controlling techniques are developed to realize the distributed manufacturing. All of the agents are tied by Internet to form a virtual manufacturing environment for tasks. With the agent technology, the resource can be redeployed easily to meet the fast changing environment, which increases the agility of the system. Furthermore, the productive resources can be adjusted according to the products' requirement, which make the system meet the customization requirements.

In the system, besides the functional decomposition, physical factors are also considered during the resource agentificaiton process. The main agents of the system include resource brokers, part brokers, and unit process brokers. Resource broker agents manage the constrained resources of the system (e.g. people, machines, facilities, etc.). Part broker agents manage material handling and inventory. Unit process broker agents utilize their knowledge of how to combine resources and parts to make other parts. These three types of agents negotiate among themselves and with the customer along the axes of possible production including price, quality, delivery time, product features, and speed of answers (Baker, Parunak et al. 1999).

DaimlerChrysler manufacturing line control system

One industrial application of agent-based manufacturing line control system is implemented in DaimlerChrysler (Bussmann and Schild 2001; Bussmann and Sieverding 2001), whose objective is to develop a flexible transportation and control system. In this project, each work piece, machine and shifting table is encapsulated into one specific agent. In the execution, the work piece agent will auction off its coming operations to machine agents. Every machine agent's bid include information about its current state of buffer. Once a work piece agent awards a machine agent, it will be the next goal of the work piece. The routing of the work piece will be negotiated by the work piece agent with the shifting tale agent.

The application of agent-based system shows two key advantages for product manufacturing. One is the distributed responsiveness, as the decision making can be much more localized. If unexpected events occur, agents have the autonomy and proactiveness to try alternatives thus can be more responsive to prevailing circumstances. The other advantage is that dynamical control mechanism, which improves the agility of the system. Because the schedules are built up dynamically through flexible interactions, they can be readily altered in the event of delays or unexpected contingencies. The implementation of the testing system has increased throughput and greater robustness to failure (Jennings and Bussmann 2003), which also shows a good prospect for the agent-based manufacturing system.

2. Multi-Agent Based Distributed Product Design and Manufacturing Planning

In this section, one agent-based distributed manufacturing system developed in the National University of Singapore (NUS) (Sun 1999; Jia 2001; Wang 2001; Jia, Fuh et al. 2002; Li 2002; Jia, Ong et al. 2004; Mahesh, Fuh et al. 2005) is presented, which studies a multi-agent based approach to integrate product design, manufacturability analysis, process planning and scheduling in a distributed manner. Under this framework, geographically dispersed entities are allowed to work cooperatively towards overall manufacturing system goals. The system model considers constraints and requirements from the different product development cycles and manufacturing.

The system adopted a federator structure to model the various manufacturing functional departments in a manufacturing process that includes design, manufacturability evaluation, process planning, scheduling and shop floor control. In the system, the different functional departments dispersed in different geographical sites are encapsulated into agents. Facilitator architecture is selected as the system architecture, which comprises a facilitator agent, a console agent and several service agents. The facilitator is responsible for the decomposition and dispatch of tasks, and resolving conflicts of system execution. The console agent acts as an interacting interface between designers and the system. The service agent models the functional modules of different product development phases, including Designing Agent, Manufacturing Resource Agent, Manufacturability Evaluation Agent, Process Planning Agent, Scheduling Agent, etc. Each functional agent represents a participant involved in a different product development and manufacturing phase. Facilitator plays the central and control roles in the whole environment, and each participant can know the status and requirements of other participants in real-time through it.

2.1 System framework design

In a multi-agent manufacturing environment, the isolated and distributed functional sub-systems can be integrated by encapsulating them as interacting agents. Each agent is specifically in charge of a particular design or manufacturing activity. The agents communicate and exchange information to solve problems in a collaborative manner. The components interact dynamically, addressing the different manufacturing planning issues collaboratively, thereby avoiding costly manual iterations. The federated structure adopted as the architecture ensures the openness of the system, which makes the functional agents can join or leave without having to halt or to reinitialize the other agents' work in progress. The different components can interact dynamically in such platform, addressing the product design and manufacturing planning issues efficiently, and the separate domains of expertise may reside at distributed sites on a network but collaborate with others on a common task, which results in great time saving in terms of data transfer and interpretation. Some legacy software tools can also be wrapped into Java-based agents having the capability of interacting with others.

