Journal of the Korean Institute
of Industrial Engineers
Vol. 18, No.2, Dec., 1992

분산시뮬레이션을 통한 FMS 구축+

김성식*, 배경한*

Realizing FMS Through Distributed Simulation+

Sung-Shick Kim*, Kyoung-Han Bae*

Abstract

This paper introduces a distuibuted simulation scheme that is useful in the top-down FMS building approach. In the scheme, we first introduce "FMS Shell" that contains basic functions and structure of FMS's. To test a proposed FMS, appropriate features of the proposed FMS are added to the shell, then distributed simulation is performed with the resulting software. This runs like a real system only without hardware devices. An real application case is stated at later part of this paper.

1. 서 론

FMS 구축과정은 당면한 회사의 환경에 따라서 달라진다. 공장에 자동화 장비와 FMC(Flexible Manufacturing Cell)들이 이미 존재하고 있다면 이들을 연결하여 FMS로 구성할 수 있다. 공장을 신설하는 경우에는 충분한 예산이 있다면 처음부터 완전한 FMS를 설치할 수 있으며, 이와는 반대로 전체적인 계획을 미리 세우고 점진적으로 기계를 도입하여 구축할 수도 있다. Greenwood[7]는 이러한 과정들을 각각 자동화섬 연결('Linked islands' of automation), 신공장 건설('Green-field' site), 진화적 접근(Evolutionary Approach)이라고 부르고 이들의 장단점을 논하였다. 이들 중 어떠한 방법을 사용하더라도 FMS 구축은 시스템의 통합 문제로 귀착된다. 기계와 통제장비의 연결, 통제장비와 컴퓨터의 연결, 컴퓨터 간의 연결, 각종 S/W 간의 대화, 이들 전체를 통제 조정하는 문제 등 H/W와 S/W들을 하나의 시스템으로 묶는 문제 자체가 FMS 구축의 핵심이라고 할 수 있기 때문이다.

통상적으로 시스템을 구축할 때에는 Top-down 접근방법과 Bottom-up 접근방법이 사용되어지고 있으며 FMS도 예외는 아니다. 자동화섬을 연결하는 경우는 분산되어진 자동화장비군을 큰 시스템으로 연결하기 때문에 자연히 H/W(Bottom)를 구성하고 이들을 조정 및 통합하는 쪽(Up)으로 진행되어진다. FMS의 역사는 그리 길지 않으므로 대

+본 연구는 학술진흥재단의 지원에 의하여 이루어졌음

*고려대학교 산업공학과

부분의 현존하는 FMS는 이의 발전과정인 수동작업, 단위기계 자동화, FMC, FMS의 구축과정을 거쳐 이루어져 왔으며 결과적으로 자동화섬을 연결하는 형태가 주종을 이루었다. Bottop-up 접근방법에서 경험하는 문제는 자동화 기계들을 Cell에게 연결할 때 기종이 서로 다르기 때문에 상호간에 연결이 어렵고 시스템의 통제와 운영이 단위기계들 또는 FMC의 특성에 따라서 종속되어지므로 전체 시스템의 효율성을 높이기가 어려운 점과 연결하는 일 자체에 막대한 노력이 수반되어지는 점 등이다.

FMS 구축기술, 능력, 적용범위에 대한 기반 정립이 잘된 현재에 와서는 신공장 건설과 진화적 구축이 가능하지만 아직도 대체로 자동화섬을 연결하는 방식을 취하는 것이 현실이다.

신공장 건설이나 진화적 구축을 택하는 경우에도 Top-down 접근방법이나 Bottom-up 접근방법 모두 사용이 가능하다. 그런데, 앞에서 언급한 것과 같은 이유 때문에 Bottom-up 접근방법은 바람직하지 않다. Top-down 접근방법이란 전체적인 운영체계를 구축한 후에 장비를 연결하고 최종적으로 시스템을 완성하는 방법이다. 이 방법은 장비를 연결하기 전에 시스템을 충분히 검토하여 최적의 시스템을 구성할 수 있으며 시행착오를 최소화할 수 있는 장점이 있는 반면에 구축대상이 존재하지 않는 상태에서 시스템설계와 운영방법을 개발해야하는 여려움이 있다. 그러나, 이러한 문제를 해결할 수 있는 FMS 구축수단이 존재한다면 Top-down 접근방법은 매우 유용한 방법이 될 수 있다.

Doumeingts[5], Doumeingts, Vallespir, Darricaul[6], Leve, Vernadat[10], Meredith[12] 등은 CIM과 NBS 등의 개념적 모형을 이용한 시스템 구축방법과 분석·설계시 사용하는 도구들을 소개하고 있다. 이들은 도구를 SADT, PETRI-net 등의 통신과 동기적 처리방법을 서려기할 때 사용하는 도구, GRAI[5]와 같이 의사결정시스템을 설계할 때 사용하는 도구, DFD, E-R Diagram, IDEFI, MERISE 등의 정보 처리 시스템을 설계할 때 사용하는 도구, GRAFCET, MFM, GEMMA 등의 물리적 시스템을 분석할 때 사용하는 도구들로 나누고 있다.

Leva와 Vernadat는 CIM용 데이타베이스의 개념적 설계방법을 제시하면서 시스템을 정확하게 표현할 때 사용하는 'AS IS'분석방법과 시스템을 평가할때 사용하는 'To Be' 분석방법을 설명하고 있다. Meredith[12]는 'AS IS'를 수행할 때 사용하는 IDEF, Node Tree, Blueprinting과 같은 도구들을 소개하고 있다.

이외에도 Huska[8]가 제안한 TESPA개념을 이용한 구축방법이 있으며 이들이 소개한 방법들은 모두 기본적으로 Top-down 접근을 하고 있다.

그러나, 이러한 방법들은 시스템 설계에 초점을 맞추고 있으므로 시스템 구축의 길잡이가 될 수는 있지만 구체적인 구축방법을 제공하지는 않는다. 그러므로, 시스템을 상세하게 설계하였더라도 실제 시스템을 구축할 때 야기되는 문제를 미리 발견하기가 어렵고 구축 중에 발견할 가능성이 높다. 이러한 문제를 해결하는 방법에는 김성식과 배경한[1, 2]이 제안한 분산시뮬레이션을 이용한 방법이 있다.

