論文 94-19-10-9

분산 FMS의 통합제어를 위한 객체지향 데이타베이스

正會員 박 장 호* 正會員 차 상 균*

Object-Oriented Database for Integrated Control of Distributed FMS

Jang H. Park*, Sang K. Cha*

要 約

FMS(Flexible Manufacturing System)는 로봇, 수치제어기기 등의 프로그램이 가능한 생산설비 하드웨어로 구성된 분산시스템이다. FMS의 효과적인 운용을 위해서는 생산설비 하드웨어를 통합 제어하기 위한 소프트웨어의 존재가 필수적이다. 그런데 FMS의 복잡성과 다양한 변화가능성은 통합제어 소프트웨어의 구축 및 유지, 보수를 어렵게 한다. 이를 해결하기 위하여 본 논문에서는 객체지향 FMS 통합제어 모델을 제안한다. 제안된 모델에서는 FMS에 실재하는 셀이 가상 셀 객체로 모델되어, 통합제어를 위하여 필요한 데이타와 연산들이 캡슐화되어 객체지향 데이타베이스에 관리된다. 또한 이밖의 FMS운용을 위해 필수적인 데이타와 이들간의 관계성도 데이타베이스에 관리된다. 본 논문은 제안된 모델을 기반으로 구현한 통합제어 소프트웨어시스템인 FREE(Fms Run-time Executive Environment)를 바탕으로 통합제어 데이타베이스에 대하여 기술한다.

Abstract

FMS is a distributed system composed of various programmable manufacturing hardware such as robots and NC machines. For the autonomous operation of such a system, an integrated software layer for the control and monitoring is needed on top of the manufacturing hardware. However, constructing and maintaining such a software layer is difficult due to the complexity of a underlying FMS and its frequently changing nature. To cope with this problem, this paper proposes an object-oriented FMS integration model, in which objects acting as virtual manufacturing cells are instantiated for each physical cell in the underlying object-oriented database. Various other entities involved in manufacturing processes and their relationship to the cell objects are also captured in the database. This paper describes the structure of this object-oriented FMS database on our prototype implementation called FREE.

*. 서울大學校 制御計測工學科
Dept. of Control and Instrumentation Eng., Seoul Nat'l University
論文番號 : 94135
接受日字 : 1994年 5月 17日

I. 서 론

FMS는 시스템 통합의 관점에서 보면, 가공, 조립 등과 같은 생산 단위공정을 수행하는 복수의 셀(FMC : Flexible Manufacturing Cell)과, 각 셀의 상태를 추적하고 제어하기 위한 제어호스트, 그리고 이들을 연결하는 통신망으로 구성된 분산시스템이다[1]. 각 셀은 다시 로봇, 수치제어기기(NC machine) 등과 같이 프로그램이 가능한 생산설비 하드웨어와 셀 제어기로 구성된다. 그림 1은 이와 같이 계층구조를 갖는 분산 FMS의 예를 보여준다. 분산 FMS의 효율적 제어를 위하여 통합제어를 위한 소프트웨어의 존재는 필수적이다. 제어호스트에 내재하는 통합제어 소프트웨어는 각 셀의 작업을 계획, 수행을 지시하고, 그 수행을 감시하기 위한 제어프로그램과 이를 지원하기 위하여 셀 및 생산설비의 상태, 생산계획 등의 정보를 관리하는 통합제어 데이타베이스로 구성된다.

그런데 FMS에는 제품의 설계 변경이나 품종 변경 등으로 인한 생산 공정의 변화, 새로운 셀의 추가와 같은 FMS 구성상의 변화와 같은 여러 변화 가능성이 존재한다. 따라서 통합제어 소프트웨어가 이러한 변화를 유연하게 수용할 수 있어야만 성공적인 FMS의 구축이 가능하다. 이를 위해서는 주어진 변화를, 가능하다면 제어프로그램의 수정없이 데이타의 추가, 삭제, 또는 최소한의 데이타베이스 스키마의 변경만으로 수용할 수 있다면 가장 바람직하고 제어프로그램의 수정이 불가피한 경우에도 그 일부만 국지적으로 수정하면 되도록 함이 바람직하다. 결국 통합제어 데이타베이스의 유연성은 성공적인 FMS 구축을 위한 선결 요구조건이 된다.

본 연구에서는 통합제어 데이타베이스 설계, 구현에 객체지향 데이타베이스 기술을 적용하여 유연성을 지원하고자 한다. 이를 위하여 FMS에 실재하는 셀(physical cell)을 가상 셀(virtual cell) 객체로 모델하여, 제어 과정을 이들 객체간의 상호작용으로 이해한 객체지향 FMS 통합제어 모델을 제안하고, 제어를 위해 필수적인 데이타를 중심으로 데이타베이스를 설계하였다. 설계된 데이타베이스는 FMS 상의 복잡한 데이타와 이들 사이의 관계를 자연스럽게 표현할 수 있고, 데이타와 핵심연산이 클래스로 정의되어 함께 관리하므로, 이를 기반으로 제어프로그램을 구현할 경우, 그 크기가 작고 대부분의 변화를 데이타베이스 레벨에서 수용할 수 있다[2][3]. 스키마를 변경할 필요가 있는 경우에도 다형성(polymorphism)을 이용하면 제어프로그램의 수정을 최소화하면서 수용할 수 있다[4]. 본 연구에서는 또한 제안된 모델을 기반으로 통합제어 소프트웨어시스템인 FREE를 구현하였다.

통합제어 데이타베이스의 설계, 구현을 위한 이전의 연구는 주로 관계형 데이타베이스[5]를 기반으로 진행되어 왔다[6][7][8]. 그런데 관계형 데이타베이스는 테이블 형태의 간단한 자료 구조만을 제공하기 때문에 FMS에서 나타나는 복잡한 형태의 데이타와 이들 사이의 관계를 표현하는데 한계를 지니며, 데이타에 대한 연산의 대부분이 제어프로그램에 분산, 중복되어 그들간의 일관성 유지가 어렵다[9]. 제어프로그램은 SQL과 C언어와 같은 절차언어의 조합으로 구현되는데, 각 언어간의 impedence mismatch로 인하여 구현 및 유지, 보수가 어렵다. 이는 결과적으로 FMS상의 변화를 통합제어 소프트웨어에 반영하는 것을 어렵게 한다. 성공적인 FMS의 구축을 위하여 이밖에도 여러 분야의 노력이 진행되고 있다. 본 연구에서 적용한 객체지향 개념은 주로 FMS의 시뮬레이션 분야에서 이용되어 왔다[10][11]. 또한, FMS 제어구조[12][13][14], 이상 회복[15][16], 스케쥴링[17][18]에 대한 연구 등이 진행되고 있다.

