컴퓨터 시스템의 시뮬레이션 모델링에 대한 정보 구조의 구축에 관한 연구

손 달 호
계명대학교

컴퓨터 시스템의 시뮬레이션 모델링에 대한 정보 구조의 구축에 관한 연구

손 달 호
계명대학교

개 요

본 논문은 IBIS(Information-Based Integrated Simulation)라 불리우는 정보 중심의 시뮬레이션 방법을 다중 처리(Multiprocessor) 컴퓨터 시스템의 시뮬레이션 모델링에 이용하였다. IBIS는 지금까지 시뮬레이션의 중요한 접근 방법이었던 언어 정의(Language-Defined) 혹은 목적 지향적인(Object-Oriented)방법의 단점을 보완한 방법으로 각각의 다른 단계에 있는 여러개의 모델들을 조합하여 정보 시스템을 구축할수 있다. 본 연구에서는 IBIS적인 접근 방법을 컴퓨터 시스템의 시뮬레이션 모델링에 국한하였으나 이를 확장하면 서비스 시스템을 포함하여 시뮬레이션이 적용될수 있는 모든 시스템에 IBIS적인 접근 방법을 이용할수 있을 것이다.

제 1 장 IBIS 접근 방법의 특징

최근 몇년동안 시뮬레이션 모델링을 위한 시뮬레이션 언어의 발달은 크게 2개의 부류로 이루어져 왔다. 그 중 하나는 고급(High-Level)및 언어 정의(Language-Defined)중심의 경향을 보여주었고 이와같은 언어로는 MAP/I, XCELL, SIMFACTORY등을 들수 있다(Miner 와 Rolston, 1983; Conway, et al., 1986; C.A.C.I., 1987). 그러나 이와같은 부류의 언어들의 한가지 단점은 시스템의 객체(Entity)들을 정의하는데 있어 경직성(Rigidity)이 존재하였다. 즉 사용자들은 객체의 속성이나 그들의 움직임을 정의하는데 있어 보다 많은 유연성을 가지지 못했었다. 2번째 부류의 접근 방법은 보다 목적 지향적인(Object-Oriented)방법으로, 이러한 방법들은 모델자가 어떤 시스템을 규정짓는데 필요한 정보 구조를 직접 구축해야만 되는 어려움이 있었다(Baskaran과 Reddy, 1984; Reddy, et al., 1985).

본 연구는 Ketcham, et al.(1989)이 개발한 IBIS(Information-Based Integrated Simulation)라 불리우는 3번째 접근 방법으로 이 방법은 정보간의 관계를 이용하여 시뮬레이션 모델을 구축하고 또한 모델의 계층적 개발시 각각의 단계에서 여러 모델들을 조합하여 어떤 모델링을 수행할수 있는 방법이다. 또한 IBIS는 D/B적인 접근방법을 이용하기 때문에 사용자는 D/B Mgt. 명령어를 사용하여 대상 모델에 대해 데이타를 입력, 회수 및 수정할수 있는 특징도 있다. 이와 더불어 IBIS는 시뮬레이션 제어 모듈을 이용하여 어느 조건을 만족하는 데이타를 회수하여 그들을 필요한 기억 장소에 불러들임과 동시에, D/B에 정의된 정보 관계를 추적함으로써 시뮬레이션을 수행할수도 있다. 따라서 시뮬레이션 모델을 개발함에 있어 모델링에 필요한 내용들을 D/B의 관계로 나타냄으로서 아래와 같은 몇가지 특징을 가질수 있다.

(1) 만들어진 D/B는 작업에 대한 새로운 통제 방법을 나타내기 위해 이용될수 있다. 예를들면 D/B를 이용하여 여러 종류의 작업의 Resource에 대한 요구, 작업 처리 순서, 작업 우선 순위등의 내용들을 가지고 있는 복잡한 시스템을 쉽게 모델링할수 있다.
(2) 만들어진 D/B는 계층적인(Hierarchical) 구조에 중점을 두기 때문에, 대상 시스템에 대한 계층적인 모델 개발의 각각의 단계에서도 여러 모델들을 조합할수 있다. 즉 어떤 계층적 단계에서도 만들어진 D/B를 이용하여 실행 가능한 모델을 만들어 낼수 있다.
(3) 끝으로, 만들어진 D/B는 D/B적 특성을 가지고 있기 때문에 사용자는 프로그래밍 코드를 이용하기보다는 D/B 호출 모듈(Query)을 이용하여 D/B에 쉽게 접촉할수 있다.