분산시뮬레이션을 이용한 방법[1, 2]은 개념설계, 상세설계, 운용방법 결정, 분산시뮬레이션을 이용한 모형, 구축, 모형에 장비를 연결하여 실제 시스템을 전환하는 과정을 거친다. 여기서 시뮬레이션을 이용한 모형이란 실제 시스템과 동일한 컴퓨터와 LAN을 이용하여 시스템을 미리 설치한 후 장비의 운영은 이산형 시뮬레이션으로 표현하고, 데이타베이스와 통신을 이용한 정보처리 시스템과 운영소프트웨어가 완성된 가동 가능한 소프트웨어를 의미한다. 이 모형은 장비와 연결이 안되었을 뿐이지 실제 시스템과 동일한 운영체계를 갖춘 것이다. 장비의 동작은 이산형 시뮬레이션으로 표현하므로 전체적인 모형은 분산시뮬레이션의 특성을 갖는다. 이 방법은 가동을 실험할 수 있어서 시행착오를 줄일 수도 있고 이로 인한 비용의 발생을

줄일 수 있다. 그리고, 모형을 통하여 완전한 시스템을 구축한 후에 장비를 도입하므로 비용에 대한 부담이 적으며, 실제 시스템으로 전환이 쉬운 장점이 있다. 특히, 모형을 이용하여 시스템의 운영방법을 다각도로 분석할 수 있고 장비와 운영체제를 충분히 검토할 수 있으므로 가능한 한 최적의 시스템을 구축할 수 있다.

이러한 방법의 또다른 장점은 일단 소프트웨어로 이루어진 시스템을 꾸며 놓으면 이들을 재사용할 수 있다는 것이다. FMS를 적용할 수 있는 제조의 형태는 다양하다. 기계부품의 가공이 대표적인 것이겠지만 기계 또는 전자부품의 조립, 섬유 제조, 또는 의복 생산 등에 이르기까지 그 적용범위는 광범위하다. 이들에 적용되는 FMS의 구조를 살펴보면 전체적인 기본 골격들은 동일하며 장비의 특성, 장비들의 구성 형태, 요구되는 응답시간과 계획 및 통제 법칙에서 차이를 보이게 된다. 특히, 동일한 종류의 제품을 생산할 경우에는 그 차이가 공장의 구성, 장비의 세부 형태와 회사 특유의 운용 정책에서 나타날 뿐 대부분의 구조에는 크게 변동이 없으며 기본적으로 요구되는 기능의 종류들은 같다. 따라서, 한 제조 부분에서 분산시뮬레이션을 통하여 FMS를 완성하였다면 그 패키지는 최소한의 변경을 통하여 그 분야의 다른 공장 건설 시에도 재사용될 수 있다.

여기서 FMS Shell이라는 개념이 자연스럽게 나오게 된다. 즉, FMS의 운용에 필요한 기능들을 추출하여 항목 별로 정리한 후 이들 간의 관계를 수립하여 FMS 운용소프트웨어의 틀을 미리 정하고 시스템의 물리적 형태가 바뀌어도 변하지 않는 부분(LAN 통신, process 통신, 데이타베이스 시뮬레이션 보조기능, 소프트웨어 MAIN의 구조)을 미리 만들어 보유하면 이것이 FMS Shell이 된다. 각 기능들은 소프트웨어 모듈로 구성되어 있으며 해당하는 곳에 있게 되며 시스템이 변하면 해당 모듈만을 변화시키면 된다.

저자들은 고려대학교에 설치된 FMS(K.U. FMS)의 설치 과정에서 Shell을 사용하였으며 그 내용은 뒤에서 설명하기로 한다.

2. 혼합 · 분산 시뮬레이션

통상 시뮬레이션에서는 주어진 모형을 수치적(Numerically)으로 평가하기 위하여 컴퓨터를 사용하며 여기에서는 수치를 표현하는 자료들을 모아 대상 모형의 특성을 추정한다. 그러나, 본 연구에서 제시하는 분산시뮬레이션의 주요 목적은 시스템 자체가 의도하는데로 동작하는가를 확인하는데 있으며 자료를 모아 평가하는 일은 그 과정에서 일어난다.

본연구에서 말하는 분산시뮬레이션의 시간의 진행을 보면, LAN으로 연결된 컴퓨터 상에서 운용소프트웨어의 수행은 물리적인 시간을 소모하면서 진행되며 장비들의 동작들은 해당장비를 담당하는 통제 컴퓨터에서 event-driven 방식의 시뮬레이션으로 처리된다. 따라서, 이는 통상의 분산시뮬레이션과는 시간진행방법이 다른 실시간 작업과 시뮬레이션이 혼합된 혼합 · 분산(Hybrid-distri-buted) 시뮬레이션이 된다. 이때 문제가 되는 것은 동기화를 위한 시계의 구성 및 전진 방법으로 본 장에서는 여기에 대하여 중점적으로 설명하기로 한다.

2.1 시계와 이벤트 리스트

FMS는 공장 통제 컴퓨터와 각 작업장(장비군)을 운용하는 컴퓨터들로 구성되어 있으며 장비들의 동작과 고장에 관한 시뮬레이션은 장비를 통제하는 컴퓨터에서 이루어진다. 따라서, 이벤트 리스트들은 네트웍크 상의 장비를 통제하는 컴퓨터들에 존재하게 되며 이벤트 리스트에는 장비들의 해당 작업 종류, 완료 시간 및 고장 종류와 발생 시간, 고장 수리 완료 시간 등이 포함된다.

각 컴퓨터들이 수행하는 기본 임무는 계획 및 통

제 임무이며 이들은 운용소프트웨어에 의하여 수행하며 물리적 시간을 소모하며 진행된다. 시뮬레이션 수행시간에 비하여 이 부분이 압도적으 많은 시간을 소모한다. 이러한 이유로 통상의 분산시뮬레이션에서 사용하는 Chandy와 Misra 등[3, 4, 13]의 timestamp와 time-warp[9] 등의 방법들은 본 연구에서 고려하지 않고 이벤트 리스트를 사용한다.

