▩ 응용논문

UML 객체지향 기법을 이용한 자동생산시스템의 분산적 운용제어와 시뮬레이터에 관한 연구

- A Study on Heterarchical Control System and Simulator for Automated Manufacturing Systems Using UML Object-oriented Technique -

조 용 탁*
Cho, Yong Tak
한 영 근**
Han, Young Geun

Abastract

Developing control functions that operate and cooperate each equipment in order to achieve a goal is one of the most important problems in the installation of automated manufacturing systems. This paper discusses the development of a control system for heterarchical architecture and a simulator to verify operations of the control system. The object-oriented paradigm that has excellent reusability, portability, and extensibility is currently being used in many application fields as a software development methodology. Especially, UML(Unified Modeling Language), the third generation object-oriented modeling methodology, has advantages such as model generalization, clearness, and so on. In this research, software objects to accomodate the real time environments of automated manufacturing systems are modeled with the diagrams of UML. Based on these models, control software is developed as a format of pseudo-codes. A simulator is implemented to validate the developed control system.

1. 서론

최근에 들어 인건비의 상승, 제품의 다양성 및 라이프사이클의 감소 등의 요인에 따라 자동생산시스템의 중요성을 평가할 수 있다. 이러한 시스템을 설치하는데 있어서, 각 구성요소들이 주어진 목표를 향해 원활하게 협력하여 일을 수행되게 해 주는 운용제어 기능의 구현은 중요한 문제 중의 하나이다. 본 연구에서는 객체지향 기법을 이용하여 분산적 구조의 생산시스템을 유연하게 운용 제어할 수 있는 모델의 설계에 관하여 논의하고, 설계된 모델로부터 운용제어 소프트웨어 시스템을 구현한다. 구현된 시스템의 적합성 여부를 검증하기 위해 시뮬레이터를 개발하여 분산적 제어 기능을 평가한다.

생산운용제어시스템을 위한 소프트웨어를 모델링할 때 과거에는 구조적 분석 방법을 이용하여 왔지만, 본 연구에서는 시스템의 분석에서부터 구현에 이르기까지 품질을 향상시키고, 개발비용과 시간을 감소시키기 위해서 재사용성, 이식성, 확장성 등이 우수한 객체지향 방법을 이용한다.

이 논문은 교육부학술연구조성비(기계공학분야)에 의한 연구(과제번호ME96-E-08) 결과임

* ACS 엔지니어링 ERP팀
** 명지대학교 산업공학과

특히, 여러 연구자들에 의해 개발된 객체지향 방법 중에서도 객체지향 방법들의 장점을 통합한 UML (Unified Modeling Language)을 이용하여 정형화된 표기법에 따른 운용제어 시스템 모델링을 시도한다. 생산시스템 운용제어 구조에 관한 기존의 대부분의 연구는 상위 제어기에 의해 시스템 구성 요소들이 제어되는 중앙 집중적 구조나 계층적 구조의 운용제어에 관하여 진행되어 왔다. 그러나, 본 연구에서는 시스템 구성 요소간의 메시지 교환과 독립적 의사결정에 의해 시스템이 자율적으로 제어되는 분산적 운용제어 구조를 위한 시스템을 구현한다.

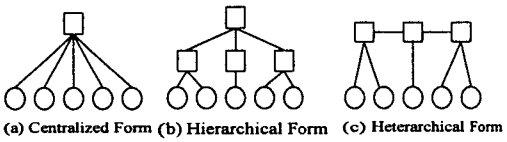
2. 자동생산시스템의 운용제어

2.1 운용제어의 정의

운용제어(Control)를 두 가지로 정의할 수 있다. 첫째는 기계나 설비 수준에서 이들이 목적에 알맞은 동작을 하도록 조절하는 것이다. 둘째는 기계나 설비가 모여 생산시스템을 이루었을 때, 시스템의 시작 및 종료, 필요한 작업을 수행하기 위한 명령 생성, 기계로부터의 피드백 정보 분석, 고장회복 등의 기능을 말한다(Groover, 1980; Han and Ham, 1995).

2.2 운용제어 구조

일반적으로 운용제어 구조의 유형은 중앙집중(Centralized) 구조, 계층적(Hierarchical) 구조, 분산적(Heterarchical) 구조로 나눌 수 있다. <그림 1>은 각각의 운용제어 구조를 보여주고 있다. 사각형은 제어기를 나타내고, 원형은 실제 제어되는 대상들을 나타낸다.



<그림 1> 운용제어 구조의 유형

2.3.1 중앙집중(Centralized) 운용제어 구조

이 구조는 <그림 1>의 (a)와 같이 한 대의 중앙컴퓨터가 현장에서 이루어지는 모든 생산 활동을 계획하고, 관련된 정보들의 처리 및 유지를 담당한다. 이 구조의 장점은 시스템에 필요한 모든 관련 정보들을 하나의 제어기에서 처리하기 때문에 전체적인 최적화를 가능하게 한다. 다른 운용제어 방식에 비해 제어기 비용을 줄일 수 있고 관리가 용이하다. 시스템에서 필요한 정보 및 상태를 쉽게 얻을 수 있다.

중앙집중 운용제어 구조의 단점은 첫째, 시스템의 규모가 커지는 경우 중앙컴퓨터에 부하가 많이 걸려. 상황에 실시간으로 대처하는 능력이 떨어지게 된다. 둘째, 중앙 컴퓨터에 이상이 생겼을 경우, 전체 시스템에 문제가 발생한다. 셋째, 한 장소에서 시스템 내의 모든 관련 자료를 다루기 때문에 운용제어 소프트웨어의 유지, 보수 및 확장이 어렵다 (Dilts, 1991).

