CIM 환경에서 프로세스 설계를 위한 시뮬레이션 코드의 자동 생성에 관한 연구

A Study on the Generation of Simulation Codes for Process Design under CIM Environments

박찬권* · 김기태** · 장성용*** · 박진우**
Chankwon Park · Kitae Kim · Seongyong Jang · Jinwoo Park

Abstract

This study deals with an autonomous generation of simulation codes for the design of processes of a CIM system based on the concept of model transformation. We assume we have a functional model of an organization which is developed using the well-known IDEF0 methodology. Then, a modeling methodology, called PROWD(PRocess flOW Description), is suggested to develop the flow processes of the organization. To derive simulation specifications for the transformed PROWD model, we define a module of a "unit system" for each activity of PROWD model. Then specifications for system entities and resources are derived from input/control/mechanism/output and entity flows of PROWD model. Entitiy queues are defined from the states and events of unit systems. Finally, SIMAN model frame is generated from those specifications through a suggested algorithm. The implementability and validity of the proposed approach is tested by developing a prototype of a computer-assisted design system on the operation processes of FMS installed at SNU-ASRI (Seoul National University-Automation and Systems Research Institute).

1. 서 론

시뮬레이션 기법의 유용성은 현실 세계의 문제에 대해서 해법을 실제로 구현하기 이전에 시스템의 동작 행태에 대한 예측과 통찰력을 확보할 수 있다는 데 있다. 뿐만 아니라 대상 시스템이나 문제 자체에 대한 명확한 관점을 제공하기도 한다. 한편 보다 향상된 컴퓨터 기능과 소프트웨어 기술은 시스템의 분석과 설계

* 영산국제산업대학교 경영정보학과
** 서울대학교 산업공학과
*** 서울산업대학교 산업공학과

과정에 보다 강력하고 사용이 편리한 시뮬레이션 패키지의 사용을 가능하게 하였다. 따라서 앞으로의 시뮬레이션 연구는 컴퓨터의 성능이나 프로그램의 구현 가능성에 구애받지 않고 수행할 수 있게 된 만큼, 문제의 체계적 표현이 점점 더 중요한 요소로 대두되게 되었다. 컴퓨터 통합 생산(CIM) 시스템과 같이 복잡하고 다양한 구성 요소를 가지고, 여러 수명주기를 거쳐서 구현되는 시스템은 시뮬레이션 모형을 생성시키는 과정에 많은 시간과 노력이 소요되기 때문에 보다 체계적인 모형 작성 방법이 요구된다. 특히, 여러 개발자가 관여하는 환경에서, 모든 개발자들이 시뮬레이션에 대한 전문지식을 가지고 있지 않은 경우에 이러한 접근이 필수적이다.

본 연구는 CIM 시스템의 초기 사용자 요구 정의 및 개념 설계 단계의 모형화에 포괄적으로 사용되는 IDEF0 기능 모형([4], [5], [6])을 기본으로 하여 프로세스 흐름 모형(process flow model)을 도출하고, 이를 이용해서 시뮬레이션 사양과 시뮬레이션 코드를 자동 생성하는 것에 관한 것이다.

다음 2장에서는 시뮬레이션 코드의 자동 생성과 관련된 연구 현황을 살펴보고, 3장에서는 본 연구에서 사용하는 프로세스 흐름 모형화 방법 및 기능 모형으로부터의 변환 방법에 대해 간단히 살펴본다. 4장에서는 본 연구에서 제안하는 PROWD(PRocess flOW Description) 모형으로부터 시뮬레이션 코드를 생성시키는 방법에 대해 설명하고, 5장에서는 이 같은 과정을 지원하는 시스템을 구현한 내용이 소개된다.

2. 관련 연구 현황

시뮬레이션 코드의 자동 생성과 관련해서, Mathewson[10]은 "상대적으로 일반적인 심볼로 표현되는 하나의 모형을 컴퓨터 코드의 형태로 변환시키는 소프트웨어 도구"라고 정의하고 있다. 그리고 이 때 사용될 수 있는 시뮬레이션 언어의 예로서 GPSS와 SIMAN을 들었다. Mathewson이 정의한 것 처럼, 사용자 입력을 최소화하면서 자동적으로 시뮬레이션 프로그램을 생성하고자 하는 노력들은 다양한 형태로 시도되어 왔다.

시뮬레이션 모형은 다양한 형태의 시스템에 적용될 수 있어야 한다. 이러한 관점에서 기존의 다양한 시뮬레이션 전용의 상용 소프트웨어들은 기능 모형을 중심으로 발전하여 왔다. 또한 전문 시뮬레이션 분석자와 실제 시스템에 대한 전문가가 공동으로 작업을 수행하는 것이 전형적인 접근방법이었다. 그러나 이런 과정은 많은 시간과 노력이 필요하다. 그러므로 많은 시뮬레이션 소프트웨어는 사용자가 최대한 쉬운 방법으로 시뮬레이션 모형을 프로그램할 수 있도록 개발되어왔다. 이런 프로그램 방법들에 대한 연구를 프로그래밍 과정을 기준으로 나누어 보면 크게 세 가지로 나누어 볼 수 있다.

제일 먼저 아이콘 기반형 시뮬레이션 프로그램 개발 방법이 있다. 이런 프로그램 개발 방법에 의하여 Pritsker사에서는 SLAM을 기반으로 하는 AIM을 개발했으며, System Modelling Corporation에서는 SIMAN을 기반으로 하는 ARENA를 개발하였다. 그외 다른 상용 시뮬레이션 소프트웨어들도 다양한 모양의 아이콘을 이용하는 시뮬레이션 프로그램으로 개발되고 있으며 점점 더 개량되고 있다. 그러나 이들 프로그램들은 그 프로그램의 사용에 대한 전문 지식이 반드시 필요하다는 단점이 있다.