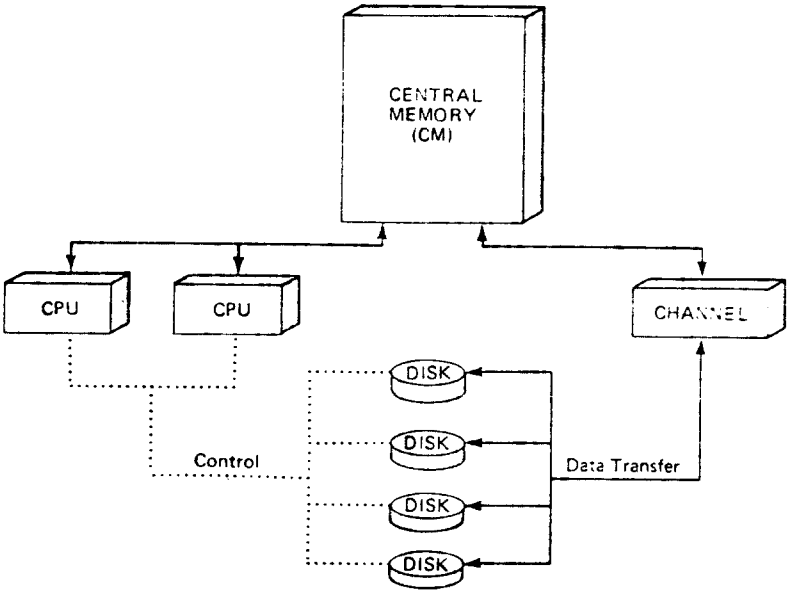
IBIS는 기본적으로 ICAM(Integrated Computer Aided Manufacturing)적인 접근 방법을 이용하여 개발되었다. 즉 ICAM의 3가지 요소인 $IDEF_0$, $IDEF_1$ 및 $IDEF_2$의 개념을 기본으로 하고 있다(Blank와 Carrasco, 1983; Miner와 Pritsker, 1980;Young, 1983). 이와 더불어 IBIS는 ICAM에서는 해결하지 못한점, 즉 시스템 모델링(System Modeling), 정보 관계(Information Relationship), 시뮬레이션 모델(Simulation Model)들간의 관계를 연구하여, 이들 3개의 접근 방법들을 총체적으로 결합하였다.
본 연구는 IBIS적인 접근 방법을 컴퓨터 시스템의 시뮬레이션 모델링에 적용하는 것을 주된 목적으로 하였다. 이와 더불어 IBIS적인 접근 방법을 컴퓨터 시스템의 시뮬레이션 모델링에 적용함으로써 지금까지 시뮬레이션에 이용된 다른 방법들과의 차이점을 비교하여 볼수 있을 것이다.

제 2 장 모델링 대상의 정의

일반적으로 모든 시스템은 입력-작업-출력(Input-Processing-Output)의 형태로 나타낼수 있다. 즉 시스템은 어떤 입력 요소를 받아들여서 작업을 하여 우리가 원하는 형태의 출력물을 만드게 된다. 약간 복잡한 시스템은 여러개의 하위시스템(Subsystem)으로 나눠질수 있으며, 이와 더불어 입력 요소들은 하위시스템 사이를 이동 할수도 있다. 물론 각각의 하위시스템도 하나의 시스템을 형성하게 되고 또다른 하위 시스템들로 나눠질수 있다. Zeigler(1984)는 시뮬레이션에서 계층적인 IPO (Input-Processing-Output) 모델의 역활에 대해 논하였다. 그의 논문에서 계층적인 시스템이나 그것을 구성하는 하위시스템들을 "작업 장소(Station)"라 불렀으며, 특히 시뮬레이션 모델링에 있어 작업장소의 계층성에 대해 논하였다. 아래는 본 연구 목적에 나타난 것처럼 컴퓨터 시스템의 시뮬레이션 D/B 모델링 및 위에서 언급한 IPO모델을 잘 설명할수 있는 예를 Pritsker(1986)로 부터 발췌하여 연구 목적에 맞게 수정하여 나타낸 것이다.

"지금 어떤 다중 처리기(Multiprocessor) 컴퓨터 시스템은 2개의 CPU(그들은 어떤 일정한 용량의 메모리를 공유함), 4개의 Disk Drive(그들은 각각 CPU에 의해 호출될수 있음), 그리고 1개의 Data Channel로 구성되어 있다. (그림-1)은 이들의 관계를 도식적으로 나타내고 있다.
이와 같은 컴퓨터 시스템에서 어떤 작업을 받아야 되는 Job들은 어떤 주어진 통계학적인 분포에 따른 도착율을 가지며, 또한 Job에 대한 CPU 작업시간은 I/O시간과 실행시간으로 이루어지며 이러한 각각의 작업 시간들도 사전에 정해진 어떤 통계학적인 분포를 이룬다고 생각하자. 시스템에 도착하는 Job들은 작업 우선 순위가 부여되며, 우