2.3.2 계층적(Hierarchical) 운용제어 구조

이 구조는 <그림 1>의 (b)와 같이 시스템을 상대적인 상부 와 하부의 계층 별로 구성하여, 상부에서 하부로 제어명령을 보내고, 하부에서 상부로 상태정보를 보내는 방식을 취하는 운용제어 구조이다. 이 계층적 구조의 장점은 첫째, 구조가 계층별로 구성되어 있기 때문에 소프트웨어를 개발하는데 있어 모듈화가 가능하다. 따라서 관리나 확장이 용이한 편이다. 둘째, 각각의 계층별로 역할이 분담되어 있기 때문에 상황에 대처하는 반응시간이 빠르다.

반면에 단점은 첫째, 하위단계는 상위 단계에 의존하게되므로 어떤 부분에 이상이 생기면 그 하위

단계의 제어기들도 기능이 마비된다. 둘째, 시스템 구조를 초기 디자인 단계에서 확정, 설치하므로, 미래의 예측하지 못한 변화에 대처해서 수정, 확장하기 어렵다 (Dilts, 1991).

2.3.3 분산적(Heterarchical) 운용제어 구조

계층적 운용제어의 단점을 극복하기 위하여 제안되었는데 통신망과 분산 컴퓨팅 분야의 기술 발전이 그 배경이 되고 있다. 이 방식은 <그림 1>의 (c)와 같이 시스템 내에 계층적 구조가 존재하지 않는 대신, 시스템을 구성하는 구성원들 사이의 협력(cooperation)을 통해 원하는 목적을 달성한다. 전체적인 정보를 최소화시키고 대신에 지역 데이터베이스들을 활용하며, 제어기간의 통신을 통한 협상(negotiation)하에 의사결정이 이루어지는 운용제어 방식이다.

수평적 운용제어 구조의 장점은 첫째, 지역 제어기들이 독립적으로 작업을 수행하기 때문에 플러그 앤 플레이(plug and play) 개념을 도입할 수 있어 시스템의 확장이 용이하다. 둘째, 한 제어기의 이상발생이 전체 시스템에 미치는 영향을 최소화시킬 수 있다. 셋째, 현재의 현장 상태를 기반으로 의사 결정이 이루어지므로, 실시간 운용제어가 가능하다. 넷째, 시스템 구성 요소별로 모듈화 됨으로써 제어기 소프트웨어 복잡성이 줄어들며, 유지, 관리가 용이하다. 다섯째, 소프트웨어 개발비용이 저렴하다. 여섯째, 오류 상황 시에도 상당 수준의 동작이 가능하게 해 주는 능력인 fault tolerance를 갖는다.

이 운용제어 구조의 단점은 첫째, 지역적인 최적화만 가능하고 전체적인 최적화는 불가능하다. 둘째, 제어기를 구현하는데 기술적인 제약이 있을 수 있다. 셋째, 통신 프로토콜과 네트웍에 관한 표준화가 부족하여 실제 복잡한 현장에서 구현되기는 어려움이 있다 (Duffie, 1988; Dilts, 1991).

3. 객체지향 기법

3.1 객체지향의 개념

1990년대 이후 컴퓨터 하드웨어 기술이 빠르게 변화되고 있음에 비해 소프트웨어 기술은 예전 수준에서 크게 개선을 이루지 못하였다. 이러한 소프트웨어의 생산성 위기를 극복하기 위해 객체지향 개념이 소개되었다. '객체지향'이라는 의미는 객체들의 집합으로 구성된 시스템을 소프트웨어의 설계와 개발에 이용하는 것으로서(Smith, 1992), 객체는 데이터와 그 데이터를 처리하는 로직인 메소드로 구성된다. 객체 자신이 가질 수 있는 데이터를 속성(Attribute)이라 한다. 객체지향 개념의 핵심은 프로그램 코드를 재사용하여 소프트웨어의 생산성을 향상시키고 보다 효율적으로 소프트웨어의 생산을 관리하는 것이다. 이를 위해 객체지향기법은 캡슐화(Encapsulation), 상속성(Inheritance), 다형성(Polymorphism)과 같은 세가지 특성을 갖고 있다.

캡슐화는 "데이터와 데이터를 조작하는 메소드을 하나로 묶는 것"을 말한다(최영근과 허계법, 1995). 상속성은 공통적인 특성을 가진 기반 객체(ancestor, supertype)를 생성한 후 그 객체의 특성을 그대로 이어받는 새로운 파생 객체(decendant, subtype)를 생성하는 것이다. 다형성은 서로 다른 객체에서 같은 이름을 가진 함수를 사용해 사용자에게 같은 이름을 제공하여 함수 이름에 대한 일관성을 제공하고 객체별로 객체 자신의 특성에 맞는 서로 다른 처리를 하는 경우에 사용될 수 있다.

3.2 기존의 객체지향 모델링 기법

Rumbaugh의 OMT(Object Modeling Technique)를 이용한 객체지향 개발 방법은 주로 응용 영역 개념(application-domain concept)을 중심으로 기술되었다.

Booch의 방법은 Ada를 이용한 소프트웨어 개발에 초점을 두었으므로 객체와 클래스, 객체간의 상호 작용에 대한 풍부한 개념을 지원하고 있다. 이 방법은 실시간 시스템이나 병행성을 지닌 시스템을 모형화하기에 가장 바람직하다.

