분산 전문가 시스템의 기능을 갖는 이산사건 시뮬레이션: 제조 공정 오류 감지와 진단에의 적용*

Discrete Event Simulation with Embedded Distributed Expert System: Application to Manufacturing Process Monitoring and Diagnosis

조대호**, 김형종**
Tae-ho Cho, Hyung-jong Kim

Abstract

One of the components that constitute the simulation models is the state variables whose values are determined by the time related simulation process. Embedding rule-based expert systems into the simulation models should provide a systematic way of handling these time-dependent variables without distracting the essential problem solving capabilities of the expert systems which are well suited for expressing the decision making function of complex cases. The expert system, however, is inefficient in dealing with the time elapsing characteristics of target system compare to the simulation models. To solve the problem, this paper provides an interruptible inference engine whose inferencing process can be interrupted when the variables' value, which are used as the parameters of the rules, are not yet determined due to the time dependent nature of the state variables. The process is resumed when the variables are ready. The elapse of time is calculated by time-advance function of the simulation model to which the expert system has been embedded. The example modeling shown exploits the embedded interruptible inferencing capability for the controlling and monitoring of metal grating process.

* 본 논문은 1997년도 한국 학술 진흥 재단 공모 과제 연구비 지원으로 수행 되었음.

** 성균관 대학교 전기 전자 컴퓨터 공학부

1. 서론

인공지능과 시뮬레이션의 복합연구의 필요성은 많은 학자들에 의해서 강조되어 왔다. 그리고, 최근 수년간 인공지능 원리를 시뮬레이션에 적용한 연구 결과가 지속적으로 발표되고 있다. 이 복합 연구의 종류를 크게 둘로 나눌 수 있다. 첫째는 인공지능을 이용한 시뮬레이션으로 인공지능 이론이 주가 되는 경우이고, 둘째는 인공지능의 도움에 의한 시뮬레이션으로 시뮬레이션 모델의 한 구성요소로 인공지능을 사용하는 경우이다[1].

여러 인공지능의 이론 중 전문가 시스템을 시뮬레이션 모델에 삽입하는 연구는 [2][3]에서 다루고 있다. 특히, Cho[4][5]에서는 객체 지향적인 시뮬레이션 환경에 분산 전문가 시스템을 삽입하는 접근을 atomic-expert model을 통해 보여 주었다. Atomic-expert model은 지식 기반 제어(Knowledge-based control)를 위한 시뮬레이션 모델의 구성과 많은 종류의 지능형 시스템의 구성에 용이성을 제공하였다.

일반적인 전문가 시스템의 추론 엔진은 전향, 후향 추론을 통해 문제를 해결한다. 그런데, 만일 추론하는데 사용되는 사실(fact)가 유효하지 않다면 추론의 결과는 나올 수 없거나 나오더라도 완전하지 못한 결과가 될 것이다. 이런 문제를 해결하기 위해서 추론 엔진은 두 가지 방법을 사용한다.

첫째는, 유효한 사실(fact)을 얻기 위해 데이터베이스를 찾거나, 외부 루틴을 실행하거나, 사람에게 그 사실(fact)을 입력할 수 있는 입력 창을 띄어주는 방법이다. 둘째는, 완전하지 않지만 기존의 주어진 사실(fact)만을 가지고 추론하는 것이다. 이런 경우 확신 지수(Certainty Factor)를 사용해서 추론 결과의 신뢰도를 표시 한다. 일반적으로 추론 기관은 이 두 가지 방법을 동시에 사용한다[12][13][15][16][17]. 이런 전문가 시스템을 시뮬레이션 모델에 사용할 경우에도 역시 가장 고려해야 할 점이 사실의 유효성에 관한 문제이다. 시뮬레이션 모델을 구성하는 가장 중요한 구성 요소 중 하나가 시간에 따라 변경되는 모델의 상태를 표시하는 상태 변수들이다. 상태 변수들은 시뮬레이션 모델에 의해서 생성된 이벤트(Event)들에 의해 실행 되는 이벤트 루틴을 통해 정해진다[10][11]. 이 상태 변수를 가지고 추론을 하는 규칙에서는 상태 변수의 유효성 여부를 점검하고, 이 유효성에 기초해서 추론해야만 한다.

기존의 atomic-expert model에서는 시뮬레이션 모델의 상태 변수의 변경으로 생기는 유효하지 않은 사실을 어떻게 처리해야 할 것에 대한 정의가 되어 있지 않다. 만일, 유효하지 않은 사실에 기초해서 추론을 할 경우 추론 결과를 제시하지 못하거나 잘못된 추론 결과를 제시하게 된다.

이런 문제를 해결하기 위해서, atomic-expert model의 추론을 위한 환경이 시뮬레이션 모델링 환경에 맞게 다음과 같은 성격을 갖게 디자인되어야 한다. 첫째, 유효하지 않은 사실(fact)을 인식할 수 있어야 한다. 둘째, 만일 유효하지 않은 사실(fact)이 존재할 경우 추론 기관의 추론을 중단할 수 있어야 하고 중단되었다는 사실을 atomic-expert model에 알려 주어야 한다. 세째, 인터럽트된 상태는 반드시 복귀되어서 추론의 결과가 제시되어야 한다.

본 논문에서는, 인터럽터블(interruptible)한 추론 엔진을 위한 추론 환경을 디자인하여 위의 문제점을 해결하였고, 이를 그레이팅 프로세스 스케줄링 시스템을 위한 DES(Dispatching Expert Model)를 구현하였다.

본 논문의 interruptible atomic-expert 모델은 이 모델 내에 삽입된 전문가 시스템이 시간 의존적인 사실에 기초해서 추론을 해야만 하는 경우에 보다 정확한 추론할 수 있도록 해준다. 여기서 제시한 모델링 방법은 시뮬레이션 모델을 이용한 실시간 제어기의 디자인 및 구현에 적용할 수 있다.

2장에서는 DEVS-formalism과 SES-formalism의 기본 개념을 살펴보고 atomic-expert 모델의 구성을 살펴본다. 3장을 통해서는 interruptible atomic-expert model의 디자인 내용을 볼 수 있다. 4장은 interruptible atomic-expert model을 이용한 그레이팅 프로세스 스케줄링 시스템의 디자인과 구현을 보여 준다. 5장은 시뮬레이션 결과를 6장은 결론과 향후 과제를 보여준다.

2. Background

이 장에서 DEVS-formalism, SES-formalism, 그리고 atomic-expert model에대한 기본 개념을 살펴 본다.

2.1 DEVS-formalism

DEVS(Discrete Event System Specification) formalism은 계층적이고 모듈화된 이산 사건 모델을 위해 정의된 이론이다[4][8][9]. 일반적으로 시스템은 시간의 흐름에 따라 입력, 상태, 출력, 상태 전이 함수들을 갖는다. DEVS-formalism은 시스템이 일반적으로 갖는 특성들을 정의하여 시스템을 모델링 할 수 있는 프레임웍(Framework)을 제공하였다. DEVS-formalism에서는 두 가지 종류의 모델을 정의하였다. 하나는 basic 모델이고, 다른 하나는 coupled 모델이다.

다음은 basic 모델의 구조이다.

$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$

where

X : a set of input events.

S : a set of sequential states.

Y : a set of output events.

$\delta_{int} : S \rightarrow S$: internal transition function

$\delta_{ext} : Q \times X \rightarrow S$: external transition function

$\lambda : S \rightarrow Y$: Output function

$ta : S \rightarrow R^{+}_{0} \rightarrow \infty$: Time advance function

where $Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$

e : 최근의 상태 전이(state transition)이후로 elapse된 time.

다음은 coupled model의 구조이다.

$DN = < D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, select >$

Where

D is a set of component names for each i in D.

M_i is a component basic model.

I_i is a set, the influences of i.

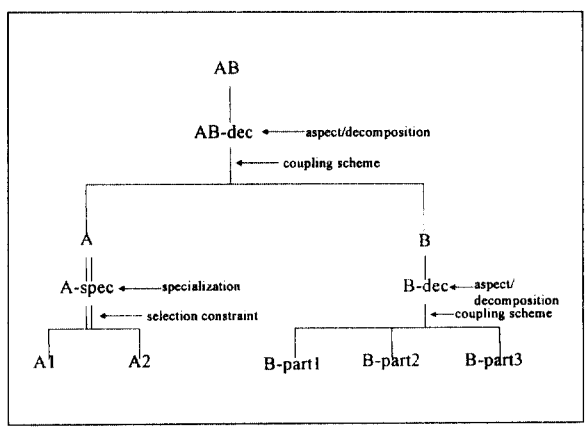
for each j in I_i , Z_{ij} is a function, the i-to-j output translation.