혼합 · 분산 시뮬레이션에는 모든 컴퓨터들의 시간을 총괄적으로 관리하는 Master clock을 두는 방법을 사용하며, Master clock은 공장 통제 컴퓨터의 내의 Timer라고 부르는 독립된 한 프로세스에서 운영한다. 시계의 전진은 컴퓨터 중 하나 이상이 물리적시간(이하 실시간)을 소모하며 운용소프트웨어를 수행하고 있을 때에는 컴퓨터에 내장된 물리적 시계에 의하여 연속적으로 실시간이 흐르게 되며 실시간 작업이 수행되지 않을 때에만 next-event 시간으로 점프하는 방법으로 이루어진다. 통제 컴퓨터들은 이산형 이벤트를 시뮬레이션 한 후에 next-event 시점을 계산하거나 다른 용도로 현재 시점을 알아야 할 때에는 Timer로부터 Master clock을 읽어오는 방법이 이용된다.

2.2 시계 운영 방법

본 연구의 혼합 · 분산 시뮬레이션은 장비의 동작을 시뮬레이션한다는 점을 제외하면 실제 시스템과 동일하다. 따라서, 시스템 내의 컴퓨터 중 어느 하나라도 장비의 동작 시뮬레이션에 관련된 작업 외에 다른 용도로 소프트웨어가 구동되고 있다면 이는 물리적 실시간의 진행을 의미한다. 그러므로, 시스템의 각 컴퓨터에는 flag를 두어 실시간 작업을 할 때는 1, 아니면 0의 값을 갖도록 하였다.

이제 전체 시스템의 시계의 동기화를 수행하는 Timer의 구조와 역할을 설명하기로 한다. 이미 앞 절에서 설명한 바 있듯이 Timer는 공장 통제 컴퓨터에 별개의 프로세스로 위치하며, 변수로는 Master-clock(간단히 CLOCK라고는 부른다.), 각 컴퓨터에서 보내온 flag 값, 컴퓨터 i(i=1, …, 장비통제 컴퓨터의 수)가 시뮬레이션 중인지를 가리키는 이진변수 gen(i), 그리고 각 컴퓨터들이 현재 보유하는 이산형 이벤트 시점들 중 가장 먼저 일어나게 될 이벤트 시점을 갖는다. 여기서 gen(i)는 컴퓨터 i 가 시뮬레이션을 수행 중이면 1, 아니면 0의 값을 갖는다.

따라서, Timer 변수를 정리하면 {CLOCK, (flag(1),……, flag(N)), (gen(1)……, gen(N)), (nevent(1)……, nevent(N))}으로 구성된다. 여기서 N은 컴퓨터의 수를 의미한다. nevent(i)는 컴퓨터 i의 가장 빠른 이벤트 시점이며 컴퓨터 i에서 이벤트가 존재하지 않으면 nevent(i)=0의 값을 갖는다.

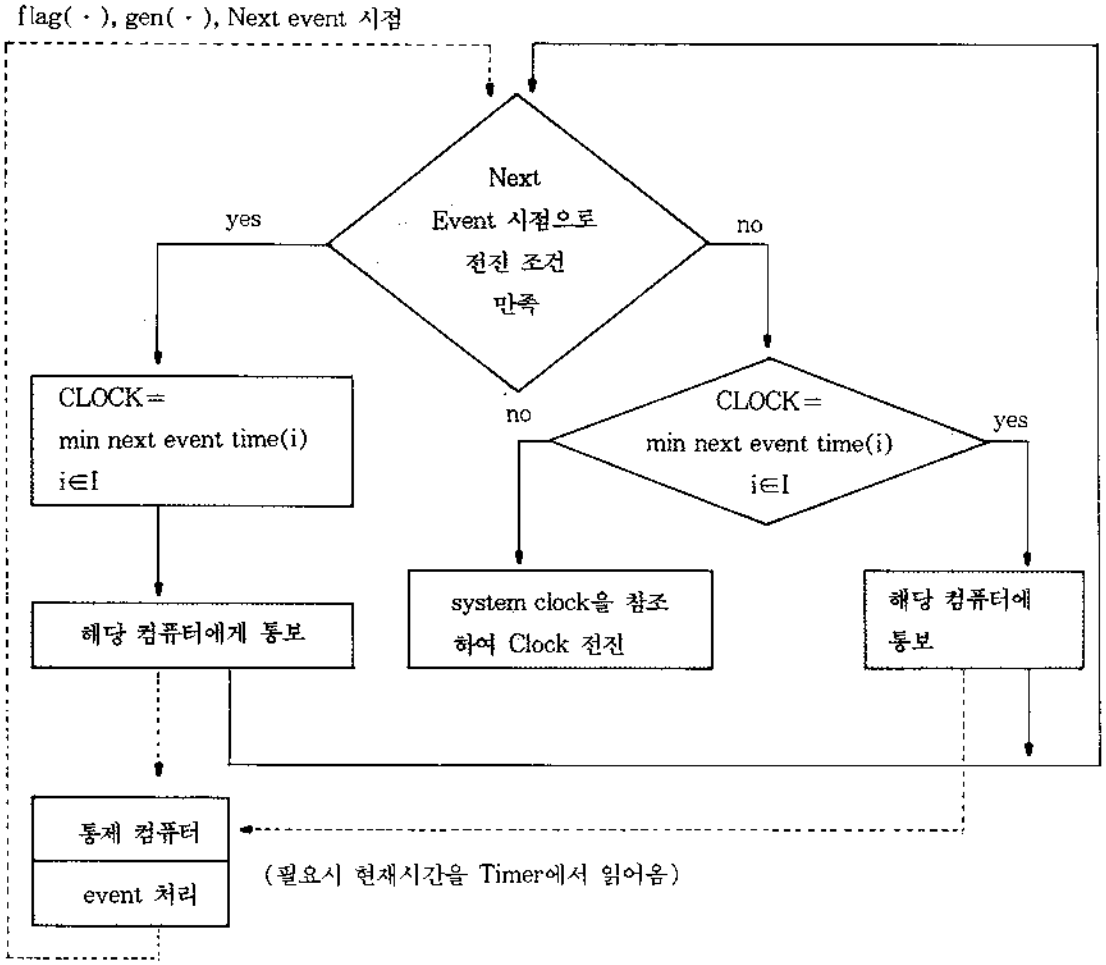
기본적으로 Timer는 모든 flag(i)=0의 값일 때 이벤트 시점들 중에서 가장 이른 시점을 갖는 컴퓨터에게 이 시점으로 전진하도록 허락하고, CLOCK을 nevent(i)로 전진시킨다.