Coad와 Yourdon의 방법의 특징은 체계적인 방법이 아닌 대화적으로 개발을 진행해 나가는데 있다. 그러나 이 방법은 객체간의 통신 지원 개념이 취약하고 대규모 프로젝트보다는 작은 시스템 개발에 적합하다는 단점이 있다.

그 외 다른 연구자들이 각기 다른 모델링 방법들을 제시하고 있는데, 위에서 기술한 바와 같이 이들마다 표기상의 차이가 있으며, 각기 다른 적용대상과 범위를 지니고 있다 (최영근과 허계법, 1995; 최성운 등, 1996).

3.3 UML(Unified Modeling Language)

최근에 일부 연구자에 의해 위에서 기술된 여러 방법들이 가지고 있는 불필요하거나 잘 사용되지 않는 모델도구들을 제거하고, 사용자들의 혼동 요인을 최소화시켜 객체지향 개발 방법론의 표준화를 시도하는 움직임이 있었다. Booch, Jacobson, Rumbaugh 등에 의해 개발된 제 3세대 객체지향 방법론인 UML은 객체지향 시스템을 분석할 뿐만 아니라, 소프트웨어 개발 목적으로 시스템을 규정, 기록, 가시화, 구축하는 언어이며, 동시적, 분산적인 시스템을 모델링하는 기법도 제공한다. 또한 향후 객체지향 시스템을 개발하기 위한 표준 모델링 방법으로 널리 사용될 것이 예상된다 (Booch, et al., 1997). UML은 복잡한 시스템의 구조, 기능, 데이터 관계 등을 모델링하기 위한 다양한 모델링 도구들을 제공하고 있다(Rational Software Corporation, 1996).

3.4 객체지향 기법의 생산시스템의 응용 사례

Adiga는 FMS 시뮬레이션을 위한 FMS 객체지향 모델을 개발하였다. Entity-relationship 모델과 메시지 흐름도(message flow diagram)를 이용하여 시스템을 구성하는 객체를 정의하고, 시스템에 관련된 데이터(data)와 절차(procedure)를 도출했다 (Adiga, 1990).

Smith는 일반적인 현장 운용제어 시스템(shop floor control system)을 개발하기 위한 방법론과 이로부터 객체지향적인 제어 소프트웨어를 작성하였다. 시스템을 제어하는 제어기 클래스를 도출한 후 Message Based Part State Graph (MPSG)를 이용하여 각 제어기가 지시할 수 있는 명령을 도출해 낸다. 제어기 클래스와 MPSG를 약간 수정함으로써 다른 시스템에 적용하여 생산시스템 운용제어 소프트웨어를 자동 생성할 수 있도록 했다 (Smith, 1992)

Lin은 로봇에 의해 공작물이 이동되는 FMC를 운용제어하는 객체지향 소프트웨어를 Rumbaugh의 OMT (Object Modeling Technique)을 이용하여 개발하였다. 셀을 구성하는 객체와 셀이 운용제어되기 위하여 필요한 객체를 도출하고, 각 객체들이 가질 수 있는 상태(state)를 정의한 후, 각 객체들이 메시지의 전달에 의해 운용제어되는 과정을 모델링하고, 그 모델을 기반으로 제어 소프트웨어를 개발한 것이다.(Lin, 1994)

4. 운용제어 시스템의 개발

4.1 분산적 운용제어 구조의 적용

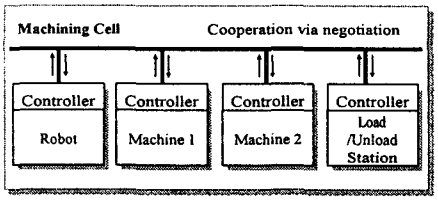
본 연구에 적용한 분산적 운용제어구조에는 상하의 계층이 존재하지 않고, 각 구성요소들을 운용제어하기 위해 동일한 수준의 제어기만 존재한다. 즉, 동일한 수준에서 필요한 정보를 요청하는 메시지와 그 메시지에 대한 회답 메시지를 전달하는 기능만 갖는다. 각 구성요소의 제어기는 각자가 해야 할 일을 결정하기 위하여 다른 구성요소의 상태 정보를 수집하여 다른 제어기들과 협상한다. 또한, 다른 제어기로부터의 상태정보 요청이 있을 경우 그 요청에 답해준다. 예를 들어 <그림 2>와 같이 각 구성요소에는 동일한 계층의 제어기가 있는 경우 제어기 사이의 상호 메시지 전달에 의해 협상(negotiation)한 후, 구성요소 자신이 수행할 작업을 결정하여 수행하게 된다.

4.2 운용제어 시스템의 UML 모델

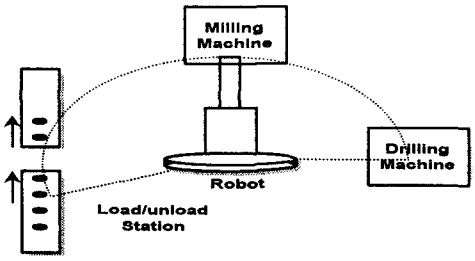
생산 운용제어 시스템은 실시간의 현상에 응답할 수 있어야 하는데, 이러한 응답 매커니즘을 UML로

처리하기가 용이하다. UML 모델은 시스템의 분석 및 설계 단계에 적합한 모델과 소프트웨어를 구현하는 단계에 적합한 모델으로 나눌 수 있다(Rational Software Corporation, 1996).

