전문가 시스템과 데이터 베이스를 사용한 지식 기반 시뮬레이션 환경 구축

Construction of Knowledge-based Simulation Environment Using Expert System and Database

김형종*, 이주용**, 조대호*
Hyung-Jong Kim, Ju-Yong Lee, and Tae-Ho Cho

Abstract

As the application domains of rule-based systems become larger and more complicated, the integration of rule-based systems within the database systems has become the topic of many research works. This paper suggests a simulation modeling using expert system and database. The integration methods employed in this research are as follows. First, we defined new states and state transition functions to interrelate simulation model and expert system. Second, we designed and implemented FCL(Fact Class Library) as a interface of expert system and database. FCL has facilities of filtering data from database, and assigning a meaning to the filtered data. Also, FCL detects the violation of the integrity rules in database, as the result of inference is reflected. Some implementation problems are pointed out and the methods to solve these problems are discussed in this paper. We developed a simulation model of the grating production line and executed it to validate the functions of the proposed method.

Keywords : Simulation Modeling, Expert System, Knowledge Base, Fact Class Library.

* 성균관대학교 전기전자 및 컴퓨터공학부
** 한국통신 프리텔 수도권 네트워크 본부

1. 서 론

시뮬레이션 모델과 전문가 시스템의 복합 연구는 그 동안 많은 학자들에 의해서 진행되어 왔다[1]. 시뮬레이션 모델은 전문가 시스템과의 인터페이스를 통해서 지식 처리 기능을 소유할 수 있다. 또한, 전문가 시스템은 시뮬레이션 모델을 통해서 시간 기반의 정보를 추론할 수 있게된다[2][3][4]. 최근 시뮬레이션 모델을 실 시스템의 제어를 위해서 사용하게되는 추세에 따라서 시뮬레이션 모델 내부의 전문가 시스템이 실 시스템의 DBMS 사용의 필요성이 제기되고 있다. DBMS는 많은 양의 자료를 일치성과 무결성을 유지하면서 효율적으로 관리 할 수 있고, 전문가 시스템은 추론기능과 유연성 있는 지식의 표현력을 갖는다. 이런 시스템들의 특징이 상호 보완을 이룰 때 지식처리 능력의 향상을 도모할 수 있다[5][6]. 대부분의 실 시스템의 DBMS가 방대한 양의 정보를 관리하기 때문에 전문가 시스템이 이 데이터들을 사실(Fact)로 사용하기 위해서는 이를 위한 인터페이스가 요구된다.

본 논문에서는 전문가 시스템과 DBMS와의 결합에서 생길 수 있는 문제점들을 해결하는 방법으로 FCL(Fact Class Library)을 제시하였다. 이를 통해 전문가 시스템과 DBMS의 인터페이스를 위한 기반 기술을 제공하고자 한다. 또한, FCL을 사용하여 GPSS(Grating Process Scheduling System) 시뮬레이션 모델을 디자인 및 구현하여 FCL의 기능적 특성을 검증하였다. 그리고, error율에 따른 시뮬레이션 결과의 차이를 통해서 FCL을 사용하는 시뮬레이션 모델의 실행 결과를 제시하였다.

2. 배경 이론

전문가 시스템은 적용 영역의 전문가들이 가지고 있는 전문 지식을 지식베이스로 구축하여 저장관리함으로써 컴퓨터가 전문가의 역할을 수행케 하는 시스템이다[11]. 전문가 시스템은 추론엔진과 지식베이스로 구성된다. 추론 엔진(Inference Engine)은 전문가 시스템의 전체적인 운영을 담당하는 기관으로 전문가들의 지식을 이용하는 방법을 정의하고 있다. 지식베이스는 전문가의 지식을 담고 있는 지식의 저장 장소로 사실과 규칙으로 구성된다[8]. 사실은 지식 베이스의 지식을 적용하고자 하는 현재의 상태이며 규칙은 문제를 풀어나가기 위한 장기 정보(long-term information)로서 전문가의 전문적 지식을 통하여 새로운 사실이나 가정을 만들어 내는데 필요한 정보이다.

데이터베이스는 서로 연관이 있는 데이터들의 모임이며, 특정 의미를 가지는 데이터의 모임이다. 데이터베이스 관리의 주요 이슈(issue)로는 동시 사용에 의한 문제 해결, 잘못된 데이터의 방지, 데이터의 중복성 제거, 데이터의 보안 문제 등을 들 수 있다[5]. 특히 데이터베이스를 관리하는 관점에서 무결성 규정은 데이터베이스내의 데이터가 일관성의 유지를 위한 것으로 데이터베이스를 유효하게 유지하여 의미적 에러를 방지하기 위한 제약조건이다.

전문가 시스템과 DBMS의 연결에 대한 연구는 많은 사람들에 의해서 이루어졌다. 전문가 시스템과 DBMS의 연결 관계는 강한 연결(tightly coupling)과 약한 연결(loose coupling)로 분류 될 수 있다. 강한 연결의 경우, DBMS가 규칙의 관리와 추론에 관여하도록 하여 두 시스템이 하나가 된 경우를 말한다. 약한 연결은 두 시스템이 서로 독립적인 시스템으로 존재하고, 두 시스템을 연결해 주는 특정 프로토콜이 있는 경우를 말한다. 특히, 약한 연결의 경우 DBMS는 전문가 시스템과의 관계에서 수동적으로 동작하며, 전문가 시스템에게 자신이 관리하는 데이터에 접근할 수 있는 인터페이스를 제공해 주게 된다. DBMS는 SQL (Structured Query Language)을 통해 관리하는 데이터에 접근할 수 있도록 허용한다. [7]에서는 전문가 시스템과 DBMS사이의 인터페이스를 설계함에 있어서, 효율적인 질의의 설계 방법을 소개하고 있다. [8]에서는 전문가 시스템과 데이터베이스의 인터페이스를 위해 전문가 시스템 부분에는 KBSI(Knowledge Based System Interface)를 두고 데이터베이스 부분에는 KBDBI(Knowledge

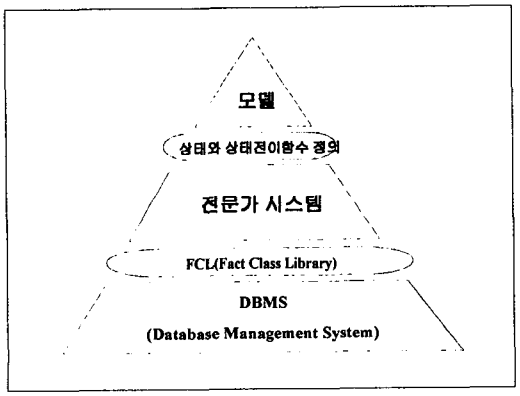
Based Database Interface)를 두었다. [9]에서 CMOS 전문가 시스템은 반도체 웨이퍼에 데이터 정보를 저장하기 위한 프레임 구조를 두었다. 여기서 프레임은 전문가 시스템의 규칙이 사용하기에 알맞은 형태의 영역 객체(domain object)의 표현 형태이다. 프레임 기반의 전문가 시스템을 구성할 때의 장점은 [10]에서 잘 소개하고 있다.