두번째로 Data-Driven Simulator 방법론이 있다. Haddock[7]은 FMC(Flexible Manufacturing Cell)의 시뮬레이션을 위하여, 기존 시뮬레이션 코드를 사용자가 입력하는 몇몇의 변수에 의해 수정하는 방법을 통하여 다양한 시스템에 대한 시뮬레이션 모형을 작성하는 의사결정시스템을 제안하였다. 또한 '신호철'[3]은 전문가 시스템을 이용함으로써 간단한 사용자 입력에 의해 약국 시스템에 대한 시뮬레이션 모형을 구축하는 시스템을 제안하였다. 그리고 '류시각'[1]은 FMS의 경제성 평가를 위해 사용자가 입력한 매개 변수를 토대로 시뮬레이션 코드를 생성시키는 프로그램을 제안하기도 하였다. 시뮬레이션 코드 생성을 위한 이들 접근 방법들은 대부분 대상 시스템에 대해 기본적인 시뮬레이션 코드를 작성해두고, 사용자가 입력하는 특정 매개 변수들을 이용해서 시뮬레이션 코드를 조금씩 수정해 나가는 방법을 사용하고 있다. 그러므로 이 경우 특정 형태의 시스템에 대해 기본적인 시뮬레이션 사양이 미리 정의되어 있어야 하고, 또한 생성된 시뮬레이션 코드가 해당 시스템에만 국한된다는 제약이 있다. 이와 같은 접근 방법을 위하여, Pidd[12]는 특정 영역을 대상

으로 하는 data driven generic simulator의 개발 지침을 제시하기도 하였다.

세번째로 시스템 모델링 도구를 이용하여 시뮬레이션 모형을 개발하는 방법이 있다. Rensburg[13]는 시스템 설계 및 분석 단계에서 IDEF0 모형을 작성한 후, IDEF0 모형의 기본 모형화 도구를 기본적인 SIMAN 블록에 대응시키는 방법으로 시뮬레이션 모형을 작성하는 방법을 제안하였다. 그러나 이 방법은 개념 설계 단계에서 작성된 추상적 형태의 IDEF0 모형으로부터 직접 SIMAN 코드를 생성하고 있기 때문에, 작성된 시뮬레이션 모형이 시스템의 동작 행태를 반영하는 데는 제약이 있다. 따라서 시스템 동작행태를 파악하기 위해 IDEF3 프로세스 흐름 모형을 별도로 작성하고 있다.

본 연구는 기존의 시뮬레이션 모형 개발 방법론중에서 시스템 모델링 도구를 이용하여 시스템 모형을 작성함과 동시에 시뮬레이션 모형을 개발하는 방법에 대한 연구이다.

3. 프로세스 흐름 모형

3.1 PROWD 모형의 개요

IDEF 방법론은 그 효용성과 대중성 때문에 시스템을 설계하고 분석하는 데 사용될 수 있는 도구들 가운데 가장 널리 이용되는 도구이다. 그 중 IDEF0 기능 모형은 시스템의 초기 개념 설계 단계에 있어 사용자의 요구를 정의하는 데 유용하게 적용될 수 있는 모형화 방법으로 널리 알려져 있다. 제약 조건 아래서 입력을 출력으로 변환시키는 활동을 묘사하는 데 사용되면서 일반적인 시스템의 활동을 표현하고 점차 자세하게 기능을 분해하는 계층적 구조를 가진다[11]. 하지만 IDEF0 기능 모형으로부터 직접 시뮬레이션 사양을 도출하기에는 IDEF0 기능 모형에서 제공하는 정보가 너무도 추상적이고 제한적이다. 예컨데, 기능 또는 활동의 시간적 순차 관계나 논리적인 연관 관계는 표현되지 않는다. 따라서 시뮬레이션 코드를 자동적으로 생

〈표 1〉 PROWD 모형의 모형화 도구(modeling constructs)

모형화 도구		의 미	그래픽 표현	속 성
기능(function)		기능/활동을 표현하기 위한 기본 모형화 도구로서, 기능(function), 활동(activity), 운용(operation), 행위(action) 등을 표현.	· box(단일실선) (입력/제어/ 메카니즘/출력)	{identifier, description, input, control, mechanism, output, layer, input/control logic, output logic}
단위 기능 (Unit Function)		별도의 변환 과정 없이 입력은 입력으로, 제어는 제어로 연결	· box(이중실선)	{identifier, input, control, output}
연결 (link)	입력	기능/활동의 수행 대상이 되는 입력의 제공을 의미	활동을 나타내는 박스의 왼쪽에 연결	{identifier, description, triggering type, start, end}
	제어	기능/활동의 수행에 대한 제약 또는 제어를 의미	활동을 나타내는 박스의 위쪽에 연결	
	메커니즘	기능/활동을 직접 수행하거나 수행을 지원하는 것을 의미	활동을 나타내는 박스의 아래쪽에 연결	
논리적 접속 (junction)	AND	분기 또는 결합되는 연결에 해당되는 입력 또는 제어가 모두 제공되어야 함을 의미	작은 사각형 안에 '&'	{identifier, type, description, rule}
	OR	분기 또는 결합되는 연결에 해당되는 입력 또는 제어 중 일부가 제공되어야 함을 의미	작은 사각형 안에 'O'	
	XOR	분기 또는 결합되는 연결에 해당되는 입력이나 제어 중 오직 하나만이 제공되어야 함을 의미	작은 사각형 안에 'X'	

성시키기 위해서는 상위 수준의 IDEF0 기능 모형을 입력 받아서 이들 정보가 반영된 프로세스 흐름 모형으로 변환한 다음, 프로세스 흐름 모형으로부터 시뮬레이션 사양을 작성하는 방법을 이용한다. 본 연구에서는 실체 흐름을 기반으로 프로세스 흐름을 모형화하는 PROWD(PRocess flOW Description) 모형[2]을 사용한다.

프로세스 흐름 모형화 방법론 가운데 가장 많이 알려진 것 중 하나가 IDEF3 방법론인데, IDEF3는 IDEF0 기능 모형과의 연계성이 미흡해서 모형을 별도로 작성해야 하는데 반해, 실체 흐름을 기반으로 한 PROWD 모형은 IDEF0 기능 모형으로부터 변환이 용이하다는 장점이 있다. PROWD 모형은 IDEF0 모형의 계층적 분해에 의한 시스템 분석을 지원하는 한편, 기능/활동 사이의 동기 관계(synchronization relationship)와 순차 관계를 보다 다양하게 표현할 수 있는 능력을 제공하고 있다. PROWD 모형에서 정의하고 있는 모형화 도구(modeling constructs)와 주요 모형화 도구에 대한 데이타 구조는 각각 〈표 1〉, 〈표 2〉와 같다.