Select is a function, the tie-breaking selector.

DEVS-scheme은 DEVS-formalism을 기반으로 계층적이고 모듈화된 시뮬레이션 모델을 만들수 있는 환경이다. DEVS-scheme 시뮬레이션 환경은 DEVS-formalism을 기반으로 만들어진 abstract simulator를 사용해서 구축되었다.

2.2 SES(Systems Entity Structure) formalism

SES[4][18]는 존재하는 모델들을 종합하여서 시스템을 구축할 때, 시스템의 구조를 나타내는 지식을 특정 형식으로 표현한 것으로서 모델들의 분할(decomposition), 분류(taxonomy)와 모델간의 연결관계에 대한 정보를 가지고 있다. SES는 트리 형태로 계층적인 모델들을 정의하고, 트리의 말단은 모델베이스의 모델로 나타낸다. SES안에는 시스템의 구조를 표현하기 위한 지식으로 entity, aspect, specialization의 3가지 형태의 노드가 있다. Entity 노드는 실 세계의 한 객체와 대응되며 여러 개의 aspect 또는, specialization을 자식으로 가질 수 있다. aspect 노드는 <그림 1>의 한 수선 사이에 있는 노드로 나타나고 aspect노드의 자식 노드들이

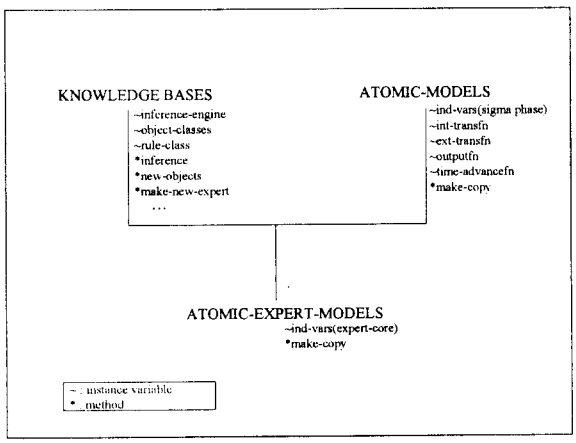


<그림 1> One Example of System Entity Structure

aspect노드의 부모 노드의 분할된 형태라는 것을 의미한다. 즉, <그림 1>의 A, B는 AB의 sub-component라는 것을 의미한다. Specialization은 <그림 1>의 수선 두개 사이에 있는 노드로서 이 노드의 자식 노드들은 부모노드의 하나의 특별한 형태들이라는 의미이다. 일반적으로 시스템은 SES의 sub-structure로 표현된다. 이 sub-structure를 PES(Pruned Entity Structure)라고 하며 SES에서 PES를 얻어내는 작업을 pruning이라고 한다. PES에는 오직 entity와 aspect만이 존재한다. 즉, specialization 노드의 자식 노드는 pruning과정에서 하나의 노드만 선택되고 specialization 노드는 사라진다.

2.3 Atomic-expert model

Atomic-expert model은 객체 지향적인 시뮬레이션 환경에 전문가 시스템을 삽입한 모델로서 <그림 2>와 같이 KNOWLEDGE BASES class와 ATOMIC-MODELS class, 두 class로부터 다중 상속을 받고 이 두개의 class의 인터페이스의 형태로 존재한다[5][6][7].



<그림 2> Atomic-expert model

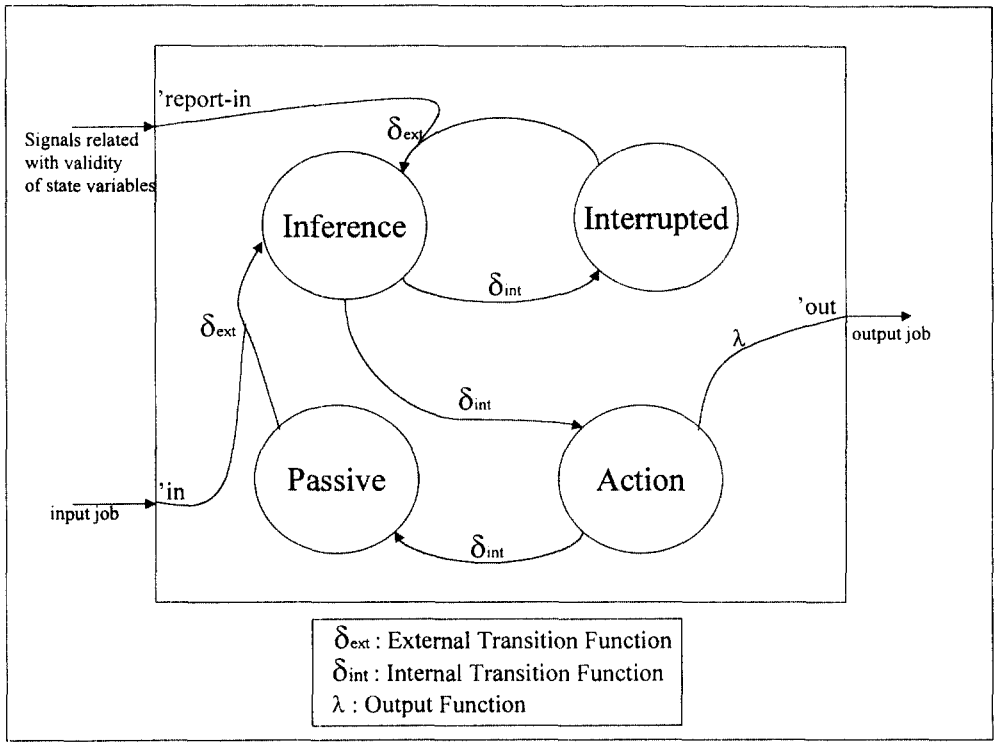
여기서 KNOWLEDGE BASE class는 시뮬레이션 모델을 위한 분산 전문가 시스템을 의미한다. Atomic-expert model을 통해서 DEVS-formalism 기반의 시뮬레이션 환경에 분산 전문가 시스템을 삽입하여 디자인, 테스트를 할 수 있는 환경이 구축되었다. Atomic-expert model은 기존의 다른 모델들과 동등하게 여겨지며 System Entity Structure에 바로 사용될 수 있다.

3. Interruptible atomic-expert model design

전문가 시스템은 사실(fact)의 유효 여부를 검사하여서 유효할 경우에만 추론을 한다. 만일 유효하지 않는 사실이 있을 경우 추론을 중단하고 유효한 사실을 얻을 때까지 기다리거나 현재 가지고 있는 사실만을 가지고 불완전한 추론을 한다. 전문가 시스템이 시뮬레이션 모델에 삽입될 때 이러한 전문가 시스템의 일반적인 특성이 반영되어야 한다. 특히, 시뮬레이션 모델에서는 상태 변수 값이 시간에 따라서 변경되기 때문에, 이 값들을 사용, 추론할 경우 이 값들이 유효한 값들인가에 대한 확인과 유효하지 않을 때 처리방법에 대한 정립이 필요하다. Atomic-expert model이 추론에 사용하는 상태 변수의 유효 여부를 점검하기 위해서 제어 대상이 되는 프로세스 모델의 현재 상태를 점검하여 알려 주는 모델이 필요하다. 본 연구에서는 이를 위해서 이벤트 기반의 제어[4] 모델을 사용하였다. 이벤트 기반의 제어 모델은 프로세스 모델의 정보를 가지고, 이 모델의 입력과 출력의 값 및 시점을 기준으로 모델의 상태를 파악한다. 본 연구에서 이벤트 기반의 제어 모델은 프로세스 모델의 상태를 atomic-expert model에 보내줌으로써 이 모델이 추론에 사용하는 프로세스 모델들의 상태 정보를 갱신해 주고, 그 정보들의 유효 여부를 알게 해준다.

<그림 3>은 interruptible atomic-expert model의 상태 전이 다이어그램(State Transition Diagram)을 보여 주고 있다.