본 논문의 구성은 다음과 같다. 2절에서는 객체지향 FMS 통합제어 모델을 제안하고, 3절에서는 설계된 데이타베이스의 객체 스키마에 대하여 기술한다. 4절에서는 FREE에 대하여 기술하고, 마지막으로 5절에서는 결론과 향후 연구방향을 제시한다.

II. 객체지향 FMS 통합제어 모델

통합제어 모델은 FMS에 실재하는 셀을 가상 셀 객체로 모델한다. 객체지향 개념에 의하여 Cell 객체에는 통합제어를 위하여 필요한 데이타와 실제 셀을 제어하고 메시지를 주고 받고 처리하기 위한 연산들이 메소드(method)로 정의되어 캡슐화(encapsulation)된다. 그림 2는 이를 개념적으로 보여준다. 그림에서 사선화살표는 실재 셀과 Cell 객체가 1:1로 대응됨을 나타내고 사선타원은 객체를 의미한다. FMS의 제어는 분산된 셀과 메시지를 주고 받는 과정을 반복하면서 이루어지는데, 통합제어 모델에서는 이 과정이 Cell 객체화의 메시지 교환을 통한 상호작용으로 이해된다. 또한, 셀과 관련된 대부분의 정보가 객체 내에 캡슐화되므로 FMS에 실재하는 셀의 변화를 통합제어 소프트웨어에

반영하고자 할 때 Cell 객체만을 수정함으로써 대응할 수 있다. 관계형 데이타베이스를 이용하는 경우에는 셀에 대한 데이타와 연산을 모델할 때 데이타의 모델링과 데이타를 해석, 이용하기 위한 연산에 대한 모델링이 분리, 진행된다. 그 결과 데이타베이스에는 데이타만이 저장되고, 연산의 대부분이 제어프로그램에 분산, 중복되게 되어 FMS의 변화를 반영하고자 할 때 유연하게 대처하기 어렵다.

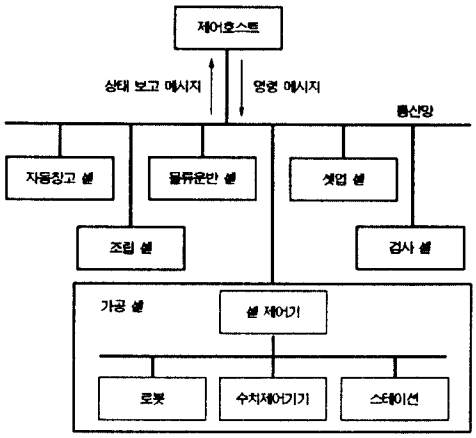

그림 1. 분산 FMS 예
Fig. 1. An example of distributed FMS

FMS 구성요소들은 서로간에 복잡한 관계를 가진다. 예를 들어, 셀은 셀을 제어하기 위한 제어컴퓨터, 셀을 구성하는 생산설비 하드웨어, 입출력 버퍼 역할을 하는 스테이션, 그리고 그 셀에서 수행되는 공정들과 관계를 가진다. 관계형 데이타베이스를 이용한 경우 관계형 데이타 모델이 테이블 형태의 간단한 자료구조만을 지원하기 때문에 이와 같이 복잡한 FMS의 데이타와 이들 간의 관계를 표현하기가 어렵고 정규화된 테이블 형태로 표현된 FMS 데이타에 대한 이해가 어렵게 된다. 반면, 객체지향 데이타베이스의 경우 이는 객체와 그들 사이의 관계성(relationship)으로 자연스럽게 표현되어 이해가 쉽다. 관계성은 두 개 이하의 클래스에 속한 객체 사이의 관계를 표현하는 이진(binary) 관계성과 서로 다른 세 개 이상의 클래스에 속한 객체들 사이의 관계를 표현하는 비이진(non_binary) 관계성으로 나눌 수 있다[9]. 그런데 대부분의 상용 객체지향 데이타베이스관리시스템에서는 객체간의 이진 관계성만을 지원하고 있으며, FMS를 구성하는 요소들간의 관계는 이것만으로 대부분 표현할 수 있으므로, 본 연구에서는 이진 관계성만을 이용하여 객체간의 관계성을 모델한다.

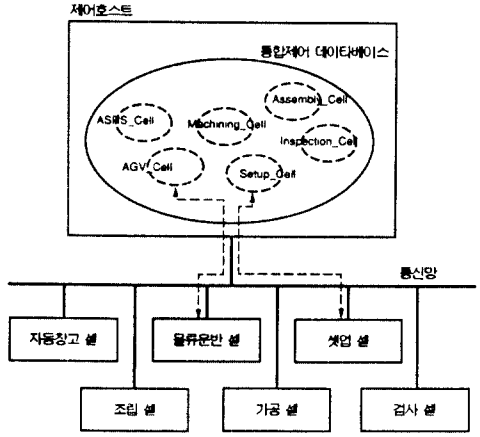

그림 2. 셀 객체
Fig. 2. Cell objects

통합제어 모델은 클래스 정의시 상속(inheritance)을 이용하여 재사용을 용이하게 한다. 예를 들어, FMS에 실재하는 셀을 클래스로 정의할 때 각 셀이 가지는 공통된 데이타와 메소드는 수퍼클래스인 Cell 클래스에 정의하고, 보다 구체적인 데이타와 메소드는 Cell 클래스의 서브클래스에 정의한다. 구체적인 정의 내용은 다음 절에 기술한다. 반면, 관계형 데이타베이스를 이용하는 경우에는 상속을 지원하지 않으므로 새로운 셀이 추가되는 경우에는 이에 대한 데이타 구조와 연산을 모두 새로 정의해야 한다.