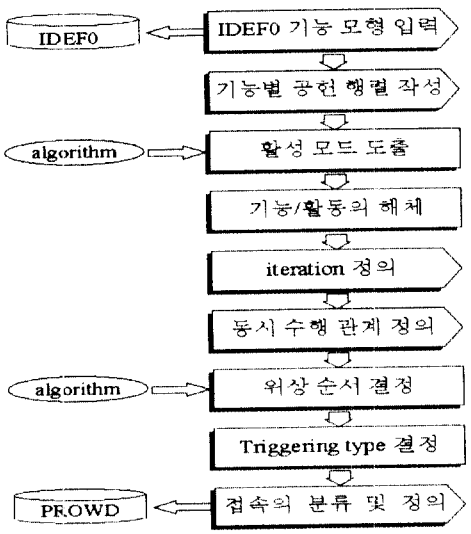
〈표 2〉 PROWD 모형의 주요 모형화 도구에 대한 데이타 구조

FUNCTION Template			비 고
identifier		Axx	기능의 노드 identifier를 사용
description		textual description	
input		Ix	
control		Cx	
input/control logic		AND | OR | XOR	입력과 제어 사이의 논리
mechanism		Mx | Vx	
output		Ox	
output logic		AND | OR | XOR	여러 출력 사이의 논리
layer		n	위상 순서
LINK Template			**비 고**
identifier		Lx	link에 identifier를 부여
description		textual description	
triggering type		TE | NTE	격발 또는 비격발 여부
tail	function	SYS | Axx[/Jn]	연결의 시작(시스템 외부 또는 기능 Axx) 접속을 통할 경우는 [/Jn]옵션
	type	NULL | Output	시스템 외부일 경우는 NULL
	identifier	NULL | Ox	시스템 외부일 경우는 NULL
head	function	SYS | [Jn/]Axx	연결의 끝(시스템 외부 또는 기능 Axx) 접속을 통할 경우는 [Jn/]옵션
	type	NULL | Input | Control	시스템 외부일 경우는 NULL
	identifier	NULL | Ix | Cx	시스템 외부일 경우는 NULL
JUNCTION Template			**비 고**
identifier		Jn	접속마다 identifier를 부여
logic		textual description	AND | XOR | OR
type		fan-out | fan-in	분기 또는 결합 여부
rule		textual description	분기 또는 결합의 선택을 위한 규칙

3.2 프로세스 흐름 모형으로의 변환

프로세스 흐름 모형으로의 변환은 IDEF0 기능 모형이 기능(function) 또는 활동(activity) 위주로 표현되고, 프로세스 흐름 모형(process flow model) 또한 활동(activity) 위주의 묘사라는 점에서 출발한다. 즉, IDEF0 모형의 중심이 되는 기능(function box)이 시스템에서 수행되는 활동(activity), 의사 결정(decision making), 행위(action), 또는 운영(operation) 등을 묘사하는 데 사용되고, 이들 기능이 프로세스 흐름 모형에서 그대로 단위 활동(unit of activity)으로 전환될 수 있기 때문이다.

또한, ICOM 구조에서 출력(output)으로 표현되는 실체로는 정보 실체와 물리적 실체가 있는데, 이것은 또 다른 기능/활동에 대해서 각각 입력, 제어, 그리고 메커니즘으로 작용할 수 있다. 따라서 기능/활동들의 활성화에 관여하는 실체들은 그 역할과 유형에 따라 정보 흐름(information flow), 자재 흐름(material flow), 제어 흐름(control flow), 그리고 자원 흐름(resource flow) 등을 구성하게 된다. 이들 흐름은 기능/활동간의 연결을 통해 표현되고, 기능/활동간의 기본적인 순서 관계를 정의한다. 따라서, IDEF0 기능 모형의 기능(function)과 ICOM의 연결은 그대로 PROWD 모형에서 각 기능과 이들 사이의 순차관계를 기본적으로 정의하게 된다.



〈그림 1〉 프로세스 흐름 모형으로의 변환 절차

한편, IDEF0 기능 모형에서는 하나의 기능이 각기 다른 조건에 의해 각기 다른 입력 조합에 대해 활성화되는 다중 활성 모드가 존재하거나, 또는 몇 개의 기능이 피이드 백과 같은 형태를 가지고 사이클을 형성하기 때문에 일관되게 순서 관계를 정의하기는 어렵다. 그러므로 이들 다중 활성 모드와 사이클은 특별한 처리과정을 통해 PROWD 모형으로 변환된다[2]. 여기에 최종적으로 각 연결의 유형을 지정하고, 논리적 관계를 정의함으로써 최종적인 PROWD 모형을 얻을 수 있다. 변환은 〈그림 1〉과 같은 과정을 거쳐서 이루어진다.

4. 시뮬레이션 코드의 자동 생성

4.1 시뮬레이션 모형을 위한 시스템 사양

시뮬레이션이 매우 다양한 형태의 실제 시스템에 적용이 되기는 하지만, 이산 사건 시뮬레이션 모형들은 논리적 체계와 구성 요소에 있어 많은 공통점을 가지고 있다. 특히 〈표 3〉과 같은 구성 요소들은 대부분의 이산 사건 시뮬레이션에서 공통적으로 발견할 수 있다[8].

이들 구성 요소는 시뮬레이션 모형 자체에 내재되어야 하는 부분과 모형 대상 시스템으로부터 정의된 입력을 바탕으로 새로이 구성되는 부분으로 나눌 수 있다. 먼저, 시뮬레이션 모형 자체에 내재되어야 할 구성 요소들로는 사건 목록, 시뮬레이션 시계, 초기화 루틴, 시간 계산 루틴, 사건 루틴, 그 밖에 라이브러리 루틴이나 보고서 생성 루틴과 같은 것들이 있다. 이들 요소들은 모형화 대상 시스템의 사양이나 특성에 관계없이 정의되는 것으로서, SIMAN이나 SLAMII와 같은 시뮬레이션 전용 언어를 사용하거나 혹은 일반 언어를 사용해서 프로그램을 작성할 때 사용자에게는 은닉되어 있는 부분이다.