그런데, 여기서 한가지 문제는 컴퓨터에서 발생중인 이벤트 시점이 Timer가 CLOCK으로 결정한 가장 빠른 이벤트 시점보다도 더 빠를 수 있다는것이다. 따라서, Timer는 비록 모든 flag이 0이더라도 gen 중 하나라도 1값을 가지고 있으면 현재시점을 전진시키지 않고 모든 flag와 gen이 0이 될 때까지 기다린 후에 CLOCK의 전진 여부를 결정한다.

현재시점을 전진시킬 조건이 아니면 실제 시계(컴퓨터 내부에서 구동되는 물리적 시계)를 구동시키고 이 시계에서 시간을 받아 연속적으로 CLOCK을 갱신하여 나간다. 이렇게 연속적으로 갱신하여 나가던 CLOCK이 nevent 중에서 가장 빠른 이벤트 시점과 일치하면, Timer는 해당 컴퓨터 i에게 이벤트 시점임을 통보하여 이벤트 시점의 전진을 허락한다. 통제 컴퓨터는 Timer로 부터 이벤트 시점 전진을 허락하는 통보를 받았을 때 이벤트를 처리하고, 다른 컴퓨터에게 정보를 전달하거

나 자신이 다음 이벤트를 발생시킨다. 컴퓨터는 자신이 발생시키는 이벤트가 실시간 작업이면 flag=1을 Timer에게 전달하고, 실시간 작업이 종료되면 flag=0으로 환원하고 Timer에 전달한다. 이벤트가 시뮬레이션이면 gen=1을 Timer에게 전달하고, 시뮬레이션이 종료되면 gen=0과 이벤트 시점을 Timer에게 전달한다. 통제 컴퓨터가 현재시점을 알아야 할 때에는 Timer로부터 CLOCK을 읽어오는 방법이 이용된다.

이러한 과정들은 [그림 1]과 같이 요약하여 표현할 수 있다. 여기에서 I는 Timer에게서 시간의 흐름을 통제받는 통제 컴퓨터들의 집합을 의미한다.

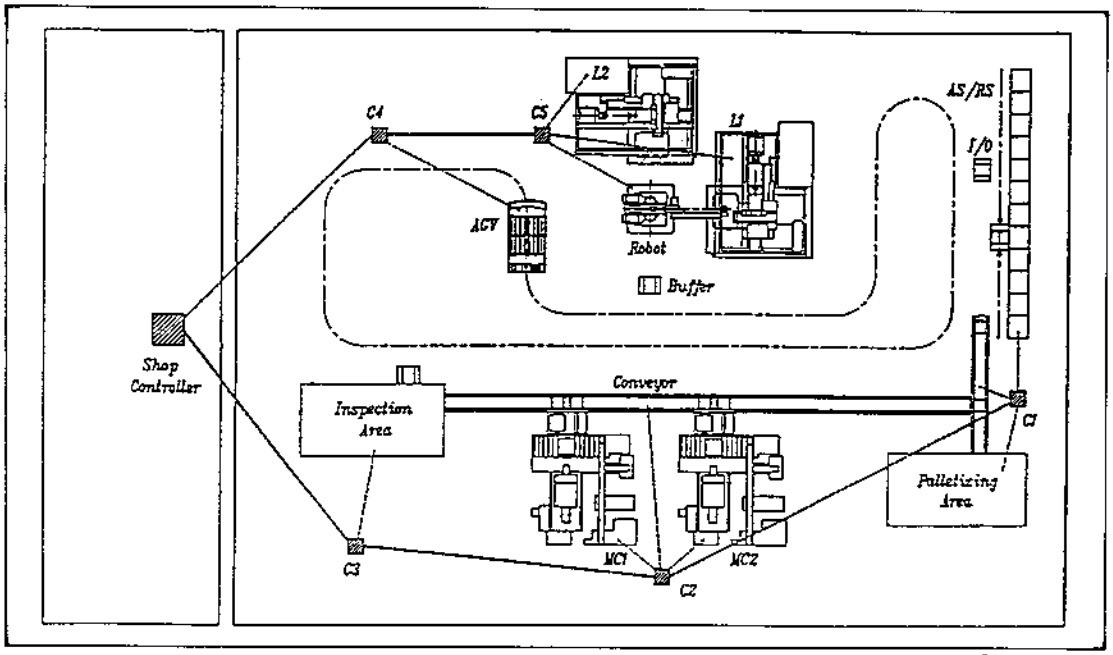


[그림 1] : Clock의 전진과 통제컴퓨터와의 정보 교환, 점선은 컴퓨터 간에 교환되는 정보의 흐름 표시.

3. 적용예

본 연구에서 제안한 "Shell의 구성 → 분산시뮬레이션을 이용한 모형의 구축 → 실제 시스템의 구축" 등의 과정에 대한 적용 가능성을 실험하기 위하여 실제 시스템을 구축하면서 이 과정을 적용하여 보았다. 이 시스템은 고려대학교 내에 위치하며 K. U. FMS[1, 2]라고 명명된 시범 FMS이다.

K. U. FMS는 부품가공 작업을 수행하는 시스템으로서 [그림 2]에서 보는 바와 같이 두개의 작업장, 자동창고, 그리고 물류시스템 등으로 구성되어져 있다. 작업장 1은 밀링 머신 2대와 컨베이어로 구성되어지며 작업장 2는 자동선반 2대, 물류용 로보트 1대, 그리고 임시저장소로 구성되어져 있다. 작업장 간의 연결과 작업장 2에 대한 팔렛의 공급은 무인운반차가 수행한다. 작업장 1과 자동창고 간의 작업물의 흐름은 컨베이어가 처리한다.