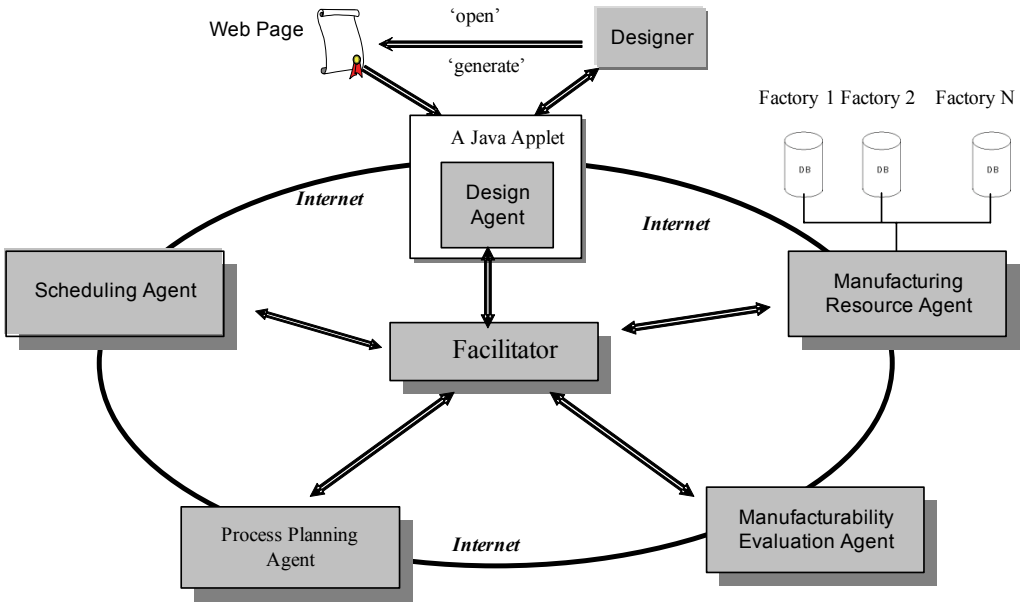

Figure 3. System architecture

The system architecture, as depicted in figure 3, which is composed of six components: Facilitator, Design Agent (D-Agent), Manufacturing Resource Agent (MR-Agent), Manufacturability Evaluation Agent (ME-Agent), Process Planning Agent (PP-Agent) and Scheduling Agent (S-Agent). The last four service agents (encapsulated pre-existing legacy tools) and the D-Agent interact with each other through the Facilitator.

2.2 Agent coordination and individual agents

The function of each agent during this framework is defined as follows:

(1) Facilitator:
It is responsible for the management of interactions and conflict resolution in the agent community. Once any agent joins or leaves the system, it needs to register with status and information to the Facilitator. Thus, the Facilitator "knows" which agent is available, and any function each agent has. Each executive agent receives tasks from the facilitator, and feedbacks the results to it after completing. The Facilitator also routes the requests information received to appropriate agents based on its knowledge of capabilities of each agent, which is known as content-based routing. In performing this task, the Facilita-

tor can go beyond a simple pattern matching by translating messages, decomposing problems into sub-problems, and scheduling the work on those sub-problems.

(2) D-Agent:
It is the interface between the system and the designers, by which the design information of product is submitted to other agents for manufacturability analysis, process plan and scheduling generation. Once the designed parts need further modifications, the information will be also sent back. It also advises the designer to make necessary modifications to the design.

(3) MR-Agent:
This agent manages manufacturing resource models from those different factories of the system, which contain information of available shop-floor resources, including machines and tools, and the capability of these resources. These models are stored in individual databases located at different local sites. The agent is in charge of looking for a suitable capability for manufacturability evaluation.

(4) ME-Agent:
This agent is responsible for the manufacturability evaluation of the product design with the help of acquiring capability information from the MR-Agent. It returns information about conflicts to the Facilitator, as well as suggestions for product redesign or a suitable capability model.

(5) PP-Agent:
This agent is responsible for the generation of an optimal process plan based on the design and selected resources.

(6) S-Agent:
This agent makes the manufacturing scheduling for parts, and feedback to the facilitator.

In order to manage the product and manufacturing information, each agent has a local database, which is used to store and manage messages received from other agents. Furthermore, with the Internet, all of these individual databases are integrated into a distributed database to improve the execution efficiency of the system.

Under such framework, the manufacturing tasks are usually executed by the cooperation of several different related agents. The takes are decomposed firstly into some sub-tasks and dispatched to the destination for process, which needs the cooperation and coordination of the agents in the system. In the project, the agents of the system make negotiations trying to find optimal trade-offs among their local preferences and other agents' preferences and make commitments based on the negotiation results. The task-completing process in the system consists of the following steps:(1)A remote designer submits design information of a product/part to the Facilitator via the D-Agent; (2)The Facilitator decomposes the task into mutually interrelated sub-tasks from a global objective point of view; (3)The Facilitator dispatches the sub-tasks to appropriate executive agents; (4)Executive agents complete their sub-tasks independently; (5)The Facilitator detects conflicts;(6)The Facilitator defines and refines the shared space of interacting agents to remove conflicts; and (7)Conflict resolution.

2.3 Agent definition

2.3.1 Manufacturability evaluation agent (MEA)

This agent is in charge of evaluating the manufacturability of a designed part during the design phrase and sending the modifying information to the design agent if necessary. The agent judges the manufacturability for one part and selects the most preferable machining plan alternatives considering the part's dimensions, tolerances, and surface finishes, along with the availability and capabilities of machine tools and tooling constraints. MEA is, firstly, to check whether the design features are defined correctly; secondly to check if design features can be machined or not based on the current available manufacturing resources; and thirdly to find out all available manufacturing resources that can fabricate the product.

Manufacturability Evaluation

After receiving the feature information from the Facilitator, the ME-Agent carries out a manufacturability evaluation process for the design. It starts with a local manufacturability evaluation on the model in terms of design flaws. Any local conflict detected in the process is notified to the D-Agent by the Facilitator for design modification. Upon the completion of local manufacturability

evaluations, the ME-Agent makes a global manufacturability evaluation on the model by acquiring a factory model from the RC-Agent.

