# 프랙탈 생산시스템을 위한 프랙탈 템플릿의 모델링 및 구현

# Modeling and implementation of a fractal template for fractal manufacturing system (FrMS)

문정태, 정무영
포항공과대학교 산업공학과 / 제품생산기술 연구소

## Abstract

In order to quickly respond to the rapidly changing manufacturing environment, it is imperative for the system to have such capabilities as flexibility, adaptability, reusability, etc. One of the promising approaches for new manufacturing paradigm satisfying those capabilities is the fractal manufacturing system (FrMS). FrMS is based on the concept of autonomously cooperating and self-reconfigurable agents referred to as fractals. In this paper, a fractal template is proposed to have both fractal-specific and agent-specific characteristics and proper techniques for implementation are also selected. The proposed template can be easily extended to the platform of FrMS which supports simulation, shop flow control, and other research issues existed in the manufacturing systems. An example of applying the proposed template to the simulation is also presented.

Keywords: fractal manufacturing system, fractal template, distributed manufacturing systems, agent-based control

## 1. Introduction

Today's manufacturing systems need to adapt to the rapidly changing environment that reflects customers' demands, unpredicted situations, incessant evolution of software and hardware, advances in infrastructures, etc. To quickly respond to the rapidly changing manufacturing environment, it is imperative for the system to have such capabilities as flexibility, adaptability, reusability, etc. Ryu and Jung (2003) proposed the FrMS architecture that could satisfy the requirements for future manufacturing systems. FrMS is based on the concept of autonomously cooperating and self-reconfigurable agents referred to as fractals. A fractal consists of five functional modules (a reporter, an organizer, a resolver, an analyzer, and a monitor) and other several auxiliary modules. On the basis of those modules, a fractal is designed to possess the characteristics of self-similarity, self-organization, goal-orientation, dynamics, and vitality. Furthermore, by adopting agent technology, FrMS has the agent-specific characteristics such as autonomy, mobility, intelligence, and cooperativeness.

Each fractal is a set of self-similar and self-reconfigurable agents, and it interacts with other fractals to achieve individually assigned goals. In FrMS architecture, 18 kinds of agents exist and they are categorized into five functional modules. Figure 1 illustrates the fractal architecture and relationships among its functional modules. An observer which has equipment monitoring agent and network monitoring agent receives messages from the environment (other fractals and equipment) thorough the sensors, and a reporter delivers messages to the environment. Other functional modules cooperate to achieve its goals.

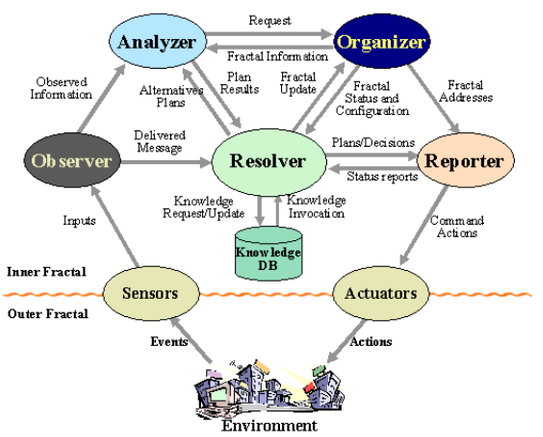


**Figure 1** Relationships among functional modules (Ryu and Jung, 2003)

FrMS has many promising characteristics. One of the main characteristics is goal-orientation. A fractal has an ability to generate its own goals

through the goal generation process (GGP) and to modify individual goals through the goal harmonizing process (GHP). With both the GGP and the GHP, a fractal can achieve individual goals and system goals simultaneously. FrMS can change its configuration automatically through the dynamic restructuring process (DRP). The GGP, the GHP, and the DRP was studied in previous research, however, because of the difficulties in considering overall systems, those are only focused on the working mechanism.