[그림 2]K. U. FMS 구성도

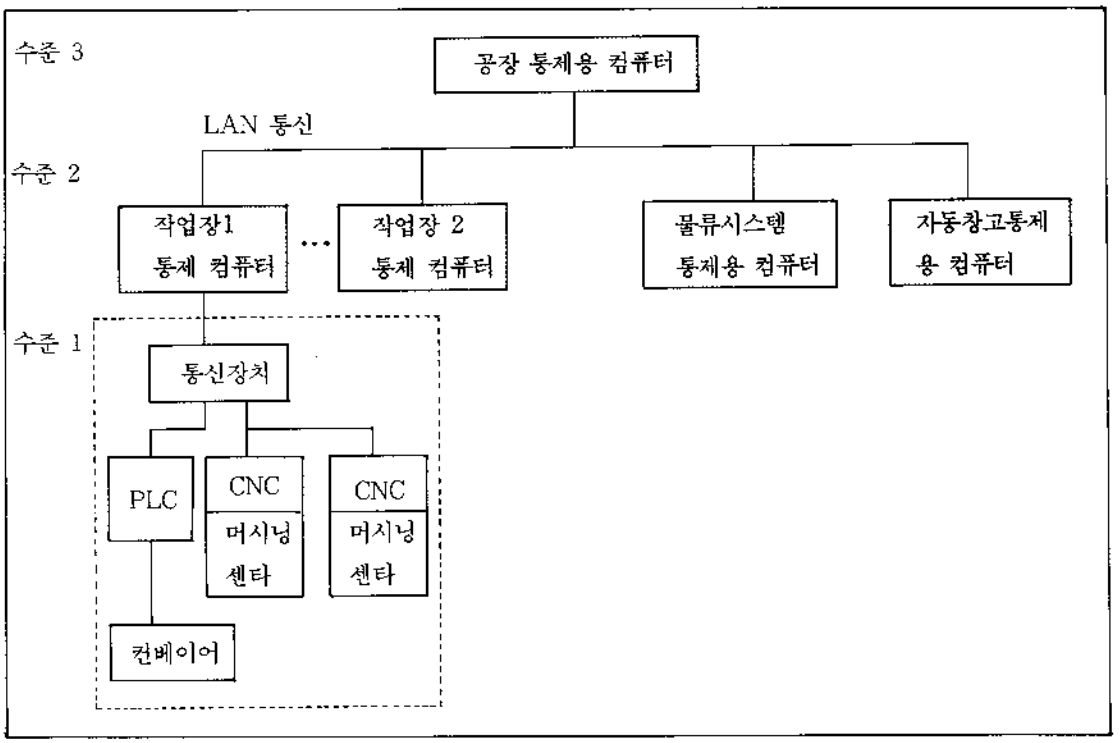
Ci : Control Computer
Li : 선반
MCi : Machining Center

작업장 2에서는 자동선반에 대한 팔렛 공급과 수급은 물류용 로보트가 처리하고, 가공대기 중인 작업물과 출고한 작업물은 임시저장소에서 보관한다. 작업장 1의 머시닝센타에게 작업물을 공급하고 가공이 완료된 작업물을 수령하는 일은 컨베이어가 맡는다.

기계부품의 가공을 목적으로 설치된 이 시스템에서는 가공품이 M/C와 선반들의 작업을 필요로 하는 부품의 종류와 수량은 기계들의 처리 능력의 한계에 의해서 제한을 받을 뿐이며, 시스템의 실시간 계획 및 운영 기능들의 능력에 의해서는 기본적으로 제한을 받지 않는다.

3.1 계층별 기능

Shell을 구성하기 위해서는 먼저 FMS를 몇개의 통제계층으로 볼 것인가가 중요하며 이에 따라 Shell의 구조가 결정된다. 본 연구에서는 FMS를 [그림 3]처럼 3수준으로 보았다.



[그림 3]K. U. FMS의 계층도]

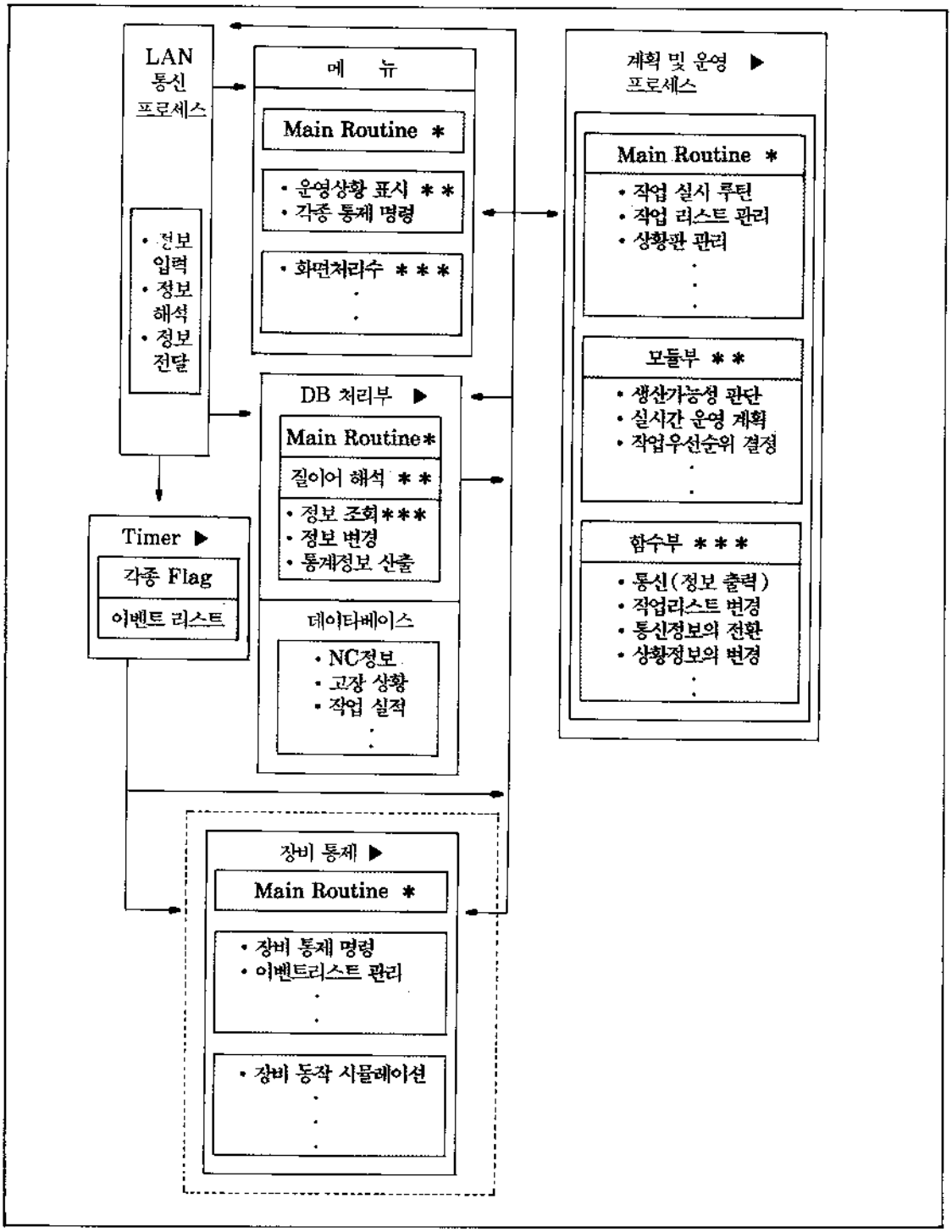
가장 상위 계층인 공장 통제 기능은 주문(order)에 대한 납기준수 가능성 여부 판단, 작업물들의 우선순위, 투입순서와 시기의 결정, 수준 2에 해당하는 통제 컴퓨터들에 대한 작업명령의 하달, 공장 상황 및 작업 관련 자료 보유 및 관리 등의 기능들이 포함되며 공장 전체 운용의 효율을 높이는 것이 주된 목적이다.