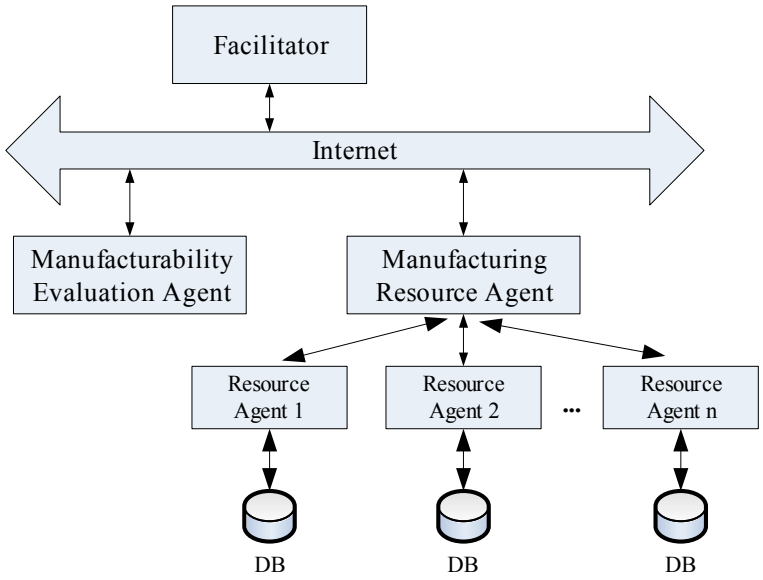

Figure 4. Manufacturability evaluation modules

The RC-Agent checks the availability of various resources required to create the part. Any global conflict is notified to the MR-Agent for it to search for a suitable factory model as a substitute to the former. The two analysis processes are repeatedly executed until no conflict is found on the part model and the agent analyses four manufacturability issues as follows:

(1)Design flaws:
The design flaws refer to those features which are difficult or impossible to machine. The ME-Agent identifies all possible flaws to avoid much higher rectification cost at an advanced stage.

(2)Tool accessibility:
The ME-Agent checks the tool accessibility of each feature. A feature may be inaccessible due to its location and orientation. For those flaws, the cutting tool may not work correctly and need to be modified. (3)Availability of cutters: The ME-Agent checks whether all the required cutting tools to machine the part are available in the factory under consideration. If some machined features ex-

ceed the manufacturing capability of the cutting tools available, then, they will need further revision on the design.

(4)Tolerance and surface finish requirements:
The ME-Agent also need to check the capability of machines contained in the factory against the tolerance and surface finish requirements in the part model.

2.3.2 Resource coordination agent (RCA)

The RCA collects the manufacturing resource information from the work shops and factories with the help of RA. As the system is open and heterogeneous, it is needed that the agent can support flexibility and extensibility in dynamic manufacturing environments, which means that resource coordination, including interaction with users via the other agents, should be sensitive to both the query context and the currently available information. The RCA has the following functionality:

- For a manufacturing resource request, multiple instantiations of the search node may be created;
- Task execution is data-driven;
- Parse the query, and decompose it if appropriate parsing involves getting an ontological model;
- Construct KIF queries based on the SQL queries' contents, and query the Resource Agent using the KIF queries to find relevant resources.

2.3.3 Resource agent

The Resource Agent (RA) manages the data contained in manufacturing information source (e.g., distributed systems database) available to retrieve and update. It acts as an interface between the local data source and other agents, hiding details of the local data organization and representation. To accomplish this task, an RA can announce and update its presence, location and the description of its contents to the RCA. There are two types of information that is of potential interest to other agents:

1. value (ranges) of chosen data objects,
2. the set of operations allowed on the data. The operations range from a single read/update to more complicated data analysis operations. The advertisement information can be sent to the RCA.

RA also needs to answer queries from other agents. It has to translate queries expressing in a common query language (such as KQML) into a language understood by the underlying system. This translation is facilitated by a mapping between the local data concepts and terms, as well as between the common query language syntax, semantics and operators, and those of the native language. Once the queries are translated, the RA sends them to the manufacturing resource database for execution, and translates the answers back into the format understood by the RCA. Additionally, RA and the underlying data source may group certain operations requested by other agents into a local transaction. In addition, RA provides limited transaction capabilities for global resource transaction.

2.3.4 Process planning agent

Process planning agent is developed to generate the optimal or near-optimal process plans for designed part based on the criterion chosen. Under the distributed environment, factories possessing various machines and tools are dispersed at different geographical locations, and usually different manufacturing capabilities are selected to achieve the highest production efficiency. When jobs requiring several operations are received, feasible process plans are produced by available factories according to the precedence relationships of the operations. The final optimal or near-optimal process plan will emerge after comparison of all the feasible process plans. In order to realize and optimize the process plan for the distributed manufacturing systems, the Genetic Algorithm (GA) methodology is adopted as an optimizing method. The GA method is composed of four operations as following: encoding, population initialization, reproduction, and chromosome evaluation and selection.

Encoding

When dealing with a distributed manufacturing system, a chromosome not only needs to represent the sequence of the operations but also indicate which factory this process plan comes from. Therefore, the identity number of the factory will be placed as the first gene of each chromosome no matter how the other genes are randomly arranged. Each other gene comprises the operation ID and corresponding machine, tool and tool access direction (TAD), which will be used to accomplish the operation. As a result, a process plan including

factory and operation information will be represented by a random combination of genes.

Population Initialization

The generation of the initial population in GA is usually done randomly; however, the initial population must consist of strings of valid sequences, satisfying all precedence relations. Once the number of initialized chromosomes is prescribed, the procedures of initialization are given as follows:

(1) Randomly select one factory ID number from the available factory list.
(2) Randomly select one operation among those, which have no predecessors.
(3) Among the remaining operations, randomly select one which has no predecessor or which either predecessor all have already been selected.
(4) Repeat step (3) until each operation has been selected for only once.
(5) Revisit the first selected operation.
(6) Randomly select machines and tools from the selected factory that can be used for performing the operation.
(7) Randomly select one amongst all possible TADs for the operation.
(8) Repeat steps (6) and (7), until each operation has been assigned a machine, tool and TAD.
(9) Repeat steps (1) to (8) until the number of prescribed chromosome is reached.

