자동화제조시스템의 객체지향적 모델링기법과 시뮬레이션구현방법: K-SIM의 사례*

이태억 · 이진규 · 임형규 · 이진환**

Object-Oriented Modeling and Simulation of Automated Manufacturing Systems : the K-SIM Case

Tae-Eog Lee, Jin-Kyu Lee, Hyeong-Kyu Lim, and Jin-Hwan Lee

(요 약)

물류시스템간의 상호작용이 복잡한 자동화제조시스템을 시뮬레이션하기 위해서는 전통적인 사건중심의 모델링이나 고객중심의 프로세스묘사방식 모델링보다 서버중심의 프로세스상호작용방식 모델링이 유리하다. 이러한 모델링방식은 기존의 동적시스템 모델링이론과도 일치하며 최근의 소프트웨어엔지니어링기법인 객체지향적 모델링기법과도 상통한다. 본 연구에서는 객체지향적 모델링기법을 바탕으로 서버중심의 프로세스상호작용방식에 기초한 시뮬레이션 모델링방법을 제안한다. 이러한 방법을 자동화제조시스템의 시뮬레이션모델링에 적용한 예를 소개하고, 순차적 처리언어인 C++를 사용하여 프로세스상호작용방식의 시뮬레이션 실행기를 구현하는 방법을 설명한다.

1. 서 론

컴퓨터시뮬레이션을 위한 모델링기법, 프로그래밍기법 및 소프트웨어엔지니어링기술, 그래픽애니매이션기술, 그래픽사용자인터페이스(GUI)기술 등은 최근 몇 년간 급속히 발전해왔다. 특히, 90년대에는 소프트웨어엔지니어링기법인 객체지향프로그래밍기법(Object-Oriented Programming (OOP))을 시뮬레이션에 적용하는 연구가 활발하게 이루어지고 있다([1], [4], [6], [7], [8], [9], [10], [22], [24], [26], [27]). 그러나, 대부분의 연구가 C++나 SMALLTALK 등과 같은 OOP언어를 활용하는 방법 등의 프로그래밍이슈에 집중되어 있다. 복잡한 대형 시스템을 시뮬레이션하기 위해서는 OOP와 같은 프로그래밍기법 이외에 대상시스템의 특성과 프로그래밍스타일에 부합되는 모델링방법론(Modeling Framework)이 중요하다.

생산시스템의 시뮬레이션에는 대기행렬네트워크 형태의 모델링에 적합한 프로세스중심적(Process-Oriented) 모델링 기법이 널리 사용되어 왔다. 이 모델링 방식은 대개 작업물(Job)과 같은 고객(Customer)이 거치는 일련의 서비스 과정(Process)을 묘사하고 기계장비 및 작업자 등 서버(Server)는 프로세스의 수행에 필요한 자원(Resource)으로 취급하여 고객중심 또는 프로세스묘사(Process-Description)방식으로 불리기도

* 이 연구는 통상산업부와 대우정보시스템(주)의 지원을 받아 이루어졌음 (G7 첨단생산시스템개발사업: 과제번호 5-1-1-4).

** 한국과학기술원 산업공학과

한다[10]. 이 방식은 가장 기초적(primitive)인 사건중심(Event-Oriented)의 모델링방식에 비해 융통성은 부족하나 모델링이 간편하다는 장점이 있다. 프로세스묘사방식을 지원하는 시뮬레이션언어로 대표적인 것은 SIMAN, GPSS, SIMSCRIPT 등이 있다. 그러나, 최근의 자동화제조시스템은 물류시스템간의 상호작용이 복잡하여 프로세스묘사방식으로 모델링하는데 한계가 있다. 일례로, SIMAN으로 자동화물류시스템을 모델링하기 위해서는 상당량의 FORTRAN 코드로 보완하는 것이 일반적이다. 자동화제조시스템을 구성하고 있는 주요 장비들의 다양한 운영방식 및 규칙을 모델링하기 위해서는 작업물과 같은 고객보다는 기계, 로보트, 물류장비 등 서버간의 상호작용에 촛점을 맞추는 서버중심의 모델링이 요구된다.

본 연구에서는 OOM을 바탕으로 서버중심의 프로세스상호작용방식의 시뮬레이션 모델링기법을 제시하고 자동화제조시스템의 시뮬레이션모델링에 적용한 예를 소개한다. 이를 위해 모델링 대상시스템을 동적시스템으로 파악하여 서브시스템의 네트워크 형태로 분할하고 각 서브시스템 또는 서버의 행태(Behavior)와 상호작용을 모델링하는 방식을 제안한다. 각 서브시스템은 OOM의 객체와 프로세스로 대응됨을 설명한다. 이러한 방법론을 G7첨단생산시스템개발사업의 세부연구과제로 개발중인 자동화제조시스템 설계용 시뮬레이터인 K-SIM의 개발에 적용한 예를 소개하고, 순차적 처리언어인 C++를 사용하여 프로세스 상호작용방식의 시뮬레이션 실행기를 구현하는 방법을 설명한다.