두번째 수준인 장비 통제 기능은 상위 기능의 명령을 목적함수로 하고 현재의 상황들을 제약조건으로 하여 자신의 통제 기능 밑에 있는 장비들의 운영 효율을 높이는 것이 주목적이다. 작업장의 통제 기능에는 작업물 처리순서의 결정, PLC, CNC 등의 제어기들의 통제, 다른 장비군과의 작업물 교환, 운영에 필요한 자료의 유지, 그리고 제어기들로부터 고장접수 및 처리 방안 지시 등이 주업무이다. 수준 2에는 가공작업장, 조립작업장, 물류 및 창고 관리자 등이 있다.

가장 하위 수준인 수준 3에는 장비들이 속해 있고 엄밀한 의미에서 각개의 장비를 통제하는 제어기가 이 수준에 속한다. 이들의 기능은 해당 작업을 제어하는 것으로 장비의 종류에 따라서 그 특성이 다르다.

3.2 Shell의 구성

FMS의 계층별 기능을 수행할 때 제기되는 문제



[그림 4] Shell의 구조, ※▶ : LAN통신 함수(정보의 출신) 표시, 점선 : 장비를 통제하는 컴퓨터에만 존재하는 프로세스, Timer : 중앙컴퓨터에만 존재

는 각 컴퓨터에서 복수의 기능이 동시(concurrently)에 수행되어야 한다는 점이다. 예를 들어서, 데이타베이스가 어떤 자료를 검색하고 있는 중에도 기계에 대한 동작 명령은 내려져야 한다. 따라서, Shell을 구성할 때 먼저 고려해야 할 사항들은 기능들을 군(group)으로 분류하여 같은 군에 속하는 기능들은 상호배타적으로 수행되고 반면에 군이 다를 때는 동시에 수행될 가능하도록 하여야 한다는 것이다.

본 연구에서는 FMS의 통제 컴퓨터들에 대한 요구기능들을 다음과 같이 군으로 나누었다.

- 계획 및 운영
- 시스템과 사람간의 인터페이스(메뉴)
- DB 처리 및 DB
- 장비 통제(장비통제컴퓨터에 만 존재)
- 통신(외부로 부터의 통신 수령)

제안된 Shell에서는 이와 같이 각 군별로 독립된 소프트웨어(이하 프로세스라고 명명함)들을 Timer(공장 통제 컴퓨터에 존재)와 함께 구성하고 이들 간의 정보교환은 컴퓨터의 내부통신을 이용하여 가능하도록 하였다.

각 프로세스의 구조를 설명하기로 하자. 기본적으로 프로세스는 다른 컴퓨터나 프로세스에서 보내온 요구사항에 Trigger되어 구동되어 진다. 따라서 각 프로세스는 이러한 요구사항을 하나씩 처리하여 나가기 때문에 이들을 쌓아 놓아야할 장소가 필요하며, 이 장소를 작업리스트라고 부른다. 메인루틴은 작업리스트의 내용을 하나씩 불러들여 이에 알맞는 일을 수행하게 되며, 이러한 행위는 해당 모듈을 차례로 불러서 수행하게 된다. 모듈은 독립기능으로 짜여져 있으며, 시스템의 구조나 운용 형태가 바뀌면 모듈에 해당하는 내용이 바뀌게 된다. 프로세스들은 한가지 일을 처리하면 그 결과나 명령을 다른 프로세스 또는 다른 컴퓨터에게 전달한다. 이때, 같은 컴퓨터 내에서 프로세스 간에 정보가 교환되는 경우에는 컴퓨터의 내부통신을 이용하고, 다른 컴퓨터와 정보를 교환하는 경우에는 LAN을 통하게 된다. 본 연구에서는 내부통신을 UNIX에서 제공하는 Message-queue를 사용하였다. Message-queue를 통하여 프로세스에 정보가 입력되면 작업리스트에 쌓이게 된다. Shell에서는 모듈들의 위치를 지정하고 구조를 정하여 주며, 모듈의 내용은 실제 Shell을 사용하는 user가 작성하게 된다. 또한, 프로세스에는 함수부가 지정되어 있는데 이는 메인루틴과 모듈부가 기능을 수행할 때 공통적으로 수행하는 루틴들로 이루어져 있다. 예를 들어서, 작업리스트의 내용 해석, 장비에 명령을 내리는 일, 다른 컴퓨터에게 정보를 전달하는 통신 등이 여기에 속한다. [그림 4]에는 Shell의 구조가 모듈, 함수들이 제시되어 있다.