Reproduction

A genetic search starts with a randomly generated initial population; further generations are created by applying GA operators. This eventually leads to a generation of high performing individuals. There are usually three operators in a typical genetic algorithm, namely crossover operator, mutation operator and inversion operator. In the proposed GA, mutation and crossover operators are used for gene recombination, which is also called offspring generation.

Crossover

In this step, a crossover operator is adopted to ensure the local precedence of operations is met and a feasible offspring is generated. The procedure of the crossover operation is described as follows:

(1) Randomly choose two chromosomes as parent chromosomes.

(2) Based on the chromosome length, two crossover points are randomly gen erated to select a segment in one parent. Each string is then divided into three parts, the left side, the middle segment and the right side according to the cutting points.
(3) Copy the left side and right side of parent 1 to form the left side and right side of child 1. According to the order of operations in parent 2, the opera tor constructs the middle segment of child 1 with operations of parent 2, whose IDs are the same as operations of the middle segment in parent 1.
(4) The role of these parents will then be exchanged in order to generate an other offspring child 2.
(5) Re-assign machines and tools to the operations in the middle segment to legalize the offspring chromosomes according to the factory id.

Mutation

Mutation operator is used to investigate some of the unvisited points in the search space and also to avoid pre-mature convergence of the entire feasible space caused by some super chromosomes. A typical GA mutation makes changes by simply exchanging the positions of some randomly selected genes. However, for the distributed manufacturing system, mutation once is not enough to explore all the feasible operation sequences, as well as compare the different selected factory combination. In the proposed GA process, mutation happens to the chromosomes twice, one is for selected factory (mutation 1) and the other is for the operations (mutation 2).

The procedure of mutation 1 is described as follows:

(1) Randomly select a factory ID from the factory ID list, which is different from the current one.
(2) In order to legalize the chromosome, machines and tools will be re- as signed for all the operations according to the new factory-id.

The procedure of mutation 2 is depicted as follows:
(1) Randomly choose a chromosome.
(2) Choose several pairs of genes stochastically and permute their positions.

Chromosome Evaluation

When all the individuals (process plans) in the population have been determined to be feasible, i.e. an operation precedence is guaranteed, they can be

evaluated based on the objective functions. The objective of the CAPP problem is to obtain an optimal operation sequence that results in optimizing resources and minimizing production costs as well as processing time. In this research, two optimization criteria, i.e. minimum processing times and minimum production cost, are employed to calculate the fitness of each process plan and measure the efficiency of a manufacturing system. After the completion of the manufacturability evaluation, the PP-Agent generates an optimal process plan for the factory supplied by the ME-Agent. The agent first constructs the solution space by identifying all the possible operation-methods (OpM's) for machining each feature and then uses a GA to search for the best plan according to a specific criterion. The criterion can be constructed by using the following cost factors:

Machine cost (*MC*)

$$MC = \sum_{i=1}^{n} MCI_i \qquad (1)$$

where n is the total number of OpM's and MCI_i is the machine cost index for using machine-i, a constant for a particular machine.

Tool cost (*TC*)

$$TC = \sum_{i=1}^{n} TCI_i \qquad (2)$$

where TCI_i is the tool cost index for using tool-i, a constant for a particular tool.

Machine change cost (*MCC*): a machine change is needed when two adjacent operations are performed on different machines.

$$MCC = MCCI \times \sum_{i}^{n-1} \Omega(M_{i+1} - M_i) \qquad (3)$$

where *MCCI* is the machine change cost index, a constant and M_i is the ID of the machine used for operation i.

Setup change cost (*SCC*): a setup change is needed when two adjacent OpM's performed on the same machine have different Tool Approaching Directions (TADs).

$$SCC = SCCI \times \sum_{i=1}^{n-1}((1-\Omega(M_{i+1}-M_i))\times\Omega(TAD_{i+1}-TAD_i)) \qquad (4)$$

where *SCCI* is the setup change cost index, a constant.

Tool change cost (*TCC*): a tool change is needed when two adjacent OpM's performed on the same machine use different tools.

$$TCC = TCCI \times \sum_{i=1}^{n-1}((1-\Omega(M_{i+1}-M_i))\times\Omega(T_{i+1}-T_i)) \qquad (5)$$

where *TCCI* is the tool change cost index, a constant.

2.3.5 Scheduling agent

In a distributed manufacturing environment, every factory has its particular niche areas and can outperform other factories in those specific aspects; therefore, one batch of products are to be finished by the most suitable factory combination with the considerations of low cost and short make span. To meet such requirement, each available candidate factory will submit a feasible process plan for a product in the batch it is capable of processing. The agent then compares all the candidate factories, selects the final factory combination for the products, and meanwhile arranges the manufacturing operations in an optimal sequence. In brief, to generate an optimal schedule in a distributed manufacturing environment, there are two determining factors: the selected factory (or process plan) for every product and operations' sequencing of the machines in the factories. Here, GA is also used as the optimization method to achieve better scheduling results.

The scheduling agent is mainly composed of four major components: the scheduling kernel including the GA engine, the stand-alone scheduling module, scheduling agent module, and the e-scheduling module. Among these four parts, the scheduling kernel can be categorized as the basic component, while the stand-alone module, scheduling agent module and e-scheduling module can be categorized as the application components. The basic component could be combined with any of the three application components to form a scheduling entity, which can carry out the scheduling tasks solely based on a specified scheduling objective.