Interruptible atomic-expert model은 Passive, Inference, Interrupted, Action의 4개의 필수적인 상태를 가지고 있다. Passive 상태는 현재 모델에 처리되고 있는 작업이 없고, 작업을 처리할 준비가 되어 있음을 나타낸다. Inference 상태는 모델 안의 전문가 시스템이 현재 작업을 처리하기 위한 추론을 하



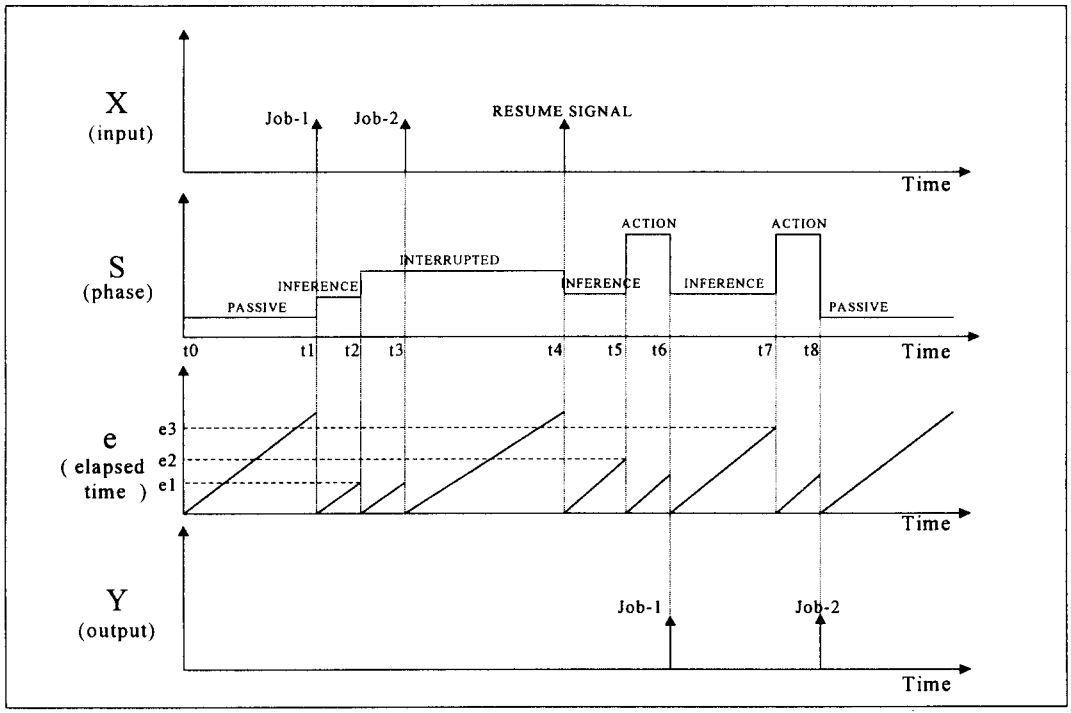
<그림 3> State Transition Diagram of Interruptible atomic-expert model

고 있음을 나타낸다. 이 상태에서는 모델 안의 전문가 시스템이 제어를 가지고 있다. Passive에서 Inference로의 상태 전이는 입력 포트인 'in를 통한 외부 입력에 의해서 구동되는 외부 전이 함수(External Transition Function)에 의해서 이루어진다.

Interrupted 상태는 모델 안의 상태 변수 값이 유효하지 않아 더 이상 추론을 진행할 수 없기 때문에, 추론 엔진이 인터럽트 된 상황을 나타낸다. Inference 상태에서 Interrupted상태로 전이되면 제어는 다시 전문가 시스템에서 DEVS-component(<그림 6> 참조)로 옮겨진다. 모델의 Interrupted 상태는 'report-in 포트를 통해서 resume signal을 받고 이에 기초해 외부 전이 함수를 구동함으로써, Inference상태로 전이되어야 한다. 여기서 resume signal은 유효하지 않았던 상태 변수 값을 유효하게 바꾸어 주는 데이터를 가지고 있다. 유효한 값을 얻게 되면 모델은 이 값을 가지고 정상적인 추론을 한다. 정상적인 추론이 끝나면 모델의 상태는 Inference에서 Action으로 전이된다.

Action상태에서 모델은 추론의 결과에 따라 동작을 한다. Action상태는 모델링 하는 대상에 따라 다른 이름의 여러 상태로 나누어질 수 있다. Action 상태가 종료되면 DEVS-component의 output 함수가 구동되어 출력을 제공한다. <그림 4>은 interruptible atomic-expert model의 timing diagram의 한 예를 보여 주고 있다.

입력을 보면 두 가지 타입으로 나타나는 것을 볼 수 있다. 첫째는, 처리되어야 할 작업이고 둘째는, 이벤트 기반의 제어 모델이 보내는 resume signal이다. t1이라는 시간에 외부 입력 Job-1이 Passive상태에서 모델 안으로 입력되면 모델의 phase는 Inference로 전이된다. 그리고, t2에서는 추론이 인터럽트 되는 것을 볼 수 있다. 이 추론 엔진이 추론을 진행하던 중 유효하지 않은 사실을 발견하고 추론을 중단한 것이다.

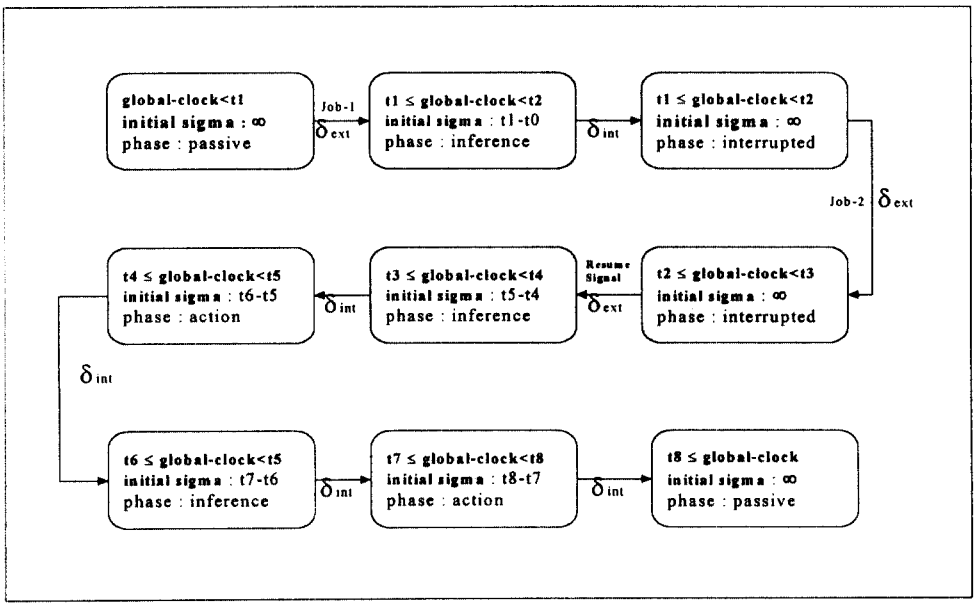


<그림 4> Timing Diagram of Interruptible atomic-expert model

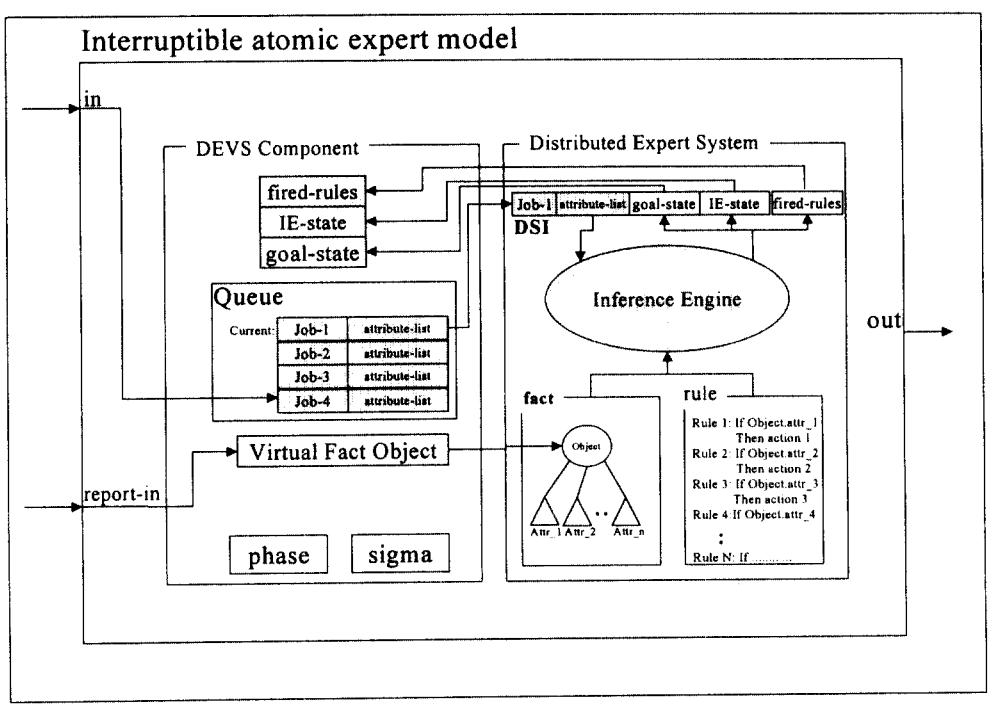
t3 시간에는 job-2가 입력으로 들어오고, 이 job은 현재 모델이 처리할 수 없는 상태이기 때문에 모델의 큐에 삽입된다. Interrupted 상태는 모델이 resume signal을 받는 시간 t4까지 지속된다. Resume signal이 입력되면 추론이 재개되면서 모델의 상태는 다시 Inference상태로 전이된다. 시간 t5에서는 전문가 시스템의 추론이 정상적으로 종료되어, 이 추론의 결과에 기초해 작업을 처리한다. Action 상태는 모델이 추론의 결과에 기초해서 작업을 처리하는 상태이다. 시간 t6에서는 Action상태가 종료되고 동시에 추론의 결과로 job-1이 출력으로 나간다.