(그림-1) 다중처리기 컴퓨터시스템의 개요



선 순위는 Job의 메모리 요구량에 반비례하여 부여된다. 그리고 메모리가 일단 Job에 할당되면 Job은 유휴 CPU를 이용하여 작업이 실행된다. 이와함께 Job들은 I/O사용을 요청하여 한개의 I/O라도 사용하고 있으면 CPU를 계속해서 사용할 수 있다. 따라서 I/O사용에 대한 요청이 허용되지 않으면 CPU사용도 허용되지 않음과 동시에 I/O사용의 요청도 대기상태에 머물게 된다.

'Job들은 CPU에서의 작업후 자동적으로 I/O이용 요청을 함과 동시에, CPU에서 Disk로의 사용 요청이 이루어진다. Disk에 Job이 저장될 장소를 찾는 시간 역시 어떤 통계학적인 분포를 이루며, 장소를 찾으면 I/O Channel을 통하여 작업을 마친 Job에 대한 Disk로의 이송이 행해진다. 작업을 마친 Job에 대한 이송이 끝나면 I/O사용 요구가 만족된 것으로 간주된다.

지금 우리는 이러한 다중처리 컴퓨터 시스템에 대해 Job이 시스템에서 작업을 받는데 걸리는 시간 및 Disk, I/O Channel, CPU와 Central Memory의 이용률 및 여기에 대기중인 Job의 수와 대기시간(Queueing Time)을 알아보기 위해 시뮬레이션 연구를 해 볼려고 한다."

아래에는 이와같은 컴퓨터 시스템의 시뮬레이션 모델링에 IBIS 접근방법을 적용할때 어떤 특징을 가지게 되는지를 살펴보았다.

(1) 주어진 작업이 어떤 작업장소(Station)에서 작업해야 되는지를 반드시 정의할 필요가 없다는 점이다. 일반적으로 시뮬레이션에서는 작업장소(Station)와 작업(Operation)간의 "1:1" 대응관계를 가정한다. 즉 주어진 작업장소는 어떤 한 종류의 작업만 수행하여 왔지만, 최근에 이르러서는 "MANY TO MANY(M:M)"의 관계로 발전하게

되었다. 즉 주어진 작업 장소는 여러 종류의 작업을 할수 있고, 또한 같은 작업이 여러 대안적인 작업장소중 하나에서 행해질 수 있다. 바꾸어 말하면 Job을 어떤 작업장소에 보내는 것은 어떤 작업이 수행되어야 하는가와는 독립적인 문제인 것이다.

(2) 객체들은 경우에 따라 능동적 혹은 수동적으로 움직일수 있다. 대개의 시뮬레이션 언어들은 일시적 객체(Temporary Entity)와 영구적 객체(Permanent Entity)들을 구별하고 있다. 일반적으로, 일시적 객체들은 능동적이며 어떤 작업을 시작하도록 하는 반면에 영구적 객체는 수동적이며 어떤 작업 요청을 기다리게 된다. IBIS시스템은 어떤 주어진 객체(Entity)가 경우에 따라 일시적 객체및 영구적 객체로 바뀌는것을 허용한다. 예를들면 어떤 Job은 I/O Channel을 이용할수 있지만(능동적), 메모리가 할당될 때까지(수동적) 기다려야만 한다. 마찬가지로 CPU는 주어진 Job의 작업에 대해서는 수동적이지만(왜냐하면, Job이 도착해야 작업을 할수 있으므로), 수리/보수작업에 대해서는 능동적이다. 왜냐하면, 어떤 정해진 시간 간격으로 컴퓨터는 수리자를 호출함으로서 수리/보수 작업을 시작할수도 있기 때문이다. 이와 같은 경우처럼 어떤 객체가 경우에 따라 성격이 바뀌어도 모델링이 가능하다. 그러나 일반적으로 Resource 객체(Entity)는 작업 조건이 달리 명시되지 않는한 수동적 객체로 간주될 것이며, 반면에 Job 객체(Entity)는 능동적 객체로 간주될 것이다.

(3) IBIS는 "Push"와 "Pull" 작업 조건을 수행할수도 있다. 즉 Push원리를 이용하면 시스템의 움직임은 어떤 입력에 의해 작업을 수행하게 되고 이와같은 움직임이 시스템 전체를 통해 전달된다. 이와는 반대로 Pull원리를 이용하면 시스템은 출력에 대한 요구로써 작업을 수행하게되고 이와같은 움직임이 시스템 전체를 통해 전달되게 되는 것이다.