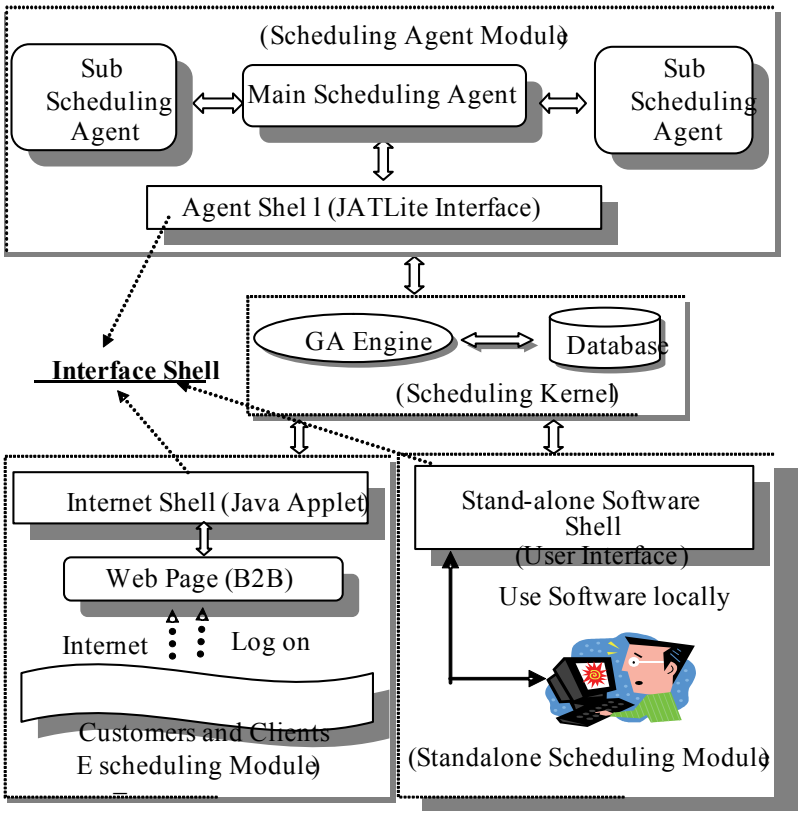

Figure 5. Scheduling agent

Integrating with the scheduling kernel, each of the three application modules has a particular working mode.

The scheduling agent module includes one main scheduling agent (MSA) and many distributed sub-scheduling agents (SSAs). The scheduling agent structure consists of not only the parallel sub-scheduling agents (SSAs) but also a main scheduling agent (MSA). Such a structure can effectively facilitate the information communication among the different production participants within the distributed scheduling system. The MSA, used by the production management department, is responsible for regulating the job production operations, collecting the factories' information from the SSAs and then, making scheduling through the scheduling kernel. After the scheduling, the details about which job is manufactured in which factory in what time and what machine is responsible for what operation can be obtained and sent to each SSA. To finish the job production cooperatively, the SSAs are distributed in the

manufacturing factories, each representing a factory, collecting the factory's working status, detecting its dispatched jobs, and checking the manufacturing progress. In addition, if any contingency happens in any factory, the SSA will send the accident information to the MSA, wait and execute the re-arranged schedule made by MSA.

Genetic Algorithm for distributed Scheduling
Chromosome representation
To handle the distributed scheduling problems, the genes in the GA must comprise the two dominant factors in the distributed manufacturing environment, i.e., the selected factory for every job with the corresponding process plan and the operation processing sequence. Here, a four-element string is used as a gene to represent an operation of a job. The first element is the identification number of the selected factory that is used to process the job, and the next three elements represent the ID of a job. Thus, a schedule can be represented by a combination of genes, which is called "chromosome" in the GA terminology, as long as the combination comprises all the operations of the jobs. Every operation processing sequence can be interpreted according to its occurrence order in the chromosome. As such, for any distributed scheduling problem, a random feasible schedule, including which job goes to which factory and what is the operation processing sequence, can be encoded using a combination of genes.

Chromosome population initialization
To begin the search for the best chromosome and correspondingly, the factory combination and the optimal schedule the chromosome represents, a number of chromosomes are initialized as follows:

(1) Create the lists of ID number of the feasible factories for every job. If job 'j03' can be processed in factories '1', '3' and '5', then the list for job 'j03' will be 1, 3, and 5.
(2) For every job, randomly select a factory ID number from the job's feasible factory list.
(3) According to the job's process plan in the selected factory, produce the job's operation genes. For example: the genes, 1j03-1j03-1j03, mean the first, second and third (last) operation of job 'j03'. All the operations will be manufactured in factory '1'.
(4) Repeat step (2) and step (3), until there is no job left.

(5) Combine and mix all the produced genes together in a stochastic way to form a chromosome.
(6) Repeat step (5), until a prescribed number of chromosome populations is formed.

Genetic operators
The power and potential of GA method come from the gene recombination, including crossover, mutation and inversion, which explore all the possible search space. In this proposed GA, two genetic operators, *crossover* and *mutation*, are employed for the gene recombination, which is called offspring generation. The procedures for the crossover operation are described as follows:

(1) Choose two chromosomes as parents and exchange a random partial string (genes) in the two parents to generate two offspring chromosomes.
(2) Regulate (delete or compensate) genes in each offspring chromosome so that it comprises the operations of all the jobs and inherits the genetic traits of their parents.