2.1 유연성 지원

통합제어 모델이 통합제어 데이타베이스의 유연성을 어떻게 지원할 수 있는지 FMS에서 발생할 수 있는 구체적인 변화 가능성을 예로 기술한다.

- FMS 구성요소의 추가

그림 1에서 용접 셀이 추가된 경우를 가정해 보자. 용접 셀은 셀제어기, 용접을 위한 로봇, 스테이션 등으로 구성될 수 있다. 제어호스트는 FMS 상태를 추적하고 있어야 하므로 용접셀의 추가에 따른 정보를 통합제어 소프트웨어에 반영하여야 한다. 통합제어 모델은 FMS 구성요소의 추가를 이미 정의된 클래스의 객체를 생성하거나, 또는 새로운 클래스를 추가하고 추가된 클래스의 객체를 생성하는 것으로 수용한다. 전자의 경우에는 주어진 변화를 데이타 레벨에서 수용한 것이므로

제어프로그램의 수정은 불필요하다. 후자의 경우에도 기존의 클래스로부터 상속을 통하여 서브클래스로 정의하면, 이미 정의된 데이타와 메소드를 기반으로 쉽게 변화를 수용할 수 있다.

- 생산공정의 변화

FMS는 제품의 설계 변경, 품종 변경으로 인한 생산공정의 변화가 빈번한 시스템이다. 기존 제품의 설계 변경은 새로운 단위공정의 추가나 기존 단위공정의 삭제, 또는 공정순서의 변화를 요구하며, 새로운 제품의 생산을 위해서는 새로운 생산공정이 필요하다. 통합제어 모델은 제품의 생산공정을 나타내기 위하여 생산공정을 Op_Seq 클래스로, 단위공정을 Operation 클래스로 정의하여 생산공정의 변화를 이들 클래스의 객체를 생성, 삭제 또는 수정함으로써 수용한다.

- 통신상의 변경

FMS 통합제어는 제어호스트와 셀제어기 사이에 메시지 교환을 통해 이루어지므로 이들간에는 주고 받을 메시지에 대한 포맷이 미리 정해져 있어야 한다. 그런데 메시지의 포맷은 필요에 따라 변경되거나 추가될 수 있다. 또는, 셀과의 통신방식이 달라질 수도 있다. TCP/IP를 기반으로 한 통신에서 MAP[10]으로의 전환은 한 예이다. 통합제어 모델에서는 Cell 클래스에 메시지 처리에 관련된 메소드가 캡슐화되어 있으므로, 셀과 주고 받는 메시지의 포맷이 변경되거나 추가된 경우 해당되는 Cell 클래스의 메시지 관련 메소드만을 수정함으로써 대응할 수 있다. 셀과의 통신방식이 변경된 경우에는 통신을 담당하도록 정의한 Comm_Server 클래스의 해당 메소드만을 변경함으로써 변화를 수용할 수 있다.

- 운용전략의 변경

FMS 통합제어를 위한 스케쥴링 방식은 다양한데, 상황에 따라 다른 스케쥴링 방식을 채택하는 것이 바람직하다. 이를 위해서는 FMS 운용중 스케쥴링 방식을 쉽게 바꿀 수 있도록 하고, 새로운 방식의 추가 및 기존 방식의 수정이 용이해야 한다. 통합제어 모델은 스케쥴링 방식을 객체로 정의한다. 따라서 여러 다른 스케쥴링 방식을 가진 Scheduler 객체가 동시에 존재하도록 함으로써 FMS의 운용중 스케쥴링 방식의 변경이 용이하다. 새로운 스케쥴링 방식을 수용하는 경우에도, 기존의 클래스로부터 상속을 통해 정의함으로써 쉬게 반영할 수 있다.

2.2 이상상태로부터의 회복 지원

제어호스트는 하드웨어 및 소프트웨어의 이상, 또는 정전 등으로 인한 이상상태 발생에도 불구하고 분산된 각 셀의 일관된 상태를 추적하고 있어야 한다. 이를 위해서는 제어호스트와 셀 사이에 교환되는 메시지의 유실이 방지되어야 하고, 메시지가 유실된 경우 재전송할 수 있는 구조를 가져야 한다. 다음으로 메시지의 처리, 스케쥴링 등의 수행중 이상상태가 발생한 경우, 수행 이전의 일관된 상태로 돌아가 재수행할 수 있는 구조가 요구된다.

통합제어 모델은 제어호스트와 셀 간에 주고 받는 메시지의 유실 방지와 재전송을 위하여 Message 클래스를 정의하여 교환되는 메시지를 보강한다. 각 셀 제어기에서도 메시지를 보강하는 것으로 가정한다. 메시지의 보강시 메시지마다 부여된 순차번호로부터 메시지의 유실을 판단하고, 유실된 경우 재전송할 수 있도록 한다.

통합제어 모델은 제어호스트가 셀로부터 메시지를 받은 후 이를 처리하는 전과정을 트랜잭션[30]으로 처리한다. 따라서 메시지 처리 과정 중 발생하는 이상상태의 경우에는 트랜잭션의 원자성으로 인하여 모두 실행되든지, 아니면 아무것도 실행되지 않은 상태 둘 중 하나를 유지하게 되어 통합제어 데이타베이스는 항상 일관된 상태를 유지할 수 있다.

III. FMS 객체 스키마

통합제어 데이타베이스는 논리적으로 각 셀과 생산설비 등의 상태정보를 추적, 관리하기 위한 FMS 상태 데이타베이스, 각 셀의 작업을 계획하기 위하여 필요한 정보를 저장, 관리하는 작업일정 데이타베이스, 그리고 셀과 제어호스트간에 주고 받는 메시지 로깅을 위한 메시지 로그 데이타베이스로 구성된다. 이 절에서는 통합제어 데이타베이스의 객체 스키마에 대하여 기술한다.