본 연구에서는 시스템의 분석 및 설계에 적합한 모델 중 Class Diagram, Sequence Diagram, State Diagram을 이용하여 시스템내의 객체를 도출하고 운용제어 기능을 모델링한다. 적용될 FMC 모델은 <그림 3>과 같이 적재/하적용 로봇 1대, 가공기계 2대, 공작물의 투입 및 반출 공간인load/unload station으로 구성되어 있다. 이와 같은 하드웨어 외에 운용제어를 위해 필요한 구성 요소로 Process Plan 데이터베이스, NC 프로그램 데이터베이스 등이 있다.



<그림 2> 분산적 운용제어 구조



<그림 3> 대상 자동생산시스템의 구성도

4.2.1 운용제어 시스템의 Class Diagram

UML의 Class Diagram은 자동생산시스템을 구성하는 클래스와 클래스간의 관계를 기존의 방식보다 더욱 상세하고 명확하게 보여 줄 수 있다. 운용제어 시스템에서 도출될 수 있는 객체를 물리적 객체와 데이터베이스로 나눌수있다. 물리적 객체는 load/unload station, robot, machine, parts이고, 데이터베이스는 process plan, NC program이다.

각 객체는 속성(attribute)와 메소드(method)로 구성되어 있다. <표 1>은 각 객체에 대한 속성들을 열거한 것이고, <표 2>는 각 객체에 대한 메소드를 열거한 것이다.

Class Diagram은 모델의 class, type, class간의 내부 구조, 다른 class와의 관계와 같은 정적 구조를 보여준다. 시간 변화에 따른 정보는 보여주지 않지만, 시스템 개발 단계에 따라 압축된 클래스 형태, 분석 단계의 클래스 형태, 구현 단계의 클래스 형태로 클래스를 표기할 수도 있다.

<표 1> 객체 클래스의 속성 정의

객체 클래스	속성
load/unload station	station_id, station_name, completed_part
robot	robot_id, robot_name, robot_state
machine	machine_id, machine_name, machine_state, NC_prog_id
part	part_id, part_name, process_id
process plan	process_id, prcess_plan_name, operation_sequence, operation_name, operation_id, NC_prog_id
NC prog DB	NC_prog_id, NC_program

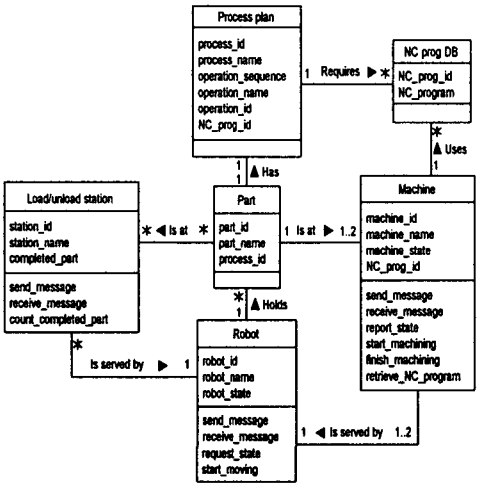
<표 2> 객체 클래스의 메소드 정의

객체 클래스	메소드
load/unload station	send_message, receive_message, count_completed_part
robot	send_message, receive_message, request_state, start_moving
machine	send_message, receive_message, report_state, start_machining, finish_machining, retrieve_NC_program

<그림 4>는 분석 단계의 Class Diagram이다. class을 세 부분으로 나뉜 사각형으로 표기하며, 사각형의 맨 위부터 아래로 class name, attributes, operations을 기입한다. 관련되지 않은 클래스들간의 의미적 연결을 association이라 한다. 각 클래스를 연결하는 선 위에 쓰인 동사 및 동사구는 association name을 나타내는데, association name 옆에 있는 작은 삼각형이 가리키는 방향으로 association name을 읽어가며 두 클래스간의 관계를 분석한다. 연결선에 사용된 숫자 및 *는 클래스로부터 상속된 인스턴스 객체(instance object) 개수를 의미한다. 예를 들어 <그림 4>에서 한 개의 part는 한 개의 process plan과

연결되고, 여러 개의 part가 한 대의 robot에 의해 처리된다. 한 대의 machine은 다수의 NC program을 처리한다. 하나의 process plan은 다수의 NC program을 필요로 한다.

<그림 4>와 같은 형태의 Class Diagram을 이용하여 일대다, 다대다 관계를 관계형 데이터베이스로 설계할 경우 중복된 레코드(record)가 발생할 수 있다. 이러한 문제점을 해결하기 위하여 일대다 관계인 경우 하나의 인스턴스를 갖는 쪽의 주키(primary key)를 여러 개의 인스턴스를 갖는 쪽의 외부키(foreign key)로 사용하고, 다대다 관계인 경우 테이블 한 개를 추가하여 세 개의 테이블로 두 테이블 간의 관계를 설계한다. 예를 들면, <그림 4>에서 하나의 process plan은 다수의 NC program을 가질 수 있다. 이 경우 process plan 클래스의 주키를 NC program DB 클래스의 외부키로 사용해야 한다.