한편 시스템 상태, 사건 루틴, 통계적 계수기 부분은 앞서 설명된 구성 요소들을 통해 시뮬레이션이 수행되기 전에 시스템의 특성에 따라 새로이 정의되어야 하는 부분이다. 그런데 시뮬레이션을 통해 연구하고자 하는 대상으로서의 시스템은 시스템 실체들의 집합을 통해 정의될 수 있다[9]. 여기서 '실체'는 '어떤 논리적인

〈표 3〉 시뮬레이션 모형의 구성 요소

구성 요소	내 용	비 고
시스템 상태 (system state)	특정 시점에서 시스템을 묘사하는 데 필요한 상태 변수의 집합	시스템 상태 변수를 정의할 수 있도록 외부에서 입력으로 제공
사건 목록 (event list)	다음 사건이 일어날 시간을 유지	시뮬레이션 모형의 내부 루틴
시뮬레이션 시계 (simulation clock)	현재의 시뮬레이션 시간 값을 유지하는 변수	시뮬레이션 모형의 내부 루틴
통계적 계수기 (statistical counters)	시스템 수행도와 관련된 통계 정보를 저장하기 위한 변수	필요한 통계치에 관한 정보 입력
초기화 루틴 (initialization routine)	초기 시작 시간(t=0)에 시뮬레이션 모형을 초기화시키는 부프로그램	시뮬레이션 모형의 내부 루틴
시간 계산 루틴 (timing routine)	사건 목록으로부터 다음 번 사건을 결정하고, 사건이 발생하면 시뮬레이션 시계를 진행시키는 부프로그램	시간의 진행은 내부 루틴이지만, 시간의 지체를 발생시키는 요소에 대한 정의는 입력시켜야 한다.
사건 루틴 (event routine)	특정한 사건이 발생했을 때, 시스템 상태를 갱신하는 부프로그램	사건과 시스템 상태의 관계를 정의해야 한다.

목표를 달성하기 위해 행위를 취하거나 상호 작용을 일으키는 것들'을 일컫는다. 또한 '상태'라는 것은 '특정 시점에서 연구 목적에 맞추어서 시스템을 설명할 수 있는 상황'으로 정의될 수 있기 때문에 시스템 상태는 실체들이 가지는 속성을 통해서 정의될 수 있다. 실체는 이밖에도 통계량이나 출력의 분석을 위한 속성을 가질 수 있다. 시간 진행 루틴에서 시뮬레이션 시간을 진행시키기 위해서는 시스템에서 시간 소요가 발생하는 부분에 대한 정의가 필요하고, 일정한 시간이 경과한 다음 시스템 상태 변화에 관한 정의도 이루어져야 한다.

그러므로 시뮬레이션을 수행하기 위해 시스템으로부터 제공되어야 하는 사양에 관한 정보가 〈표 4〉와 같이 정리되었으며, 〈표 2〉와 같은 PROWD 모형의 데이터 구조로부터 이들 정보를 도출한다.

〈표 4〉 시뮬레이션 모형에 필요한 시스템 정보

구 분		사 양	내 용	비 고
process unit	활동	process id.	실체의 속성(attribute) 변화를 일으키도록 하는 부분	시간의 지체 발생
		time duration	실체가 process에 머무는 시간	parameter로 정의
	입력	entity-event	process에 실체를 투입시키는 사건과 해당 활동을 정의	
	출력	entity-event	process의 종료에 의해 발생하는 사건과 실체를 정의	
실체(entity)		entity id.	시스템으로부터 정의된 서비스를 받는 실체	
속성	state	state variable	실체와 시스템 상태를 정의	
	static	static variable	통계량을 얻기 위해 실체와 시스템 상태를 정의	
사건(event)		event id.	시스템의 상태 변화에 따라 실체의 처리 또는 이동을 정의	사건별 출발/도착 process를 정의
Queue	사건 que.	event que. id.	프로세스에 대한 사건의 대기행렬	
	실체 que.	entity que. id.	프로세스에 들어갈 실체의 대기행렬	

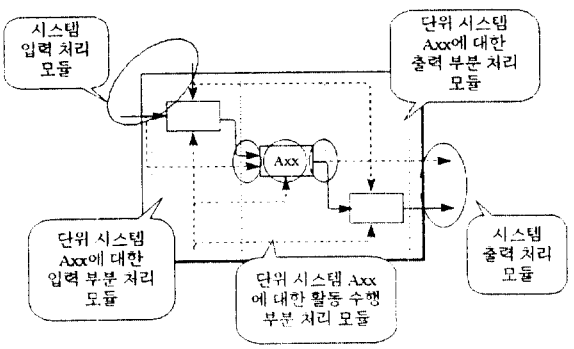
4.2 시뮬레이션 사양의 작성

시뮬레이션 코드의 자동 생성은 IDEF0 기능 모형으로부터 도출된 PROWD 모형을 시뮬레이션 사양에 맞도록 자동적으로 대응(mapping)시키는 한편, 추가되는 사용자 정의 정보를 입력 받는 방법으로 이루어진다. 이를 위해서 먼저, 단위 시스템을 정의하고, 단위 시스템을 중심으로 입력과 출력, 관련되는 실체, 그리고 대기 행렬 등을 정의한다.

4.2.1 단위 시스템과 자원

단위 시스템(unit system)은 기본적인 프로세스 단위(process unit)를 정의하기 위한 것으로서, 〈표 3〉에서 보는 것처럼 프로세스 흐름 모형에서 기능/활동을 수행하는 기능(function)으로부터 정의된다. 이렇게 정의된 단위 시스템(unit system)은 시뮬레이션 모형의 기본적인 모듈을 구성하며, SIMAN 코드로는 하나의 STATION에 해당된다. 각 기능(function)들 사이에 정의된 실체의 흐름은 단위 시스템(unit system)으로 구성된 모듈들 사이에서 실체의 흐름으로 구현되어 전체 시스템의 동작 행태 및 성능에 대한 시뮬레이션을 수행할 수 있다.

시뮬레이션 코드를 자동으로 생성시키기 위해서 하나의 단위 시스템은 다시 〈그림 2〉와 같이 세 개의 부 시스템으로 나누어서 정의된다. 즉, 하나의 단위 시스템은 각 단위 시스템에 대한 입력 부분을 처리하는 '입력 시스템', 각 단위 시스템에서 발생하는 시간 지연을 위한 '작업 수행 부시스템', 그리고 각 단위 시스템으로부터 다른 단위 시스템이나 전체 시스템으로의 출력을 처리하는 '출력 부시스템' 등 3개의 부시스템으로 나뉘어지고, 이를 중심으로 시뮬레이션 코드가 작성된다.