3.1 FMS 상태 데이타베이스

3.1.1 Cell

Cell 클래스에는 통합제어 소프트웨어가 필요로 하는 상태 정보, 그리고 실제 셀로 메시지를 보내고, 실

제 셀로부터 들어온 메시지를 처리하기 위한 메소드 등이 정의된다. Cell 클래스는 그림 3에서 보는 바와 같이 네 개의 클래스와 관계성을 가진다. 즉, 셀을 제어하는 Computer, 입출력 버퍼의 역할을 하는 Station, 셀을 구성하는 생산 설비 하드웨어를 총칭하는 Equipment, 그리고 그 셀에서 수행되는 Operation과 1:1 혹은 1:m의 관계성을 가진다. 예를 들어, 하나의 셀은 복수의 생산설비로 구성되므로 Cell 클래스와 Equipment 클래스 사이에는 1:m의 관계성이 있다. 마찬가지로 그림은 셀이 오직 하나의 컴퓨터에 의하여 제어됨을 나타내고 있다.

FMS에는 서로 다른 공정을 수행하는 복수의 셀이 존재하는데, 이를 모델하기 위하여 각 셀이 갖는 공통된 데이타와 메소드는 수퍼클래스에서 정의하고, 여러 다른 종류의 셀이 갖는 보다 구체적인 데이타와 메소드는 서브클래스에서 정의하도록 한다(그림 4). 그림에서 직사각형은 클래스를 나타내고, 위로부터 클래스 이름, 데이타, 메소드의 순서로 나타냈다. 그림에서 보는 바와 같이 각각의 셀들이 가지고 있는 공통된 데이타와 메소드는 수퍼클래스인 Cell 클래스에 정의되어 있다. 이로부터 상속된 서브클래스에서는 수퍼클래스에서 정의된 데이타와 메소드를 다시 정의할 필요가 없으므로 소프트웨어의 구성이 용이하게 된다. 새로운 셀이 추가된 경우에도 마찬가지이다. 예를 들어 용접셀이 생산시스템에 추가된 경우 Cell 클래스로부터 상속을 통해 용접셀 클래스를 정의하면 된다.

객체지향 개념에 의하면, 메소드는 경우에 따라 메소드 이름만 상속받아 같은 이름을 가진 메소드들이 서로 다른 계산을 수행하도록 정의할 수 있는데, 이런 메소드는 그 대상 객체의 클래스에 따라 적절한 메소드 정의가 수행된다(run_time polymorphism). 그림 4에서 Cell 클래스의 메소드중 * 표시가 된 것들은 서브클래스에서 재정의 할것을 의미한다. 이들 메소드는 제어호스트에서 각 셀로 메시지를 만들어 보내거나, 혹은 각 셀로부터 전달되어 온 메시지를 처리하는 메소드들이다. 이와 같이 하는 이유는 각 셀과의 통신방식이나 주고받는 메시지의 포맷이 다를 수 있기 때문에 이에 맞도록 재정의할 수 있게 하기 위해서이다.

Make_Msg 메소드는 셀로 보낼 메시지를 만들기 위한 메소드이다. 각 셀과 제어호스트 사이에 미리 약속된 포맷에 따라 *Make_Msg* 메소드는 셀로 보낼 메시지를 만든다. 셀과 주고 받을 메시지의 포맷이 변경된 경우에는 해당 셀의 *Make_Msg* 메소드만을 수정함으로써 변화를 반영할 수 있다. *Send_Msg* 메소드는 *Make_Msg* 메소드에 의해 만들어진 메시지를 셀로 보내는 메소드이다. 이를 위하여 Send_Msg 메소드는 만들어진 메시지를 데이타베이스에 저장한 후, 이 사실을 Comm_Server 객체에 알린다. 실제 셀에 메시지를 전송하는 것은 Comm_Server 객체가 담당한다. *Send_Msg* 메소드를 이와 같이 정의함으로써 통합제어 소프트웨어는 셀에 메시지가 전달될 때까지 기다리지 않고 다음 과정을 계속해 나갈 수 있다. *Receive_Msg* 메소드는 셀로부터 제어호스트로 전달되어, Comm_Server 객체에 의해 데이타베이스에 저장되어 있는 메시지를 읽어 내는 메소드이다. *Process_Msg* 메소드는 *Receive_Msg* 메소드를 통해 읽어 낸 메시지의 내용을 해석하여 데이타베이스 갱신과 같은 적절한 조치를 취하게 된다.

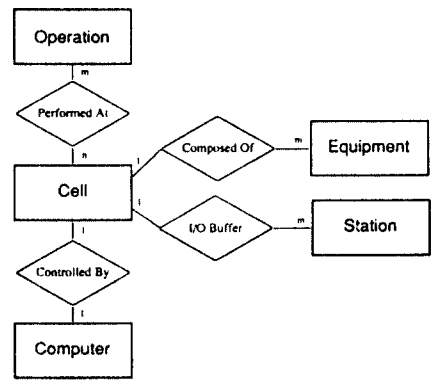

그림 3. Cell 클래스와 다른 클래스들과의 관계성
Fig. 3. Relationship among Cell class and other classes

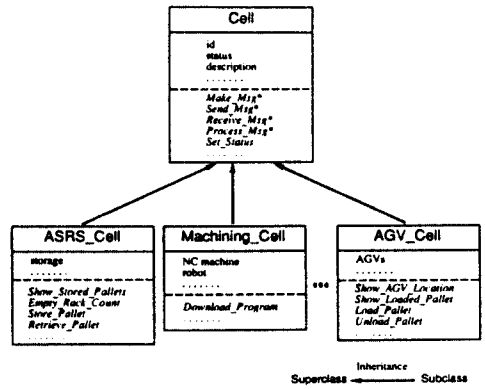

그림 4. Cell 클래스 상속구조
Fig. 4. Cell class inheritance hierarchy

3.1.2 Plant_Controller

Plant_Controller 클래스는 제어호스트 자신을 표현한다. 각 셀과 메시지 교환을 통하여 전체시스템을 제어하는 제어호스트를 Plant_Controller 객체로 정의하여 FMS 통합제어환경을 Cell 객체들과의 상호작용으로 모델한다. Plant_Controller 객체는 현재 채용된 스케쥴링 방식을 구현한 Scheduler 객체와 관계성을 가지는데, 이를 통해 스케쥴링이 필요한 경우, 채택된 Scheduler 객체에 이를 요구할 수 있다.