3. 모델 설계 및 적용

본 논문에서는 FCL(Fact Class Library)을 제안하여 시뮬레이션 모델링에서 전문가 시스템과 DBMS의 인터페이스문제를 해결하고자 한다. 본 장에서는 FCL의 디자인 및 그레이팅 생산 공정의 시뮬레이션 모델링을 소개한다.

3.1 전문가 시뮬레이션 모델의 정보 처리 계층

<그림 1>는 전문가 시뮬레이션 모델의 정보 처리 계층도를 나타낸 그림이다. 전문가 시스템과 DBMS 사이, 시뮬레이션 모델과 전문가 시스템 사이에 각각 인터페이스를 위한 모듈이 존재한다. 전문가 시스템과 DBMS간의 결합은 논문에서 제시하는 FCL이 인터페이스 역할을 하며, 시뮬레이션 모델과 전문가 시스템간의 연계는 모델 구축에서의 상태와 상태전이함수를 정의함으로 연계가 된다.



<그림 1> 정보 처리 계층도

3.2 FCL기반의 모델 아키텍쳐

<그림 2>는 FCL을 사용하는 시뮬레이션 모델의 구조이다. 전문가 시스템과 시뮬레이션 모델 사이에는 추론에 사용되는 정보들을 서로 교환할 수 하기 위해 IT(Inference Table) 클래스를 정의하고 있다. 이 클래스에는 현재 처리중인 작업에 대한 정보, 지금까지 추론된 규칙 정보, 추론 기관의 상태 등이 저장되어 있다. 이러한 값들을 서로 교환하면서 시뮬레이션이 진행된다. <그림 2>를 통해 볼 때 데이터베이스는 시뮬레이션 모델의 외부에 존재하는 것을 볼 수 있다. 또한, FCL은 시뮬레이션 모델의 내부에 존재하며, 전문가 시스템이 추론을 하기 위해 필요한 사실은 데이터베이스에서 FCL을 통해 얻는다. 그리고 전문가 시스템의 추론 결과에 의해 생성, 삭제, 또는 갱신된 데이터는 FCL을 통해 데이터베이스에 반영된다. 곧, FCL은 전문가 시스템을 구성요소로 갖는 시뮬레이션 모델과 데이터베이스 사이의 인터페이스 역할을 하게 된다.

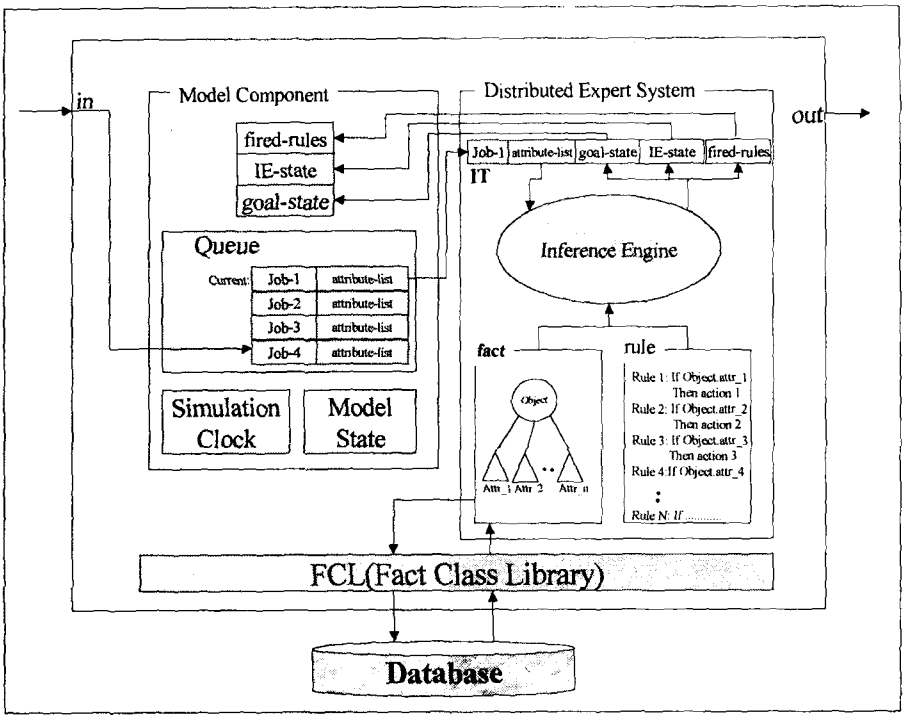
3.3 인터페이스 설계

가. 시뮬레이션 모델과 전문가시스템

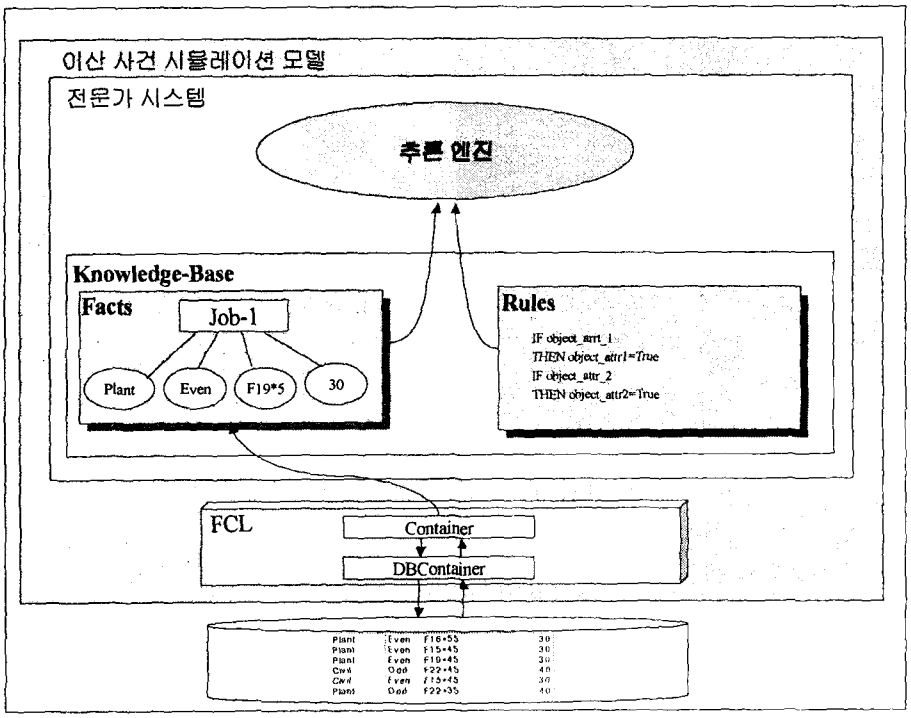
이산사건 시뮬레이션모델과 전문가 시스템간의 연계는 모델에 전문가 시스템을 위한 새로운 상태와 상태 전이 함수를 정의함으로써 이루어진다. 전문가 시스템의 추론 엔진의 상태와 현재까지 추론된 규칙의 집합 등의 정보는 전문가 시스템의 상태를 대표하는 값이 될 수 있다. 이러한 값들에 기초해서 시뮬레이션 모델에는 적절한 상태 값이 정의되어야 한다. 그리고, 해당 상태로 전이하기 위한 상태 전이 함수의 정의가 요구된다. 이러한 상태와 상태전의 함수의 정의는 시뮬레이션 모델과 전문가 시스템의 자연스러운 연결관계를 맺어 준다.