그리고, 모델은 현재의 큐의 상태를 점검하여 t3에 입력 된 job-2를 처리하기 위한 Inference상태로 전이된다. 시간 t7에 추론이 정상적으로 종료되고 모델의 상태는 다시 Action상태로 전이된다. Action상태가 종료되는 t8에는 job-2를 출력한다. <그림 4>를 볼 때, job-1을 처리하기 위해서 사용된 Inference 시간은 e1+e2로 나타낼 수 있다. 이는 인터럽트 전의 추론 시간과 그 후의 추론 시간을 합한 것이다. 이 추론 시간은 전문가 시스템의 규칙을 실행(firing)하는데 걸리는 시간이라고 할 수 있다. 인터럽트가 일어나기 전에 이미 여러 규칙이 실행되었기 때문에 다시 Inference상태로 돌아온 경우에는 이미 실행된 규칙에 대한 정보가 있으면 그 규칙들은 다시 실행할 필요가 없고 그 만큼의 추론 시간이 절약되는 것이다. 이와 같이 이미 실행된 규칙에 대한 정보가 있을 경우, e1+e2와 e3의 차이는 단지, 유효하지 않았던 사실을 가지고 추론하는 규칙의 실행 시간만큼이 된다.

<그림 4>의 Timing Diagram은 <그림 5>의 순차적인 형태의 상태 전이 다이어그램으로 나타낼 수 있다. <그림 5>의 사각형 안에는 각 상태별로 모델이 갖는 중요한 속성들을 표시하고 있다. Global-clock은 시뮬레이션 환경에서 모든 모델들에게 공통적으로 사용되는 시간이다. Sigma값은 모델의 다음 이벤트 스케줄링 시점 즉, 현재 상태가 지속되는데 남은 시간을 표시한다. Phase는 모델이 갖는 상



<그림 5> State Transition Diagram in sequential form.



<그림 6> The components of Interruptible atomic-expert model

태를 의미하며 DEVS-component와 전문가 시스템의 상태가 모두 반영된 상태이다. <그림 5>에서 각각의 상태 전이는 외부 전이 함수와 내부 전이 함수를 통해서 이루어진다. 또한, Interrupted상태의 초기 sigma값이 무한대(infinitive)로 정해져서 resume signal에 의한 외부 전이 함수의 실행이 있기까지 이 상태는 지속된다.

<그림 6>은 interruptible atomic-expert model의 구성 요소를 보여주고 있다. 이 모델은 DEVS-component와 분산 전문가 시스템(Distributed Expert System)으로 구성되어 있다. 두개의 입력 port를 가지고 두 가지 종류의 입력을 받아 드린다. 그 중 하나는 모델에 의해서 처리되는 작업을 입력 받는 'in 포트이고 다른 하나는 프로세스 모델의 상태를 받는 'report-in 포트이다.

출력 포트로는 'out 포트를 갖고 추론 결과에 기초한 실행 결과를 내보낸다. Interruptible atomic-expert model의 전문가 시스템은 DSI(Data Structure for Inference)를 갖는다. 이는 현재 전문가 시스템의 상태를 표시하기 위한 메모리 공간이다.

DSI는 job-id, attribute-list, goal-state, IE-state와 fired-rules로 구성되어 있다. Job-id는 DSI의 인덱스로 현재 추론 대상이 되는 작업을 고유화 해주기 위한 용도로 사용된다.

Attribute-list는 추론 대상이 되는 작업의 속성을 담고 있으며 추론 엔진은 이 값들을 가지고 추론을 한다. Goal-state에는 추론의 결과를 저장한다. 만일 추론이 정상적으로 종료 되었을 경우에는 추론의 결과가 저장 되지만 정상적으로 종료되지 않은 경우, null값을 갖게 된다. IE-state는 추론 엔진의 현재 상태를 가지고 있다.

추론 엔진이 갖을 수 있는 상태로는 Inference, Interrupted, Passive 상태가 있다. Fired-rules에는 추론의 결과로 실행된 규칙의 이름들이 저장된다.

추론이 인터럽트 되었다가 재개 될 때 이 정보들은 이미 실행한 규칙에 대해서 다시 실행하지 않을 수 있도록 한다.

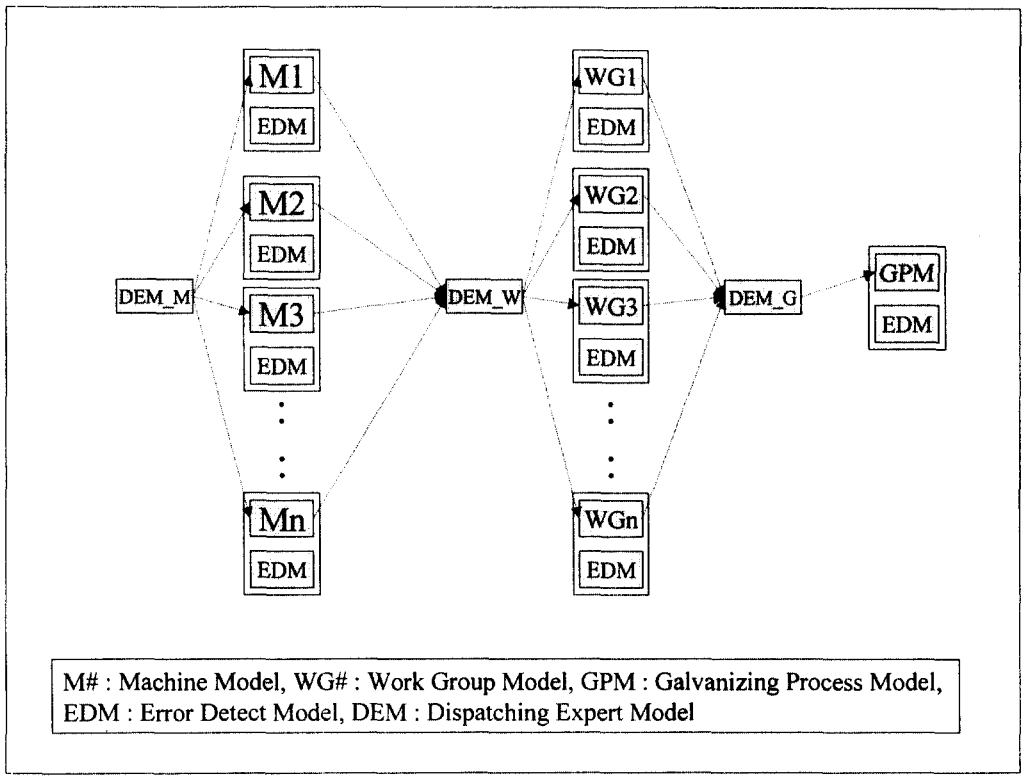
<그림 6>에 나타난 것처럼 전문가 시스템의 사실은 객체 지향적인 fact object의 형태로 관리 된다. Fact object들은 프로세스 모델의 현재 상태를 나타내어서 추론을 하는데 사용된다.