In this research, therefore, we design and implement the fractal template for FrMS. The designed template can be used in both verifying the results of research issues and finding other problems not considered in FrMS. In the next section we present FrMS architecture related to this research. The subsequent main section we illustrate fractal template including its structure and functions with selection of proper techniques. The following section presents our scenario of applying template to FrMS. The paper ends with some conclusions and short reference to our future activities.

## 2. Fractal architecture

The fractal architecture is based on a hierarchical structure and the design of a basic unit incorporates a set of pertinent attributes that can fully represent any level in the hierarchy (Tirpak et al., 1992). This means that a fractal can represent every part of the system from an entire manufacturing shop at the highest-level to a physical machine at the bottom-level (Shin et al., 2003). Each fractal which is a set of autonomous agents provides services required to accomplish individual goals and acts independently while attempting to achieve the shop-level goals. If its hierarchical level is changed (it means that the fractal should handle other level goals), through the fractal evolution process, the low level fractal can be a high level fractal and vice versa. Figure 2 shows inner structure of a fractal. Whatever its hierarchical level, inner structure of a fractal is same as the figure.

In FrMS, many kinds of agents exist and according to their main functions they are classified into six functional modules as shown in the Table 1.

## 3. Fractal template

The main purpose of developing the template is to provide suitable abstractions and support fractal specific characteristics such as reconfigurability, flexibility, etc. In this research, we call a fractal an agent when the fractal is created but not yet ready to perform its roles. When the agent is ready to perform as a fractal, then we call it a fractal. Whatever its hierarchical level, before fabricated, all agents have same structure and modules. At first, each agent has the following three functional modules.

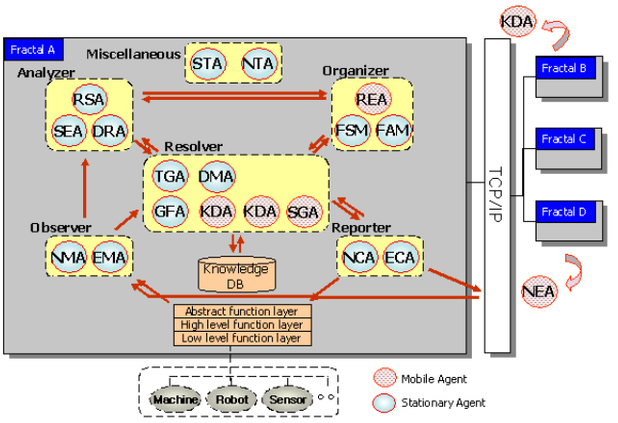


**Figure 2** Inner structure of a fractal

**Table 1** Fractal agents in FrMS (Ryu and Jung, 2003)

| Functional Module | Corresponding Agents |
|---|---|
| Observer | - Network Monitoring Agent(S)<br>- Equipment Monitoring Agent(S) |
| Analyzer | - Schedule Evaluation Agent(S)<br>- Dispatching-rule Rating Agent(S)<br>- Real-time Simulation Agent(S) |
| Resolver | - Schedule Generation Agent(M)<br>- Goal-Formation Agent(S)<br>- Task Governing Agent(S)<br>- Negotiation Agent(M)<br>- Knowledge Database Agent(M)<br>- Decision-Making Agent(S) |
| Organizer | - Fractal Status Manager(S)<br>- Fractal Address Manager(S)<br>- Restructuring Agent(M) |
| Reporter | - Network Command Agent(S)<br>- Equipment Command Agent(S) |
| Miscellaneous | - System Agent(S)<br>- Network Agent(S) |

- Communication module: This module has basic functions to communicate with other agents and fractals. It receives messages and interprets them. When its high level fractal assigns roles to the agent, this module delivers them to the goal-formation module.
- Goal-formation module: The main function of this module is to generate sub-goals and determine whether the agent performs them or the agent should assign generated sub-goals to other agents or fractals. Basically this module has functions of fuzzification of defuzzification

to interpret assigned goals.

- System module: This module helps the agent use other agents and modules like plug-in styles.