(4) 특별한 경우의 모델링 요구 사항을 만족시키기 위해서는 사용자가 정의한 속성(Attribute)이 일시적 객체와 더불어 시스템에 있는 모든 대상물에 부여되어야 한다는 점이다. 예를들면 컴퓨터 시스템에서 각각의 작업장소에 대해 작업을 요구하는 Job에 대한 준비 작업 및 작업 회수를 통제해야 할 경우, IBIS에서는 "CURRENT SETUP"과 "CYCLE"의 속성(Attribute)이 작업 장소의 객체(Station Entity)에 부여될수 있고 "LOADING"의 속성(Attribute)이 작업물 객체(Job Entity)에 부여될수 있다. 이와 함께 사용자가 정의한 속성들은 일반적으로 D/B에 정의된 데이타 필드처럼 DBMS 명령어를 이용하여 입력, 회수 및 수정할수도 있다.

위의 사항들을 정리하면 어떤 연구 대상의 시스템은 시뮬레이션 정보 구조를 만들기 위해서는 아래와 같은 사항들에 대해 명확한 정의를 필요로 함을 알수 있다.

(1) 어떤 주어진 작업 장소(Station)들은 계층적으로 하위의 작업 장소들로 나눠 질수 있어야 한다. 예를 들면, 다중처리 컴퓨터 시스템은 CPU, I/O Channel, Disk 등의 각각의 요소들로 나눠질수 있고, 또한 이러한 각각의 요소들은 그들을 구성하는 또 다른 요소로 나눠질수 있다. 이와같이 정의된 종속적(Parent-Child)관계를 통하여 D/B에서는 모든 요소들이 계층적인 관계를 유지할수 있도록 한다.

(2) 연구 대상 시스템에 대해 작업 장소들간의 Job의 작업 경로와 주어진 작업 장소에서 Job에 수행되는 작업의 내용을 명확히 확인하여야 한다.

(3) 연구 대상 시스템에 대해 어떤 작업 장소와 그러한 작업 장소에서 행해지고 있는 작업사이의 "M:M(Many to Many)" 관계를 확인하여야 한다.

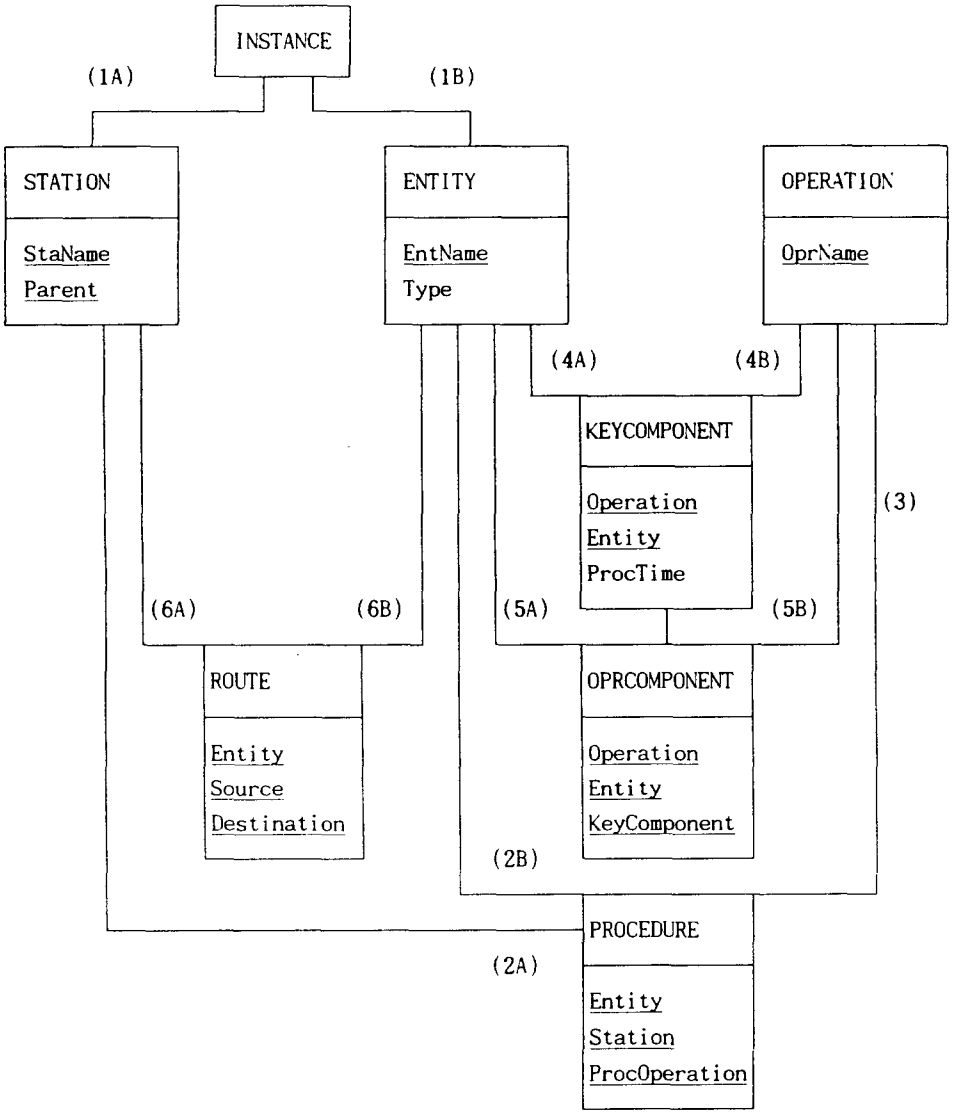
(4) 연구 대상 시스템에 대해 작업(Operation)과 그와 같은 작업들에 필요한 객체사