Because the sequence of the genes in the chromosome expresses the jobs' operation processing sequence, after the crossover, the processing sequence for every operation (gene) in the schedule (chromosome) is changed. It is important to note that the precedence of a job's operations will not be affected by the crossover because every gene (operation) of a job in the chromosome is not fixed to a specific operation of the job and it is interpreted according to the order of occurrence in the sequence for a given chromosome.

The crossover is carried out under the assumption that a factory has been selected for every job. Yet, in the distributed manufacturing environment, the randomly selected factory is, by and large, not necessarily the most suitable one for the specific job. Therefore, another genetic operator, gene mutation, is employed twice for modifying the operation processing sequence again as well as changing the selected candidate factory for the jobs. The mutation procedures are as follows:

(1) In a chromosome, choose several pairs of genes stochastically and permute their positions (Mutation1).
(2) Select one gene (job) from the chromosome in a random manner. In the meantime, randomly select a factory ID number from the job's feasible fac

tory list, which has been created in step (1) of chromosome population ini tialization.

(3) In the selected chromosome, replace the first element (represents the ID of the selected factories) of all the genes (the gene selected in step 2) with the newly selected factory ID number (Mutation2). The aim of the step is to change the factory selection used to manufacture the job.

(4) To keep the consistency of the chromosomes, apply the same change of fac tory selection for the gene (job) to all the other chromosomes in their gen eration.

In step (1), the first mutation changes the jobs' operation sequences, while from step (2) to step (4), the second mutation changes the factory ID, which is used for a randomly selected job.

After gene crossover and mutation, the parent chromosomes can produce a generation of offspring. In the same way, the offspring could reproduce the next generation of offspring. Thus, through this iteration, numerous chromosomes are produced and can be compared. Correspondingly during this process, many possible factory combinations and job operation processing sequences are formed and analyzed. The application of such gene crossover and mutation in this GA ensures each product (job) to be manufactured in its most suitable factory. In addition, the production schedule of each of the factories in the distributed environment can be generated concurrently.

3. Prototype Implementation

In order to verify the effectiveness of the system, a prototype of the proposed system has been developed for the integration of design, manufacturability analysis, and process planning. The developed system includes a unique facilitator, and several functional agents, which are organized according to the framework of Figure 3. JATLite is selected as the template for the agents' development.

Each agent is composed of the following components: network interface, local knowledge model and domain knowledge model. All the agents in the system use the common communication protocol, KQML, for concurrent negotiations. KQML is conceived as both a message format and a message-handling protocol to support run-time knowledge sharing among agents. It is essentially a wrapper to encode context sensitive information. The KQML language is di-

vided into three layers: the content layer, the message layer, and the communication layer, as shown in figure 6.

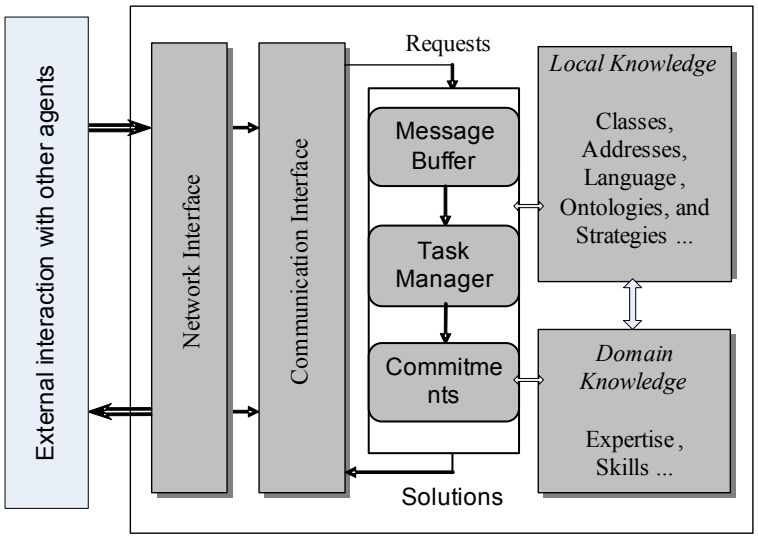

Figure 6. Internal structure of an agent

In the system, agents are autonomous cognitive entities, with deductive, storage and communication capabilities. Autonomy in this case means that an agent can function independently from any other agent. There are three kinds of functional agent in the system. Each has different internal structure, and can be decomposed into the following components:

(1) A network interface: It couples the agent to the network.
(2) A communication interface: It is composed of several methods or functions for communicating with other agents.
(3) A local function module: Resources in this model include Java classes to perform the desired functions, other agent names, messaging types in KQML syntax. The functional module also provides the facility of inference and collaboration facilities. The collaboration facilities are the general plans for coordination behaviour that balances the agent's local interests with global (community) interests.
(4) Agent Knowledge base: The model comprises expertise that is required for an agent to perform functional tasks, and skills that may be methods for ac tivating actions corresponding to the received requests.

A snap shot on the prototype system (PPA and MSA) is shown in figure 7 below.