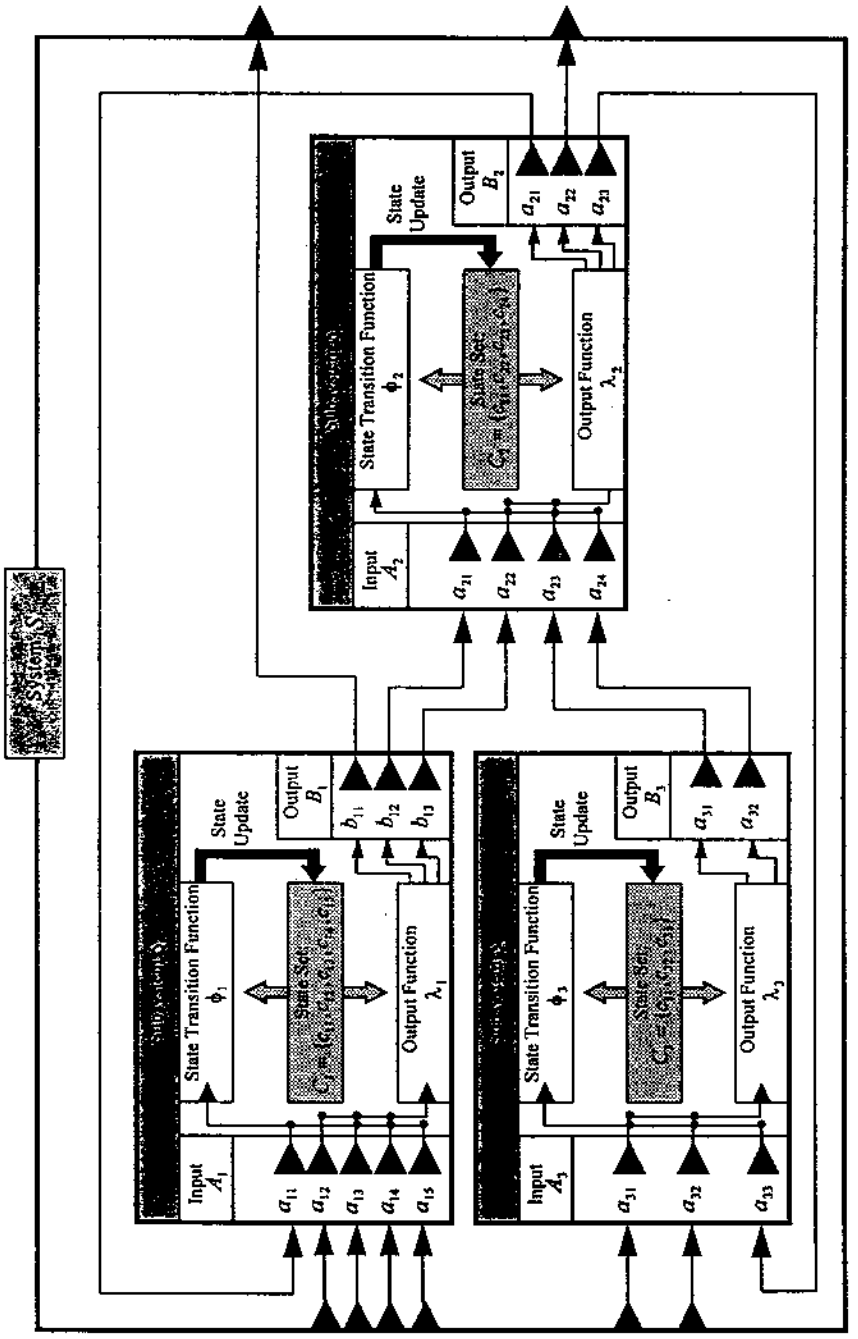
2. 객체지향형 시뮬레이션모델링 방법론

자동화제조시스템과 같이 이산적인 사건(Event)에 의해 상태가 불연속적으로 변화하는 이산사건형 시스템(Discrete Event Dynamic System)의 모델링 및 분석방법에 대한 연구는 활발히 진행되고 있다. Mesarovic과 Takahara[21], Ackoff[4] 등에 의해 연구된 General System Theory, Zeigler[28]의 DEVS(Discrete Event System Specification), Automata, Petri Net 등이 이산시스템의 모델링 및 분석에 널리 사용되고 있다. Mesarovic과 Takahara[21]는 미분방정식이나 전통적인 선형시스템(Linear System)과 같은 연속상태변화형 시스템 뿐만 아니라 이산사건형 시스템도 모델링할 수 있는 방법론으로, 집합이론에 기초하여 다음과 같은 일반형동적시스템모델(General System Specification (GSS))을 제안하였다.

$$\begin{aligned} S &= \langle A, B, C, \varphi, \lambda \rangle, \\ &\varphi: C \times A \rightarrow C, \\ &\lambda: C \times A \rightarrow B. \end{aligned}$$

여기서, A, B, C는 각각 입력, 출력, 시스템 상태를 나타내는 집합이며 φ와 λ는 각각 상태전이함수와 출력함수를 표시한다. 상태전이함수 φ는 시스템의 현재 상태에서 입력에 따라 다음 상태로의 시스템 자체적 변화를 나타내기 위한 것이고, 출력함수 λ는 시스템의 현재 상태에서 입력에 의한 시스템 외부로의 반응(출력)을 표현하는 것으로 이 두 함수는 시스템의 동적특성을 반영한다. 이들 함수는 각기 몇 개로 분할될 수도 있으며 시스템 특성과 모델링 목적에 따라 수식, Petri Net, Automata 등의 정식표현방법(Formal Specification)이나 Activity Cycle Diagram (ACD)[11], Block Diagram 등과 같은 비정식표현방법(Informal Specification)을 이용하여 구체적으로 정의될 수 있다. 경우에 따라서는 컴퓨터프로그램으로 직접 구현될 수도 있다. GSS와 같이 집합이론에 바탕을 둔 시스템 표현방법은 Automata[13], DEVS[28] 등의 이산시스템의 모델링방식에도 유사하게 활용되고 있다. 시스템 모델을 하위의 여러 개의 서브시스템으로 분할하여 각 서브시스템을 GSS로 표현하고 서브시스템간의 상호작용을 출력함수와 입력집합으로 모델링하여 서브시스템의 네트워크형태로 모델(〈그림 1〉 참조)을 구성할 수 있다([20], [28]). 이러한 다계층(Multi-Level) 또는 계층구조적(Hierarchical) 모델링방법은 이미 다양한 종류의 대형시스템을 모델링하는 데 널리 이용되고 있다(High-Level Net인 Colored Petri Net (CPN)[14], IDEF[18], 인간-기계시스템 모델링[22], DEVS[28] 등).

한편 최근에 급속히 보급되고 있는 OOM 또는 OOP는 프로그램을 여러 개의 객체단위로 분할하고 각 객체가 수행하는 행위와 상태변화는 객체내부에 정의되



〈그림 1〉 GSS에 의한 시스템모델링 개념도

는 메쏘드(Method)로 모델링하고, 객체간의 상호작용은 메세지교환(Message Passing)에 의해서만 이루어지도록 한다. OOM은 OOP를 지원하기 위한 소프트웨어모델링 또는 설계방법이다([20], [25]). 이 방법은 객체분할방법과 속성상속(Inheritance) 등의 정태적인 구조 및 상호관계를 명시하는 객체간관계(Object Relationship), 계층상속구조(Generalization Hierarchy), 객체구성관계(Composed-Of) 등을 정의하는 객체스키마(Object Schema)와 객체가 수행하는 오퍼레이션(또는 프로세스, 메쏘드)의 동태와 상호작용을 정의하는 Event Schema, Finite State Machine, Object-Flow Diagram 등의 Activity Schema(또는 Behavior Schema)로 이루어진다. 예를 들어 Martin과 Odell[20]이 제시한 Event Schema는 메쏘드를 사건과 오퍼레이션의 상호관계로 모델링한다. 이는 사건중심의 시뮬레이션모델링에 널리 활용되고 있는 Event Graph[26]와 유사하나 사건과 오퍼레이션의 순서가 반대이다. OOM/OOP는 GSS와 개념적으로 유사하여 주요 구성요소들을 다음 표와 같이 대응시킬 수 있다.

〈표 1〉 GSS와 객체지향프로그램과의 관계

GSS	OOM/OOP
System/Subsystem	Object
State	Data Structure/State Variables
State Transition Functions	Methods/Processes
Output Functions	Message Passing

시뮬레이션 대상인 자동화제조시스템은 동적시스템으로서 GSS로 개념적인 모델링이 가능하다. 더구나, 〈표 1〉과 같은 대응관계를 활용하면 OOM을 시뮬레이션모델링기법으로 활용할 수도 있다. Martin과 Odell[20], Rumbaugh 등[25]이 제시한 OOM은 정태적인 객체구조 및 관계의 정의 뿐만 아니라 Event Schema나 Finite State Machine 등을 활용하여 각 객체의 프로세스 또는 시스템 전체의 동태적 특성을 모델링하고 객체간의 상호관계는 Object Relationship Diagram이나 Object Flow Diagram 등으로 묘사하도록 권고하고 있다. 그러나, 이러한 모델링 방법은 서브시스템간의 입출력관계가 명확하지 않고 너무 일반화되어 있다. 실제 대부분의 "객체지향적 시뮬레이션" 연구([4], [6], [9], [22])는 OOM의 객체스키마를 이용하여 정태적 특성을 표현하는 데 치중하고 있다. 한편, 자동화제조시스템의 모델링에서는 서브시스템간의 입출력 관계, 동적인 상호작용을 명확히 정의하는 것이 핵심이며 이를 위한 체계화된 프로토콜이 필요하다.