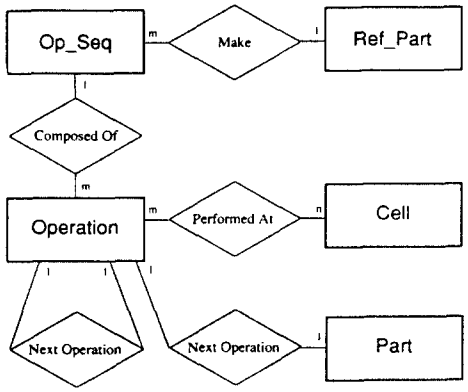

그림 5. Operation과 Op_Seq 클래스
Fig. 5. Operation class and Op_Seq class

3.2 작업일정 데이타베이스

3.2.1 Operation과 Op_Seq

하나의 제품을 만들 때에도 서로 다른 공정 순서에 따라 작업이 진행될 수 있다. 즉, 순서가 바뀌어도 상관이 없는 공정의 경우에는 두 공정의 순서를 바꿀 수 있다. 따라서 하나의 제품을 생산하는 데도 여러 개의 공정 순서가 있을 수 있다. 이를 표현하기 위하여 생산 제품의 품종을 나타내는 Ref_Part 객체는 여러 다른 Op_Seq 객체와 관계성을 가질 수 있다. 또한, Op_Seq 객체는 이를 구성하는 Operation들과 1:m의 관계성을 갖는다. 그림 5는 이를 보여준다.

3.2.2 Scheduler와 Task

FMS에서 상황에 따라 적절한 스케쥴링 방식을 이용할 수 있도록 하기 위하여 이를 클래스로 정의하고, 원하는 객체를 선택함으로써 변경이 가능하도록 한다. 그림 6은 Scheduler 클래스의 상속 구조를 보여준다. MWR_Sched(Most Work Remaining) 클래스는 가장 많은 작업량이 남은 가공품부터 작업하는 스케쥴링 방식을, SPT_Sched(Shortest Processing Time) 클래스는 전체 작업시간이 가장 적게 소요되는 가공품부터 작업하는 스케쥴링 방식을 정의하고 있다. 새로운 스케쥴링 방식을 추가할 경우에는 기존의 클래스로부터 상속을 통해 서브클래스로 정의할 수 있다.

Task 객체는 Scheduler에 의해서 생성되는데, 다음 작업에 대한 지시사항을 가지고 있다. 즉, 어느 Part를 어느 셀에서 가공할 것인지에 대한 정보를 가진다. Part에 가해질 공정은 Part 객체와 "Next Operation" 관계성을 갖는 Operation 객체를 통해 알 수 있다. 생성된 Task 객체는 Cell 객체에 전달되어 셀에 보낼 메시지를 생성하는데 이용된다.

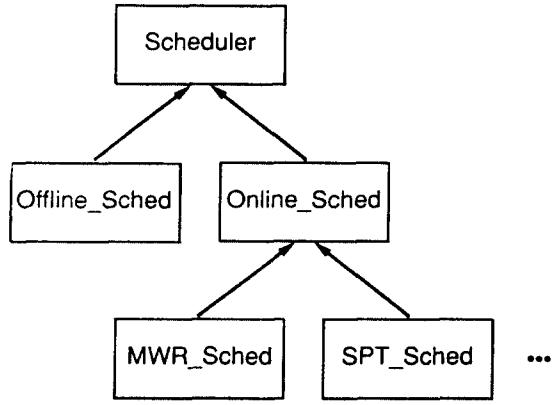

그림 6. Scheduler 클래스
Fig. 6. Scheduler class

3.3 메시지 로그 데이타베이스

3.3.1 Message와 Msg_Handler

Message 클래스는 제어호스트와 FMS 내에 실재하는 각 셀 간의 통신시 주고 받는 메시지를 정의하는 클래스이다. 그림 7은 Message 클래스의 정의와 이로부터 상속된 두 클래스의 정의를 보여 준다. Outgoing_Msg 클래스는 제어호스트로부터 각 셀로 보내는 메시지를, Incoming_Msg 클래스는 각 셀로부터 제어호스트로 들어오는 메시지를 정의한다. Message 클래스는 메시지 유실 방지와 유실시 재전송을 위하여 메시지마다 유일한 순차번호를 부여한다. 제어호스트나

각 셀의 셀제어기에서는 이를 이용하여 메시지의 유실 여부를 판단하고 유실된 경우 재전송을 요구한다.

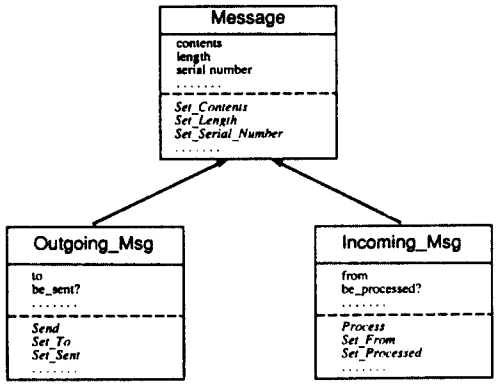

그림 7. Message 클래스
Fig. 7. Message class

Outgoing_Msg 클래스는 be_sent?라는 데이타를 가지고 있는데 이는 메시지가 실제 셀로 보내졌는지 여부를 나타낸다. 3.1.3 절에서 기술한 바와 같이 Cell 클래스의 Send_Msg 메소드는 셀로 보낼 메시지를 데이타베이스에 저장하고 Comm_Server 객체에 이 사실을 알리는 것으로 메시지를 보내는 일을 끝낸다. 따라서 Send_Msg 메소드가 성공적으로 수행되었더라고 메시지가 실제 셀에 전달된 것은 아니므로 be_sent? 데이타는 "NO" 값을 가지고 있다. 이 값은 Comm_Server 객체에 의하여 실제 셀로 메시지가 보내진 후"YES"로 갱신된다. Incoming_Msg 클래스의 경우에는 be_processed? 데이타를 가지고 있다. 이는 셀로부터 전달되어 온 메시지가 처리되었는지 여부를 나타낸다. Cell 클래스의 *Receive_Msg* 메소드는 be_processed? 데이타를 검사하여 "NO"인 메시지를 읽어 *Process_Msg* 메소드가 처리한 후 이를 "YES"로 갱신한다.