즉, 추론을 하는데 입력 사실로는 DSI의 attribute-list와 fact object의 값이 입력 정보가 된다. Interruptible atomic-expert model의 DEVS-component는 DSI의 구성 요소들이 갖는 모든 값과 같은 값을 갖는 상태 변수들을 가지고 있어야 한다. 그래서, DSI의 내용이 변경되면 동시에 그 상태 변수들의 값도 변경이 되어야 한다. Phase변수는 모델의 현재 상태를 나타내 주며, 이 현재상태는 DEVS-component와 전문가 시스템의 현재 상태를 반영한 것이다. 특히, DEVS-component의 Virtual fact object는 전문가 시스템의 추론 자료가 되는 fact object의 값을 변경하기 위한 통로로 제공되는 상태 변수이다. 특히, 시간에 따라서 변경되는 fact object의 상태를 virtual fact object를 통해 갱신해 줌으로서 프로세스 모델의 상태의 변화를 반영하는데 좀더 자연스러운 환경을 얻게 된다.

4. 그래이팅 프로세스 스케줄링 및 오류 진단에의 적용

이 시뮬레이션 모델 구축의 목적은 interruptible atomic-expert model을 사용하여서 작업 할당 환경을 구축하는 데 있다. 본 연구에서 atomic-expert model은 이벤트 기반의 오류 감지 모델의 프로세스 모델에 대한 상태정보 레포트에 기초해서 추론을 한다. 모델링의 대상이 되는 도메인은 그래이팅 생산 공정이다. 그래이팅 생산 공정은 다음 세 개의 단위 공정으로 나눠진다. 첫 공정은 베어링 바와 베어링 바에 크로스바를 가로질러 용접하여 그 크기를 원하는 데로 잘라내는 공정이다. 이 공정은 가공 머신에 의해서 진행된다. 두번째 공정은 첫공정에서 만들어진 그래이팅의 끝 부분에 붙일 엔드 바를 용접하는 작업이다. 3번째 공정은 두번째 까지 마쳐진 그래이팅을 도금하는 작업이다. 도금 공정에서는 각 작업의 무게를 기준으로 작업 배치(batch)를 만든다.

<그림 7>은 본 연구에서 구현한 시뮬레이션 모델의 전체적인 구조이다. 모델을 보면 DEM (Dispatching Expert Model), EDM(Error Detect



<그림 7> Overall Structure of Simulation Model

Model), M#(그래이팅을 만들기 위한 머신 모델), WG#(엔드 바 용접을 위한 가공 팀 모델) , GPM(도금 공정)으로 구성된 것을 알 수 있다.

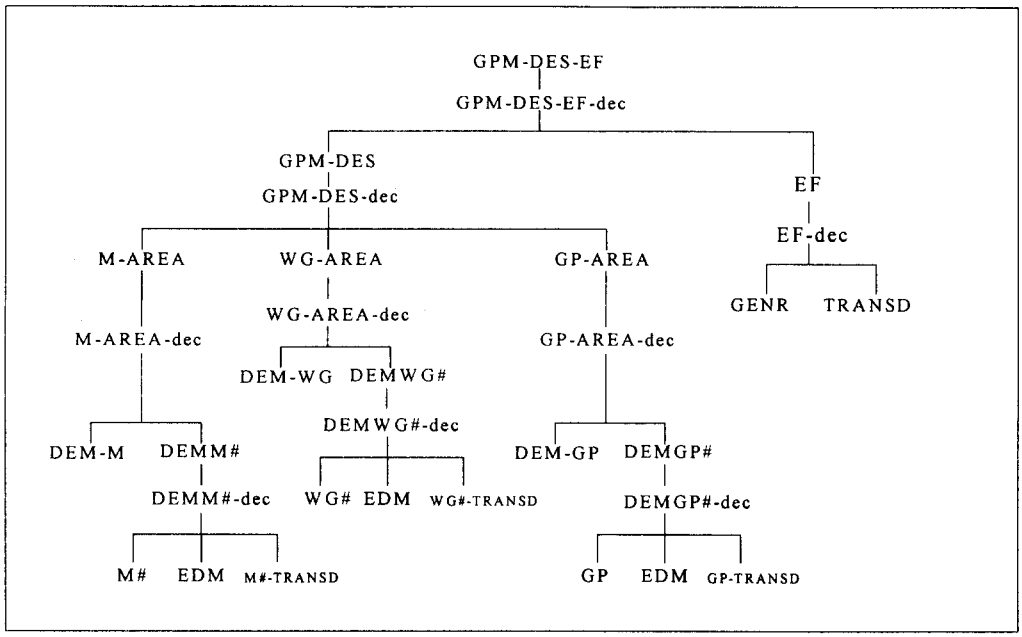
<그림 7> 모델은 목적에 따라 분류하면 프로세스 모델을 구성하는 M#, WG#, 그리고 GPM과 프로세스 모델의 지능형 제어를 위해 존재하는 EDM, DEM모델로 구분된다.

그레이팅 프로세스 스케줄링을 위한 모델들과 experimental frame의 구조는 <그림 8>에 나타나 있다.

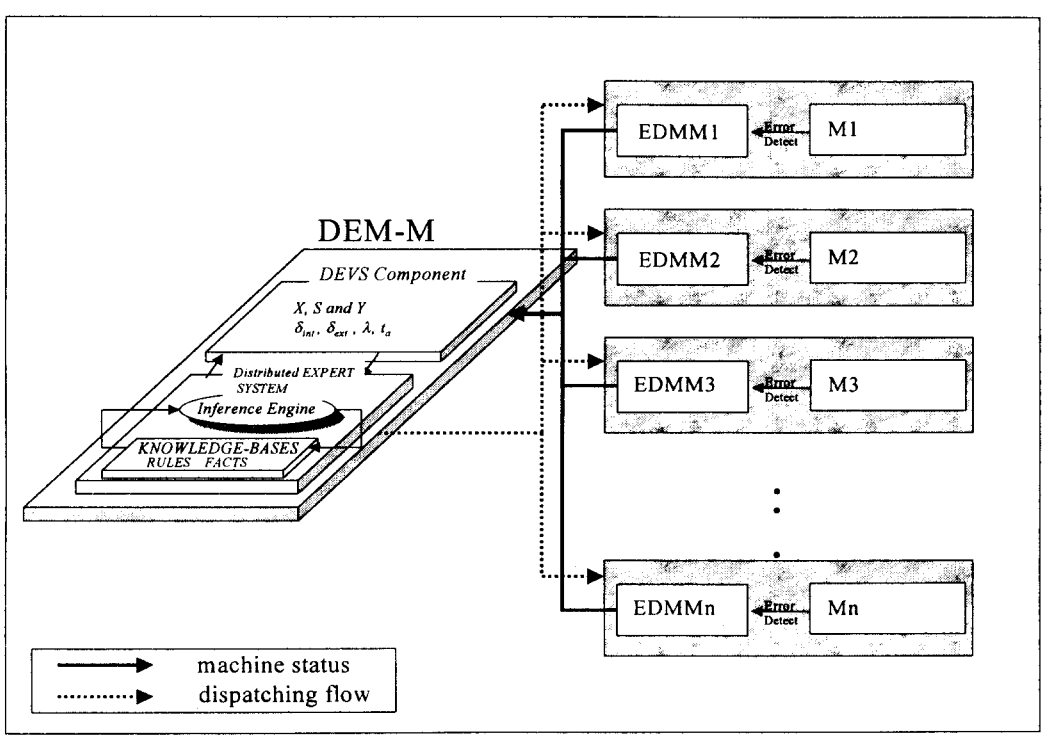
Root-entity인 GPM-DES-EF는 Experimental Frame(EF)와 GPM-DES로 구성되어 있다. GPM-DES는 <그림 7>의 모델의 구조를 의미한다. GPM-DES entity는 M-AREA, WG-AREA, 그리고 GP-AREA, 3의 구성 요소로 이루어 지고, 각각의 구성 요소들은 하나의 DEM(Dispatching Expert Model), 프로세스 모델과 같은 개수의 EDM(Error Detect Model)과 각 공정의 프로세스 모델로 구성 된다. 특히, 각각의 프로세스 모델에는 성능을 점검하기 위한 M#-Transd, WG#-Transd, GP-Transd모델을 연결하였다.

<그림 9>는 머신 공정에서의 M#, EDMM#, 그리고 DEM-M 모델들의 상호 작용을 보여 주고 있다. EDMM# 모델은 M# 모델 하나에 쌍으로 연결되어서 M# 모델의 상태를 점검하고 현재 상태를 DEM-M 모델에게 보내 준다. DEM-M 모델은 여기서 받은 각 머신의 현재 상황과 작업의 내용을 보고 최적의 머신에 작업을 할당한다[14]. <그림 9>의 상호 관계는 가공팀 공정, 도금 공정에도 동일하게 적용된다.