본 연구에서는 자동화제조시스템의 모델링을 위해 OOM을 GSS의 개념을 바탕으로 서브시스템간의 상호작용을 명확히 모델링할 수 있도록 보완하고 상호작용을 체계적으로 표현하는 프로토콜을 정의하였다. 또한, 서브시스템을 시스템내의 가공, 운반 등의 물리적인 상태변화를 초래하는 자체 프로세스를 갖는 물리적 서브시스템(가공장비, AGV(Automatic Guided Vehicle), 로보트 등의 서버), 상태변화를 초래하는 프로세스를 갖지 않고 단순히 정보의 저장만을 위한 정보서브시스템(프로세스플랜, 공구 데이타베이스 등), 시스템의 상태에 따라 의사결정을 하고 운영방식 및 제어로직 등을 구현하는 제어서브시스템(AGV 컨트롤러, 시스템 컨트롤러)으로 구분하였다([5], [23]). 특히, 종래의 공장 모델링에서처럼 흔히 가공장비, 작업자 등의 물리적 서브시스템 등을 자원으로 취급하는 경향과는 달리 가급적 서버로서 모델링하였다. 이러한 모델링방식을 편의상 서버 및 객체지향적 시뮬레이션모델링방법론(Server and Object-Oriented Simulation Modeling Framework(SOOSM))이라고 부르기로 한다. SOOSM은 다음과 같이 요약된다.

(1) GSS와 같이 시스템을 서브시스템의 네트워크로 구성하여 각 서브시스템의 상태변수를 정의하고 서브시스템이 수행하는 프로세스를 파악한다. 한 서브시스템은 복수 개의 프로세스를 가질 수 있다. 각 서브시스템이 수행하는 프로세스간의 입출력관계 및 상호작용을 화살표를 이용하여 표시하고, 각 화살표 위에는 전달되는 메세지나 상호작용 내용을 표시한다. 각 서브시스템을 다시 하위의 서브시스템으로 재분할하는 다계층적 분할법은 경우에 따라 유용할 수 있으나 자동화시스템의 모델링에는 반드시 활용할 필요는 없다.

(2) 서브시스템의 상호작용을 표현하는 메세지는 체계화된 프로토콜을 사용하도록 권고한다. 자동화공장의 모델링을 위한 서브시스템간의 프로토콜은 서브시스템간에 정보를 전달(예, AGV 도착을 가공장비에게 알림)하는 정보전달(Inform), 다른 서브시스템에게 정보를 요청(예, 가공장비가 프로세스플랜에 작업물의 작업시간을 문의)하는 정보조회(Query), 다른 시스템에게 작업을 요청(예, AGV 컨트롤러가 AGV에 이동을 명령)하는 작업요청 및 명령(Ask), 시스템의 상태가 특정 조건이 만족될 때까지 또는 지정된 시간까지 프로세스 진행을 보류 또는 해제하는 대기(Wait) 및 허락(Trigger), 서브시스템간에 객체를 전달(예, AGV가 팔렛을 가공장비에 전달)하는 객체전달(Give) 등으로 정의해 볼 수 있다. 그러나, 모델러가 모델링단계에서 파악할 수 있는 수준에 따라 상호작용의 이해에 도움이 되는 수준으로만 모델링할 수도 있다. 동일 기계처럼 같은 서브시스템이 중복되는 경우에도 각각을 개별적으로 표시하되 중복표시를 적절하게 사용한다.

(3) 서버중심으로, 쉽게 이해되는 자연스러운 인식단위의 서브시스템으로 모델링한다[6]. 생산시스템의 예를 들어 기존의 프로세스묘사방식의 모델링기법처럼 작업자, 가공장비 등을 프로세스를 수행하는 데 필요한 부차적인 자원으로 취급하는 방식을 지양하고 서버로서 모델링하여 서브시스템으로 정의한다. 만약, 특정 장비가 사용하는 조립용 볼트나 부품과 같이 자원으로 모델링함이 불가피한 경우에는 해당 장비의 상태변수(예로 가용 부품수)로 모델링하거나 별도의 자원서브시스템(예로 가용부품재고시스템)으로 모델링한다.

(4) 가급적 물리적 서브시스템과 제어 서브시스템으로 구분한다. 이는 Job Scheduling, AGV의 Dispatching, Routing, Traffic Control, Robot의 Work Cycle 등의 운영방식 및 제어로직은 특정 서브시스템 단독보다는 여러 서브시스템간의 상호작용을 통합조정하는 경우가 많기 때문이다. 또한, 운영방식 및 제어로직을 수정, 유지보수하기가 간편하기 때문이다.

(5) 각 서브시스템의 프로세스는 필요 또는 편의에 따라 Petri Net, Automata, Event Graph 등의 정식표현방법(Formal Specification)이나 Event Schema, ACD, Block Diagram 등과 같은 비정식표현방법(Informal Specification)을 이용하여 모델링한다. 정식표현방법은 배우고 적용하는 데 노력이 필요하고 Formalism의 융통성 부족으로 모델링하기 어려운 상황이 발생할 수 있는 반면에 분석 및 코드생성 등이 가능하다. Event Schema는 OOM에서 제안된 다소 비정식적인 방법이나 프로세스 수행에 연관된 세부사건들의 상호관계를 파악하는 데 편리하다. SOOSM에서는 특정의 방법을 지정하지는 않지만 모델링 경험에 비추어 다소 비정식적인 Event Schema를 사용할 것을 권고한다. 특히, Event Graph, Petri net 등에 익숙하지 않은 모델러에게는 Event Schema가 적합한 것으로 생각된다.

(1), (2), (3), (4)에 의해 작성되는 모델을 "서브시스템구성도"라 부르고 (5)의 모델은 "프로세스개념도"라고 부른다. 이를 자동화제조시스템에 적용한 예는 다음 절에서 소개한다.

3. 자동화제조시스템의 SOOSM 모델링

K-SIM은 자동화공장의 설계용 시뮬레이터로서 GUI, 모델링라이브러리, 애니메이션 모듈 등으로 구성된다. 사용자가 GUI로 모델링라이브러리를 이용하여 대상시스템의 레이아웃을 구성하고 시스템 운영에 필요한 정보를 입력하면 시뮬레이션 모델이 완성될 수 있도록 설계되었다. 모델링이 명확하고 유지보수 및 확장이 용이하도록 SOOSM을 이용하여 모델링라이브러리를 설계하고 OOM/OOP를 이용하여 프로그램을 개발하였다. 또한, 개발된 모델링라이브러리를 활용하여 서브시스템간의 상호작용 프로토콜을 정의함으로써 새로운 공장 모델을 손쉽게 개발할 수 있다.

자동화제조시스템은 서버시스템(가공장비, 로보트, AGV, AGV Guide Path Segment 및 Control Point, Conveyor Splitter 및 Merger, Conveyor Segment, Loading 및 Unloading Station, Buffer, 작업자 등), 고객시스템(작업물, 팔렛, 공구 등), 정보시스템(Process

Plan, 공구데이타베이스 등), 제어시스템(시스템컨트롤러, AGV 컨트롤러 등), 자원서브시스템(부품, Jig, Fixture 등) 등으로 구분된다. 이들을 이용한 Mazak-Type FMS(Flexible Manufacturing System)의 서브시스템 구성도는 〈그림 2〉와 같다.