나. 전문가 시스템과 데이터베이스

전문가 시스템과 DBMS간의 결합에서 생길 수 있는 문제점은 다음과 같다. 첫째, 데이터베이



<그림 2> FCL기반의 시뮬레이션 모델



<그림 3> 전체 추론 과정

스내의 테이블의 각 필드(field)들이 전문가 시스템의 지식베이스의 사실(fact)에 반영될 때, 각 시스템의 지식 표현 방법이 다르기 때문에 지식의 타입(type)에 따른 클래스의 속성들과 올바른 사상이 되어야 한다. 둘째, 데이터베이스에서 추출된 데이터 중 추론에 필요한 데이터가 부재(Null)이거나, 그 데이터가 적합한지를 파악해야 한다. 전문가 시스템과 데이터베이스의 결합을 위해서는 우선, 전문가 시스템의 지식 표현 방법과 데이터베이스의 자료 표현 방법이 서로 다르기 때문에 시스템의 특성에 맞게 표현된 자료를 서로 사상하여 전환하는 작업이 이루어져야 한다. 두 구조의 대응 관계는 <표 1>과 같다[11].

<표 1> 데이터베이스와 사실의 사상

데이터베이스		사실 베이스
테이블(Table)	↔	사실클래스(Fact Class)
레코드(Record)	↔	사실(Fact)
필드(field)	↔	속성(Attribute)

데이터베이스에서 테이블은 사실베이스의 사실 클래스에 사상된다. 그리고 데이터베이스 의 테이블을 구성하고 있는 행, 열, 셀 들은 각각 지식베이스에 객체, 속성, 객체 슬롯(object slot)으로 사상된다.

<표 2> 데이터베이스의 테이블과 전문가 시스템의 자료 표현 방법

데이터베이스 테이블		지식 베이스
행 (Row)	↔	사실 객체 (Object)
열 (Column)	↔	속성 (Property)
셀 (Cell)	↔	객체 슬롯(Object Slot)

또한 <표 2>와 같이 지식베이스에 있는 사실 객체, 속성, 객체 슬롯은 데이터베이스의 테이블에 행, 열, 셀(cell)로 사상시킬 수 있다.

다. FCL(Fact Class Library) 디자인

FCL은 데이터베이스에서 추론에 필요한 데이터를 추출하여 이 데이터를 전문가 시스템에서 추론하기 위한 사실(Fact)로 사상하기 위한 것이다. 이를 통해 데이터 레벨(level) 정보를 지식 레벨 정보로 한 단계 높이는 처리 과정이 수행된다. FCL은 다음 4가지의 기능을 수행한다.

a. 추론에 필요한 정보를 데이터베이스에서 추출: 추론을 위해 필요한 정보를 데이터베이스에서 가져오기 위한 작업을 한다.
b. 추론에 필요한 데이터에 대한 의미부여: 데이터베이스에서 추출해온 정보를 추론을 위한 사실로 변환하는 작업을 한다. 데이터베이스의 제한된 데이터 타입으로 인해 표현할 수 없었던 정보를 추론을 위해 정의된 데이터의 형태로 변환한다.
c. 추론에 필요한 정보를 얻기 위한 저장 공간: 추론에 사용되는 정보를 얻고, 추론 결과를 저장하기 위한 저장 공간이 된다.
d. 추론 결과 저장 시 데이터베이스 객체의 무결성, 일치성, 식별성의 유지: 데이터베이스에 추론 결과를 저장할 때 무결성, 일치성, 식별성을 유지 시켜주는 기능을 한다.

FCL은 <그림 3>과 같이 Container Class와 DBContainer Class로 구성된다. DBContainer는 데이터베이스 입출력에 관련된 모든 프리미티브(primitive) 함수들을 정의해 놓았고 질의(query)에 의해 필터링된 데이터를 가져오는 기능을 갖는다. 또한, 추론 결과를 저장할 경우 데이터베이스 객체들의 제약조건을 맞추어주는 기능을 갖는다. Container에는 전문가 시스템의 추론에 필요한 지식을 얻는 지역 메모리처럼 사용하기 위한 저장 버퍼가 존재하며, 동시에 데이터베이스의 각 데이터의 의미를 해석한다. 또한, 전문가 시스템의 사실로 사용될 수 있도록 데이터베

<표 3> DBSet 클래스의 주요 멤버 함수들

함수 명	기능
Retriv_ManufactSpec();	데이터베이스로부터 ManufactSpec 테이블의 정보를 읽어오기 위한 함수. 입력으로는 SQL의 where절과 sort절을 주고, 출력은 조건을 만족하는 정보를 버퍼로 가져오는 작업을 한다.
Update_ManufactSpec();	갱신된 ManufactSpec 테이블의 정보를 데이터베이스에 반영하기 위한 함수이다. 정보의 추가, 삭제, 갱신이 끝났을 때 호출하여 데이터베이스의 테이블을 갱신한다.
Add_ManufactSpec();	레코드 하나를 추가
Del_ManufactSpec();	레코드하나를 삭제
Write_ManufactSpec();	특정 레코드를 갱신
Check_Integrity();	무결성 제약 조건에 어긋나는 경우를 점검하는 기능을 수행하는 멤버함수 이다.

이스의 정보를 변환하는 기능을 갖는다. 예를 들어, 특정 머신의 정보로 추론을 해야 할 경우 DBContainer는 주어진 조건에 맞게 데이터베이스에게 질의를 하여 가져오는 기능을 수행하고, 가져온 데이터 중에 베어링바(bearingbar)의 타입이 'f19*3'이라는 데이터는 각 토큰 별로 'f'는 flatbar, 높이는 '19' 그리고 두께는 '3'의 형태로 변환하는 일을 Container가 한다. 이렇게 변환된 사실들은 Container에 저장되어 추론에 사용된다. DBContainer Class는 MFC(Microsoft Foundation Class)의 CRecordSet으로부터 상속받아 만들어진 클래스의 인스턴스를 멤버 변수로 갖는다. CRecordSet 클래스가 갖는 데이터베이스 관련 함수들을 모두 상속받아 갖고 있는 멤버 변수를 사용하기 때문에 데이터베이스에 대한 접근이 용이하다. 또한, CRecordSet 클래스의 하위 클래스를 정의할 때는 각각의 테이블이나 뷰에 하나의 클래스를 정의한다. 즉, DBContainer내부에는 전체 시뮬레이션 모델에서 사용하는 테이블과 뷰의 개수만큼 해당 클래스의 인스턴스가 존재하게 된다. 이러한 클래스의 이름을 DBSet 클래스라고 정의하였다. <표 3>은 Table ManufactSpec과 사상되는 DBSet 클래스의 주요 멤버 함수를 보여 주고 있다.