〈그림 2〉 단위 시스템에 대한 부 시스템의 정의

한편, 프로세스 흐름 모형에서 기능(function)을 수행하거나 기능의 수행을 지원하도록 정의된 메커니즘으로부터 단위시스템의 identifier, 자원 identifier를 정의하고 필요할 경우에는 사용자에 의하여 용량 혹은 작업소요시간이 입력될 수도 있다. 여기서 identifier는 프로세스 흐름 모형의 내용으로부터 시뮬레이션 코드에 자동적으로 대응되어가는 과정에서 내부적으로 각각의 단위 시스템이나 자원 등을 구분하기 위해서 PROWD 모형의 데이타 구조로부터 상속받는 정보이다.

4.2.2 실체의 정의

시뮬레이션은 대상 시스템을 흐르는 객체 즉, 실체의 흐름을 통하여 수행된다. 따라서 프로세스 흐름 모형에서는 각 기능(function)별로 정의된 입력과 출력이 실체의 흐름을 형성하게 된다. 프로세스 흐름 모형에서의 입출력은 크게 전체 시스템의 관점에서 정의되는 입출력과 개별 기능(function)에 정의되는 입출력-시스템내의 단위시스템들간의 입출력-이 있다. 그러므로 시뮬레이션 모형에서의 실체의 흐름은 전체 시스템에 대한 입출력을 정의하는 실체 흐름과 각 단위 시스템(unit system) 사이의 실체 흐름으로 나누어 정의될 수 있다.

또한 프로세스 흐름 모형에서 각 기능(function) 사이의 실체 흐름에는 연결(link)의 종류에 따라 해당 기능을 직접 가동시킬 수 있는 격발 실체 흐름과 직접적으로는 기능의 프로세스를 가동시킬 수는 없는 비격발 실체 흐름이 있다. 서로 다른 특성을 가지는 두 가지의 실체흐름은 다음절에서 서로 다른 대기행렬을 이용한다. 이들 실체 흐름은 단위 시스템의 입-출력 부시스템 부분의 코딩과 관련된다. 하나의 단위 시스템에서의 입력과 출력 부시스템에는 여러 개의 실체들이 입출력될 수 있다. 이들 실체 사이의 논리적 관계를 위하여 프로세스 흐름 모형의 접속(junction)으로부터 단위 시스템들간의 실체 흐름을 정의할 수 있다.

4.2.3 사건의 정의와 실체 대기 행렬(entity queue)

실체의 흐름에 따라 각 단위 시스템(unit system)에서는 기능의 수행과 관련하여 4종류의 사건이 정의될 수 있다. 즉,

- NTE : { set of non-triggering entities } arrived
- MA : { set of mechanism } available
- TE : { set of triggering entities } arrived
- FT : { set of mechanism : function terminate} perform on { set of entities } during [parameter of processing time].

NTE는 독립적으로 다른 실체에 영향을 주지 못하지만 다른 특정한 사건에 의하여 다른 실체에 영향을 줄 수 있다. MA는 단위 시스템에서 작업을 수행하는 시간 지연 현상을 일으킬 수 있는 사건이 가능한 상태를 말한다. TE는 특정한 사건을 실행할 수 있는 실체의 도착을 의미한다. FT는 단위 시스템에 특정한 기능이 수행되는 사건을 의미한다.

각 사건 집합은 또 여러 개의 사건들로 구성된다. 이런 사건들 중 MA와 FT는 단위시스템의 3가지 부시스템 가운데 '작업 수행 부시스템'의 상태에 의해 결정되지만, NTE와 TE는 '입력 부시스템'과 '출력 부시스템'에서 처리된다. TE는 직접 작업 수행이 이루어지도록 하는 사건이지만 NTE는 직접으로는 작업수행이 이루어지도록 하는 사건이 아니다. 그러므로 서로 다른 성질의 사건들을 제어하기 위해서 단위시스템의 입력 부시스템에는 서로 다른 두 개의 대기 행렬(queue)이 정의된다.

- QTE : QUEUE for triggering entities
- QNE : QUEUE for non-triggering entities

QTE로 정의되는 대기 행렬로 들어오는 실체들은 TE의 특성을 가지게 되며, 대응하는 자원을 즉시 할당받아서 실행된다. 이에 대하여 NTE로 정의되는 대기행렬로 들어오는 실체들은 NTE의 특성을 가지게 되며 NTE로 들어오는 몇 개의 실체들이 모여서 실행되게 된다.

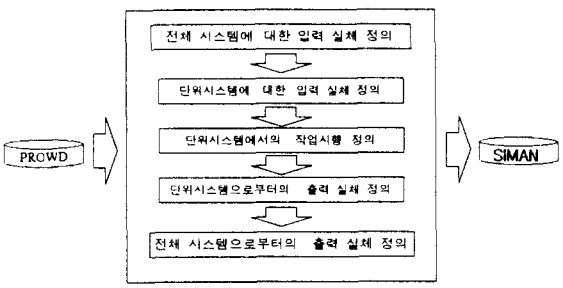
4.2.4 사용자 정의 및 매개 변수의 정의

프로세스 흐름 모형으로부터 정의될 수 없는 다양한 정보는 사용자가 입력하도록 한다. 일반적으로 사용자가 입력하는 사항에는 다음과 같은 것이 있다.

- 단위 시스템의 기능/활동에 대한 수행 시간
- 단위 시스템의 수행을 위한 자원의 용량
- 시스템 입력 실체의 발생 또는 도착 빈도 및 간격
- 논리적 분기 및 결합에 대한 확률(필요한 경우)
- 시뮬레이션 수행 시간
- 기타 시스템 수행도 분석을 위한 실체별 속성 및 통계량

4.3 시뮬레이션 코드 생성 알고리즘

앞서 정의한 방법을 통합하여, 실제로 SIMAN에 적용될 수 있는 시뮬레이션 코드를 자동적으로 생성하는 알고리즘의 개괄적인 수행과정은 〈그림 3〉과 같다.