위 〈그림 2〉의 FMS 구성 요소 중 하나인 스태커크래인에 대한 프로세스개념도를 Event Schema를 이용하여 나타내면 〈그림 3〉과 같다. 스태커크래인은 〈그림 4〉의 K-SIM 모델링라이브러리의 Vehicle 클래스로부터 생성할 수 있으며, 스태커크래인 콘트롤러는 AGVCon으로부터 생성한다. 〈그림 2〉에서 처럼 스태커크래인은 스태커크래인 콘트롤러, 기계의 입출력 버퍼, 로딩/언로딩 스테이션과 상호작용을 한다. 스태커크래인 콘트롤러와 스태커크래인과의 상호작용은 〈그림 3〉에 명시되어 있으나, 입출력 버퍼 및 로딩/언로딩 스테이션과의 상호작용은 단순하게 입출력 리소스로 추상화되어 있다. 이는 스태커크래인의 프로세스 관점에서는 스태커크래인이 구체적으로 어떤 객체와 팔렛을 주고받는지는 중요하지 않기 때문이다.

SOOSM에 의한 모델링이 완료되면 이 모델을 바탕으로 OOM으로 프로그램을 설계한다. 즉, SOOSM 모델을 구현하기 위한 Object Schema를 구성하고 각 Object의 Behavior Schema를 작성한다. SOOSM에서 상태전이함수를 Event Schema로 이미 모델링한 경우에는 Behavior Schema로서 Event Schema를 재작성할 필요가 없다. 예로 K-SIM의 모델링 라이브러리를 구현하는 시뮬레이션 프로그램에 대한 OOM의 Object Schema를 도해하면 〈그림 4〉와 같다.

4. 프로세스상호작용방식에 의한 시뮬레이션실행기의 구현방법

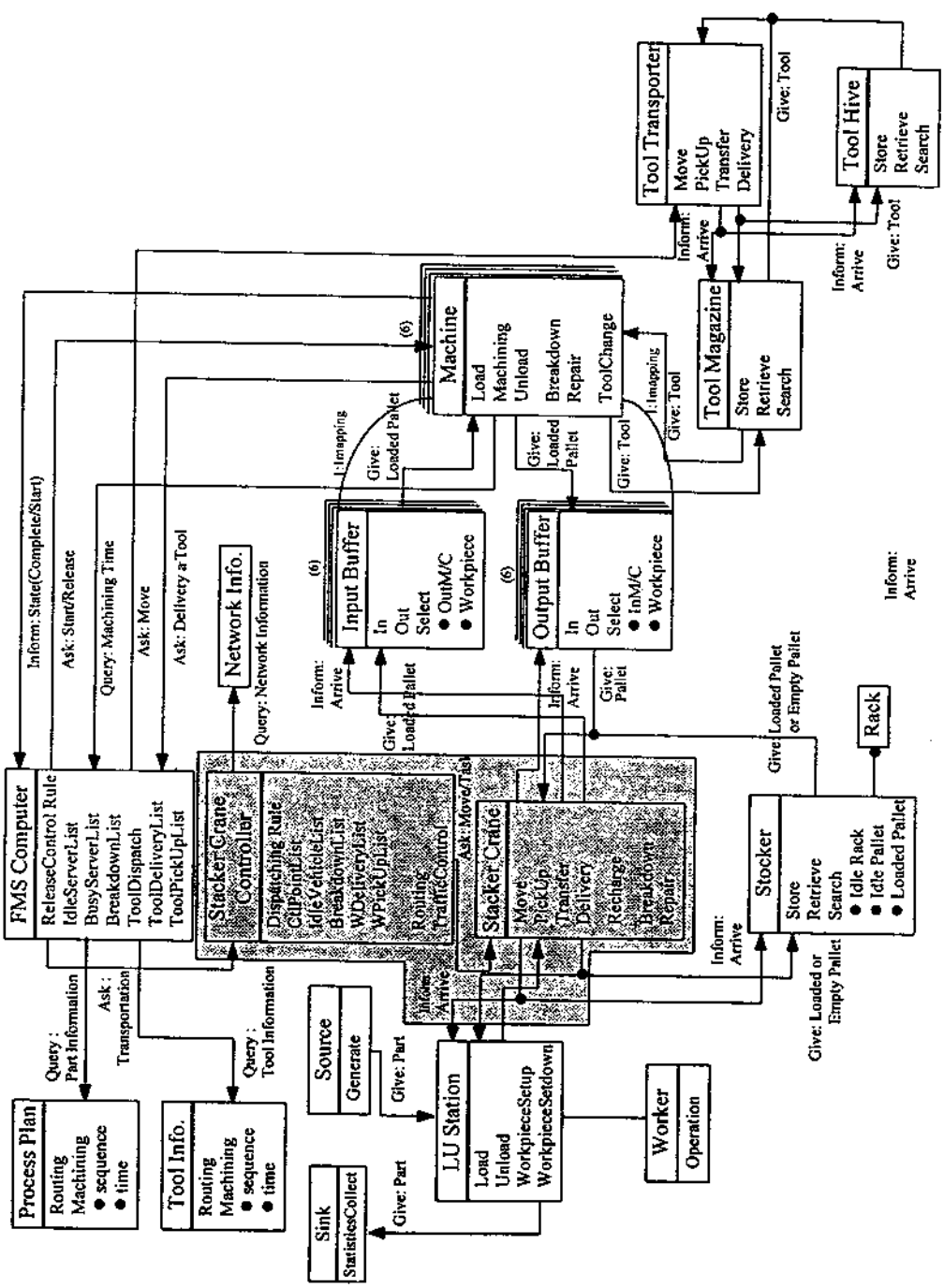
SOOSM에 의한 모델에서 각 서브시스템의 프로세스는 동시에 병렬적으로 진행될 수 있으며 프로세스간에는 상호정보 조회 및 전달, Request, 동기화(Synchronization) 등의 상호작용이 있다. 프로세스는 어떤 서브시스템이 수행하는 일련의 Activity와 이에 연관되어 생성되는 사건들로 구성된다. 예를 들어 머시닝센터가 터닝(Turning)작업, 공구교환, 보링 등의 작업을 반복하는 경우에 머시닝프로세스를 다음과 같이 도해해 볼 수 있다.

〈그림 5〉의 공구교환은 프로세스간 상호작용이 일어날 수 있는 작업이다. 즉, 터닝작업이 완료된 시점에서 보링작업에 필요한 공구가 가용하지 않거나 혹은, 터닝작업중 공구가 파손된 경우 머시닝센터는 공구관리 시스템에 공구운반을 요구한다. 머시닝센터로부터 공구운반 요청을 받은공구관리시스템은 공구이송장치에게 공구이송을 명령하고, 명령을 받은 공구이송장치는 적합한 공구를 운반하여 머시닝센터에 전달함으로써 머시닝프로세스가 원활히 진행되도록 한다. 이 예에서 알 수 있듯이 머시닝센터의 머시닝프로세스와 공구이송장치의 공구운반프로세스 사이에는 공구를 매개로 상호작용이 이루어진다. 특히 자동화제조시스템에서는 물류시스템과 제조시스템간의 이러한 상호작용이 빈번하게 이루어진다. 〈그림 6〉은 4 대의 기계와 2 대의 로보트로 구성된 작업장을 예로 들어 프로세스의 병존성 및 병존하는 프로세스간의 상호작용을 간트챠트를 이용하여 설명한 것이다. 4 대의 기계는 가공작업을 완료한 후 로보트에게 가공이 끝난 작업물의 언로드 및 (만약 가공할 작업물이 있다면) 작업물의 로드를 요청(Ask)한다. 로보트는 주어진 규칙에 의하여 기계의 요청에 따라 작업물의 로딩, 언로딩 또는, 언로딩/로딩 작업을 한다. 로보트가 기계에 이러한 작업을 하는 동안 기계는 수동적인 상태로 있으며, 작업이 끝난 로보트로부터 신호(Trigger)를 받아야 비로소 능동적인 상태가 되어 가공작업을 수행한다. 막대로 표시된 것은 각 객체의 프로세스이고, 프로세스 사이의 화살표가 로보트와 기계와의 상호작용을 나타내는 것이다.