The structure seems to be simple; however, it guarantees flexibility, reconfigurability, adaptability and other possibilities of adopting new technologies. With the structure a fractal can perform any task by having appropriate plug-in modules or agents. The structure is based on the concept of building architecture with two entities: components and connectors. Components (agents and modules) act as the primary units of computation in a system and connectors specify interactions and communication patterns between components. This guarantees the structure to support all the reconfigurable processes in FrMS.

Figure 3 illustrates the simple view of the template. When the communication module receives message which contains fuzzy goals, it delivers them to the goal-formation module. The goal-formation module analyzes goals by defuzzification and determines whether the fractal can perform them or not. If it determines that it cannot handle some received goals, the fractal makes connection to other fractals and assigns goals to them. The system module supports those processes. For example, if a fractal is assigned to control equipment (various roles may delivered), from this point on, it prepares performing given roles. When the fractal finishes preparing processes, its structure is similar to that of a fractal shown in figure 2.

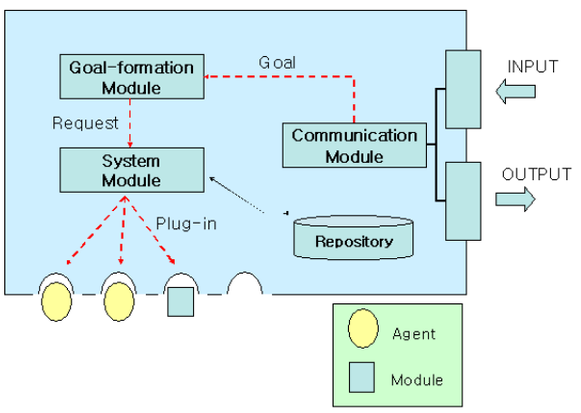


**Figure 3** Simple view of the template

Requirements of a fractal template are follows.

- A fractal should receive mobile agents from other fractals.
- A fractal should change its configuration during run-time.
- Any agent in a fractal could communicate with other agents.
- A mobile agent could share system resources even though it comes from other fractals.
- A fractal should fully support all activities of agents.

In previous researches, researchers developed agents with Java™ language and Aglets™ which is a Java-based agent development tool developed at the IBM Research Laboratory in Japan. The platform of Aglets™ has extensive support for security and agent communication, and provides an excellent package of documentation. However, there are some difficulties in applying the platform to FrMS because it is hard to control local machine and adopt fractal concept by using that platform. Therefore, the overall system consists of two systems, java application for both system control and stationary agents and Aglets™ server referred to as Tahiti for mobile agents at that time. The configuration has many unnecessary things and it gives rise to unexpected problems of synchronization, redundancy, interoperability, conflicts between platforms, etc.

To develop the template, we consider above requirements and try to eliminate those superfluous things. We select Java™ language suitable for our purpose. Java™ language has many advantages for us to develop FrMS. We use Remote Procedure Call (RPC), Remove Method Invocation (RMI), Java Management Extensions (JMX), Java Native Interface (JNI), etc. served by Java platform. Those technologies enable us to implement the fractal template. We use heterogeneous architecture style which mixed with layered architecture and blackboard architecture in the template. Software architecture that adapts to changes in requirements (Eracar and Kolar, 2000) is used in modeling controllers. As a communication protocol, we adopt mobile agent-based negotiation protocol (Shin and Jung, 2004) for efficiently distributing the communication loads of agents.

## 4. Applications

Because of realizing FrMS in its early stage, we intend to phase the template in. At first, for verifying proposed template we prepare the scenario of shop floor simulation using the template. Figure 4 illustrates the layout which is similar to the FrMS test bed. In the layout there are 11 fractals as equipment controllers and one fractal as a high level controller. For convenient we simplify this shop floor by the following several assumptions.

- There is only one type of raw material which comes from the AS/RS.
- There are three types of part, referred to as part A, B, and C. Each part is produced by only one operation.
- Machine 1 and 3 can produce part A and B, and machine 2 and 4 can produce part B and C as well.