〈그림 3〉 시뮬레이션 코드의 생성 과정

이 때 PROWD 모형으로부터 시뮬레이션 코드를 생성하기 위해 이용되는 정보는 크게 분류하여 세 가지로 나뉘어진다.

- 전체 시스템에 대한 입-출력
- 전체 시스템을 구성하는 단위 시스템
- 단위 시스템간의 입-출력 실체

이들 세 가지 정보를 이용하여 다음과 같은 과정을 거쳐서 시뮬레이션 코드가 생성된다. 고딕으로 쓰인 부분은 시뮬레이션 원시 코드로 사용된다. 부록에서는 알고리즘의 보다 구체적인 내용과 〈그림 2〉에서 정의하고 있는 부시스템의 유형별 generic SIMAN code가 소개되어 있다.

Algorithm

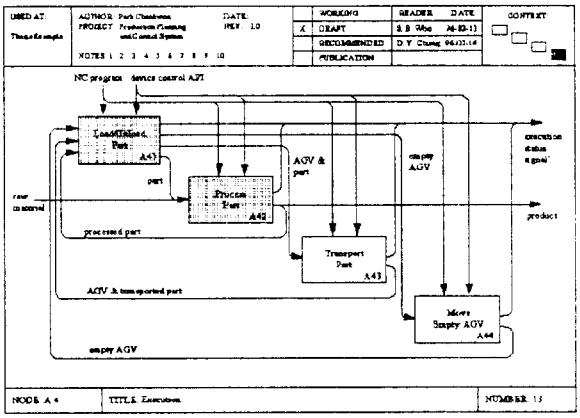
```
단계 0 : 해당 시스템에 대한 정보를 프로세스 흐름 모형으로부터 읽어 들인다.
단계 1 : 전체 시스템에 대한 입력 실체 정의
    while( i < SysInNO )            ; 전체 시스템에 대한 입력 실체 설정
    {       CREATE                  ; 입력 실체 생성
            ASSIGN                  ; 입력 실체의 유형 정의
            ROUTE                   ; 입력 실체의 목적지 할당
    }
단계 2 : 단위 시스템(unit system)에 대한 입력 실체 정의
    while( i < UnitSysNO )          ; 단위 시스템의 개수 만큼 입력 부시스템 설정
    {       STATION                 ; 단위 시스템에 대한 입력 부시스템 정의
            while( j < PIiNO )
            {       QUEUE           ; 단위 시스템에서의 입력 실체별 대기 행렬 정의
            }
            MATCH                   ; 단위 시스템에서의 입력 실체에 대한 사건 수행 조율
            ROUTE                   ; 단위 시스템내의 작업 수행 부시스템으로 실체 할당
    }
단계 3 : 단위 시스템에서의 작업 수행 정의
    while( i < UnitSysNO )          ; 단위 시스템의 개수 만큼 작업 수행 부시스템 설정
    {       STATION                 ; 단위 시스템에 대한 작업 수행 부시스템 정의
            DELAY                   ; 단위시스템에서의 작업 수행
    }
단계 4 : 단위 시스템으로부터의 출력 실체 정의
    while( i < UnitSysNO )          ; 단위 시스템의 개수 만큼 출력 부시스템 설정
    {       STATION                 ; 단위 시스템에 대한 출력 부시스템 정의
            while( j < POiNO )
            {       ASSIGN          ; 출력 실체의 유형 정의
                    ROUTE           ; 출력 실체의 목적지 할당
            }
    }
단계 5 : 전체 시스템으로부터의 출력 실체 정의
    while( i < SysOutNO )           ; 전체 시스템에 대한 출력 실체의 정의
    {       STATION                 ; 전체 시스템에 대한 출력 부시스템 정의
            TALLY & COUNT           ; 시뮬레이션 결과 획득
            DISPOSE                 ; 출력 실체 삭제
    }
```

여기서 SysInNO는 전체 시스템에 대한 입력 실체 개수를, SysOutNO는 전체 시스템에 대한 출력 실체 개수를, UnitSysNO는 전체 단위 시스템의 수를, PIiNO는 i번째 단위 시스템으로의 입력 실체 개수를, 그리고 POiNO는 i번째 단위 시스템으로부터의 출력 실체 개수를 각각 의미한다. 알고리즘에 의해 각 단계별로 SIMAN 언어에 기반을 둔 시뮬레이션 프로그램이 생성된다. 단계 1에서는 전체 시스템에 대한 입력 실체의 갯수만큼 실체를 생성시켜 실체 ID를 부여하고, 입력 실체의 유형에 따라 Type을 부여하도록 한다. Type과 ID가 부여된 실체는 해당 단위 시스템의 입력 처리부 시스템을 나타내는 STATION으로 보내진다. 단계 2에서는 단위시스템으로의 입력 실체에 대한 입력 제어를 수행한다. 개별 입력 실체들은 각 유형(NTE I TE)별 대기행렬로 보내진 후, 조건이 만족되면 triggering된다. Triggering된 입력 실체는 단위시스템에 대한 작업 수행 부시스템으로 보내어진다. 단계 3에서는 단위 시스템에서 수행되는 작업에 의한 시간의 지체가 이루어진다. 소요 시간과 자원에 관한 정보로는 사용자가 입력한 추가 정보가 사용된다. 시간 지체가 이루어진 후에는 단위 시스템에서의 출력을 처리하는 출력 부시스템으로 보내어진다. 단계 4에서는 단위 시스템으로부터 외부로 나가는 출력 실체의 유형을 설정하고, 출력 실체의 ID와 다음 목적지를 할당한다. 단위 시스템의 외부로 나가는 출력 실체는 각 단위 시스템의 입력 부시스템 혹은 전체 시스템의 출력 부분으로 보내어진다. 단계 5는 전체 시스템으로부터 외부로 나가는 출력 실체를 정의하는 부분으로, 시스템 외부로 나가는 실체를 이용하여 필요한 통계 정보를 획득한다. 통계 정보를 획득한 후에는 출력 실체를 시스템 밖으로 보내면서 실체를 삭제한다.