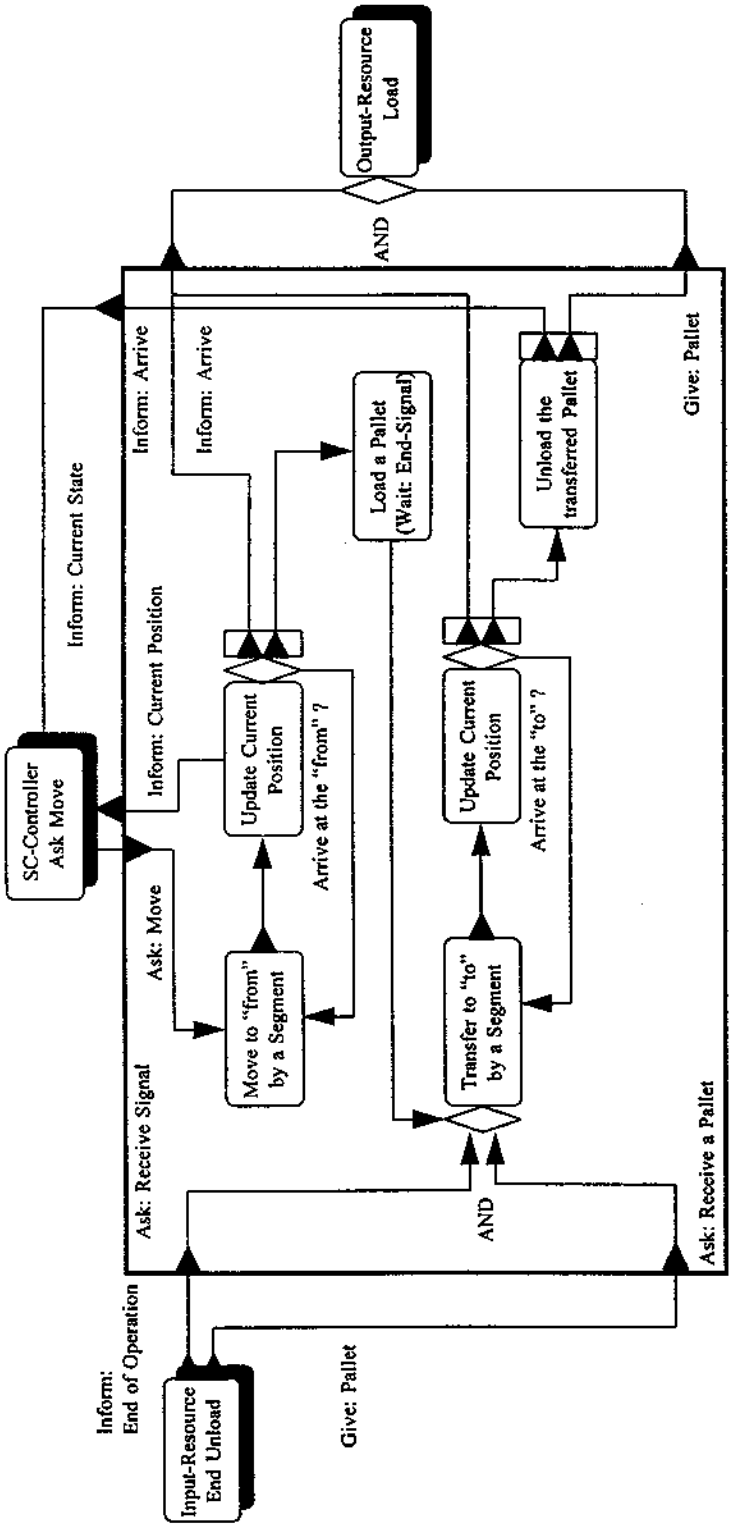
프로세스의 병존성과 상호작용을 구현하기 위해서는 다음과 같은 기능이 필요하다.

(1) 통신기능: 프로세스간에 정보요청, 전달, 프로세스의 수행요청 등을 Global Data Structure를 거치지 않고 프로세스간의 직접 통신으로 구현하는 기능.

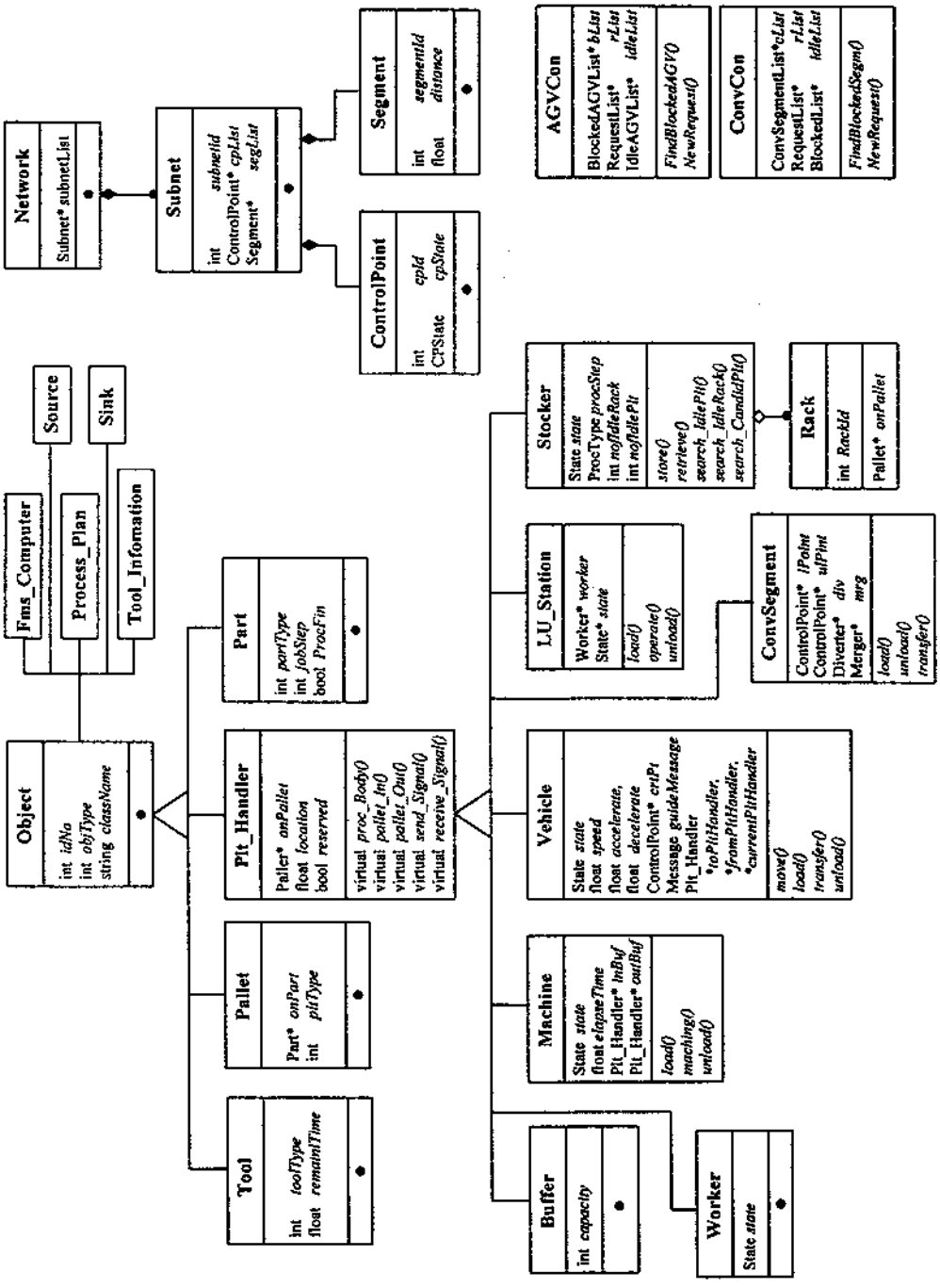
(2) 코루틴(Coroutine): 순차처리형 컴퓨터와 C++와 같은 순차처리형(Single-Threaded) 프로그래밍언어로 병렬 프로세스간의 상호작용을 구현하기 위한 프로그래밍기법([15], [16]).