<표 3>의 Check_Integrity() 멤버함수는 하나의 테이블에서의 무결성 제약조건의 위배를 검사하는 함수이다. 여러 개의 테이블에서의 무결성을 점검해 주기 위해서 DBContainer 클래스 내에는 Check_InterTIntegrity() 멤버함수가 존재한다. Container 클래스는 DBContainer 클래스에 있는 레코드들을 자신의 버퍼로 옮겨와 이들의 의미를 분석한다. 이러한 버퍼의 기능을 담당하는 클래스로 DBContainer 내의 DBSet 클래스와 사상될 수 있는 클래스를 만들고 이 클래스의 인스턴스를 Container 클래스의 멤버 변수로 정의하였다. DBSet과 사상되는 클래스를 FD(Fact Data) 클래스라고 정의하였다. FD 클래스는 DBSet 클래스의 각 레코드들의 값을 옮겨 넣을 수 있는 자료구조의 리스트를 멤버 변수로 갖는다. 이 자료 구조는 데이터베이스에서 표현할 수 없는 자료형이나 의미적 분석이 요구되는 경우를 고려하여 디자인 되어야 한다. 또한, FD 클래스의 내부에는 DBSet의 데이터들을 가져와 전문가 시스템이 사용할 수 있는 형태로 변환시키기 위한 멤버 함수

<표 4> FD 클래스의 주요 멤버 함수

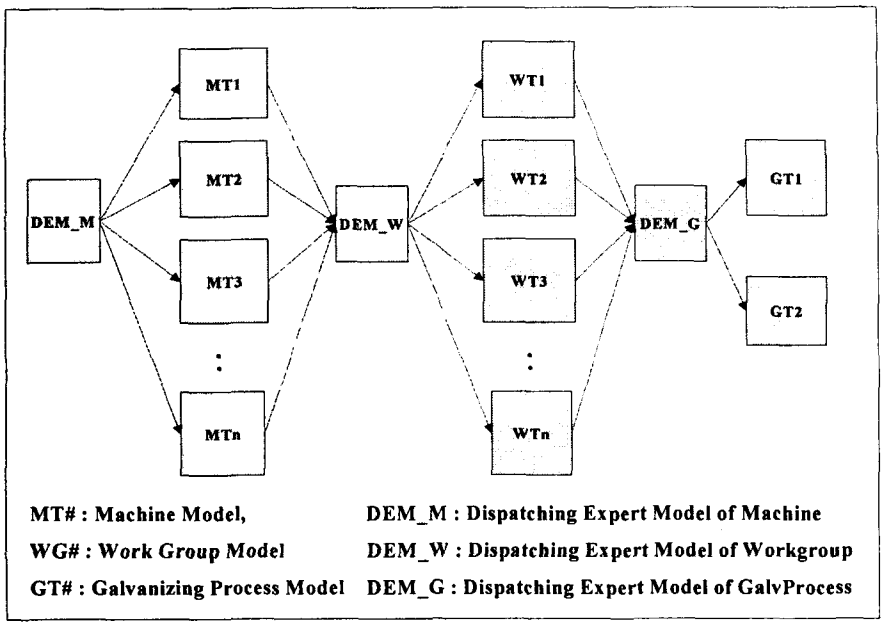
함수 명	기능
Import_ManufactSpec();	DBSet으로부터 가져온 정보를 FD 클래스 인스턴스의 리스트에 읽어들이기 위한 멤버 함수이다. 이 함수가 실행되면 리스트에는 DBSet의 Retieve된 정보를 정의한 자료구조에 맞게 변환하여 리스트에 저장한다.
Export_ManufactSpec();	전문가 시스템에 의해서 갱신된 데이터의 내용을 DBSet에 전달하여 주기 위한 멤버함수이다. 이 함수가 실행되면 FD 내부의 리스트 정보를 DBSet에게 전달해 주는 일을 한다. 만일 추가, 삭제, 또는 갱신의 작업이 발생하면 이에 근거해서 DBSet의 내용을 갱신해주는 기능을 또한 갖는다.
AddFact_ManufactSpec();	리스트에 새로운 원소를 추가
DelFact_ManufactSpec();	리스트에 하나의 원소를 삭제
WriteFact_ManufactSpec();	리스트에 특정 원소의 정보를 갱신

가 존재한다. 그리고, 리스트의 정보를 읽고 쓰는 데에 무리가 없도록 하기 위한 인터페이스 함수들이 존재한다. <표 4>는 ManufactSpec DBSet 클래스와 사상되는 FD 클래스의 주요 멤버 함수를 보여주고 있다.

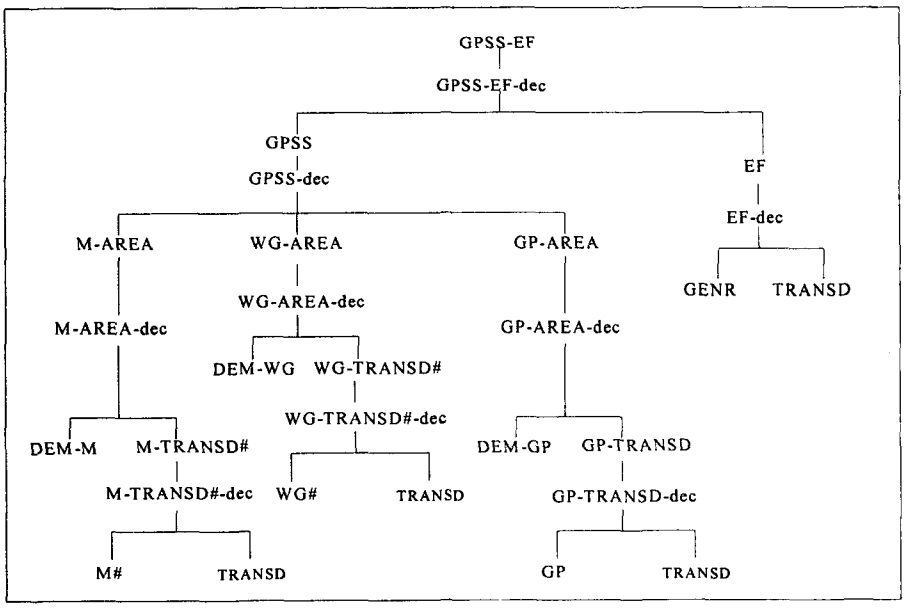
3.4 GPSS 모델에의 적용

<그림 4>는 본 연구에서 구현한 시뮬레이션 모델의 전체적인 구조이다. 모델을 보면 DEM (Dispatching Expert Model), M#(그래이팅을 만들기 위한 머신 모델), WG#(엔드 바 용접을 위한 가공 팀 모델) , GP#(도금 공정)으로 구성된 것을 알 수 있다. <그림 4> 모델은 목적에 따라 분류하면 프로세스 모델을 구성하는 M#, WG#, 그리고 GP#과 프로세스 모델의 지능형 제어를 위해 존재하는 DEM모델로 구분된다. DEM-M 모델은 데이터베이스에서 추출한 각 머신의 현재 상황과 작업의 내용을 보고 최적의 머신에 작업을 할당한다. DEM은 모델 안에는 작업을 작업 수행 모델에 할당하는 데 필요한 규칙들이 rule-base에 저장되어 있다. DEM 모델은rule-base의 규칙과 작업 정보에 기초해서 최적의 모델에게 작업을 할당한다. GPSS-EF의 DEM모델 내부의 전문가 시스템은 자신이 할당해야할 작업의 명세와 작업 할당 대상이 되는 머신 정보를 데이터베이스에서 가져와 이를 사실(Fact)로 사용한다. DEM 모델은 데이터베이스의 정보를 사실로 변환하는 과정에 FCL을 사용한다. 또한, DEM 모델의 추론 결과는 데이터베이스의 테이블에 저장이 되는데 이 때 FCL이 데이터베이스에 추론 결과를 저장하기 위한 인터페이스가 된다. 그레이팅 프로세스 스케줄링을 위한 모델들과 EF(Experimental Frame)의 구조는 <그림 5>에 나타나 있다. Root-entity인 GPSS-EF는 EF와 GPSS로 구성되어 있다. GPSS는 평가 대상이되는 모델을 의미한다.