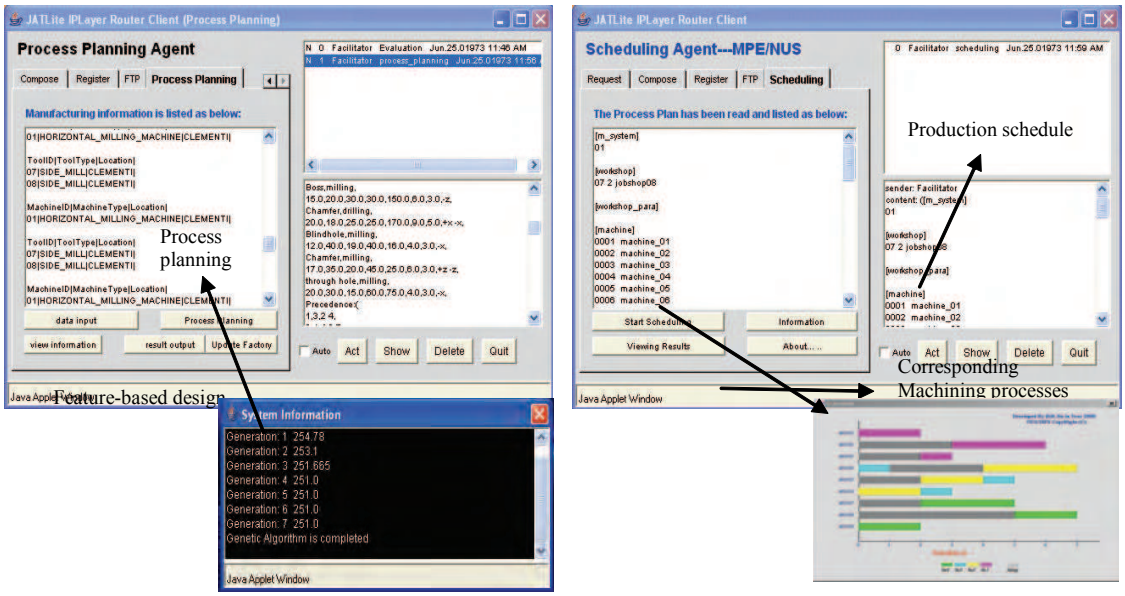

Figure 7. Optimal process plans and production schedules generated from the PPA and MSA respectively

4. Conclusion and Future Work

Agent theory was developed from the distributed artificial intelligence around 20 years before. As its nature and characteristic are suitable for distribute problems solving, agent theory has been viewed as a promising theory and methodology applied to a distributed environment. It has now achieved some promising results in industrial applications. The rapid development of Internet has also provided a good tool and suitable platform for agent's application. With the use of Internet and advancement of communication methods, agents can be dispersed in different geographical places, which make it easy for collaboration of different partners in a manufacturing supply chain.

Except the description to some basic theories and key issues in agent-based systems and the introduction to some typical manufacturing systems, this chapter also introduces one agent-based distributed manufacturing system developed in NUS. The objective of this research is to develop a distributed

collaborative design environment for supporting cooperation among existing engineering tools organized as independent agents on different platforms. A facilitator multi-agent system architecture is discussed for distributed collaborative product design and manufacturing planning, and one prototype for collaborative design and planning on machining processes, which has been developed as a proof-of-concept system, demonstrates the capability of such a multi-agent approach to integrate design, manufacturability evaluation, process planning and scheduling efficiently. This approach could be extended to include other product life cycle considerations, collaboratively, at the design stage. Some of the advantage of the system over the traditional mode can be achieved in several facets as follows:

1) Distributed function execution improves the efficiency and the reliability of the system, thus to increase the responsiveness of the enterprise to the market requirements. In the system, the major functional tasks are distrib uted, and each one agent needs to focus on one task execution, which can improve the efficiency of the process. Furthermore, as the functions are executed dispersedly, once some functional agent malfunction, the rest can still work, which also improves the reliability of the whole system.

2) Open architecture to make the system good adaptability and easy exten sion. As the system adopts a facilitator structure, the newly added agent can execute the system tasks if only registering with the facilitator; at the same time the functional agent can leave the system only need to inform the facilitator and not influence other's progress. Moreover, the newly functional agents can be also integrated into the system without interfering other agents' function. All of these can make the system good adaptability and easy extension, thus to improve the agility of the system.

3) The agent-based system provides a platform to realize the concurrent func tion in the product development. The different departments from design through manufacture to customer service can join together for one product design, thus to improve the quality and efficiency of the final product de velopment.

Although some promising results have been achieved in the prototype and previous research work on the agent theory, there are still difficulties to be

overcome for its wider application in industry. Some of the challenges faced include:

1) Effective coordination and negotiation methods for MAS. Coordination and negation methods are the basis and key issues for intelligent agent sys tems. It has been under study for a long time and a variety of methods have been proposed, but one effective and efficient method for the agent-based system is still needed.

2) Methods to incorporate and agentify the legacy manufacturing systems and tools. Now, there are various computer-aided software systems ap plied in manufacturing system and industrial scenario, but there is still no successful methodology to agentify these legacy modules in the agent sys tems. This is one bottleneck that impedes a wider development of agent-based methods for industrial applications.

3). Agent theory provides a decentralized solution for complex systems, de composing and conquering make the agents easy to deal with sub-tasks. But one problem emerges is that the local optimization can not result in a global optimal result for the whole system. How to achieve a global opti mal solution in the agent-based system still needs a further study.

5. References

Baker, A. D., H. V. D. Parunak, et al. (1999). *Internet-based Manufacturing:* A Perspective from the AARIA Project, Enterprise Action Group.

Bussmann, S. and K. Schild (2001). An agent-based approach to the control of flexible production systems. ETFA 2001. *The 8th International Conference on Emerging Technologies and Factory Automation. Proceedings*, 15-18 Oct. 2001, Antibes-Juan les Pins, France, IEEE.