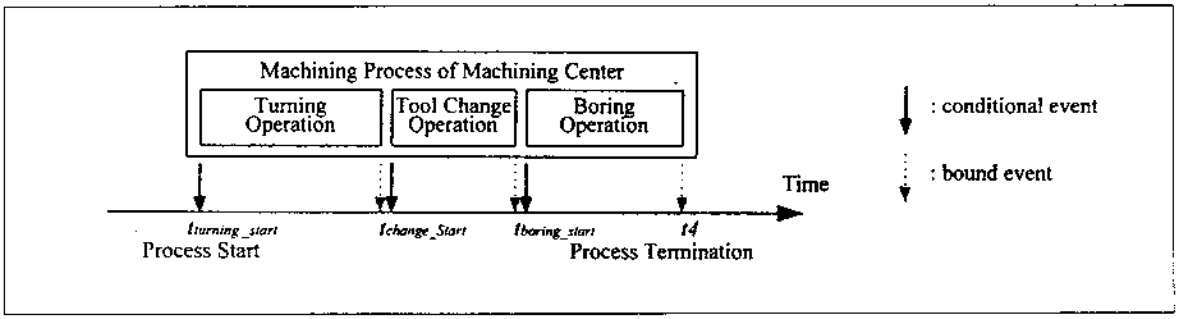
〈그림 2〉 Mazak-Type FMS의 서브시스템 구성도



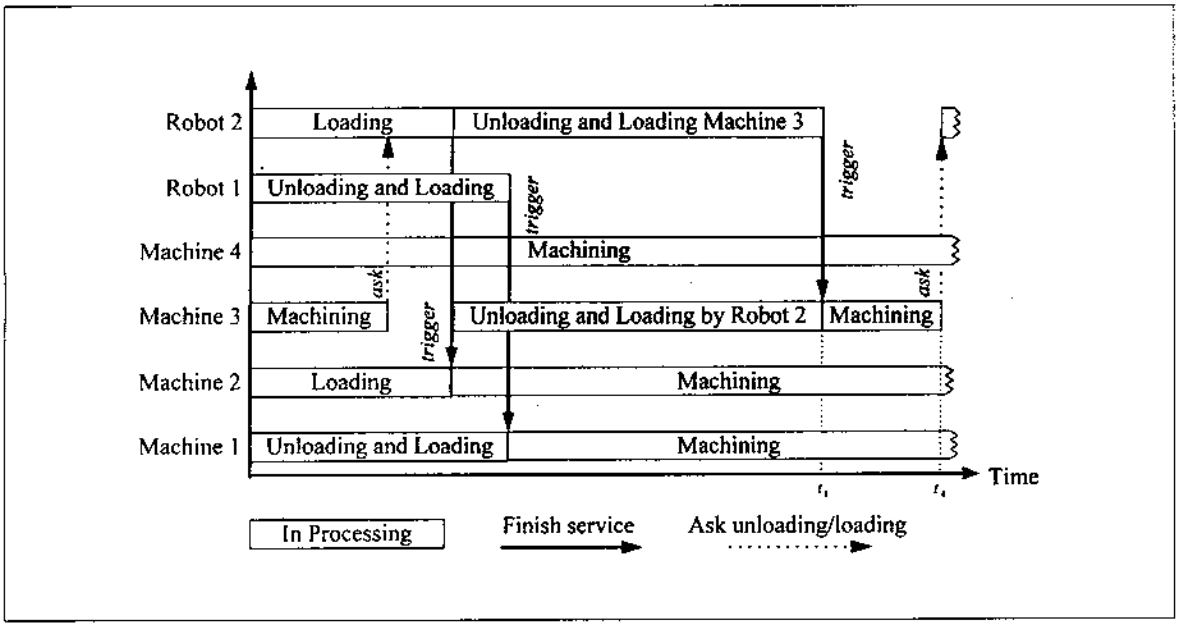
〈그림 3〉 스태커크레인 서브시스템의 프로세스개념도



〈그림 4〉 K-SIM 모델링 라이브러리의 Object Schema



〈그림 5〉 머시닝프로세스



〈그림 6〉 서버 프로세스의 병존성과 상호작용

C++는 메세지교환에 의한 프로세스(즉, 메쏘드) 간의 통신기능을 갖고 있으며 객체지향적 소프트웨어 엔지니어링을 위한 상속기능(Inheritance) 등을 갖추고 있으나 병렬프로세스간의 상호작용을 처리하기 위한 코루틴기능은 별도로 개발하여야 한다.

코루틴은 어떤 프로세스(또는 해당되는 프로그램 세그먼트)를 진행시키다가 다른 프로세스의 요청에 의해 또는 실행조건이 만족되지 않아서 프로그램실행권한을 다른 프로세스에게 넘겨주었다가 조건이 만족된 후에 정지된 지점부터 재실행(Resume)시키는 데 필요한 장치이다. 즉, 프로세스의 정지한 위치 및 재실행 점을 기억, 관리하는 기법이다. 코루틴은 각 프로세스를 구현하는 객체의 메쏘드 내부에 포함되어 프로세스의 코루틴간에 상호생성(Creation), 같은 계층의 코루틴간의 제어권한의 전이(Resume), 상하위 코루틴간의 제어권한전이(Call, Detach) 등을 구현하게 된다([13], [17], [19]). 코루틴구현은 대칭구조형과 비대칭구조형으로 구분된다. 비대칭구조에서는 코루틴간의 제어권한전이(상호작용)가 반드시 마스터코루틴(상위코루틴)을 통하여 이루어지도록 제약한다. 이에 반하여 대칭구조의 코루틴에서는 제어권한전이가 하위의 코루틴간에 자유롭게 이루어진다. 그러나 대칭구