GPSS-DES entity는 M-AREA, WG-AREA, 그리고 GP-AREA 3의 구성 요소로 이루어지고, 각각의 구성 요소들은 하나의 DEM(Dispatching Expert Model)과 각 공정의 프로세스 모델로 구성된다. 특히, 각각의 프로세스 모델에는 성능을 점검하기 위한 Transd모델을 M#, WG#, GP 각



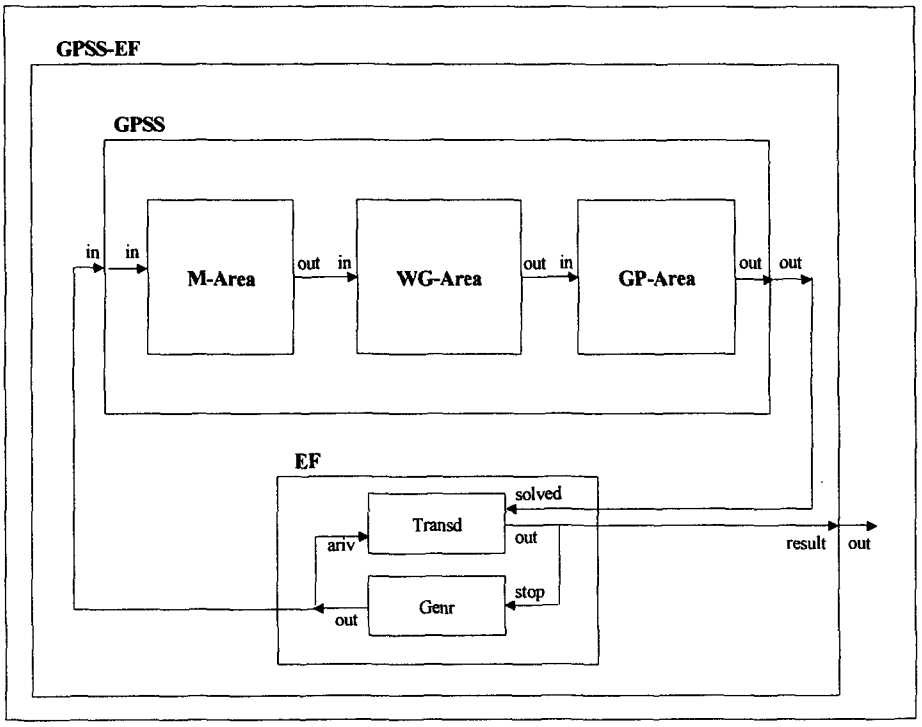
<그림 4> GPSS Model의 전체 구조



<그림 5> GPSS-EF의 System Entity Structure

각에 연결하였다.

<그림 6>은 GPSS-EF 모델에서 GPSS 모델과 EF 모델과의 컴포넌트간의 연결 관계를 볼 수 있다. GPSS-EF 모델은 GPSS모델과 EF 모델로 구성되어 있으며, GPSS 모델은 Machine -Area 모델, WorkGroup-Area 모델, GalvProcess

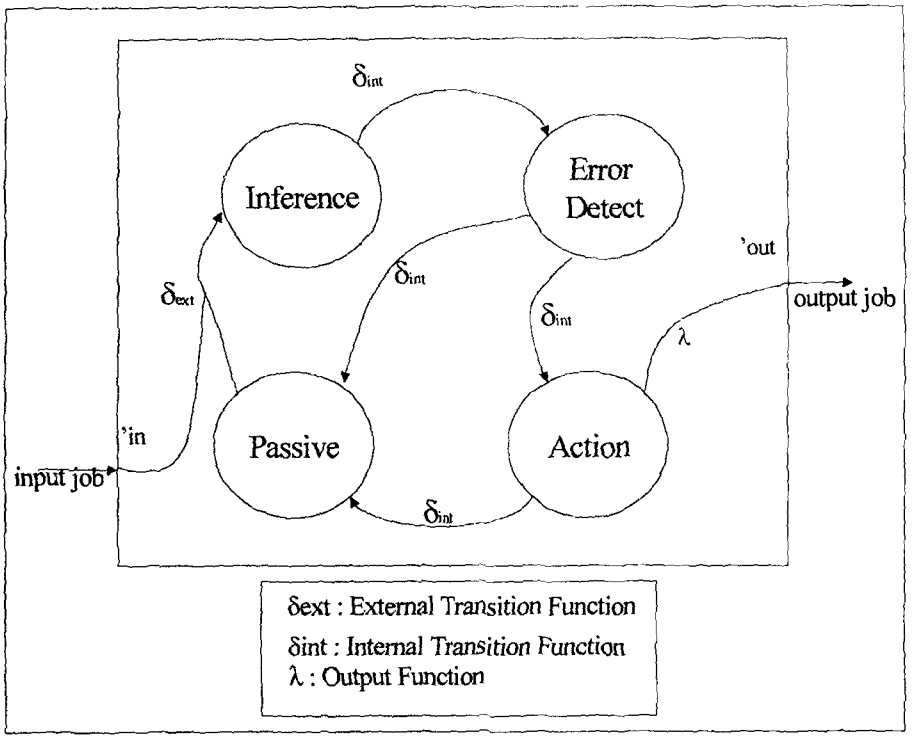


<그림 6> GPSS-EF의 구조

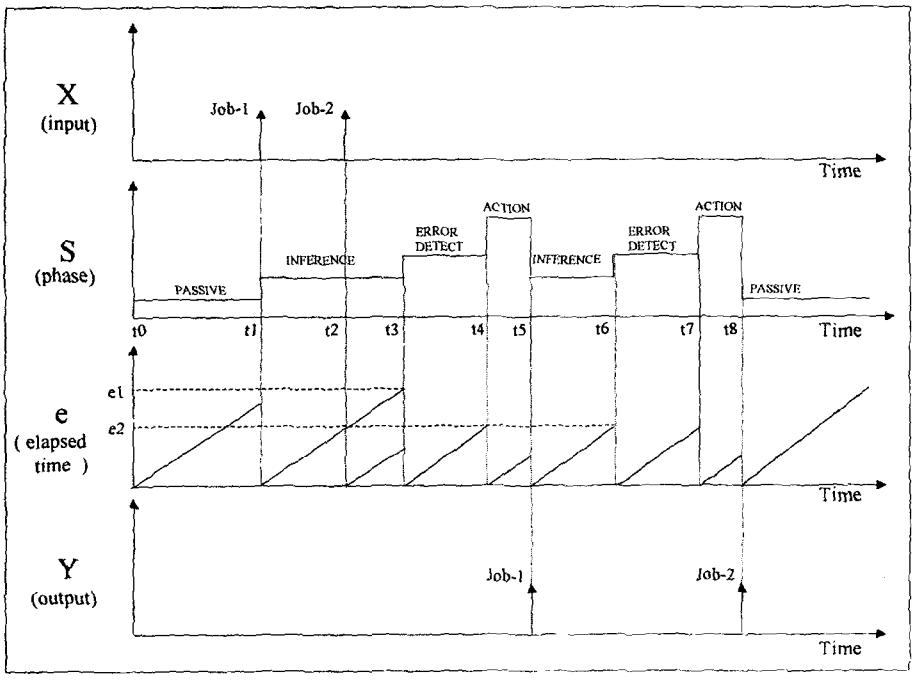
-Area 모델로 구성되어 있다. EF 모델에서는 Transd와 Genr로 구성된다.