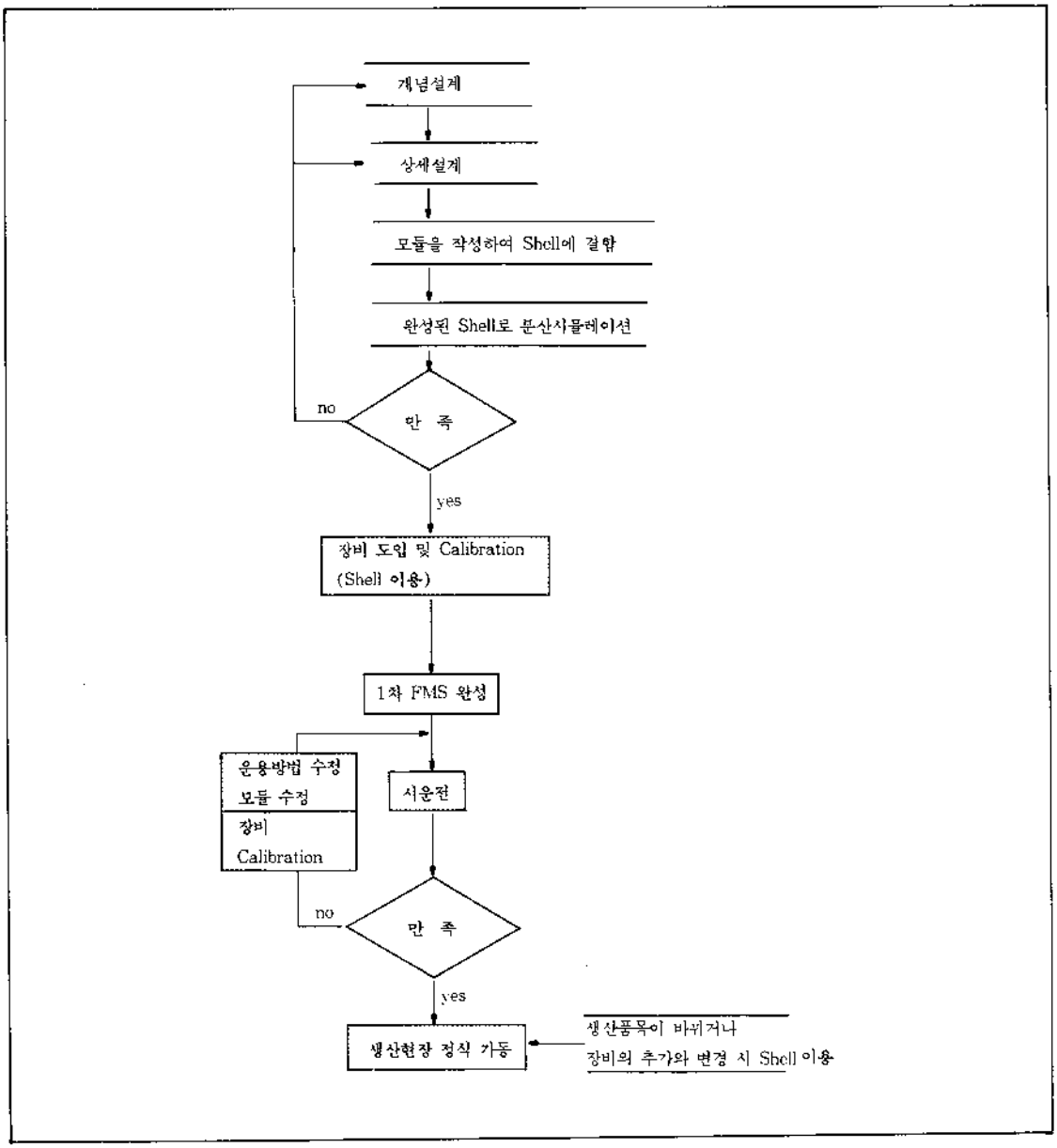
이제 Shell을 구축하는 환경에 대하여 설명하기로 하자. Shell은 한 컴퓨터에서 한 프로세스 만이 구동되는 환경 하에서 시분할(Timer Sharing)을 하여 구축할 수도 있다. 그런데, 이 방법은 프로세스 간의 대화가 어렵고, 입력되는 정보량이 커지면 시스템이 불안해지는 단점이 있다. 또한, Shell은 여러 프로세스가 동시에 구동되며, 프로세스간의 정보 교환량이 많으므로, 이 방법을 사용하는 것은 무리가 있다. 다라서, 본 연구에서는 Shell을 멀티타스킹이 가능하고 프로세스 간의 대화가 용이한 UNIX와 XENIX시스템 하에서 구축하였다.

3.3 Shell을 이용한 FMS구축

일반적으로 FMS 구축은 개념 설계, 상세 설계, 구축 및 실험, 운영 및 유지의 과정을 거친다. 서론에서 언급한 바와 같이 각 단계에서 사용할 수 있는 도구들은 많지만 이것들이 시험과 검증 방법을 제공하지 못하므로 구축 단계와 운영 및 유지 단계에서 많은 시행착오를 거치게 된다. 그런데, 본 연구에서 제안하는 방법은 장비가 존재하지 않는 상태에서 완전한 운영시스템을 구성할 수 있는 Shell이 존재하므로 Shell에 시스템에 필요한 모듈

들만 구축해 넣고 완전한 실험을 할 수 있다. Shell을 이용하면 시행착오를 인한 위험부담을 줄일 수 있고, 시스템을 구축하는 마지막 단계에서 고가의 장비를 도입하게 되므로 FMS 구축에 대한 비용 부담이 적은 장점이 있다.

Shell을 이용하여 FMS을 구축하는 과정은 [그림 5]에서 보는 바와 같다. 개념설계 단계에서 FMS 도입 시의 경제성, 생산성, 시스템의 신뢰성 등을 분석하고, 이들을 종합하여 개념적인 시스템을 설계한다. 상세 설계 단계에서는 개념 설계에서 제시된 결과들을 바탕으로 기계의 사양을 결정하고, 공정 설계, 생산 계획의 수립, scheduling, 작



[그림 5]Shell을 이용한 FMS 구축 과정

업 우선순위와 작업물 투입순서의 결정, 공구(Tool)의 선택 등의 생산과 운영에 관련된 기능들을 결정하고, 이들 기능을 바탕으로 데이타베이스의 내용 결정, LAN 선정 들의 작업을 수행한다.

상세 설계 단계에서는 일반적인 시뮬레이션을 통하여 설계된 FMS의 작동상태를 확인하고 수정하는 작업을 거치게 된다. 상세 설계가 완료되면 이 때부터 Shell에 들어갈 모듈을 작성한다. 모듈의 작성은 Shell에서 지정된 형식에 따라 이루어지며, 상세설계에서 계획된 내용을 모두 반영한다. 또한 메뉴화면의 작성, 통신코드의 작성, 장비의 시뮬레이션을 위한 특성 분석, 분산시뮬레이션의 결과 제시 방법의 결정 등 모듈과 공용함수 외에도 여러가지 사항이 프로그램되거나 결정되어야 한다.