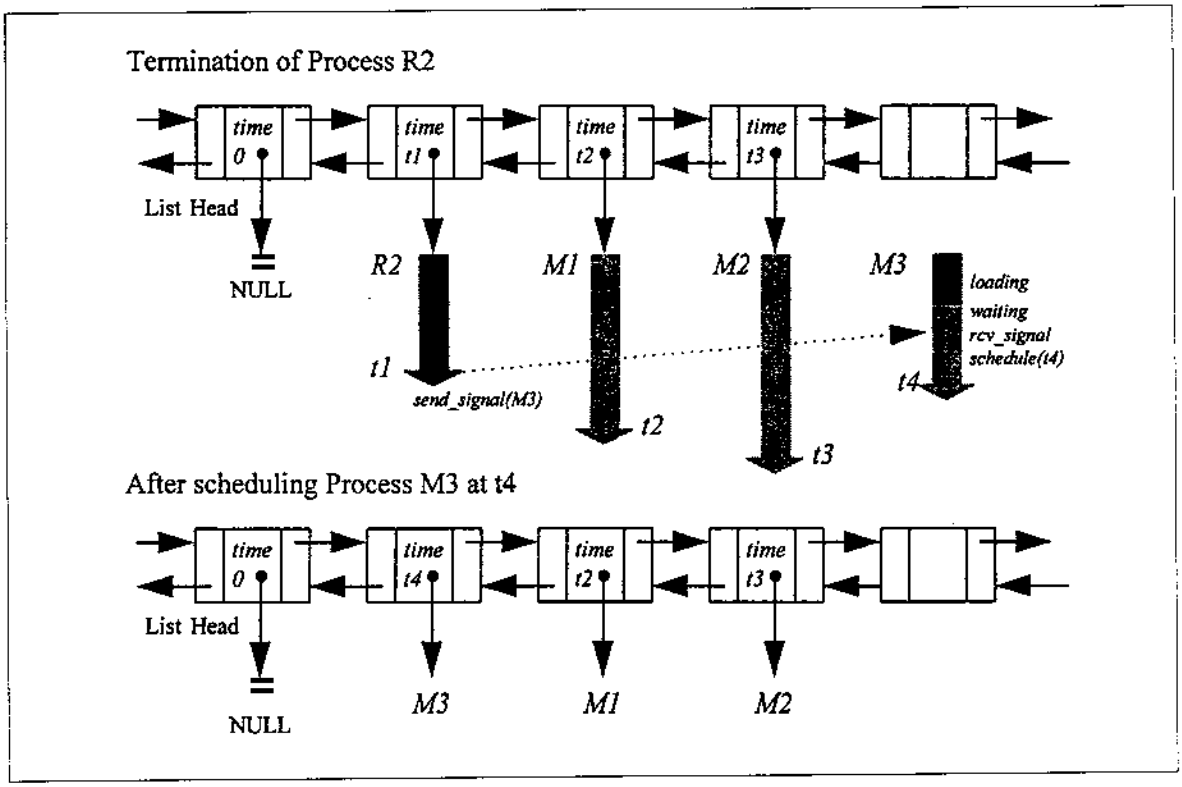
조의 코루틴은 실제 프로그램으로 구현하여 실행하는 데 있어 함수 Overflow를 초래할 수 있으므로, K-SIM의 시뮬레이션엔진은 각 객체의 프로세스를 통합관리하는 칼렌더(Calendar)를 마스터코루틴으로 하는 비대칭구조형으로 개발하였다.

칼렌더는 프로세스가 Activation되도록 스케쥴링된 객체들의 리스트를 유지관리한다. 칼렌더는 다양한 객체를 Activation 예정시점순서로 관리할 수 있는 순환이중연결리스트로 구현되었다. 일반적으로 칼렌더의 리스트 처리속도가 시뮬레이션프로그램의 효율을 크게 좌우한다. 리스트가 커질수록 오름차순 정렬 등의 처리속도가 급속히 늦어지므로 스케쥴링되는 객체 및 사건들의 상호연관성에 따라 칼렌더를 분할하여 운영하고, 분할된 칼렌더들은 필요시 상호조정되도록 하는 것이 모델이 대형화될수록 유리할 것으로 판단된다. 따라서, K-SIM에서는 시뮬레이션 처리속도를 높이기 위해 3가지 종류의 칼렌더를 사용하였다. 첫째는 서버(Pallet-Handler, AMS내의 작업물들은 팔렛에 실려 운송, 가공, 보관된다는 의미로 명명)들의 프로세스를 스케쥴하는 데 사용되는 칼렌더이다. 두번째는 대상시스템 외부로부터 내부로 작업물을 공급하는 Source라는 객체를 스케쥴하기 위한 칼렌더이다. Source는 작업물의 종류, 도착형태, 뱃치크기를 자유롭게 정의할 수 있도록 설계되었다. 다른 하나의 칼렌더는 서버들의 고장을 스케쥴하기 위한 것이다. 이를 따로 분리한 이유는 고장발생은 다른 사건에 비하여 아주 드문 사건(Rare Event)임에도 불구하고 항상 스케쥴되어 있어야 하므로 리스트 처리속도에 영향을 줄 수 있기 때문이다.

칼렌더를 구성하는 주요한 메쏘드는 다음과 같다([2], [3], [16]). 첫째는 칼렌더의 리스트에 언제, 어떤 객체에 제어권한을 전이할 것인가를 기록하는 Schedule 메쏘드이다. 스케쥴된 각 객체의 프로세스의 재실행점은 프로세스 자신이 기억하고 있다. 둘째는 스케쥴된 객체의 프로세스에 제어를 전이하는 역할을 담당하는 Run 메쏘드이다. Run에 의하여 제어권한을 넘겨받은 객체는 자신의 상태를 갱신(Update)한 후, 다음에 발생할 확정된 사건이 있으면 이를 칼렌더에 기록하고, 그렇지않으면 제어권한을 그대로 칼렌더에 반납한다. 셋째는 칼렌더의 리스트에 스케쥴된 내용을 리스트로부터 제거하는 Cancel 메쏘드이다. Cancel은 서버가 고장나거나 공구 등의 파손에 의하여 처음에 계획된대로 작업이 완료될 수 없을 때 사용된다.

칼렌더의 기능 및 상하위 코루틴간의 제어전이를 통한 시뮬레이션 진행 예를 설명하기 위하여 〈그림 6〉의 2 대의 로봇과 4 대의 기계가 함께 작업하는 경우를 예를 들어 설명한다. 〈그림 7〉의 순환이중연결리스트는 〈그림 6〉을 구성하는 객체들을 Activation 시간순서의 오름차순으로 저장하기 위한 것이다. 초기상태는 로봇 2, 기계 1, 기계 2의 순서로 해당 프로세스를 수행하도록 스케쥴되어 있다. 아래 방향으로 향하고 있는 음영된 화살표는 〈그림 6〉에서 설명한 각 객체들의 프로세스를 의미한다. 화살표 머리 부분은 현재 스케쥴된 객체의 프로세스 중 앞으로 일어날 구체적인 사건을 의미하며, t_i는 그 사건이 일어날 시간을 의미한다. 작업이 끝난 하위의 코루틴(또는 상위의 다른 코루틴일 수도 있음)이 제어권한을 칼렌더(마스터코루틴)에 반납하면 칼렌더는 시간순으로 정렬된 객체의 프로세스 중 가장 앞의 객체에 제어권한을 넘겨주고 제어권한을 넘겨받은 객체를 리스트에서 제거한다. 현재의 상태에서 제어권한은 로봇 2가 받게 된다. 제어권한을 받은 로봇 2는 언로딩/로딩 작업이 완료되었음을 알리는 신호(점선으로 표시된 화살표)를 기계 3에게 보내게 된다. 로봇 2로부터 작업완료 신호를 받은 기계 3은 머시닝 프로세스가 완료되는 시점 t_4를 칼렌더에 스케쥴하고 제어권한을 반납한다(기계 3이 칼렌더에 머시닝 프로세스 완료시점을 스케쥴한 순간부터 기계 3의 머시닝 프로세스는 진행중인 것으로 간주됨). 이 때 칼렌더는 t_4를 현재 t_2, t_3와 비교하여 리스트에 오름차순으로 저장한다. 스케쥴이 완료된 후의 리스트의 모습은 〈그림 7〉의 아래 부분과 같다.