DEM은 Interruptible atomic-expert model로 구현되었다. 이 모델 안에는 작업을 프로세스 모델에



<그림 8> 그래이팅 프로세스 스케줄링시스템과 EF모델의 System Entity Structure(SES) representation.



<그림 9> DEM-M과 EDM사이의 상호 작용

할당하는데 필요한 규칙들이 rule-base에 저장되어 있다. DEM 모델은 rule-base의 규칙과 EDM 모델에게서 받은 프로세스 모델의 현재 상태와 작업 정보에 기초해서 최적의 모델에게 작업을 할당한다. <그림 10>, <그림 11>은 EDM과 DEM모델의 timing diagram의 한 예를 보여주고 있다.

DEM 모델이 작업 처리를 위해 추론 중에 유효하지 않은 사실이 존재하는 것을 발견하면 추론은 인터럽트 된다.

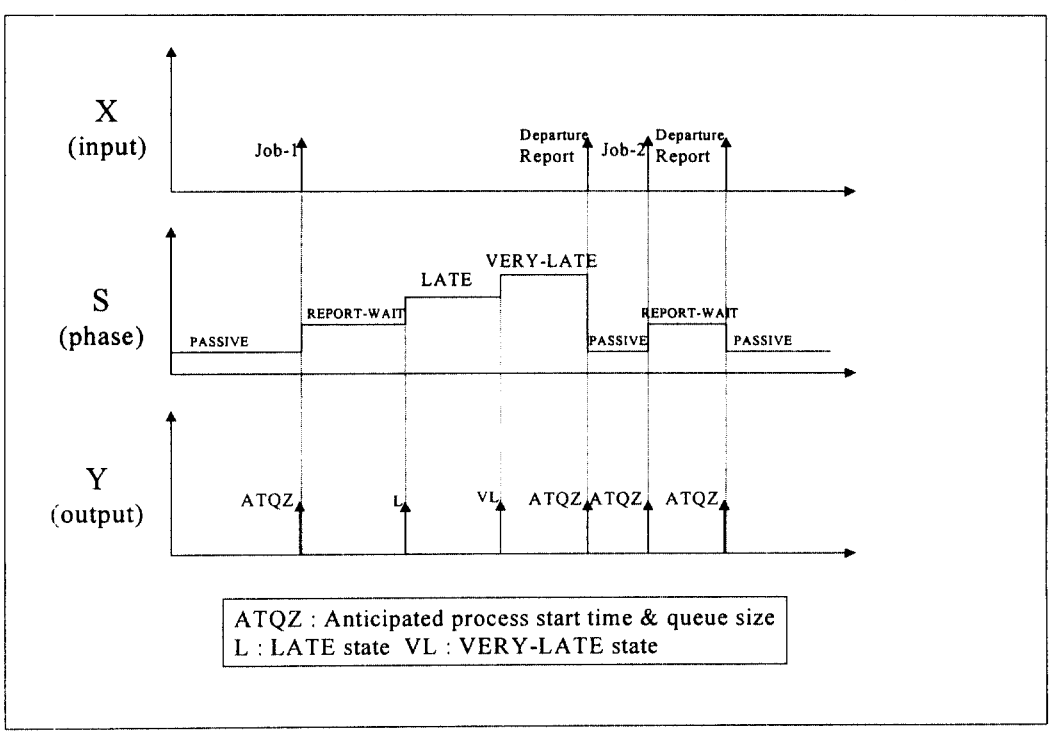
이런 유효하지 않은 사실의 유무는 EDM 모델이 자신이 갖고 있는 정보에 근거해 프로세스 모델의 상태를 점검하고, 예상대로 동작하지 않

을 경우 오류가 생겼음을 DEM 모델에게 알려준다. 본 적용에서 정의한 오류 상태는 프로세스 모델에 입력된 작업이 예상 종료시간을 지나서도 종료되지 못하고 있는 경우이다.

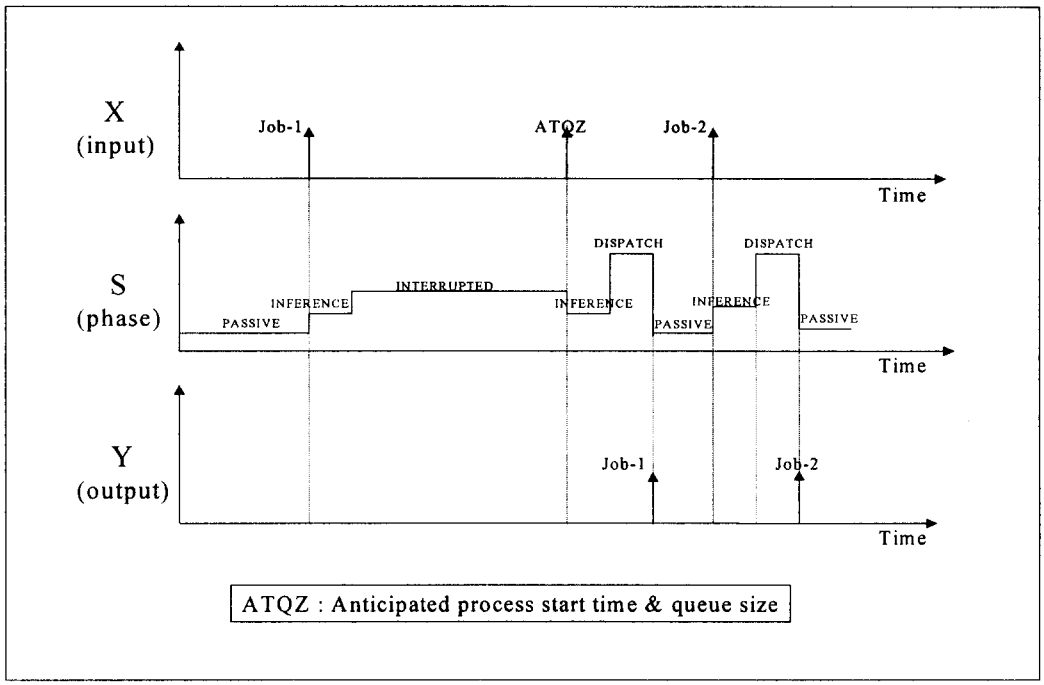
<그림 10>에서 REPORT-WAIT상태는 프로세스 모델에게서 작업이 종료 되었다는 신호를 얻기위한 상태이다. EDM 모델은 이 상태에 머무르면서 프로세스 모델의 출력 포트를 감지한다. 이 상태에서 프로세스 모델의 출력이 발생하면 EDM 모델은 자신이 담당하는 프로세스 모델의 큐에 있는 작업량과 모델의 성능에 기초해서 예상 작업 시작 시간과 큐의 크기 (ATQZ : Anticipated process start Time and Queue siZe)를 보내 준다. <그림 10>을 보면 예상 작업 시작 시간과 큐의 크기(ATQZ)는 프로세스 모델에 새 작업이 도착한 시점과 작업 이 끝났을 시점에서 보내지는 것을 알 수 있다.

예상보다 작업 종료 시간이 늦어질 경우 EDM은 LATE상태로 상태전이가 되면서 LATE상태를 DEM모델에게 전달한다.

DEM모델은 할당할 작업을 받았을 때 작업 할당의 대상이 될 수 있는 프로세스 모델들의 상태에 기초해 추론을 하다가 LATE상태가 발견되면 Interrupted상태로 상태 전이가 이루어진다.