셀로부터 전달되어 온 메시지를 처리하기 위해 Cell 객체에 전달하는 것이 Msg_Handler 객체이다. Msg_Handler 객체는 Comm_Server 객체로부터 셀에서 메시지가 전달되어 데이타베이스에 저장되었음을 통보받으면 데이타베이스로부터 방금 도착한 메시지를 읽어 이로부터 메시지를 처리할 Cell 객체를 찾아 처리를 요구한다.

3.3.2 Comm_Server

Comm_Server 클래스는 셀과의 통신을 담당하기 위하여 정의한다. 셀로부터 메시지를 받으면, 이를 Incoming_Msg 클래스의 객체로 만들어 데이타베이스에 저장한다. 이때 메시지 유실 여부를 판단하고, 유실이 확인되면 셀에 메시지의 재전송을 요구한다. 문제가 없는 경우에는 Msg_Handler 객체에 도착한 메시지의 처리를 요구한다. 또한 Outgoing_Msg 객체로부터 실제 셀로 보내기 위한 데이타를 얻어 셀에 전달하는 역할도 담당한다.

IV. FREE

FREE는 객체지향 FMS 통합제어 모델을 기반으로 구현된 통합제어 소프트웨어시스템이다. 이는 실제로 서울대학교 자동화시스템공동연구소(ASRI:Automation and Systems Research Institute)에 구축되어 있는 FMS 모델플랜트를 대상으로 하였다[21]. 통합제어 데이타베이스로는 상용 객체지향 데이타베이스관리시스템인 Objectivity/DB[22]를 이용하였고, 제어프로그램 작성은 C++언어를 이용하였다. 셀과의 통신은 자동화시스템공동연구소에서 TCP/IP를 기반으로 하여 개발한 API(Application Program Interface)를 이용하였는데, 현재 C언어로 구현되어 있다.

4.1 구조와 제어흐름

그림 8은 FREE의 구조를 보여 주는데 통합제어 데이타베이스를 중심으로 각 셀로부터 전달되어오는 메시지를 받아 처리하고 스케쥴링을 수행하는 주제어 프로세스, 실제 셀과의 통신을 담당하는 통신 프로세스, 그리고 FMS 상태감시를 위한 모니터링 인터페이스로 구성된다. 이중에서 모니터링 인터페이스는 객체지향 개념을 적용하여 모니터링 인터페이스의 생성 및 유지, 보수가 용이하도록 하였는데[23] 본 논문에서는 기술하지 않았다.

그림 9는 FREE의 제어흐름을 보여 주는데, 이는 객체들간의 상호작용으로 이해된다. Comm_Server 객체가 셀로부터 명령의 수행결과를 보고하는 메시지를 전달받으면, 메시지의 유실 여부를 판단하여 유실된 경우 셀에 재전송을 요구한다. 문제가 없는 경우에는 도착한

메시지를 Incoming_Msg 객체로 만들어 로깅한 후

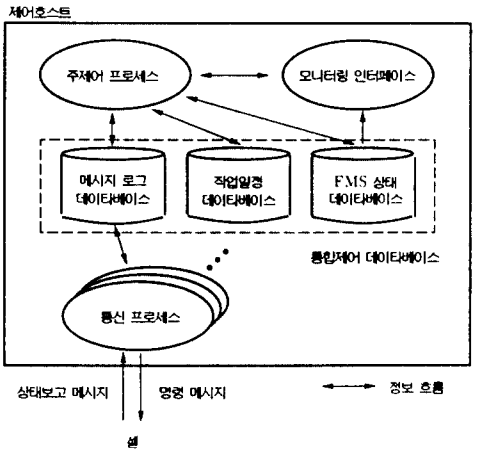

그림 8. FREE 구조
Fig. 8. FREE architecture

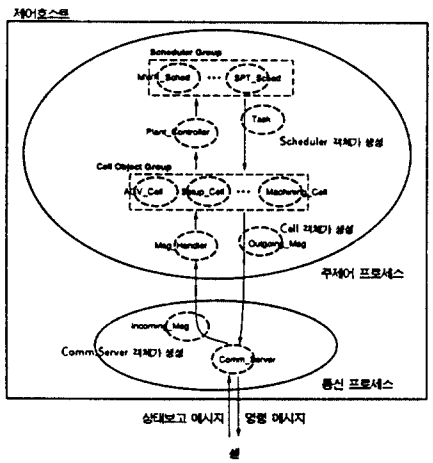

그림 9. FREE 제어흐름
Fig. 9. FREE control flow

Msg_Handler 객체에 새로운 메시지가 도착했음을 알려 이를 처리하도록 요구한다. Msg_Handler 객체는 Incoming_Msg 객체들을 검색하여 아직 처리되지 않은 객체를 찾아 처리해야 할 셀을 알아내고, 이를 Cell 객체에 알린다. Cell 객체에 의한 메시지 처리내용은 크게 세가지로 나누어 볼 수 있다. 셀 또는 셀 구성요소들의 상태 변화에 대한 정보 등을 담고 있는 경우 그 내용을 데이타베이스에 반영한다. 메시지의 처리 결과 스케쥴링이 필요한 경우 Plant_Controller 객체에 이를 알린다. 메시지의 내용이 모니터링 인터페이스에 의해 현재 모니터되고 있는 데이타와 관련된 경우 이를 모니터링 인터페이스에 전달한다. Plant_Controller 객체는 스케쥴링이 필요한 경우 현재 선택된 스케쥴링 방식을 구현한 Scheduler 객체에 스케쥴링을 요구한다. Scheduler 객체는 구현된 방식에 따라 다음 작업을 계획하여 Task 객체를 생성하여 Cell 객체에 전달한다. Cell 객체는 Task 객체의 내용으로부터 Outgoing_Msg 객체를 생성하여 로깅한 후 이를 실제 셀에 전송하도록 Comm_Server 객체에 전달한다. Comm_Server 객체는 전달받은 Outgoing_Msg 객체로부터 실제 셀에 전송할 데이타를 얻어 이를 전송한다.