<그림 7>은 GPSS시스템의 DEM모델의 상태 전이 다이어그램을 보여주고 있다. Passive, Inference, Error Detect, Action의 4개의 상태를 가지고 있다. Passive상태는 현재 모델에 처리되고 있는 작업이 없고, 작업을 처리할 준비가 되어 있음을 나타낸다. Inference상태는 모델 안의 전문가 시스템이 현재 작업을 처리하기 위한 추론을 하고 있음을 나타낸다. 이 상태에서는 모델 안의 전문가 시스템이 제어를 가지고 있다. Passive에서 Inference로의 상태 전이는 입력포트인 'in을 통한 외부 입력에 의해서 구동되는 외부 전이 함수에 의해서 이루어진다. 정상적인 추론이 끝나면 모델의 상태는 Inference에서 Error Detect 상태로 전이된다. Error Detect상태에서는 추론 결과를 데이터베이스에 저장하기 전에 FCL에 있는 데이터의 무결성을 검사한다. 무결성 검사는 주키(primary key)가 널(Null)이거나 동일 주키에 2개 이상의 결과 값이 있는지를 검사하게 된다. 정상적으로 추론되었다면 Action 상태로 전이하고, 만일 비정상적으로 추론되었다면 에러가 발생하였다는 정보를 log에 기록하고 Passive 상태로 전이한다. Action상태에서 모델은 추론의 결과에 따라 동작한다. Action상태는 모델링하는 대상에 따라 다른 이름의 여러 상태로 나누어질 수 있다. Action상태가 종료되면 output함수(λ)가 구동되어 출력을 발생 시킨다.

<그림 8>는 DEM모델의 timing diagram의 한 예를 보여주고 있다. <그림 8>의 X축은 입력을 나타내는데 두 개의 작업이 입력으로 들어오는 것을 볼 수 있다. 첫 번째 작업 job-1은 t1이라는 시간에 외부 입력으로 들어와 모델의 상태를 Passive상태에서 Inference상태로 전이하도록 만든다. t2에서는 job-2가 입력으로 들어오고, 이 job은 현재 모델이 처리할 수 없는 상태이기 때



<그림 7> State Transition Diagram

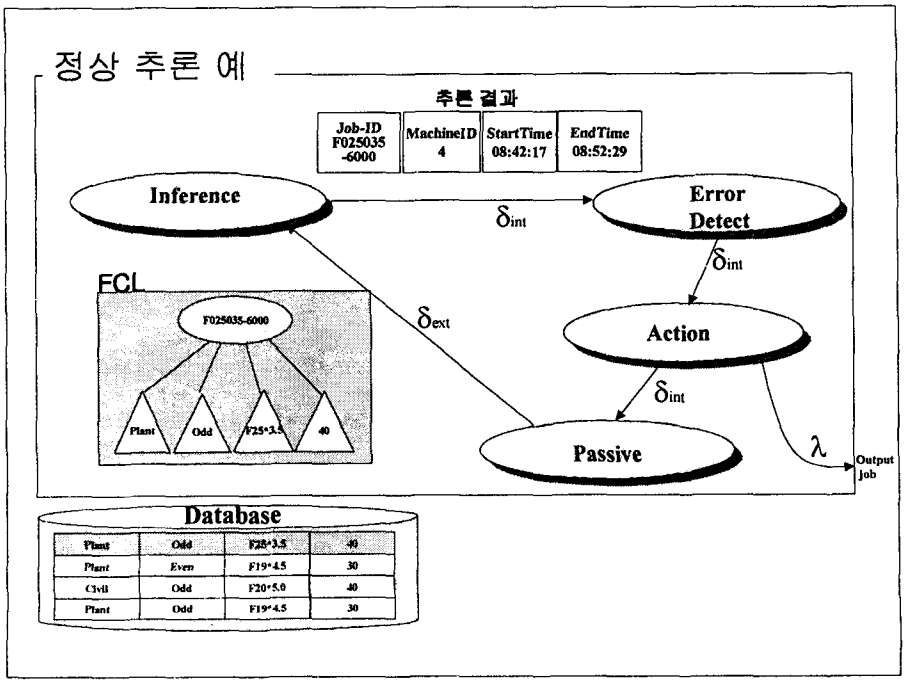


<그림 8> Timing Diagram of GPSS

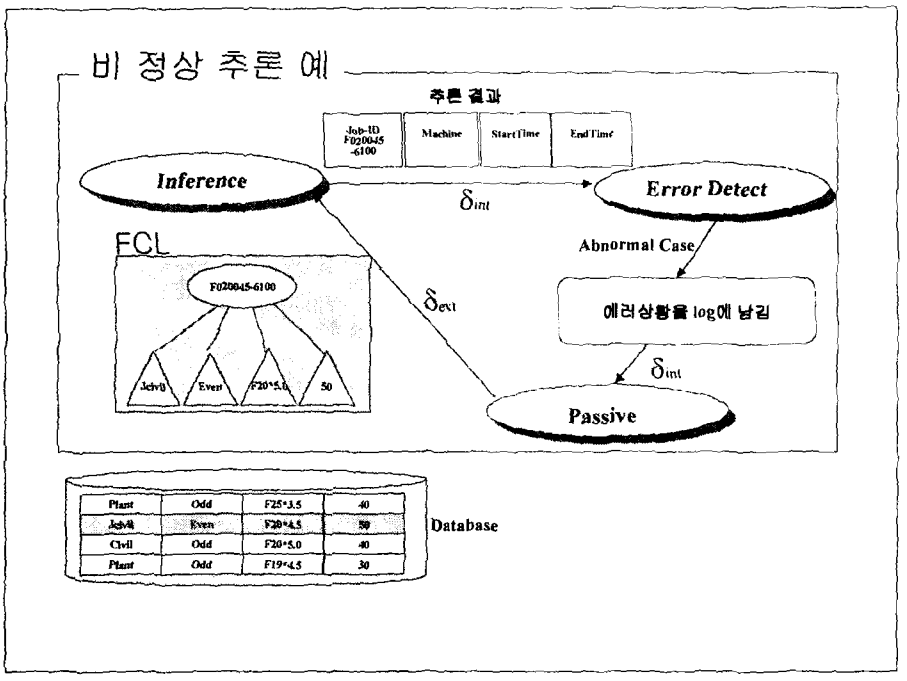
문에 모델의 큐에 삽입된다. Inference 상태에서 추론이 정상적으로 종료되면, 이후 t3에서 Error Detect상태로 전이한 후 추론 결과에 대한 에러를 검사한다. t4에서 Action상태로 전이한 후 t5에 종료되고 동시에 추론의 결과로 job-1이 출력으로 나간다. 그리고, 모델은 현재의 큐의 상태를 점검하여 t2에 입력된 job-2를 처리하기 위한 Inference상태로 전이된다. 시간 t6에 추론이 정상적으로 종료되고 Error Detect상태로 전이한 후 추론 결과에 대한 에러를 검사한다. 모델의 상태는 다시 Action상태로 전이되어 Action상태가 종료되는 t8에는 job-2를 출력한다. <그림 8>을 볼 때 job-1과 job-2를 처리하기 위해서 사용된 시간 e1과 e2는 전문가 시스템이 규칙을 실행(firing)하는데 걸리는 시간이라고 할 수 있다.