<그림 4> 운용제어시스템의 Class Diagram

4.2.2 FMC 운용제어 시스템의 Sequence Diagram

UML의 Sequence Diagram은 객체간에 교환되는 메시지를 시간적 순서에 따라 명확히 보여준다. 따라서 전체 시스템이 작동되는 과정을 순차적으로 쉽게 분석할 수 있다. 다이아그램에서 세로 막대는 객체가 존재하는 기간을 나타내고, 세로 막대의 하단부로 갈수록 시간이 경과되는 것을 보여준다.

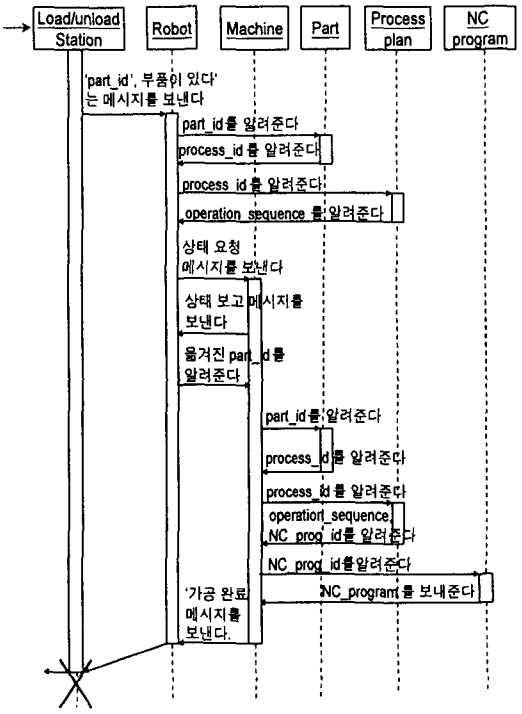
본 연구에서 대상으로 하는 자동생산시스템의 운용제어 순서와 메시지 교환 형태를 Sequence Diagram으로 나타낸 것이 <그림 5>이다. load/unload station에 새로운 공작물이 도착하면, load/unload station은 robot에게 part_id와 공작물이 있다는 메시지를 보낸다. robot은 part에게 part_id를 알려주어 process_id를 전달받고, process_id를 process plan 객체에게 알려주어 operation sequence를 전달받는다. 전달받은 operation sequence에 따라 robot이 공작물을 이동시켜야 할 machine을 결정한다. machine이 결정되면 robot은 machine의 상태정보를 요청하는 메시지를 보내고 machine으로부터 상태정보를 받은 후 machine에게 part_id를 알려준다. machine은 part_id를 part에게 전달하여 part로부터 process_id를 전달받은 후 process_id를 process plan에게 보내어 operation_sequence와 NC_program_id를 전달받는다. machine은 NC program 객체에게 NC_program_id를 보내어 NC program을 전송 받은 후 공작물을 가공하고, 가공이 완료되면 가공완료 메시지를 robot에게 보낸다. 이와 같은 순서에 의해 한 공작물에 대한 메시지 흐름을 <그림 5>와 같이 모델링할 수 있다.

4.2.3 운용제어 시스템의 State Diagram

시스템 구성요소의 상태(State)

load/unload station은 '공작물이 있다(part)', '공작물이 없다(no part)'의 두 가지 상태를 갖는다.

machine은 '공작물이 없고 기계 비가동(idle, no part)', '가공 중(machining)', '공작물이 장착된 상태로 기계 비가동(idle, part)'이라는 상태를 갖는다. robot은 '정지(idle)', '이동중(moving)'이라는 상태를 갖는다.



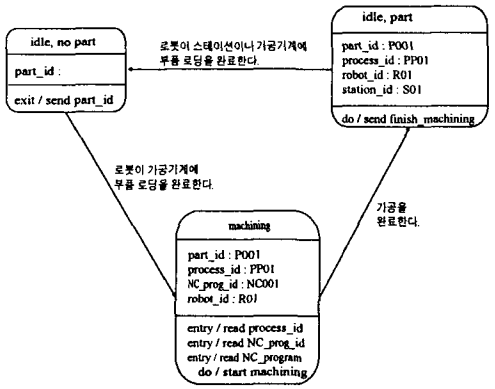
<그림 5> 운용제어시스템의 Sequence Diagram

State Diagram

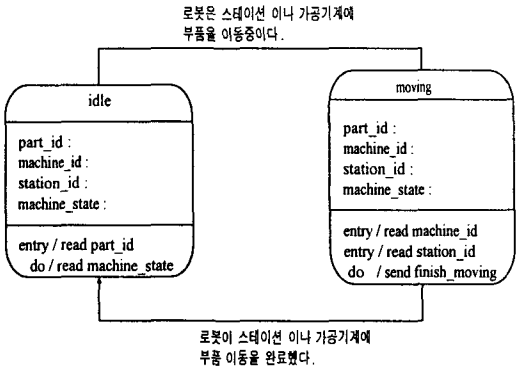
State는 한 객체가 시스템 내에서 갖는 상황 또는 조건이고, 객체의 상황 또는 조건이 변경되기 전까지는 임의의 한 상태에 있게 된다. UML의 State Diagram은 외부의 요청이나 메시지에 의해 반응하여 객체들과 이들의 상호작용이 취할 수 있는 state의 순서를 도시화한 것이다. state는 둥근 사각형으로 표현되고, state name, state variable, internal activity로 구성되어 있다. state variable은 attribute와 같은 형태와 의미를 갖으며, internal activity는 임의의 객체가 어떤 state에 있는 동안에 수행할 수 있는 action이다. 임의의 state로 바뀌기 직전에 수행해야 할 action은 'entry'이고, 임의의 state에서 나가는 동안에 수행해야 할 action은 'exit'이며, 임의의 state에 있는 동안에 수행해야 할 action은 'do'이다.