이의 "M:M" 관계를 확인하여야 한다.
(5) 어떤 경우에 객체가 능동적 혹은 수동적으로 바뀌는가를 확인하여야 한다.
(6) 사용자가 정의한 속성(Attribute)들이 일시적 객체 및 모든 대상물에 정의 되어야한다.

제 3 장 데이타 베이스 설계

IBIS는 시스템간의 관계를 나타내는데 네트웍(Network) D/B구조를 이용하고 있다. 비록 관계적인(Relational) D/B는 정보를 회수하고 나타내는데 보다 많은 유연성을 가지고 있지만, 네트웍 D/B는 D/B가 만들어질때 데이타 경로가 미리 정해지는 장점이 있다(Kroenke, 1983). 이렇게 사전에 정해진 데이타 경로는 시뮬레이션이 수행될때 D/B로 부터 데이타의 경로를 효과적으로 추적하는데 도움을 줄 수 있다.
네트웍 D/B는 파일들간의 "One-to-Many(1:M)"와 "Many-to-Many(M:M)"의 관계를 이용하여 파일, 레코드, 필드를 명시할수 있다. 즉, D/B에서 파일들은 소유(Owner)레코드와 한개 이상의 소속(Member)레코드로 구성되어 있고 따라서 소유 레코드와 소속 레코드들은 1:M의 관계를 가지게 된다. 이와함께 계층적 관계(Hierarchy)는 특별한 형태로써 소유 레코드는 그 자신과 같은 형태의 여러 소속 레코드를 가질수 있다.
또 다른 형태의 소속(Member) 레코드들은 2개 혹은 여러개의 소유(Owner) 레코드들을 연결할수 있다. 왜냐하면 소속(Member) 레코드는 여러개의 다른 소유(Owner) 레코드들에 속할수 있기 때문이다. 이러한 연결(Bridging) 레코드들은 2개의 소유 레코드들 사이의 관계를 2개의 1:M 관계를 이용하여 M:M 관계로 나타낼수 있다. 일단 이러한 관계가 만들어지면 어떤 정해진 모듈(Query)에 의해 D/B로 부터 원하는 정보를 회수할 수 있다. 예를들면 정해진 모듈(Query)에 의해 어떤 Job을 컴퓨터에서 작업하는데 필요한 모든 작업(Operation)과 그러한 작업에 필요한 모든 객체(Entity)들을 찾을 수 있다.
따라서 IBIS에서는 어떤 객체는 여러 작업의 요소가 될수 있고, 또한 어떤 작업은 여러 객체를 요소로 가질수 있기 때문에, 객체와 작업은 M:M의 관계를 가지게 된다. 예를들면 다중처리 컴퓨터 시스템에서 I/O 장치(객체)는 Job의 입력 및 출력 작업의 요소가 될수 있고, 이와함께 Job의 작업은 I/O객체와 CPU객체를 요소로 가질수 있다. 이러한 M:M 관계는 (그림-2)에 나타난 것처럼 2개의 1:M 관계로 나타내어질수 있다. (그림-2)에서 화살표는 소유(Owner)와 소속(Member)의 관계를 보여주고 있다. 즉 소유 레코드(Owner Record)부터 소속 레코드(Member Record)로 화살표가 나타나게 된다. (그림-2)에서 "OPERATION COMPONENT" 레코드는 "ENTITY"와 "OPERATION" 레코드 모두에 공유되기 때문에 객체(Entity)와 작업(Operation) 사이의 관계는 그들에게 공유되는 OPERATION COMPONENT 레코드를 이용하여 추적될 수 있다. 그림에서 밑줄친 필드(Field)는 소유 레코드로부터 소속 레코드로 이동된 정보를 나타내고 있다. 따라서 OPERATION COMPONENT 레코드의 필드 "OPERATION"은 OPERATION COMPONENT를 어떤 정해진 작업(Operation)과 관계지우게 된다. 일반적으로 작업(Operation)은 시스템에 있는 객체들을 변환시키거나, 그룹으로 만들거나, 나누거나, 발생시키거나, 없애는데 필요한 논리들을 말하는 것으로, 작업 요소(Operation Component)와 관련된 사항들은 작업 요구의 우선순위 및 작업이 끝난후의 객체의 처리와 관련된 내용들이 될것이다. 예를들면 컴퓨터 시스템에서는 Job의 작업 우선 순위나, 작업이 끝난 Job의 처리와 관련된 내용이 여기에 해당된다.