4. 시뮬레이션 모델의 기능 검증

이 장에서는 디자인 및 구현된 시뮬레이션 모델이 정상적인 추론을 했을 때와 비정상적인 추론을 했을 때의 모델의 검증 과정을 설명한다. 아래 <그림 9>은 시뮬레이션 모델의 정상적인 추론 과정을 검증하는 그림이다. <그림 9>에서 보는 바와 같이, 모델이 추론을 하려면 데이터베이스에서 추론에 필요한 데이터를 추출한다. 추출된 데이터가 FCL에서 DBContainer와 Container를 거치면서 데이터가 지식베이스의 사실로 사상되어 사실클래스를 이루는 객체가 된다. 이러한 객체는 Inference상태에서 추론되고 추론된 결과는 Error Detect상태에서 추론이 정상적으로 되었는지 검사를 하게 된다. 정상적으로 추론되었다면 Action상태로 전이하여 결과 값을 출력하게 된다. <그림 9>에서처럼 추론의 결과로 4번 Machine으로 추론 결과가 나오게 된다. 이러한 과정은 데이터베이스에서 추출된 데이터를 사용하여 진행된다. Error.Detect 상태에서는 이러한 추론 결과에 대한 검사를 하게 된다. 이때 행하는 검사는 정상적인 추론이 되었는가와 무결성 검사이다. Error Detect 상태에서 에러가 검출되지 않으면 Action상태로 전이되며 추론의 결과를



<그림 9> 정상 상태 추론



<그림 10> 비정상 상태 추론

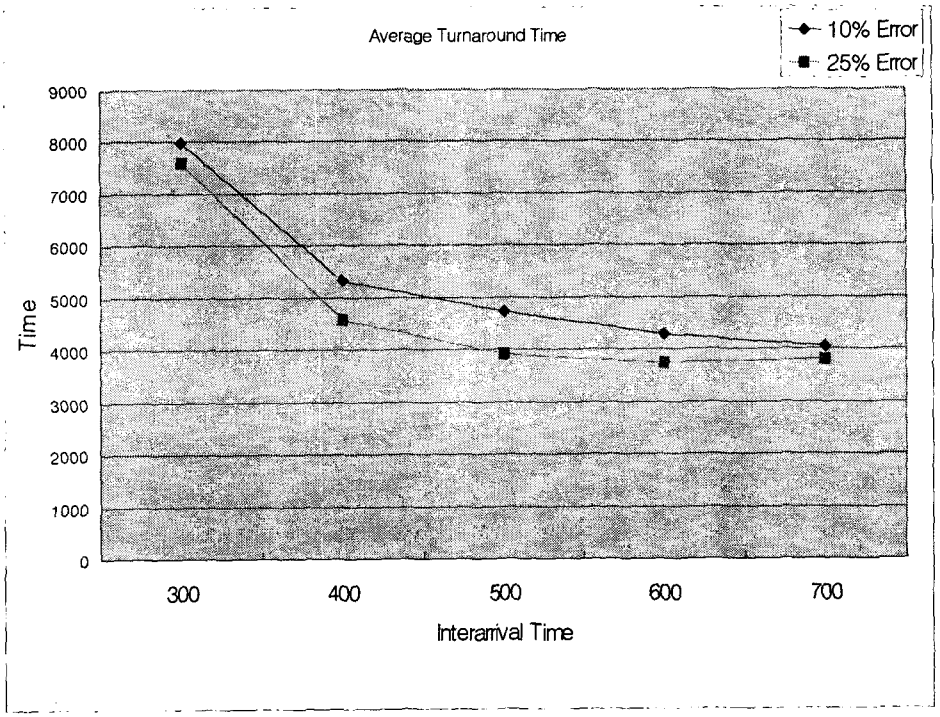
출력하게 된다.

만일 <그림 10>에서처럼 추론한 데이터가 없거나 일관성이 지켜지지 않아 Error Detect상태에서 비정상적인 추론이라고 감지되면 Log에 에러가 발생하였음을 남기고 Passive상태로 전이하게 된다. GPSS에서는 Machine의 용도가 'Plant' 이거나 'Civil'인 것만 처리하므로 <그림 10>에서 보는 바와 같이 Machine의 용도가 'Jcivil'라면 Machine에 대한 추론이 불가능하므로 추론 결과는 없다. <그림 10>에서 보듯이 추론의 중간 결과 부분이 빈칸으로 되어 있는 것은 추론할 수 없는 입력 정보를 받아 들였기 때문이다. 이렇게 비정상적인 추론 결과를 Error Detect상태에서 에러임을 감지하면, 에러처리를 해당 job의 에러를 log에 쓰고 Passive상태로 전이한다.

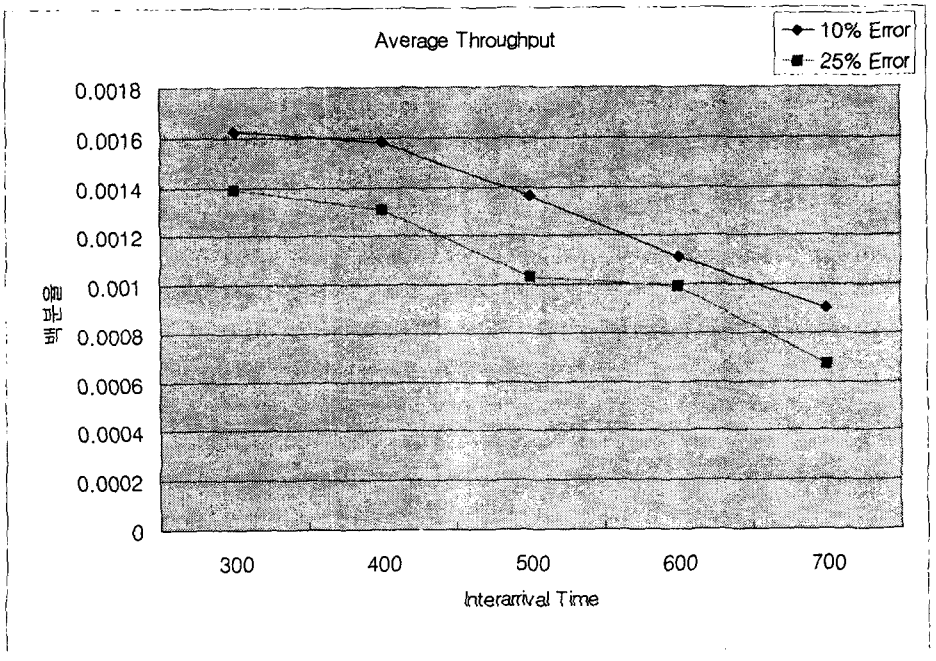
5. 실험 결과

<그림 11>과 <그림 12>는 시뮬레이션을 수행한 결과이다. 시뮬레이션 실행에서의 비교 대상은 error율에 따른 평균 turnaround 시간과 평균 throughput이다. 또한, 각 시뮬레이션 수행의 interarrival 시간은 300에서 700시간을 평균으로 지수분포 사용하여 작업을 발생 시켰다. <그림 11>에서 error율이 클 경우에는 turnaround 시간이 더 작게 나타나고, error율이 작은 경우 turnaround 시간은 크게 나타나는 것을 볼 수 있다. 이는 error가 발생하면 작업을 처리해주는 서버를 점유하지 못하기 때문에 발생하는 시간적 차이이다. 그런데, 이것은 error가 발생하는 위치에 따라서 그 비율이 틀려 질 수 있다. 3개의 공정 중에서 어떤 공정에서 error가 발생하느냐에 따라서 평균 turnaround 시간에는 차이가 나타나게 된다. 또한, interarrival 시간이 작을수록 작업들의 turnaround 시간이 급격히 커지는 것을 볼 수 있다.