<그림 6>과 <그림 7>은 각각 machine과 robot의 동적인 state 변화를 모델링한 것이다. <그림 6>을 예를 들어 설명하면 robot이 공작물을 machine에 옮겨 놓으면 machine의 상태가 'idle, no part'에서 'machining'으로 변경된다. 'idle, no part' 상태에서 'machining' 상태로 변경되기 전에 machine은 part_id를 part에게 보내고, 'start_machining'이라는 가공명령을 내린다. 또한 'machining' 상태로 변경되기 직전에 part_id에 해당하는 process_id, NC_progeam_id, NC_program을 읽어들이고, 'machining' 상태에서 가공을 하고, 'machining' 상태에서 'idle, part' 상태로 변경되기 전에 'finish_machining'라는 명령을 내리면, 'machining' 상태에서 'idle, part' 상태로 변경된다. robot이 machine에 있는 공작물을 다른 machine이나 load/unload station에 옮기면 'idle, part' 상태에서 'idle, no part' 상태로 바뀐다.

이와 같이 복잡하고 실시간 시스템에서의 운용제어과정을 정확히 묘사할 수 있고, 실제 기계 제어코드를 생성하기 위해서도 정확한 state 표현이 요구되어 State Diagram을 사용하게 되었다.



<그림 6> 운용제어시스템의 State Diagram 1

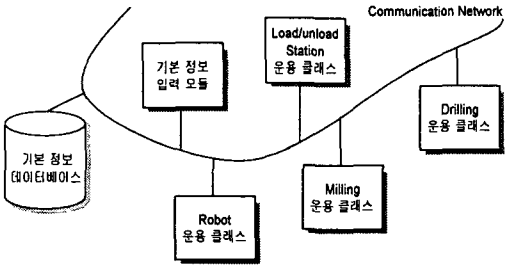


<그림 7> 운용제어시스템의 State Diagram 2

4.2.4 운용제어 시스템의 구현

본 연구에서 구현된 운용제어 시스템은 로봇제어 클래스, load/unload station 제어클래스, milling machine 제어클래스, drilling machine 제어클래스가 수평적, 분산적으로 위치하고, 이들의 제어 기능에 필요한 데이터를 위한 기본 정보입력 모듈과 데이터베이스로 구성된다 <그림 8>. 앞에서 개발된 UML 모델을 기반으로 운용제어를 위한 네 가지 객체의 가상코드(Pseudo-code)가 구현되었다. 가상코드란 프로그램 수행의 일반적인 형태를 보여주는 코드로서, 실제 활용을 위해서는 공급업체에 따라 상이한 제어기에 맞추어 후처리가 필요하다.

Robot 운용제어 객체의 주요 기능은 공작물의 가공순서에 대한 정보를 파악하여 공작물이 옮겨질 위치를 machine들과 협상을 통해 결정한 후, 해당위치에 공작물을 옮기는 것이다. <그림 9>는 로봇이 어떤 공작물을 load/unload station에서 process plan에 있는 가공 순서에 의해 우선 가공해야할 기계로 이동시킬 경우의 가상코드의 일부를 나타낸 것이다.



<그림 8> 운용제어 시스템 구조도

```
retrieve(part_id, process_plan);
if operation_sequence = machine_A
  { request machine_A_state;
    receive machine_A_state;
    decide location;
      if machine_A = idle, no part
        moving (part, from station, to machine_A)
      if machine_A = idle, part
        retrieve(part_id, process_plan)
          if next operation = machine_B
            request machine_B_state
              if machine_B = idle, no part
                moving (part, from machine_A, machine_B)
              if machine_B = idle, part
                moving (part, from machine_B, to station)
              if machine_B = machining
                wait for machining
        if next operation = null
          moving (part, from machine_A, to station)
      if machine_A = machining
        wait for machining }
```

<그림 9> 로봇 운용 클래스의 가상코드

Load/unload station 운용제어 객체의 주요 기능은 가공할 공작물이 있다는 신호와 모든 대기 공작물의 가공이 끝났다는 메시지를 로봇에게 전해주는 것이다.

Milling machine 운용제어 객체와 Drilling machine 운용제어 객체의 주요 기능은 공작물이 가공 완료되었다는 메시지와 자신의 상태 정보를 로봇에게 전달해 주고, 로봇으로부터 공작물이 자신에게로 장착되었다는 메시지를 받으면, 필요 가공 데이터를 공급받아 공작물을 가공하는 것이다.

각 객체의 가상코드를 이용하여 Robot 객체, Load/unload 객체, Machine 객체가 가져야 할 메소드를 구현한다. 객체지향의 장점을 활용하자면, 이렇게 구현된 메소드를 재사용할 수도 있고, 기존의 메소드를 약간 수정하여 사용할 수도 있다. 예를 들면, machine 객체가 동일한 기능을 갖는 machine 객체로 바뀌었을 경우 기존의 메소드를 재사용할 수 있고, 시스템 구성 요소로서 다른 기능을 갖는 상이한 machine 객체가 추가되었을 경우 기존의 메소드를 약간 수정하여 사용할 수 있다.

4.3 UML 기법의 도입 효과

UML이라는 객체지향 기법을 도입하여 생산운용제어 시스템을 분석/설계함으로써 다음과 같은 효과를 얻을 수 있다.