<그림 10> The Timing Diagram of EDM (Error Detect Model)



<그림 11> The Timing Diagram of DEM(Dispatching Expert Model)

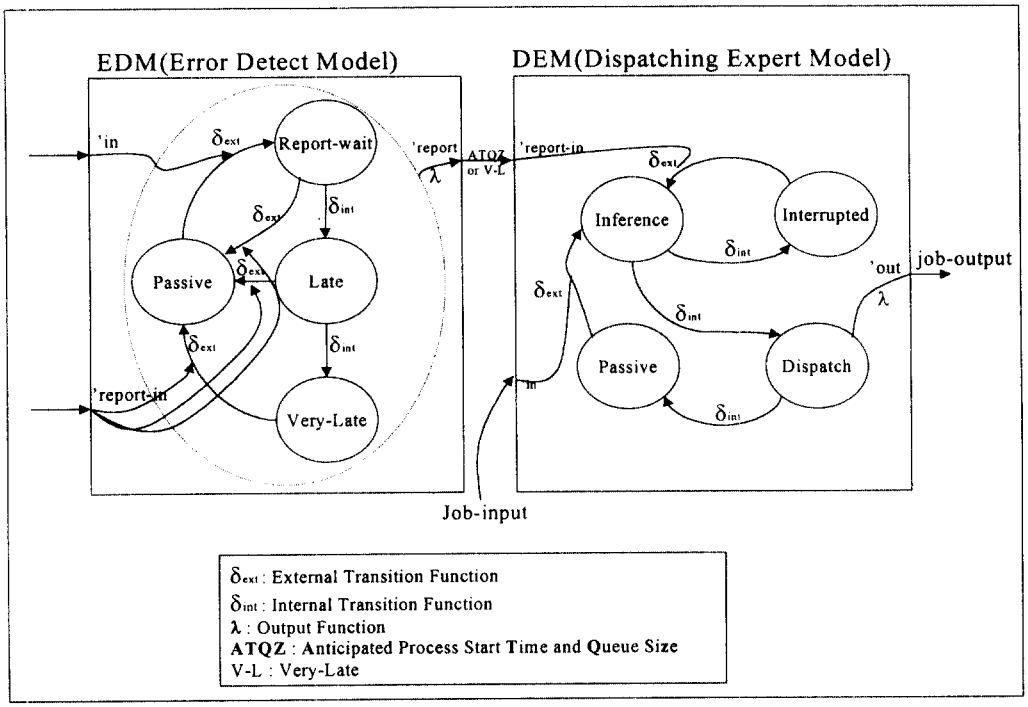
EDM은 LATE상태에서 다시 프로세스 모델의 상태를 점검하고 있다가 일정 시간 후에도 계속 출력을 내어 보내지 않으면 EDM은 VERY-LATE상태로 상태 전이가 일어난다.

VERY-LATE상태는 프로세스 모델이 일종의 이상 상태에 있는 것을 의미한다. DEM은 추론 도중 VERY-LATE상태의 모델을 발견하면 그 모델은 작업 할당대상에서 제외시키고 추론을 진행한다. 또한, 작업 할당 대상이 되는 모든 머신이나 가공팀이 VERY-LATE 상태가 되면 DEM은 Interrupted 상태로 들어간다. 이 상태에서는 EDM으로부터 받은 ATQZ 메시지를 resume signal로 사용한다. 이와 같은 메커니즘으로 프로세스 모델은 제어된다.

<그림 12>는 DEM과 EDM의 상태 전이를 보여주고 있다. EDM은 REPORT-WAIT상태에서 LATE, LATE 상태에서 VERY-LATE, 그리고, 모든 상태에서 ATQZ 메시지를 전달 한다. 이 메시지들은 interruptible atomic-expert model안의 fact object의 상태를 변경해 준다. 하나의 작업이 DEM에 도착하게 되면, 전문가 시스템은 작업의 attribute-list의 내용과 fact object의 내용을 가지고 추론을 하게 된다. 만일 LATE상태가 존재하면 DEM은 인터럽트 된다. 인터럽트된 DEM의 'report-in 포트를 통해 ATQZ나 VERY-LATE상태가 보고되면 이 입력을 인해서 외부 전이 함수가 실행 되면서 다시 추론이 재개된다. 여기서 ATQZ와 VERY-LATE메시지는 인터럽트된 DEM의 resume signal로 사용된다. DEM은 Action 상태가 종료되면서, 작업을 자신의 'out 포트를 통해서 작업 할당을 한다.

5. Simulation 결과

이 장에서는 interruptible atomic-expert model의 기여도를 기존의 atomic-expert model과 비교하여서 살펴보고자 한다. 4장에서 디자인 및 구현된 시뮬레이션 모델과 비교하기 위해서 시간 의존 변수(Time Dependent Variable)의 유효성을 검사하지 않고, 프로세스의 큐의 크기에 기초한 예상 작업 시간을 가지고 추론하는 atomic-expert model을 구현

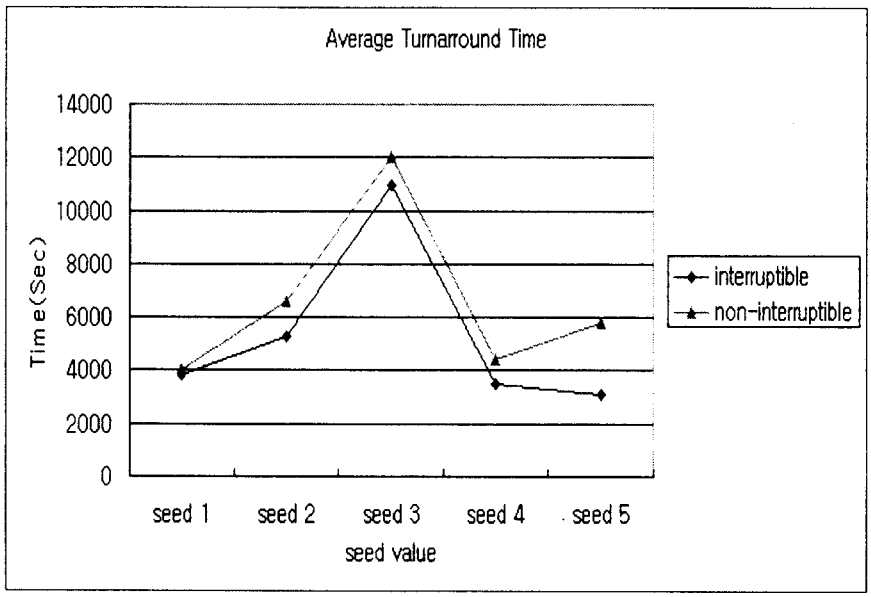


<그림 12> The state transition diagram of EDM and DEM

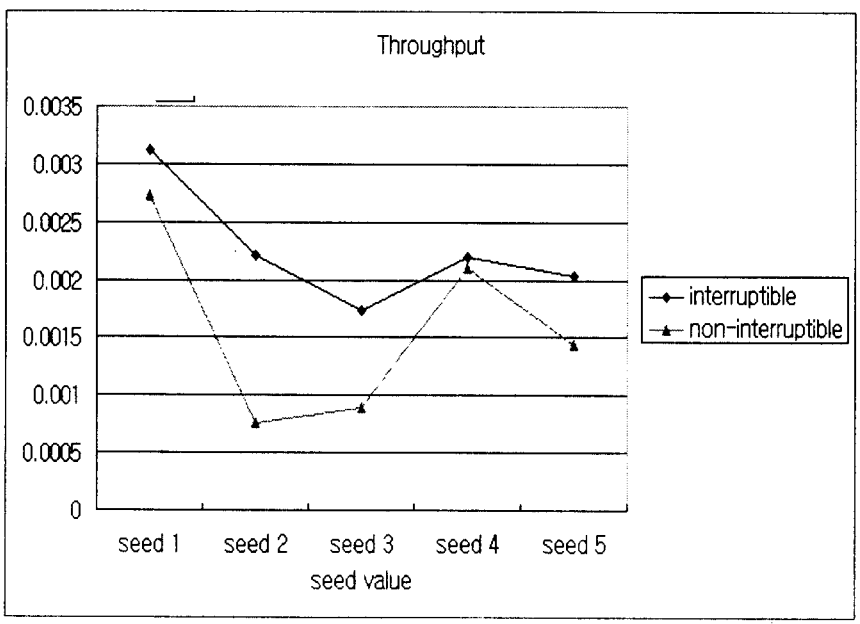
하였다. 즉, 추론 당시에 이용할 수 있는 사실을 가지고 추론하기 때문에 인터럽트가 없이 추론을 할 수 있도록 했다. 반면에 4장에서의 구현 환경에서는 추론 을 위한 값이 준비될 때 까지 인터럽트 된다. 시뮬레이션에서는 평균 400의 지수 분포로 interarrival-time을 발시켜 5개의 seed값을 가지고 테스트하였다. 작업의 크기와 머신, 가공 팀, 그리고 도금 공정 각각의 작업 처리 시간은 정규 분포를 사용하였다. 이 정보들은 그레이팅 프로세스 정보에 기초한 것이다. <그림 13>, <그림 14>은 두 가지 모델들의 average turnaround time 과 throughput을 보여주고 있다. 각각에서 interruptible-atomic expert model이 성능 면에서 우위에 있는 것을 볼 수 있다. 이것은 작업 할당에 사용되는 정보의 유효성 여부에 의해서 생긴 차이이다. 즉, 인터럽트가 가능한 경우에서는 LATE상태를 만나면 그 상태에서 일정 시간 추론을 멈추고 있어야 한다는 단점이 있지만 작업 할당 대상 의 상태 정보에 근거해 시간 의존 변수의 유효성을 점검하여 정확한 추론을 한다. 그리고, non-interruptible의 경우는 비록 LATE상태에 대해 반응할 수 없기 때문에 interruptible에 비해서 그 시간을 절약하게 되지만 결국 유효하지 못한 정보에 기초한 추론으로 전체적인 성능은 떨어진 것을 볼 수 있다.