<그림 12>는 error율이 큰 경우에 throughput은 작게 나타나며, error율이 작을 경우 throughput은 크게 나타나는 것을 보여준다. 이것은 error가 발생할 경우 해당 작업은 모델의 출력으로 나타나지



<그림 11> Error 비율에 따른 평균 Turnaround 시간 차이



<그림 12> Error 비율에 따른 평균 Throughput 차이

않게 때문이다. 여기서도, interarrival 시간이 커질수록 서버들의 대기 시간이 많아지기 때문에 throughput은 작게 나타난다.

6. 결론 및 향후과제

본 연구에서 전문가 시스템과 데이터베이스를 사용하는 GPSS 모델을 구축하고, 모델의 실행을 통해 시뮬레이션 모델 내부의 지식 처리 모듈이 데이터베이스 관리 시스템과 제대로 연계 동작하여 결과를 내는 지를 검증하였다. 모델 내부의 전문가 시스템과 데이터베이스의 결합에 생길 수 있는 문제를 지적하고, 이를 해결하기 위한 FCL (Fact Class Library)를 디자인하고 이를 사용하여 모델링 하였다. 또한, 시뮬레이션 결과를 통해서 error 검출 비율에 따른 turnaround 시간과 throughput을 살펴봤다. 본 논문에서 제시한 FCL은 기타 다른 데이터 소스와 전문가 시스템과의 인터페이스에도 발전시켜 적용 가능 할 것으로 본다.

향후 연구 과제는 다음과 같다. 첫째, Error Detect상태에서 현재는 error를 log로 남기고 바로 Passive상태로 전이를 하는데, 이에 대한 더욱 세부적인 디자인이 요구된다. 예를 들면 error의 종류를 여러 단계로 나누어서, log로 남겨야 할 것, 교정 해야할 것, 실행을 중단하고 사용자에게 알려야 할 것 등의 단계를 나누어 이에 대한 모델의 상태의 세분화에 대한 연구가 필요하다. 둘째, 본 연구에서 사용한 FCL의 확장을 통해서 전문가 시스템과 데이터베이스사이의 연계를 위한 일반적인 환경의 구축이 요구된다.

참고 문헌

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

[2] 조대호, 김형종 "분산전문가 시스템의 기능을 갖는 이산사건 시뮬레이션" 한국시뮬레이션학회 논문지, Vol. 7, No. 2, December 1998.

[3] Bernard P. Zeigler, Tae H. Cho, Jerzy W. Rozenblit , "A Knowledge-Based Simulation Environment for hierarchical Flexible Manufacturing" IEEE Transaction On Systems, Man, and Cybernetics-Part A : System and humans, Vol. 26, No.1 January 1996

[4] Tae H. Cho and Bernard P. Zeigler, "Simulation of Intelligent Hierarchical Flexible Manufacturing : Batch Job Routing in Operation Overlapping" IEEE Transaction On Systems, Man, and Cybernetics-Part A : System and humans, Vol. 27, No.1 January 1997.

[5] Stonebraker, M., "Implementation of Rules in Relational Database System," Database Engineering, Vol.6, No.4, 1983.

[6] 김형종, 조대호, 이칠기, 김훈모, 노용한 "반도체 생산라인에서의 이탈처리 추적 전문가 시스템의 지식베이스 구축," 제어 자동화 시스템공학 논문지 제5권 제1호 1999.1.

[7] K. Whang, S. Brady, "High-Performance Expert System-DBMS Interface for Network Management and Control," IEEE Journal on Selected Area in Communications, Vol. 7, NO. 3, APRIL, 1989.

[8] H. C. Howard, D. R. Rehak, "KADBASE: Interfacing Expert Systems with Databases," IEEE Expert, Vol. 4, Issue. 3, Fall, 1989.

[9] R. A. Perez, S. Koh, "Integrating Expert Systems with a Relational Database in Semiconductor Manufacturing," IEEE Tran-sactions on Semiconductor Manufacturing, Vol. 6 No. 3, Aug. 1993.

[10] R. Fikes, T. Kehler, "The role of frame-based representation in reasoning," Communications of the ACM, Vol. 28, No. 9, Sep. 1985.

[11] R. Rodriguez, et. al, "Efficient Expert System: Rule-Base management Via

Relational Database Techniques," Advances in Artificial Intlligence Research, Vol.1 pp. 65-79, 1989.

[12] Kellogg,C., "From Data Management to Knowledge Management," IEEE Computer, Jan.,1986.

[13] Stonebraker, M., Woodfill, J., and Andersen, E., "Implementation of Rule in Relational Data Base Systems," Database Engineering, Vol. 6, No. 4, 1983, pp.65-74.

[14] Cremers, A. and Domann, G., "AIM-An Integrity Monitor for the Database System INGRES," Proc. Int. Conf. on Very Large Databases, 1983, pp.167-170.

[15] Dittrich, K.R. Kotz, A.M., and Mülle, J.A., "An Event/Trigger Mechanism to Enforce Complex Consistency Constraints in Design Databases," ACM SIGMOD RECORD, Vol. 15 No. 3, Sept. 1986, pp. 22-36.

[16] E. N. Gehani, et. al, "Event Specification in an Object-Oriented Database," Proceedings of ACM-SIGMOD, pp. 81-90, 1992.

저자소개

김형종
1996년 성균관대학교 정보공학과 학사
1998년 성균관대학교 정보공학과 석사
2000년~현재 성균관대학교 전기전자 및 컴퓨터공학과 박사과정
관심분야 : 시뮬레이션 모델링, 인공지능, 보안 시뮬레이션.

이주용
1998년 성균관대학교 정보공학과 학사
2000년 성균관대학교 전기전자 및 컴퓨터공학부 석사
2000년~현재 (주) 한국통신 프리텔 수도권 네트워크 본부
관심분야 : 시뮬레이션 모델링, 전문가 시스템, 데이터 마이닝.

조대호
1983년 성균관대학교 전자공학과 학사
1987년 Univ. of Alabama 전자공학 석사
1993년 Univ. of Arizona 컴퓨터공학 박사
1993년~1995년 경남대학교 전자계산학과 전임강사
1995년~현재 성균관대학교 전기전자 및 컴퓨터공학부 부교수
관심분야 : 시뮬레이션 모델링 방법론, 지능형 시스템, 공장 자동화, 보안 시뮬레이션.