첫째, UML에서 제공하는 모델링 도구인 Sequence Diagram과 State Diagram을 이용함으로써 실시간 처리가 요구되는 시스템도 명확히 모델링할 수 있게 되었다. 둘째, UML에서 제공하는 모델링 도구의 명확성 때문에 다른 시스템 분석가나 시스템 사용자일지라도 시스템을 쉽게 분석하고 재설계할 수 있게 되었다. 셋째, UML은 여러 객체지향 기법들의 장점을 통합한 것이므로 모델링 방법 재교육에 많은 노력이 필요하지 않다.

5. 분산 운용제어 시뮬레이터

5.1 시뮬레이터의 구성

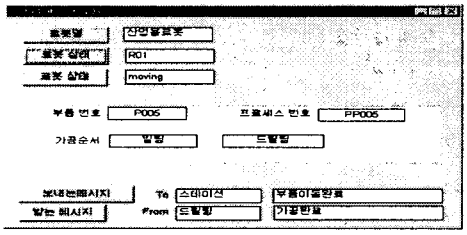
본 연구에서 개발된 분산 운용제어 시스템의 적합성 여부를 검증하기 위하여 운용제어 시뮬레이터를 개발하였다. 이 시뮬레이터는 네 개의 운용제어 모듈과 한 개의 기본정보 입력 모듈로 구성되어 있다. 각 운용제어 모듈은 서로 통신하며, 서로 다른 모듈로부터 상태정보를 받아서 자신이 수행해야 할 일을 스스로 결정한다. 시뮬레이터 개발 과정에서 주로 구성요소의 가공상황과 구성요소의 상태변화에 중점을 두었고, 치공구와 공구 교환을 위한 준비 시간, 고장 수리에 대한 상태 등은 고려되지 않았다. 시뮬레이터는 PC 상에서 응용프로그램 개발 툴인 PowerBuilder를 이용하여 개발되었다.

Robot 운용제어 모듈을 이용하여 <그림 10>과 같이 로봇의 상태, 로봇이 이동될 출발지와 목적지, 현재 이동 중인 공작물에 대한 정보, 프로세스 번호, 가공 순서, 전송/수신 메시지 등을 알 수 있다. Load/unload station 운용제어 모듈을 이용하여 <그림 11>과 같이 가공된 공작물 수와 load/unload station에 있는 공작물 총 수, 전송/수신 메시지를 알 수 있다. Milling machine 운용제어 모듈과 Drilling machine 운용제어 모듈을 이용하여 <그림 12, 13>과 같이 가공 기계의 상태정보, 전송/수신 메시지를 알 수 있다. 기본 정보 입력 모듈을 이용하여 <그림 14>와 같이 시스템의 운용제어를 위해 필요한 기본 정보인 공작물 정보, process plan 정보, NC program 정보, 기계 정보를 입력, 수정, 삭제, 조회할 수 있다.

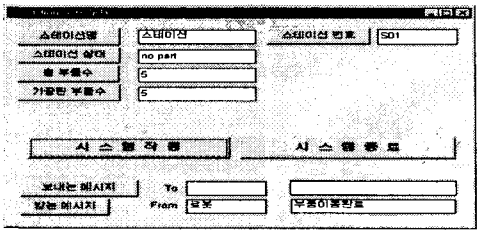
5.2 운용제어의 모의 실험 사례

운용제어 시스템은 유사한 가공 공정을 갖는 다양한 공작물에 대하여 유연하게 가공할 수 있어야 하므로, <그림 15>와 같이 밀링 가공만 필요한 부품, 드릴링 가공만 필요한 부품, 밀링 가공 후 드릴링 가공이 필요한 부품, 드릴링 가공 후 밀링 이 필요한 부품 등의 네 가지 공작물을 임의적 순서로 가공해야 하는 상황을 대상으로 모의 실험을 수행하였다.

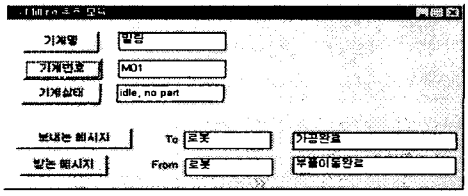
시간 변화에 따른 각 구성요소의 상태 변환 내용과 메시지 교환 내용이 파일의 형태로 저장되는데, <표 5>는 실험 결과 중 <그림 15> (c)의 공작물을 위한 운용제어의 내용을 보여준다. <표 5>에서 'From'은 메시지를 보내는 모듈이고, 'To'는 메시지를 받는 모듈이다. '상태'는 메시지가 전달되기 전의 구성요소 상태를 나타낸다. 또한 '메시지'란에 전달되는 메시지가 기입되지 않은 부분은 메시지 전달 직전 또는 직후의 구성요소의 대기 상태를 나타낸 것이다.