4.2 통합제어 데이타베이스의 구현

이 절에서는 구현에 이용한 Objectivity/DB에 대하여 간략하게 기술하고, 설계된 데이타베이스가 어떻게 구현되는지 예를 통해 기술한다. Objectivity/DB는 C++ 객체의 지속성(persistence)을 지원하기 위하여 클래스 선언시 시스템에서 정의한 지속 클래스의 서브클래스로 정의하도록 한다[21]. 객체간의 이진 관계성은 사용자가 정의하는 연관(association)을 통해 표현된다. 연관은 탐색(navigation)이 가능한 방향에 따라, 양방향 연관과 단방향 연관이 있고, 관계된 객체의 수에 따라 1:1, 1:m, m:n의 연관으로 구분할 수도 있다. Objectivity/DB의 논리적 저장구조는 기본 객체, 컨테이너, DB(database), FDB(federated database)의 네 단계로 나누어진다. 기본 객체의 모임인 컨테이너는 메모리나 디스크에서 물리적인 집중화(clustering)가 가능하며, 잠금(locking)의 최소단위이다. DB는 컨테이너의 모임으로 각각의 DB는 하나의 파일에 대응된다. FDB는 논리적 저장구조의 최상위 단계로 다수의 DB로 구성된다.

모델플랜트를 대상으로 한 통합제어 데이타베이스는 하나의 FDB로, FMS 상태 데이타베이스, 작업일정 데이타베이스, 메시지 로그 데이타베이스는 각각 하나의 DB로 구현하였다. 메시지 로그 데이타베이스는 각 셀별로 컨테이너를 갖도록 하여, 셀과 주고 받는 메시지는 컨테이너 단위로 저장하도록 하였다. 이는 Objectivity/DB의 잠금의 최소단위가 컨테이너이기 때문에 여러 셀과의 통신이 동시에 이루어질 수 있도록 하기 위함이다.

그림 10은 구현된 통합제어 데이타베이스 중 Cell 클래스와 이로부터 상속된 ASRS_Cell 클래스 정의의

일부를 보여준다. 그림에는 그림 3과 4에서 보인 데이타와 메소드, 그리고 다른 객체와의 관계성이 정의되어 있다. Cell 클래스는 객체의 지속성을 지원하기 위하여 ooObj라고 하는 시스템 정의 지속 클래스의 서브클래스로 정의되었다. 그림에서 ooHandle은 괄호안 클래스의 객체에 접근하기 위한 포인터 역할을 하는데, 예를 들어 Cell 객체는 Equipment 객체와 Composed_Of라는 이름의 연관을 가짐을 나타내고 있다. []의 존재는 관계된 객체가 여러 개임을 나타낸다. 그림에서 일부 메소드들은 가상함수(virtual function)로 선언되어 있는데, 이는 Cell 클래스의 서브클래스에서 같은 이름의 메소드를 재정의하여 사용할 것을 나타낸다.

V. 결 론

본 논문에서는 성공적인 FMS 통합제어 소프트웨어의 구축을 위하여 필수적인 통합제어 데이타베이스의 유연성을 지원하기 위하여 먼저 객체지향 FMS 통합제어 모델을 제안하였다. 제안된 모델은 FMS에 실재하는 셀을 가상 셀 객체로 모델하여 제어과정을 이들 객체간의 상호작용으로 이해함으로써 개념적으로 쉽다.

```
class Cell: public ooObj{ // Inherit persistence from ooObj
protected:
    char          id[IDSIZE];
    int           status;
    ooVArray(char)   description; // ooVArray: Variable-size array

    // User-defined associations
    ooHandle(Computer) Controlled_By <-> CellA;
    ooHandle(Station) IOBuffer[] <-> CellA;
    ooHandle(Equipment) Composed_Of[] <-> CellA;
    ooHandle(Operation) Performed_At[] <-> CellA;

Public:
    // Public methods
    int Status(void);
    ooStatus Set_Satus(int stat); // ooStatus: Indicates whether an error occurred or not
                                  //          Return values: oocSuccess or oocError
    ooStatus Show_Description(void);

    virtual ooStatus Make_Message(ooHandle(Outgoing_Msg) &messageH);
    virtual ooStatus Send_Message(ooHandle(Outgoing_Msg) &messageH);
    virtual ooStatus Receive_Message(ooHandle(Incoming_Msg) &messageH);
    virtual ooStatus Process_Message(ooHandle(Incoming_Msg) &messageH);
    .......
};

class ASRS_Cell: public Cell{
protected:
    ooHandle(Storage) StorageA <-> ASRS_CellA;

Public:
    void Show_Stored_Pallets(void);
    int Empty_Rack_Count(void);
    ooStatus Store_Pallet(ooHandle(Pallet) &palletH);
    ooStatus Retrieve_Pallet(ooHandle(Pallet) &palletH, char* name);

    ooStatus Make_Message(ooHandle(Outgoing_Msg) &messageH);
    ooStatus Send_Message(ooHandle(Outgoing_Msg) &messageH);
    ooStatus Receive_Message(ooHandle(Incoming_Msg) &messageH);
    ooStatus Process_Message(ooHandle(Incoming_Msg) &messageH);
    .......
};
```

그림 10. 통합제어데이타베이스 스키마 예

Fig. 10. An example of integrated FMS control database schema

셀 객체에는 통합제어에 필요한 데이타와 연산들이 대부분 캡슐화되어 있고, 구성요소간의 복잡한 관계도 자연스럽게 표현되어 FMS상의 변화를 쉽게 반영할 수 있다. 다음으로 통합제어 모델을 기반으로 설계한 통합제어 데이타베이스는 생산시스템을 구성하는 셀들과 각 생산설비의 상태 정보, 생산계획 정보, 제어호스트와 셀 간에 주고 받는 메시지 등과 이에 대한 핵심연산을 저장, 관리함으로써 FMS의 변화를 대부분 수용할 수 있고, 제어프로그램의 변경이 필요한 경우에도 그 범위를 최소화할 수 있다.