Bussmann, S. and J. Sieverding (2001). Holonic control of an engine assembly plant: an industrial evaluation. *Proceedings of IEEE International Conference on Systems, Man & Cybernetics*, 7-10 Oct. 2001, Tucson, AZ, USA, IEEE.

Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. *Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents*, Amsterdam, the Netherlands, Springer-Verlag New York, Inc.

Davis, R. and R. G. Smith (1983). "Negotiation as a metaphor for distributed problem solving." *Artificial Intelligence* 20(1): 63-109.

F. Bellifemine, G. Caire, et al. (2006). JADE: A White Paper.

FIPA (1997). FIPA 97 Part 2 Version 2.0: *Agent Communication Language Specification*, FIPA.

FIPA (2000(1)). *FIPA Interaction Protocol Library Specification*, FIPA.

FIPA (2000(2)). *FIPA ACL Message Structure Specification*, FIPA.

Gensereth, M. R. and S. P. Ketchpel (1994). "Software Agents." *Communications of the ACM* Vol. 37(No. 7): 48-53.

JADE (2005). *JADE:Java Agent DEvelopment Framework*, http://jade.tilab. com/.

JATLite http://java.stanford.edu/ .

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

Jennings, N. R. and M. Wooldridge (1998). *Applications of intelligent agents*, Springer-Verlag New York, Inc.

Jia, H. Z.,2001, *Internet-based multi-functional scheduling for distributed manufacturing systems*, *M. Eng Thesis* ,National University of Singapore,Singapore.

Jia, H. Z., J. Y. H. Fuh, et al. (2002). "Web-based Multi-functional Scheduling System for a Distributed Manufacturing Environment." *Concurrent Engineering* 10(1): 27-39.

Jia, H. Z., S. K. Ong, et al. (2004). "An adaptive and upgradable agent-based system for coordinated product development and manufacture." *Robotics and Computer-Integrated Manufacturing* 20(2): 79-90.

Li, L.,2002, Agent-based computer-aided process planning for distributed manufacturing systems., *M. Eng Thesis*, National University of Singapore,Singapore.

Mahesh, M., J.Y.H.Fuh, et al. (2005). "Towards A Generic Distributed and Collaborative Digital Manufacturing", *Proceedings of the International Manufacturing Leaders Forum on Global Competitive Manufacturing*, Adelaide, Australia.

Maturana, F., W. Shen, et al. (1999). "MetaMorph: an adaptive agent-based architecture for intelligent manufacturing." *International Journal of Production Research* 37(10): 2159-73.

Parunak, H. V. D., A. D. Baker, et al. (1998). "The AARIA Agent Architecture: from Manufacturing Requirements to Agent-Based System Design".

WorkshopProc. on Agent-Based Manufacturing, ICAA'98, Minneapolis, MN.

Parunak, H. V. D., R. Savit, et al. (1998). Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, Springer-Verlag, London, UK.

Peng, Y., T. Finin, et al. (1998). "A Multi-Agent System for Enterprise Integration." *International Journal of Agile Manufacturing,* vol. 1(No. 2): 201-212.

Shen, W. (2002). "Distributed manufacturing scheduling using intelligent agents." *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]* 17(1): 88-94.

Shen, W., F. Maturana, et al. (1998). Learning in Agent-Based Manufacturing Systems. *Proceedings of AI & Manufacturing Research Planning Workshop*, IAlbuquerque, NM, The AAAI Press,.

Shen, W., D. Xue, et al. (1998). An Agent-Based Manufacturing Enterprise Infrastructure for Distributed Integrated Intelligent Manufacturing Systems. *Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems*, London, UK.

Smith, R. G. (1980). "The contract net protocol: high level communication and control in a distributed problem solver." *IEEE Transactions on Computers C*-29(12): 1104-1113.

Smith, R. G. (1988). The contract net protocol: high-level communication and control in a distributed problem solver *Distributed Artificial Intelligence* Morgan Kaufmann Publishers Inc.**:** 357-366

Sun, J.,1999, Agent-based product design and planning for distributed concurrent engineering, *M. Eng Thesis*, National University of Singapore,Singapore.

Wang, G.,2001, Agent-based manufactuirng service system, *M. Eng. Thesis*, National University of Singapore.

Weiss, G. (1999). *Muliagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, Massachusetts, The MIT Press.

Wooldridge, M. and N. R. Jennings (1995). "Intelligent Agents: Theories and Practices." *Knowledge Engineering Review*: 115-152.

Manufacturing the Future

Edited by Vedran Kordic, Aleksandar Lazinica and Munir Merdan

ISBN 3-86611-198-3

Hard cover, 908 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, July, 2006

Published in print edition July, 2006

The primary goal of this book is to cover the state-of-the-art development and future directions in modern manufacturing systems. This interdisciplinary and comprehensive volume, consisting of 30 chapters, covers a survey of trends in distributed manufacturing, modern manufacturing equipment, product design process, rapid prototyping, quality assurance, from technological and organisational point of view and aspects of supply chain management.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

J. Li, J.Y. H. Fuh, Y.F. Zhang and A.Y.C. Nee (2006). Multi-Agent Based Distributed Manufacturing, Manufacturing the Future, Vedran Kordic, Aleksandar Lazinica and Munir Merdan (Ed.), ISBN: 3-86611-198-3, InTech, Available from: http://www.intechopen.com/books/manufacturing_the_future/multi-agent_based_distributed_manufacturing

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821