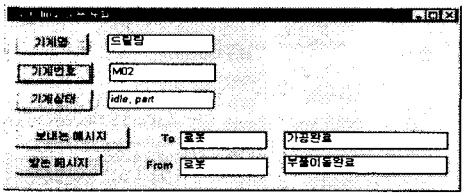
<그림 10> Robot 운용제어 모듈



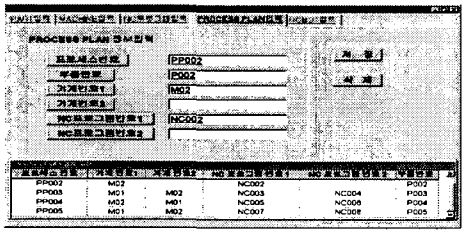
<그림 11> L/UL station 운용제어 모듈



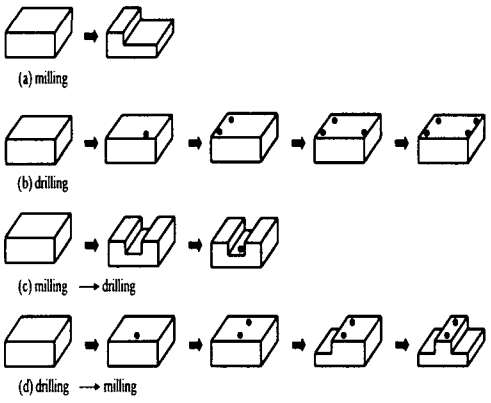
<그림 12> Milling machine 운용제어 모듈



<그림 13> Drilling machine 운용제어 모듈



<그림 14> 기본정보입력 모듈



<그림 15> 실험대상 부품의 공정

<표 8> 시뮬레이션 결과의 일부

공작물번호	P003		
	구성요소명	메시지	구성요소상태
From	스테이션	공작물있음	part
To	로봇		idle
From	로봇	상태정보요청	
To	밀링		idle, no part
From	밀링	상태정보회답	
To	로봇		
From	로봇	공작물이동완료	moving
To	밀링		
From	밀링	가공완료	machining
To	로봇		
From	밀링		idle, part
To			
From	로봇	상태정보요청	
To	드릴링		idle, no part
From	드릴링	상태정보회답	
To	로봇		
From	로봇	공작물이동완료	moving
To	드릴링		
From	드릴링	가공완료	machining
To	로봇		
From	드릴링		idle, part
To	로봇		moving
From	드릴링		idle, no part
To			
From	로봇	공작물이동완료	moving
To	스테이션		
From	로봇		idle
To			

본 사례와 몇 가지 다른 경우를 가정하여 시뮬레이터를 구동시켜 본 결과, 본 연구에서 객체지향 방법으로 모델링되어 구현된 분산 운용제어 시스템이 계획, 설계된 방향으로 동작함을 검증할 수 있었다.

6. 결론

본 연구에서는 객체지향 기법을 적용하여 자동생산시스템을 위한 운용제어 모델을 설계한 후 분산적 구조의 운용제어 시스템을 개발하였으며, 개발된 운용제어 시스템의 기능 적합성을 PC상에서 검증하기 위한 시뮬레이터를 개발하여 모의 실험 결과 개발된 제어 시스템의 동작을 확인할 수 있었다.

UML을 시스템 개발에 적용함으로써 객체지향 기법들간의 상이한 표기법과 적용 범위의 제한을 극복할 수 있게 되었다. 또한 UML에서 제공하는 Sequence Diagram과 State Diagram을 이용하여 시스템을 모델링하면 본 운용제어 시스템과 같이 실시간 처리를 요구하는 시스템 분석 및 설계에서는 그 효과가 더욱 크다고 할 수 있을 것이다.

향후에는 고장 및 오류 발생 감지 및 처리에 관한 사항, 객체지향 데이터베이스 적용, 기계 레벨의 프로그램 코드를 생성할 수 있는 후처리기(post-processor)등에 대한 연구가 진행되어야 할 것이고, 그 결과를 통해 실제 구축된 자동생산시스템에 적용하려는 시도를 하여야 할 것이다.

참고문헌

[1] 최영근, 허계법, 객체지향 소프트웨어 공학, 한국 실리콘, 1995.

[2] 최성운, 도홍석, 계원경, "객체지향 소프트웨어 개발 방법론 동향", 정보과학회지, 14(10), pp. 4-11, 1996.

[3] Adiga, S. and Gadre, M., "Object-oriented Software Modeling of a Flexible Manufacturing System," Journal of Intelligent Robotic Systems, Vol. 3, pp. 147-165, 1990.

[4] Booch, G., Jacobson, I., and Rumbaugh, J., "The Unified Modeling Language for Object-oriented Development," Documentation Set Version 1.0, 1997.

[5] Dilts, D. M., Boyd, N. P., and Whorms, H. H., "The Evolution of Control Architectures for Automated Manufacturing Systems", Journal of Manufacturing Systems, Vol. 10, No. 1, pp.79-93, 1991.

[6] Duffie, N. A., Chitturi, R. and Mou, J., "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities", Journal of Manufacturing Systems, Vol. 7, No. 4, pp. 315-328, 1988.

[7] Groover, M. P., Automation, Production Systems, and Computer-Integrated Manufacturing, Prentice-hall, 1980.

[8] Han, Y. G. and Ham, I., "Petri Net Based Modeling of Heterarchical FMS Control," Transactions of NAMRI, Vol. 23, pp299-305, 1995.

[9] Lin, L., Masatoshi W., and Adiga, S., "Object-Oriented Modeling and Implementation of Control Software for A Robotic Flexible Manufacturing Cell," Robotics & Computer-Integrated Manufacturing, Vol. 11, No. 1, pp.1-12, 1994.

[10] Rational Software Corporation, Unified Modeling Language for Real-Time Systems Design, Rational Software Corporation, 1996.

[11] Smith, J., "A Formal Design and Development Methodology for Shop Floor Control in Computer Integrated manufacturing", Ph.D. Dissertation, The Pennsylvania State University, University Park, Pennsylvania, U.S.A., 1992.