본 연구에서는 FMS 운용을 위해 필수적인 데이타와 연산을 대상으로 통합제어 데이타베이스의 설계를 진행하였으나, FMS의 효용성을 극대화시키기 위해서는 실제 생산부분 뿐 아니라, 제품의 기획에서 판매에 이르는 생산의 전과정에 걸쳐 유관 단위활동들을 유기적으로 연결하여 효율적으로 수행할 수 있도록 함이 바람직하다. 컴퓨터통합생산시스템은 이를 염두에 둔 개념이다[25]. 따라서 본 연구의 연장선상에서 객체지향 개념을 컴퓨터통합생산시스템을 구성하는 CAD/CAM, MRP(Material Requirement Planning) 등의 소프트웨어 모듈의 통합에 적용하기 위한 연구를 진행 중이다.

참 고 문 헌

[1] W. W. Luggen, *Flexible Manufacturing Cells and Systems*. Prentice_Hall. 1991.

[2] F. Bancilhon, "Object-oriented database system," in *Proc. ACM SIGACT SIGMOD Symposium on Principles of Database Systems*, Mar. 1988.

[3] E. Bertino and L. Martino, *Object-Oriented Database Systems : Concepts and Architectures*. Addison-Wesley, 1993.

[4] S. L. Osborn, "The role of polymorphism in schema evolution in an object_oriented database," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, pp.310-317, Sep. 1989.

[5] E. F. Codd, "A relational model for large shared data banks," *Communications of the ACM*, vol. 13, pp.377-387, Jun. 1970.

[6] L. Lin and Y.-J. Fang, "Database model for hierachical control of manufacturing cell systems in a manufacturing shop," *Computer Integrated Manufacturing Systems*, vol.6, pp.185-194, Aug. 1993.

[7] J. R. Moyne, L. C. McAfee, Jr., and T. J. Teorey, "An application of entity_relationship data modeling techniques to the automated manufacturing proce-

sses," in *Proc. of 2nd Int'l Conf. on Data and Knowledge Systems for Manufacturing and Engineering*, (Gaithersberg, MD), pp.206-215, IEEE Press, Oct. 1989.

[8] H. Wedekind and G. Zoentlein, "Conceptual basis for database applications in flexible manufacturing systems," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, (Raleigh, NC), pp.551-557, 1987.

[9] R. G. G. Cattell, *Object Data Management: Object-Oriented and Extended Relational Database Systems*. Addison-Wesley, 1991.

[10] D. Breugnot, M. Gourgand, D. Hill, and P. Kellert, "GAME: An object-oriented approach to computer animation in flexible manufacturing systems modelling," in *Proc. of the 24th Annual Simulation Symposium*(A. H. Ruta, ed.), (New Orleans, Louisiana), pp.217-227, Apr. 1991.

[11] A. Levas and R. Jayaraman, "WADE : An object-oriented environment for modeling and simulation of workcell applications," tech. rep., Manufacturing Research Department, IBM Research Division, T.J. Watson Research Center, Nov. 1987.

[12] D. M. Dilts, N. P. Boyd, and H. H. Whorms, "The evolution of control architectures for automated manufacturing systems," *Journal of Manufacturing Systems*, vol. 10, no. 1, pp.79-93, 1991.

[13] A. T. Jones and C. R. McLean, "A proposed hierarchical control model for automated manufacturing systems," *Journal of Manufacturing Systems*, vol.5, no. 1, pp.15-25, 1986.

[14] S. P. Rana and S. K. Taneja, "A distributed architecture for automated manufacturing systems," *Int'l Journal of Advanced manufacturing Technology*, vol. 3, no. 5, pp.81-98, 1988.

[15] P. R. Chintamaneni, P. Jalote, Y. B. Shieh, and S. K. Tripathi, "On fault tolerance in manufacturing systems," *IEEE Network*, vol. 2, pp. 32-39, May 1988.

[16] A. Adlemo, S. A. Andreasson, T. Andreasson, and C. Carlsson, "Achieving fault tolerance in factory automation systems by dynamic configuration," in *Proc. of The First Int'l Conf. on Systems Integration*, (Morristown, New Jersey), pp.396-402, Apr. 1990.

[17] J. R. Perkins and P. R. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Transactions on Automatic Control*, vol. 34, no. 2, pp. 139-148, 1989.

[18] K. G. Shin and Q. Zheng, "Scheduling job operations in an automatic assembly line," in *Proc. of IEEE Int'l Conf. on Robotics & Automation*, (Cincinnati, Ohio), pp.176-181, May 1990.

[19] A Valenzano, C. Demartini, and L. Ciminiera, *MAP and TOP Communications*. Addison-Wesley, 1992.

[20] J. Gray and A. Reuter, *Transaction Process-ing : Concepts and Techniques*. Morgan Kaufmann, 1993.

[21] 박장호, "객체지향 FMS 통합제어 데이타베이스 설계와 구현," 석사학위논문, 서울대학교, 1994년 2월.

[22] Objectivity, Inc., *Objectivity/DB System Overview, ver. 2.1*, 1993.

[23] 권학일, "FMS 모니터링을 위한 객체지향 그래픽 UIMS의 구현," 석사학위논문, 서울대학교, 1994년 2월.

[24] R. G. G. Cattell, ed., *The Object Database Standard : ODMG-93*. Morgan Kaufmann, 1994.

[25] D. D. Bedworth, M. R. Henderson, and P. M. Wolfe, *Computer-Integrated Design and Manufacturing*. McGraw-Hill, 1991.

차상균(Sang Kyun Cha) 정회원

1958년 2월 19일생

1980년 : 서울대학교 전기공학과 졸업

1982년 : 서울대학교 대학원 계측제어공학과 석사

1991년 : 미국 Stanford대학교 전기공학과(컴퓨터 시스템) 박사

데이콤, Texas Instruments, IBM Palo Alto, Hewlett-Packard Laboratories 근무

1992년 2월 ~ 현재 : 서울대학교 제어계측공학과 조교수

박 장 호 (Jang Ho Park) 준회원

1969년 7월, 16일생

1992년 : 서울대학교 제어계측공학과 졸업

1994년 : 서울대학교 대학원 제어계측공학과 석사

1994년 3월 ~ 현재 : 동대학원 제어계측공학과 박사과정