실행기(Executer)는 시뮬레이션모델을 구성하는 객체의 프로세스 실행과 연관된 모든 사건이 올바른 순서로 처리되도록 관리한다. 실행기의 첫째 기능은 세 종류의 칼렌더 중에서 프로세스 발생 예정시간이 가장 빠른 사건이 속한 칼렌더를 찾아내는 것이다. 둘째는, 시뮬레이션시간의 진행이다. 이는 위에서 선택



〈그림 7〉 칼렌더의 구조 및 변화 예

된 칼렌더에서 첫번째로 스케줄링된 사건발생 예정시간으로 시뮬레이션 시간을 갱신함으로써 이루어진다. 마지막으로, 선정된 객체의 해당 프로세스를 실행되도록 제어권한을 넘겨줌으로써 이와 연관된 다른 사건이 연속적으로 발생하여 시스템의 상태가 갱신되도록 한다.

5. 결론

물류시스템의 상호작용이 복잡한 자동화제조시스템의 시뮬레이션모델링에 서버를 중심으로 모델링하는 기법과 프로세스상호작용방식의 시뮬레이션진행방법을 적용하는 방법을 제시하였다. 소프트웨어엔지니어링방법론으로 제시된 OOM방법론을 동적시스템모델인 GSS의 서브시스템네트워크의 개념으로 보완하여 서버 중심의 시뮬레이션모델링방법론인 SOOSM을 제시하고 자동화제조시스템의 설계용 시뮬레이터의 개발에 적용한 예를 소개하였다. SOOSM으로 정의된 모델을 시뮬레이션 프로그램으로 구현하기 위해 프로세스상호작용방식의 시뮬레이션진행기를 C++로 개발하는 방법을 제시하였다.

SOOSM은 아직 보완되어야 할 점이 많이 있으나 자동화공장의 객체지향적 시뮬레이션 모델링을 위한 방법론으로 발전시킬 예정이다. SOOSM, OOM 및 OOP에 의해 개발된 모델링라이브러리에 기반한 K-SIM은 상용화가 추진중이고 유지보수 및 확장성이 좋아 경쟁력을 갖게 될 것으로 기대한다.

【참고문헌】

[1] 김경섭, "객체지향 시뮬레이터, AgvTalk의 가능성", 한국시뮬레이션 학회 논문지, 제3권, 제1호, pp. 1-16, 1994.7.

[2] 이진규, 프로세스상호작용 방식의 자동화제조시

스템 시뮬레이터, 한국과학기술원 산업공학과 석사논문, 1995.

[3] 이태억, 임형규, 이진규, "프로세스상호작용에 의한 객체지향형 공장설계용 시뮬레이션 엔진의 개발", 제 2회 G7 첨단생산시스템 Workshop, pp. 65-68, 1994.

[4] Ackoff, R. L. and Emery, F. E., On Purposeful Systems, Aldine Atherton, 1972.

[5] Adiga, S., Object-oriented Software for Manufacturing System, Chapman & Hall, 1993.

[6] Ball, P. and Love, D., "Expanding the Capabilities of Manufacturing Simulators Through Application of Object-Oriented Principles", Journal of Manufacturing Systems, Vol. 13, No. 6, pp. 412-423

[7] Bischak, D. P. and Roberts, S. D., "Object-Oriented Simulation", Proceedings of the 1991 Winter Simulation Conference, pp. 194-204, 1991.

[8] Borgen, E. and Strandhagena, J. O., "An Object-Oriented Tool Based on Discrete Event Simulation for Analysis and Design of Manufacturing Systems", Optimization of Manufacturing Systems Design, pp. 195-220, 1990.

[9] Davies, R. M and O'Keefe, R. M., Simulation Modeling with Pascal, Prentice Hall International, 1989.

[10] Fishwick, P. A., "SIMPACK: Getting Started with Simulation Programming in C and C++", Proceedings of the 1992 Winter Simulation Conference, pp. 154-162, 1992.

[11] Hlupic, V. and Paul, R. J., "Simulation Modeling of Flexible Manufacturing Systems Using Activity Cycle Diagrams", Journal of the Operational Research Society, Vol. 45, No. 9, pp. 1011-1022, 1994.

[12] Hopcroft, J. W. and Ullman, J. D., Introduction to Automata Theory, Languages, and Computation

[13] Horton, I., A. and Turner, S., J., "Using Coroutines in Pascal", Software-Practice and Experience, Vol. 16, No. 1, pp. 45-61, 1986.

[14] Jensen, K and Rosenberg, G. (Eds.), High-level Petri Nets Theory and Application, Springer-Verlag, 1991.

[15] Kreutzer, W., System Simulation Programming Styles and Languages, Addison-Wesley, Reading, MA 1986.

[16] Lee, J. K. and Lee, T.E., "A Process-Interaction Based Simulator for Automated Manufacturing Systems", 한국과학기술원 산업공학과 IE 연구교류 세미나 논문집, 제 1권, pp. 93-105, 1995. 3.

[17] Malloy, B. and Soffa, M. L., "Conversion of Simulation Process to Pascal Constructs", Software-Practice and Experience, Vol. 20, No. 2, pp. 191-207, 1990.

[18] Mandel, "Graphical Process Description-Views and Diagrams", Int. J. of Computer Integration Modeling, Vol.3, No. 5, pp. 314-327, 1990.

[19] Marlin, C. D., Coroutines, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1980.

[20] Martin, J. and Odell, J. J., Object-Oriented Analysis & Design, Prentice Hall, pp. 67-119, 1992.

[21] Mesarovic, M. D. and Takahara, Y., General Systems Theory: Mathematical Foundations, Academic Press, 1975.

[22] Miller, R. A., "A Systems Approach To Modeling Discrete Control Performance", Advances in Man-Machine Systems Research, Vol. 2, pp. 177-248, 1985.

[23] Mize, J. H., Bhuskute, H. C., PRATT, D. B. and KAMATH, M., "Modeling of Integrated Manufacturing Systems Using Object-Oriented Approach", IIE Transactions, Vol. 4, No. 3, pp. 14-26, 1992.

[24] Pidd, M., "Object Orientation & Three Phase Simulation", Proceedings of the 1992 Winter Simulation Conference, pp. 689-693, 1992.

[25] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorenson, W., Object-oriented Modeling and Design, Prentice-Hall, 1991.

[26] Sargent, R. G., "Event Graph Modeling for Simulation with an Application to Flexible Manufacturing Systems", Management Science, Vol. 34, No. 10, pp. 1231-1251, 1988.

[27] Shewchuk, J. P., "An Approach To Object-oriented Discrete Event Simulation of Manufacturing Systems", Proceedings of the 1991 Winter Simulation Conference, pp. 302-311, 1991.

[28] Zeigler B. P., Theory of Modeling and Simulation, Robert E. Krieger Publishing Company, 1976.

이태억
1982 한국과학기술원 산업공학 석사
1982-1986 대우조선
1991 미국 Ohio State University 산업공학 박사
현재 한국과학기술원 산업공학과 조교수
관심분야: 확률 및 이산사건시스템 모델링, Periodic Scheduling

이진규
1995 한국과학기술원 산업공학 석사
현재 한국과학기술원 산업공학과 박사과정
관심분야: 제조시스템 모델링

임형규
1993 한국과학기술원 산업공학 석사
현재 한국과학기술원 산업공학과 박사과정
관심분야: 제조시스템 모델링 및 시뮬레이션

이진환
현재 한국과학기술원 산업공학과 석사과정
관심분야: 제조시스템 모델링