5. 시스템의 구현과 적용

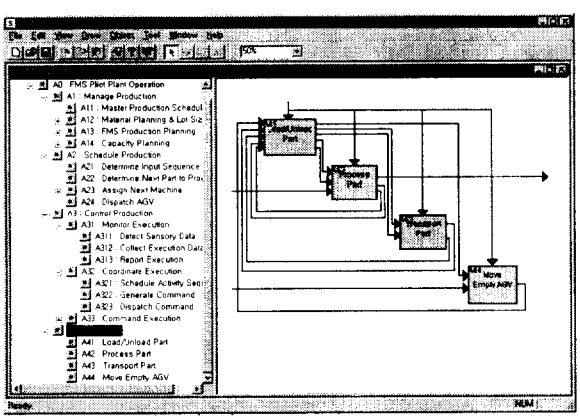
기능 모형으로부터 시뮬레이션 사양을 작성해가는 과정을 살펴보기 위하여 FMS의 운영과정에 대해 먼저 IDEF0 기능 모형을 작성하면 〈그림 4〉와 같다. 예제는 서울대학교 자동화시스템공동연구소(ASRI)에 설치된 FMS에 대한 IDEF0 기능 모형의 일부이다[2].

제안된 방법을 통하여 프로세스 흐름 모형으로부터



〈그림 4〉 IDEF0 기능 모형 예제

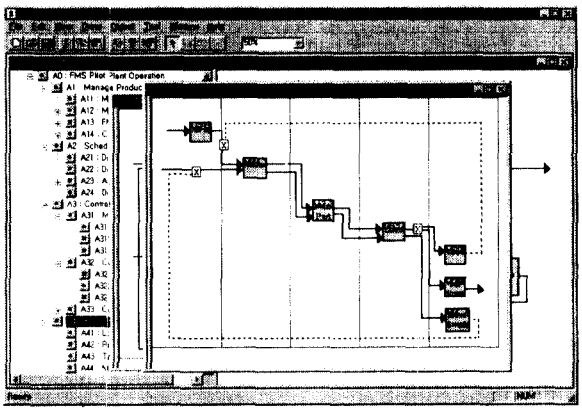
의 시뮬레이션 사양을 바탕으로 SIMAN의 Model Frame을 자동적으로 생성시킬 수 있는 지원 시스템을 구현하였다. 지원 시스템은 마이크로소프트 윈도우 환경에서 Visual C++을 이용하였다. 지원시스템에서는 IDEF0 기능모형을 입력시키고, 편집할 수 있는 기본적인 기능을 제공하고 있는데 〈그림 5〉는 〈그림 4〉의 예제 모형을 입력시킨 것이다.



〈그림 5〉 IDEF0 기능 모형 예제

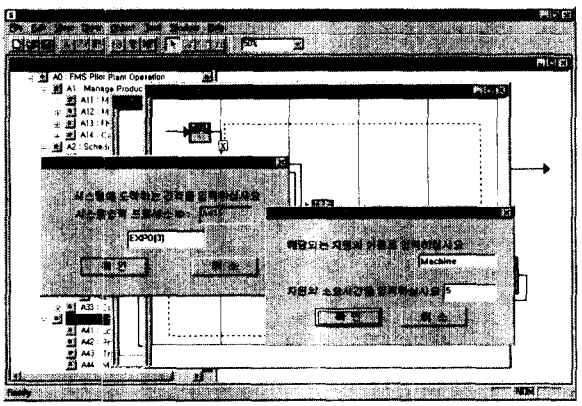
〈그림 4〉에 나타나는 것처럼 화살표에 텍스트 정보는 메뉴를 통하여 별도의 화면으로 제공하도록 처리하였다. 〈그림 4〉와 같이 입력된 IDEF0 기능 모형은 〈그림 1〉에 나타난 것과 같은 과정을 거쳐 〈그림 6〉과 같

은 PROWD 모형으로의 변환이 이루어진다[2]. 변환된 PROWD 모형은 〈그림 6〉과 같은 결과를 화면상에 보여주지만, 시뮬레이션 모형으로의 변환을 위해서 별도의 파일로 저장함으로써 〈표 2〉와 같이 정의되는 데이터베이스를 유지한다[2].

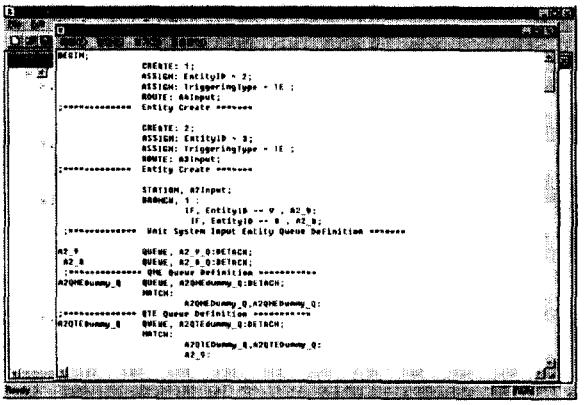


〈그림 6〉 변환된 PROWD모형

한편 실행 가능한 시뮬레이션 코드를 생성시키기 위해서는 4.2.4에서 설명한 바와 같이 추가적인 사용자 정보를 입력받아야 한다. 지원 시스템에서는 기본적으로 두 가지 종류의 사용자 정의 정보를 입력받도록 하고 있는데, 전체 시스템에 도착하는 실체의 도착 간격



〈그림 7〉 사용자 정의 정보의 입력



〈그림 8〉 생성된 시뮬레이션 SIMAN Model Frame

및 분포에 대한 정보와 각 기능/활동을 수행하는 데 걸리는 소요시간 및 자원에 대한 정보이다. 따라서 전체 시스템으로부터의 입력에 해당하는 각 화살표에 대해서 실체의 도착 간격 및 분포에 대한 정보를 입력시키고, 각 기능/활동을 선택해서 해당 자원과 소요시간에 대한 정보를 〈그림 7〉과 같은 대화 상자를 이용해서 입력시킨다. 사용자 정의 정보가 입력되면, 4.3절에서 제안된 알고리즘을 통해서 SIMAN model frame을 생성시킬 수 있는데, 〈그림 8〉의 내용은 지원 시스템에 의해 생성된 예제 시스템에 대한 SIMAN model frame이다.