<그림 15>을 보면 각 seed값의 변경에 따른 interrupt rate를 볼 수 있다. Interrupt rate는 하나의 작업이 interruptible atomic-expert model안에서 인터럽트가 발생할 확률을 의미한다. <그림 15>과 <그림 14>를 비교해 볼 때 interrupt rate와 두 모델의 throughput 차 사이에 양의 상관관계가 있는 것을 볼 수 있다. 인터럽트가 발생할 확률이 높다는 것은 머신, 가공 팀, 도금 공정의 오 동작이 있을 확률이 높아진다는 것인데, 이런 상황에서는 interruptible atomic-expert model의 성능이 더 잘 발휘된다는 것을 알 수 있다.

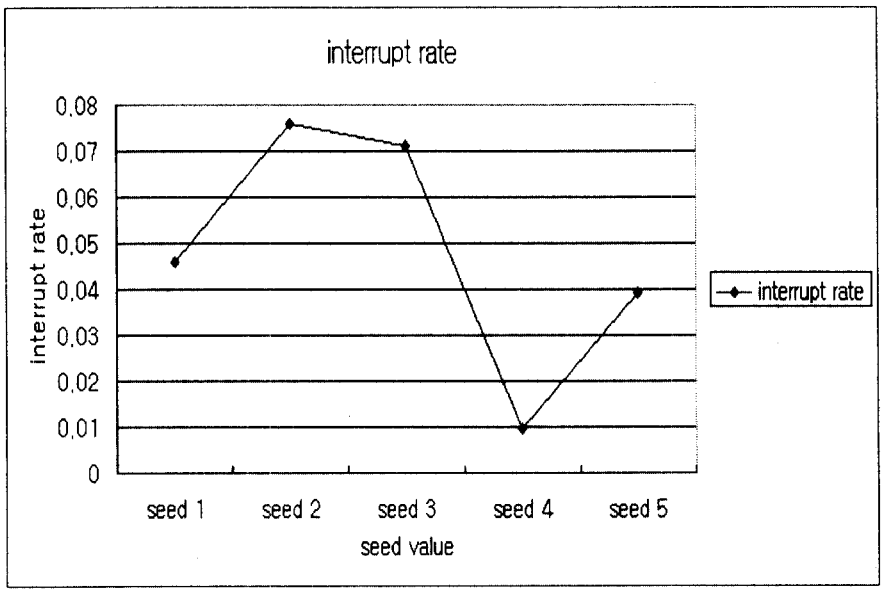
이러한 성능의 차이를 가져오는 이유는 정확한 정보를 가지고 추론하기 위한 인터럽트 기능의 유무에 따라서 나타난 것이다.



<그림 13> Average Turnaround Time of each model



<그림 14> Throughput of each model



<그림 15> The interrupt rate of interruptible atomic-expert model

6. 결론

시뮬레이션 모델의 상태 변수 값은 시간이 경과하면서 진행되는 시뮬레이션의 실행에 의해서 결정된다. 시뮬레이션 모델에 전문가 시스템을 삽입하고자 할 때, 시간 기반의 변수들의 유효 여부에 기초해서 정확한 추론을 하는 것이 요구된다. 이를 위해서 본 논문에서는 기존의 atomic-expert model에 인터럽트 기능을 추가하기 위한 interruptible atomic -expert model을 디자인하였다. 또한, 이런 디자인을 그레이팅 프로세스 스케줄링 시스템을 위한 DEM (Dispatching Expert Model)에 적용하였다. 그리고, 시간 의존 변수의 유효성을 점검하기 위해서 사건 기반 제어 모델을 사용하였다. 또한, non -interruptible 한 경우와의 비교를 통해서 interruptible atomic -expert model의 기여도를 제시하였다.

향후 연구 과제로는 첫째, 현재는 하나의 시간 의존 변수를 사용하는 간단한 예에 대하여 유효성을 검증하였으나, 추후로는 여러 개의 시간 의존 변수를 사용하는 경우에 대한 연구가 필요하고 그 효과는 더욱 클 것으로 예상된다. 둘째는, 전문가 시스템의 규칙과 사실이 많아 추론 시간이 길어져 문제가 되는 경우 인터럽터블한 전문가 시스템이 어떻게 효과적으로 적용되는 가에 대한 연구가 필요하다.

참고문헌

[1] Tsatsoulis, C., "A Review of Artificial Intelligence In Simulation," SIGART Bulletin, Vol. 2, No.1, pp. 115-121, 1991.

[2] Arons, H. De Swann, "Expert Systems in the Simulation Domain," Mathematics and Computers in Simulation, Vol. XXV, 1983

[3] O'Keefe, R., "Simulation and Expert Systems-A Taxonomy and Some Examples," Simulation, Vol. 46, pp. 10-16, 1986

[4] Bernard P. Zeigler, Object-Oriented Simulation with Hierarchical, Modular Models. San Diego, CA, USA: Academic Press, 1990.

[5] T. H. Cho, Bernard P. Zeigler, "Simulation of Intelligent Hierarchical Flexible Manufacturing: Batch Job Routing in Operation Overlapping,"

IEEE Trans. Syst. Man, Cybern. A, Vol. 27, pp. 116-126, Jan. 1997.

[6] T. H. Cho, "A Hierarchical, Modular Simulation Environment for Flexible Manufacturing System Modeling," Ph. D. dissertation, Univ. Of Arizona, Tucson, 1993.

[7] B. P. Zeigler, T. H. Cho, and J. W. Rozenblit, "A Knowledge-based Environment for Hierarchical Modeling of Flexible Manufacturing System," IEEE Trans. Syst. Man, Cybern. A, Vol. 26, pp. 81-90, Jan. 1996.

[8] B. P. Zeigler, Multifacetted Modeling and Discrete Event Simulation. Orlando, FL: Academic, 1984.

[9] B. P. Zeigler, Theory of Modeling and Simulation, John Wiley, NY, USA, 1976, reissued by Krieger, Malabar, FL, USA, 1985.

[10] Paul A. Fishwick, "Simulation Model Design and Execution : Building Digital Worlds," Prentice-Hall, 1995.

[11] Averill M. Law and W. David Kelton, "Simulation Modeling & Analysis," McGraw-Hill, 1991.

[12] Rich, Knight, "Artificial Intelligence : The Second Edition," McGrawHill, 1991, chap. 6, chap. 7, chap. 8.

[13] J. Giarratano, G. Riley, "Expert Systems - Principles and Programming : Second Edition," PWS Publishing Company, 1994, chap. 1, chap. 2, chap. 3, chap. 4, chap. 5, chap. 12.

[14] R. G. Askin and C. R. Standridge, Modeling and Analysis of Manufacturing System. New York: Wiley, 1993.

[15] P. H. Winston, Artificial Intelligence : The Third Edition, Addison Wesley, 1992, chap. 7.

[16] F. Zahedi, "Intelligent Systems for Business - Expert Systems with Neural Networks," Wadsworth Publishing Company, 1993, chap. 2, chap. 3, chap. 4, chap. 8, chap. 11.

[17] 김형종, "전문가 시스템을 이용한 반도체 생산 라인에서의 이탈처리 추적 시스템 구축," 성균관대학교 석사학위 청구 논문, 1997.

[18] 조대호, "지식 표현 기법을 이용한 모델 구조의 표현과 구성 : 단편구조 유연생산 시스템 예," 한국시뮬레이션학회 논문지, Vol. 4, No. 1, June 1995.

저자소개

조대호

1983년 성균관대학교 전자공학과 학사.
1987년 Univ. of Alabama 전자공학 석사.
1993년 Univ. of Arizona 컴퓨터 공학 박사.
1993년~1995년 경남대학교 전자계산학과 전임강사.
1995년~현재 성균관대학교 전기 전자 컴퓨터 공학부 조교수.
관심 분야 : 시뮬레이션 모델링 방법론, 지능형 시스템, 공장 자동화, 인공지능

김형종

1996년 성균관대학교 정보공학과 학사.
1998년 성균관대학교 정보공학과 석사.
1998년~현재 성균관대학교 전기 전자 컴퓨터 공학부 박사과정.
관심 분야 : 컴퓨터 시뮬레이션, 전문가 시스템