(그림-2) IBIS 개념도의 일부(Ketcham, et al., 1989)



IBIS에서 주된 역활을 하는 파일들은 작업 장소(Station) 파일, 객체(Entity) 파일, 작업(Operation) 파일의 3개이다. 여기서 작업장소(Station)란 객체들이 작업을 받는 장소를 말하며, 객체란 시스템내에서 움직이거나 혹은 다른 객체와 상호작용을 하는 물체를 말하며, 또한 이들은 일시적 객체(Temporary Entity)와 영구적 객체(Permanent Entity)로 나눠질수 있다. 예를들면 Job은 일시적 객체이고, CPU, Data Channel, Disk, Central Memory는 영구적 객체인 것이다. 작업은 객체들간의 상호 작용을 의미한다. 즉, 작업 장소(Station)에서 발생하는 객체들간의 상호 작용을 말한다. 3개의 파일을 제외한 나머지 파일들은 작업장소, 객체, 작업들 간의 상호관계를 나타내는 파일이다. 예를들면 이러한 파일들은 Job의 작업 경로및 순서, 각각의 작업 장소에서의 작업 절차, Job의 저장 메모리 용량, Job에 대한 CPU의 작업속도등의 내용을 명시하게 된다.

(그림-2)는 전체의 시스템에 대한 IBIS 모형의 일부분을 보여주고 있다. 그림에 나타난 것처럼 작업장소(Station)와 객체(Entity) 사이의 관계는 2개의 연결파일 즉, 객체의 경로 명시("ROUTE" 레코드), 객체의 작업 절차("PROCEDURE" 레코드)로 나타내어 질수 있다. 그림에 나타난것처럼 "PROCEDURE" 레코드는 3개의 파일과 관계지워져 있는데 이러한 3개의 파일 내용들을 조합하면 우리들은 어떤 작업 장소에서의 작업 순서등과 관련된 내용을 알수 있을 것이다. 이와같이 구축된 IBIS D/B는 아래와 같은 특징을 가지게 된다.

(1) "ROUTE" 레코드와 "PROCEDURE" 레코드를 구별함으로써 모델자로 하여금 어떤 작업 장소에서의 상세한 작업 내용을 알지 못하고도 객체를 어떤 주어진 작업 장소에서 다른 작업 장소로 이동시킬수 있다. 마찬가지 방법으로, 모델자는 어떤 작업 장소에 대한 입력과 출력만을 알고서도 작업 순서를 나타낼수 있다. 따라서 각각의 작업 장소들은 모델화되고 시뮬레이션될수 있는 각각의 독립적인 모형으로 생각되어 질수 있다. 이와 더불어 시스템에 대한 전체적인 개략을 알아보기 위해 여러개의 작업 장소들을 동시에 시뮬레이션할수도 있다.

(2) "PROCEDURE" 레코드를 이용하여 IBIS 모델은 여러 종류의 Job들이 어떤 주어진 한개의 작업 장소에서 행해질수도 있고, 또한 같은 종류의 Job이 여러 종류의 작업 장소에서도 행해질수 있도록 한다.

(3) "KEY COMPONENT" 레코드를 이용하여 각각의 다른 Job에 대해 작업 시간과 관련된 사항및 작업 요구 사항을 기록할수 있도록 해준다. 따라서 각각의 Job들은 작업 시간과 관련된 사항및 작업 요구 사항을 가지고 여러개의 "KEY COMPONENT"에 의해 작업이 시작될수 있으며, 또한 각각의 객체(Entity)들도 여러 작업을 시작하도록 할수 있다.