- Producing time follows normal distribution.
- AS/RS keeps the balance of the number of parts on Conveyor. The balanced number of each part is three respectively.
- Finished part is unloaded to the destination place along with one of following routes;
  - Conveyor → L/U station → Robot 1 or 4 → AGV → Destination
  - Conveyor → Robot 1 or 4 → AGV → Destination
- Each robot spends same time to transfer parts.
- Buffer size is unlimited and equipment is never malfunctioned.

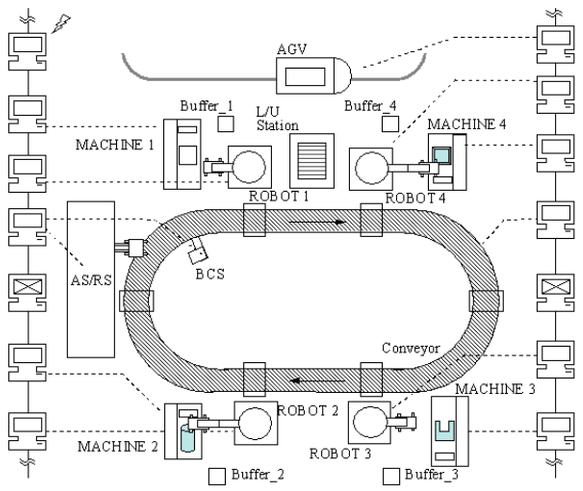


**Figure 4** Exemplary layout

Second, we create equipment control modules with layered architecture and abstraction. If a fractal is assigned to control machine 1, the fractal just plugs the equipment control module of machine 1. All equipment control modules serve same high level interfaces. Third, we develop some necessary agents and modules for the simulation such as a negotiation agent, a fractal status monitoring agent, etc. All the fractals are connected with each other through the TCP/IP. Therefore, it is possible to locate some fractals in other area.

To operate this shop floor controlled by fractals, it is necessary to realize goal-formation process, however; the research about this goal-formation process in on going. Therefore, we assume the process working and we assign well-defined goals to each fractal.

## 5. Conclusion and further research

We have briefly described the architecture of the FrMS, which is a new paradigm for evolving manufacturing environment. As one of the efforts of integrating research activities of FrMS, we design and develop the fractal template. The template is designed to create both agents and fractals. It supports dynamic restructuring process, fractal evolution process and other activities of fractals.

However, still there are lots of uncovered areas in FrMS, which is related to normally exiting problems in manufacturing systems. We will reform the template to be suitable for manufacturing systems and devise appropriate algorithms to improve productivity of overall systems.

## Acknowledgement

This work was supported by Korea Research Foundation Grant (KRF-2003-041-D00622).

## Reference

Eracar, Y.A. and Kolar M.M. (2000), An architecture for software that adapts to changes in requirements. Journal of Systems and Software, 50(3), 209-219.

IBM Japan, Aglets homepage, http://www.trl.ibm.com/aglets/index_e.htm.

Ryu, K-Y. and Jung M-Y. (2003), Agent-based fractal architecture and modeling for developing distributed manufacturing systems. *International Journal of Production Research*, 41(17), 4233-4255.

Shin, M-S. and Jung, M-Y. (2004), MANPro: Mobile Agent-based Negotiation Process for distributed intelligent manufacturing. *International Journal of Production Research*, 42(2), 303-320.

Shin, M-S, Cha, Y-P, Ryu, K-Y, and Jung, M-Y, (2003), In: Proceedings of the 8th Annual International Conference on Industrial Engineering, 1034-1039, Las Vegas.

Sun Microsystems, Java homepage, http://java.sun.com/

Tirpak, T. M., Daniel, S. M., LaLonde, J. D., and Davis, W. J. (1992), A Note on a Fractal Architecture for Modeling and Controlling Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 22; 564-567.