이 과정이 끝나면 "장비는 없지만 실제와 같이 작동되는" FMS가 완성된 것이다. 이러한 분산시뮬레이션 패키지를 가지고 실험을 계속하여 작동상황을 살피고 만족스럽지 않을 때는 기종과 운영방법들을 수정하여 나간다. 일단 만족할 만한 수준에 도달하면, 이 시스템을 현장에 옮기고 실제 장비들을 연결하게 된다. 이때 장비들에 장착되어 있는 각 컨트롤러마다 특성과 기능이 다르기 때문에 장비들을 한대씩 연결하여 작동여부를 점검하여 나간다. 점검할 때는 이미 연결된 장비들은 실제로 작동시키고 연결되지 않은 장비들은 시뮬레이션시키는 방법을 택하게 되며, 이로서 각 장비가 전체 시스템안에서 다른 장비와 조화를 이루며 제대로 역할을 하는지를 실험한다. 이러한 과정을 통하여 모든 장비들이 연결되고 작동이 확인되며 FMS가 완성된 것이며, 이때부터 시 운전을 시작할 수 있다. 생산계획에 따라서 실제 제품을 생산하게 되지만 여기서 시스템이 완전히 구축되었다고 볼 수는 없다. 아직도 예상하지 않았던 문제가 기계적인 부분, 운용 부분, 자료 부분에서 발생할 가능성이 높으며, 계속적인 수정을 통하여 안정화시켜야 한다. 이때 새로운 운용방법이나 운용방법의 변화는 현장에서 직접 실험하지 않고 FMS분산시뮬레이션 패키지 상에서 이루어질 수 있으며 이로서 공장의 가동중단을 최소화할 수 있다. 시운전 결과가 만족한 수준에 도달하면 드디어 신뢰성 있는 FMS가 완성되어진다. 완전한 FMS가 완성되어졌다 하더라도 FMS 분산시뮬레이션 패키지의 활동이 중단되는 것은 아니다. 시스템의 운용방법 변경, 신제품의 생산, 장비의 변화가 일어날 때마다 현장에서 실현하기 전에 FMS 분산시뮬레이션 패키지를 이용하여 먼저 실험을 한 후에 문제점을 보완할 수 있다.

K. U. FMS 구축을 통한 경험에 의하면 모듈의 작성 부분이 기계를 연결하는 부분보다 월등히 많은 시간과 노력이 소요되었다. 이는 모듈 작성이 기계의 특성을 파악해야만 가능하고 기계 간에 부품을 교환하는 방법, 이를 위한 기계적인 미캐니즘 등도 규명되어야 하기 때문이다. 즉, 모듈의 작성이 끝나고 나면 장비에 대한 상당히 많은 사항들이 모듈에 반영된 상태이며, 기계의 연결은 컴퓨터와 장비간의 대화 확인, 기계적인 조정문제의 해결로 완성되기 때문이다.

현재 K. U. FMS에서는 FMS 분산시뮬레이션 패키지를 이용하여 시스템을 계속 테스트하면서 부품의 생산계획, Scheduling와 전문가시스템을 구축 중에 있다.

4. 결 론

본 연구에서 제안한 FMS 구축방법은 신공장을 건설하거나 진화적인 접근을 할 때 사용이 용이한 방법이며, 여기서 사용된 혼합-분산시뮬레이션은 시뮬레이션을 통한 실제시스템으로 진화하는 방법을 취할 때 매우 유용한 수단이 된다. 분산시뮬레이션에 기초한 Shell은 여러 종류의 FMS의 구축에 높은 응용성을 갖추고 있으며, 저렴한 비용으로 FMS를 구축할 수 있으므로 예산이 적은 중소기업에서도 사용할 수 있는 방법이다.

현재 고려대학교에서는 user의 요구에 쉽게 대

응하기 위해 Shell에 전문가시스템을 비롯한 다양한 기법들을 적용할 수 있는 방법을 계속 연구 중이다.

참 고 문 헌

[1] 김성식, 배경한, "컴퓨터를 이용한 실제에 준하는 FMS구축", 「산업공학」, Vol. 4, No. 1, pp. 83-91.

[2] 김성식, 배경한, "CIM 구축 Tool로서의 네트워크 상의 분산시뮬레이션", 「'91한국자동제어학술회의 논문집」, Vol. 1, pp. 799-803, 1991.

[3] Chandy, K. M. and Misra, J., "Distributed Simulation : A Case Study in Design and Verification of Distributed Programs", *IEEE Trans. on Soft. Engin.*, Vol. SE-5. t, pp. 440-452, September 1979.

[4] Chandy, K. M. and Misra, J., "Asynchronous Distributed Simulation via a Sequence of Parallel Computations" *Comm. of the ACM*, Vol. 24, Mo. 11, pp. 198-206, April 1981.

[5] Doumeingts, G., "GRAI Approach to Designing and Controlling Advanced Manufacturing System in CIM Environment", *Advanced Information Technologies for Industrial Material Flow Systems*, Comp. and Syst. Sci., Vol. 53, pp. 461-529, 1989.

[6] Doumeingts, G., Vallespir, B., Darricau, D., and Roboam, M., "Design Methodology for Advanced Manufacturing Systems", *Computers in Industry*, Vol. 9, pp. 271-296, 1987.

[7] Greenwood, N. R., *Implementing Flexible Manufacturing Systems*, MACMILLAN EDUCATION LTD, 1988.

[8] Huska, A. M., "The TESPA-Concept of Flexible Manufacturing System/Evolutionary Implanting of Flaxibility into Manufacturing System", *Modelling and Design of F. M. Ss*, edited by Kusiak, A., ELSEVIER, pp. 137-156, 1986.

[9] Jefferson, D. R., "Virtual Time,"*ACM Trans. on Prog. Lang. and Sys.*, Vol. 7, No. 3, pp. 404-425, July 1985.

[10] Leva, A. D., Vernadat, F., "Information System Analysis and Conceptual Database Desinmg in Production Environments with M*", *Computers in Industry*, Vol. 9, pp. 183-217, 1987.

[11] Levi, S. T., Agrawala, A. K., Real Time System Design, *McGraw-Hill Computer Science Series*, 1990.

[12] Meredith, J. R., "Implementing the Automated Factory", *Jou. of Man. Sys.*, Vol. 6, No. 1, pp. 1-13.

[13] Misra, J., "Distributed Discrete-Event Simulation", *Computing Surveys*, Vol. 18, No. 1, pp. 39-65, March 1986.