(4) "KEY COMPONENT" 레코드를 "OPERATION COMPONENT" 레코드와 분리함으로써 주어진 객체를 어떤 작업에 대해서는 능동적으로(KEY COMPONENT) 혹은 어떤 작업에 대해서는 수동적으로(OPERATION COMPONENT)움직이도록 하여준다. 즉 "KEY COMPONENT"는 어떤 작업을 시작하도록 하는데 비해, "OPERATION COMPONENT"는 작업을 시작하기 전에 작업 요구를 수동적으로 기다리게 된다. 따라서 객체들은 어떤 작업을 수행하기 위해 그 자신이 시스템에 나아갈수도 있고 혹은 어떤 작업의 수행을 기다리기 위해 대기장소에 수동적으로 대기할수도 있다.

(5) 끝으로, 사용자가 정의한 속성(Attribute)들이 작업 장소(Station), 작업(Operation), 일시적 객체(Temporary Entity) 및 영구적 객체(Permanent Entity)들을 포함한 대개의 IBIS레코드에 첨가 될수 있다.

제 4 장 IBIS의 운영 내용

지금까지 어떤 주어진 시스템의 시뮬레이션에 IBIS접근방법을 적용할때의 고려사항 및 IBIS의 특징에 대해 살펴 보았다. 아래에는 구축된 IBIS가 어떤 방법으로 운용및 통제되는가에 대해 살펴 보기로 하자.

(1) 사용자는 D/B에 여러가지 방법으로 정보를 넣을수 있으며, 이와 더불어 사전에 정해진 D/B 호출 모듈(Query)이나 호환적인 화면(Interactive Display)을 이용하여 정보를 D/B로 부터 회수할수도 있다. 또한 D/B 그 자체가 데이타 항목들 사이의 관계를 유지하고 있기때문에 데이타는 어떤 순서로든지 입력및 회수도 할수 있다.
(2) 시뮬레이션이 수행될때, D/B는 프로그램이 요구하는바에 따라 D/B의 일부분을 회수하여 원하는 장소에 끊임없이 제공할수 있다. 따라서 D/B에 대한 사용자의 초기 입력 사항들은 D/B의 "정적" 인 부분을 구성한다. 예를들면, Job과 Resource의 형태, Job의 작업 순서및 작업 시간과 같은 사항이 될것이다. 이와 더불어 시뮬레이션 작업을 수행하게 되면, 각각의 Job의 작업상태 및 Resource의 이용 상태, 작업 요청, 대기중인 Job의 관리와 같은 "동적" 인 D/B를 생성하게 된다.
(3) 시뮬레이션의 수행을 통제하기 위해, 즉 작업중인 Job의 작업 요구 조건을 만족하기 위해, D/B는 데이타 네트웍을 추적 관리하게 된다. 「그림-2」는 IBIS 네트웍을 통하는 움직이는 객체(Entity)의 이동을 개념적으로 나타내고 있다. 각각의 움직이는 객체(Entity)들은 (1A)와 (1B)를 통하여 객체의 형태 및 현재의 작업 위치를 알린다. Job이 작업 장소에 도착하면 IBIS는 현재의 작업 장소에서의 작업 순서를 정의하기위해 (2A)와 (2B)를 통하여 "PROCEDURE" 레코드를 찾는다. 만일 "PROCEDURE" 레코드를 찾지 못하면, IBIS는 객체를 다른 작업 장소로 보낼것인지 혹은 하위 네트웍으로 보낼것인지를 결정한다. 그리고 "PROCEDURE" 레코드가 존재하면 객체는 정해진 작업 순서를 따른다. 만일 "PROCEDURE" 레코드가 객체로 하여금 작업을 수행하도록 요청하면, IBIS는 (3)을 통하여 수행해야될 작업을 확인하고, (4A)와 (4B)를 통하여 현재의 객체에 대해 작업 시간과 관련된 사항을 확인하고(이때부터 "KEY COMPONENT" 관점으로 바뀜), 추가적인 "OPERATION COMPONENT"는 (5A)와 (5B)를 통하여 확인한다. 또한 각각의 "OPERATION COMPONENT" 레코드를 이용하여 작업에 필요한 객체들을 다른 장소로 부터 끌어 올수 있다. 그리고 관계되는 "PROCEDURE" 레코드를 더 이상 발견하지 못하면 IBIS는 객체가 현재의 작업 장소을 떠나야 된다고 간주하게 되고, 따라서 객체는 (6A)와 (6B)를 통하여 앞으로의 진행 경로를 확인하게 될것이다. 마침내 IBIS가 더 이상 객체에 대한 진행 경로를 찾지 못하면, 일시적 객체는 없어지고, 영구적 객체는 대기소에 저장될 것이다.
(4) 끝으로 시뮬레이션을 수행한후 얻은 결과들은 다시 D/B에 저장되며 이렇게 저장된 내용들은 D/B 호출 모듈(Query)에 의해 호출될수 있다. 예를들면, 시스템에서 소요된 객체의 시간(Entity Time in System), 객체 이용률(Entity Utilization), 대기중인 작업의 수(Queue Length), 대기 시간(Time in Queue)등과 같은 내용에 대해 원하는 통계량(평균, 표준편차, 최대, 최소등)으로 결과를 모을 수도 있다. 이와함께 사용자가 정의한 속성(Attribute)들에 대한 통계량도 함께 요청하면, IBIS는 자동적으로 이러한 속성들의 값들을 관찰하여 이러한 속성들의 값들이 바뀔때 마다 관계된 통계량의 값들도 함께 바꾸어 D/B에 저장하게 된다.

