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.

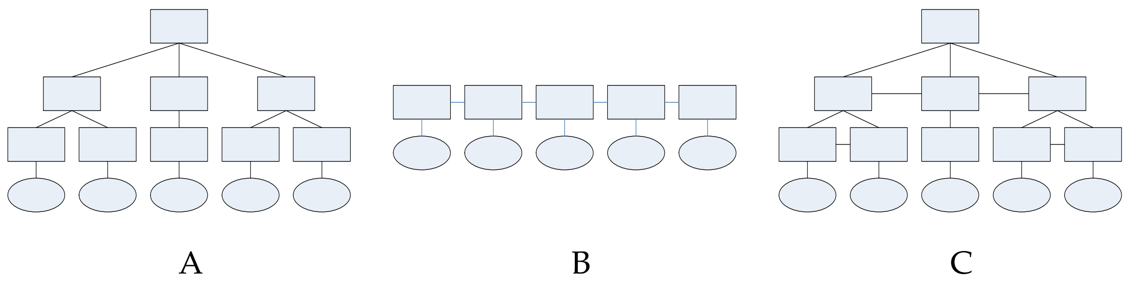

Figure 1. Architecture of MAS

In the hierarchical architecture (A), the agents are connected with layered relationship and all of the control modules are organized into a hierarchical manner. 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 feed-back 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:

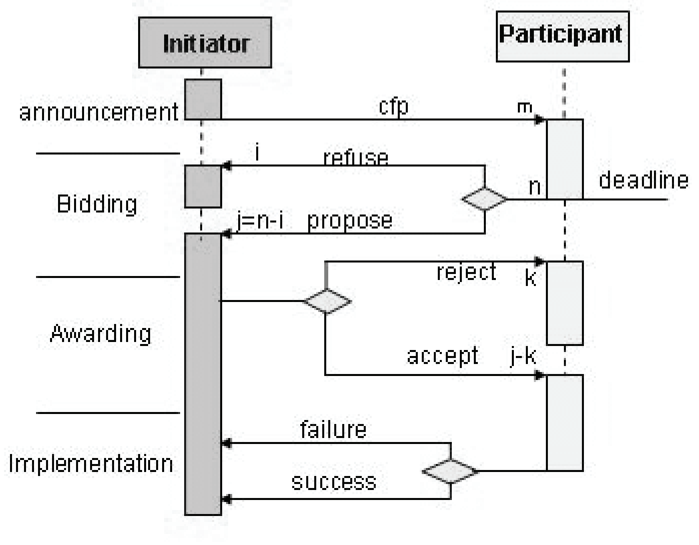

Figure 2. FIPA Contract Net Protocol (FIPA 2000(1))

• 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.

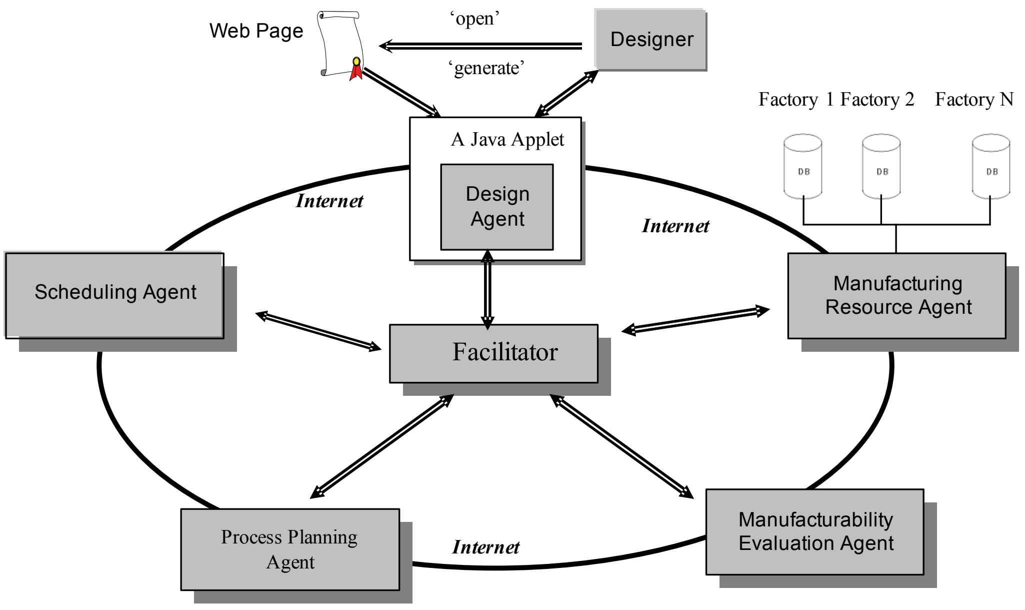

Figure 3. System architecture

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 Facilitator 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 tradeoffs 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.

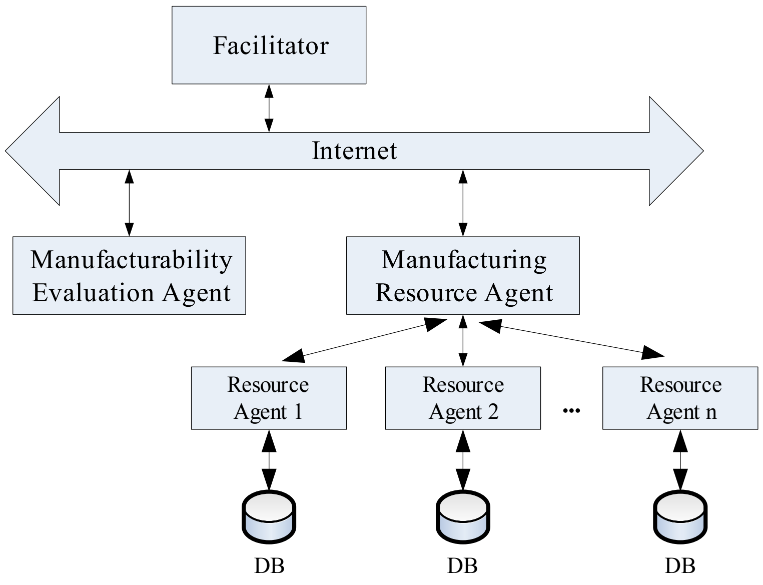

Figure 4. Manufacturability evaluation modules

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.