제 5 장 결 론

IBIS에 이용된 "정보중심(Information-Based)" 적인 접근 방법은 시뮬레이션 모형의 개발에 D/B개념 및 목적 지향적인 프로그래밍(Object-Oriented Programming)의 개념을 결합하여 이용하였다. 시뮬레이션이 수행되면 IBIS는 문제가 요구하는 바에 근거하여 모델링 요소들을 선택적으로 회수하여 D/B의 전체적인 형태를 구성하게 된다. 이와함께 IBIS는 시뮬레이션의 수행을 통제하기위하여 D/B에 정의된 정보 관계를 추적함과 동시에 시뮬레이션 결과를 D/B에 재 저장하게 된다.

IBIS는 시스템에 대한 모델링과 시뮬레이션 실험을 정보관계로 정의함으로써 시뮬레이션 모델링과 설계에 대해 아래와 같은 특징을 나타내고 있다.

(1) 구축된 D/B는 대개의 시스템을 시뮬레이션하는데 필요한 정보관계를 나타내고 있다. 예를들면, 작업 장소과 객체에 대한 계층적인 정의, 자유로운 작업 경로 및 순서, 능동적 객체 및 수동적 객체의 구별, 작업에 대한 "Push" 와 "Pull" 방법등의 구축이다.

(2) 구축된 D/B는 계층적인 관점 및 총체적인 관점에의 각각의 다른 계획 수준에 대해서도 쉽게 이용될수 있다.

(3) IBIS 시스템은 개념적인 D/B로써 실제 적용을 위해서는 코드화가 필요하다. 예를들면, C 언어의 "Pointer" 기능등을 이용하여 데이타 네트웍의 모든 부분을 효과적으로 추적및 관리 할수 있을 것이다.

참 고 문 헌

1. Baskaran, V. and Y.V. Reddy, "An Introspective Environment for Knowledge Based Simulation", Proceedings of the 1984 Winter Simulation Conference, Society for Computer Simulation, San Diego, Ca., 1984, pp. 645-651.

2. Blank, L.T. and H. Carrasco, System Development Methodologies(SDM), 1983, Texas Engineering Experiment Station, Texas A&M University, College Station, Tx.

3. C.A.C.I., SIMFACTORY Promotional Material, 1987, C.A.C.I., La Jolla, Ca.

4. Conway, R., W.L. Maxwell, and R.E. Shannon, "Memory Management, Data Structures, and the Development of Advanced Simulation Software in a Microcomputer Environment", Modeling and Simulation on Microcomputers: 1985, Society for Computer Simulation, La Jolla, Ca., 1985, pp. 67-70.

5. Ketcham, M.G., R.E. Shannon, and G.L. Hogg, "Information Structures for Simulation Modeling of Manufacturing Systems", Simulation, 1989, pp 59-67.

6. Kroenke, D.M., Database Processing: Fundamentals, Design, Implementation, 2nd ed., 1983, SRA, Chicago.

7. Miner, R. and A.A.B. Pritsker, "Dynamics Modeling: The ICAM Definition Language($IDEF_2$) Approach", Proceedings of the 1980 AIIE Fall Conference, Norcross, Ga., 1980, pp. 15-23.

8. Miner, R.J. and L.J. Rolston, MAP/1 User's Manual, 1983, Pritsker and Associates, West Lafayette., In.

9. Pritsker, A.A.B., Introduction to Simulation and SLAM II, 1986, Halsted Press, N.Y.

10. Reddy, Y.V., M.S. Fox, K. Doyle, and J. Arnold, "INET: A Knowledge Based Simulation Model of a Corporate Distribution System", Proceedings of the IEEE Conference on Trends and Applications, Los Angeles, Ca., 1983, pp 109-118.

11. Young, R.E, $IDEF_1$: The ICAM Information Modeling Methodology, Prepared for Manufacturing Technology Division, 1983, Air Force Materials Laboratory, Texas A&M University, College Station, Tx.

12. Zeigler, B.P., Multifacetted Modelling and Discrete Event Simulation, 1984, Academy Press, N.Y.