6. 결론 및 추후연구

컴퓨터 통합 생산시스템처럼 복잡한 시스템에 대한 프로세스의 설계에 많이 이용되는 IDEF0 모형과 같은 상위 수준의 모형 단계에서 시뮬레이션을 수행할 수 있는 방법을 제시하였다. 이와 같은 방법은 시스템 설계시, 초기 단계에서부터 시스템의 개략적인 수행도 분석을 가능케하고, 시스템 개발 과정에 관여하는 여러 개발자들 사이에서 의사소통 증진과 체계적인 연계를 통해 개발에 소요되는 시간과 노력을 줄이는 데 기여할 수 있으리라 기대된다. 뿐만 아니라 시뮬레이션 모형의 생성 방법이 기존의 방법에 비해 보다 일반적(generic)이라는 장점을 가지고 있기 때문에, 계층적인 시스템 분석만 수행하면 해당 시스템에 대한 시뮬레이

션을 수행할 수 있게 된다.

그러나 이러한 일반성을 획득하는 대신 기존 시뮬레이션 소프트웨어를 사용하는 경우에 비해 프로그램의 규모가 커지는 단점이 있다. 이러한 문제점은 기존의 ARENA와 같은 CASE Tool을 이용할 때 발생하는 문제점으로 generic code generator를 이용하게 될 경우 공통적으로 발생할 수 있는 단점이라 할 수 있다. 따라서 제안된 접근 방법의 이점을 살리면서 프로그램의 규모를 줄이는 방법이 요구된다. 한편 시뮬레이션 수행 결과를 통해서 상위 수준의 기능 모형이나 프로세스 흐름 모형을 변경시킴으로써, 프로세스를 개선하고 재설계 할 수 있는 방안에 대해서도 추가적인 연구가 이루어져야 할 것이다.

참고문헌

[1] 류시각, 시뮬레이션에 의한 유연생산시스템(FMS)의 운영정책 결정에 관한 연구, 연세대학교 대학원 경영학과 박사학위 논문, 1990년 6월

[2] 박찬권, 컴퓨터 통합생산을 위한 프로세스 설계 지원 시스템에 대한 연구, 서울대학교 대학원 산업공학과 박사학위 논문, 1996년 8월

[3] 신호철, 시뮬레이션 모델구축을 위한 전문가 시스템의 설계에 관한 연구, 연세대학교 대학원 경영학과 석사학위 논문 , 1991년 12월

[4] Bravoco, R. R., and S. B. Yadav, "A Methodology to Model the Functional Structure of an Organization," Computers in Industry, Vol. 6, No. 5, pp345-361, 1985

[5] Busby, J. S., and G. M. Williams, "The Value and Limitations of Using Process Models to Describe the Manufacturing Organization," Int. J. of Production Research, Vol. 31, No. 9, pp2179-2194, 1993

[6] Colquhoun, G. J., R. W. Baines, and R. Crossley, "A State of the Art Review of IDEF0," Int. Journal of Computer Integrated Manufacturing, Vol. 6, No. 4, pp252-264, 1993

[7] Haddock, J. ,"An Expert Framework Based On a Simulation Generator", SIMULATION, Vol. 48, No. 2, pp. 45-53, 1987

[9] Law, A. M. and W. D. Kelton, Simulation Modeling and Analysis, 2nd Ed., McGraw-Hill, 1991

[10] Mathewson, S. C., "Simulation Program Generators", SIMULATION, Vol. 23, No. 6, pp 181-189, 1975

[11] NIST, Integration Definition for Functional Modelling (IDEF0), Draft Federal Information Processing Standards Publication(FIPS) 183, 1993

[12] Pidd, M., Computer Simulation in Management Sceinece, 3rd Ed. Wiley, 1992

[13] Rensburg, A.V. and N. Zwenmstra, "Implementing IDEF Techniques as Simulation Modelling Specification", Computer & IE, Vol. 29 No. 1-4, pp 467-471, 1995

〈 부 록 〉

A1. 시뮬레이션 코드 생성 알고리즘

Notation

SysInNO : 전체 시스템에 대한 입력 실체 갯수
SysOutNO : 전체 시스템에 대한 출력 실체 갯수
TYPEj : 단위 시스템에 대한 입력 실체j의 유형
UnitSysNO : 전체 단위 시스템의 수
PIiNO : i번째 단위 시스템으로의 입력 실체 갯수
POiNO : i번째 단위 시스템으로부터의 출력 실체 갯수
Process_i_j_Next : i단위 시스템으로부터의 j출력 실체의 다음 작업 수행 단위 시스템

Algorithm

단계 0 : 해당 시스템에 대한 정보를 프로세스 흐름 모형으로부터 읽는다.
단계 1 : 전체 시스템에 대한 입력 실체 정의

```
i = 0
while( i < SysInNO )
{       CREATE & MARK
        ASSIGN : ENTITY ID
        if( TYPEi == NTE ) ASSIGN : Type = NTE
        else ASSIGN : Type = TE
        ROUTE : process_ID_input[i]
        I++;    }
```

단계 2 : 단위 시스템(unit system)에 대한 입력 실체 정의

```
i = 0 ;
while( i < UnitSysNO )
{       STATION : process_i_input
        j = 0 ;
        while( j < PIiNO )
        {       i_processs_j QUEUE, i_process_j_Q:DETACH;
                j++ ;       }
        j = 0 ;
        MATCH :
        while( j < PIiNO )
        {       if( TYPEj == NTE ) i_process_j :
                j++ ;       }
        i_process_NTE;
        j = 0 ;
        MATCH :
        while( j < PIiNO )
        {       if( TYPEj == TE ) i_process_j:
```

```
            j++         }
        i_process_NTE ,  i process Activation;
        i process Activation ROUTE: process_i_operation;
        i++;        }
단계 3 : 단위 시스템에서의 작업 수행 정의
    i = 0 ;
    while( i 〈 UnitSysNO )
    {       STATION : process_i_operation ;
            QUEUE, process_i_Q;
            SEIZE: process_i_R;
            DELAY: process_i_DelayTime;
            RELEASE : process_i_R;
            ROUTE: process_i_output
            i++;        }
단계 4 : 단위 시스템으로부터의 출력 실체 정의
    i = 0 ;
    while( i 〈 UnitSysNO )
    {       STATION : process_i_output
            j = 0 ;
            while( j 〈 POiNO )
            {       ASSIGN : Type = TE || NTE
                    ASSIGN : ENTITY ID
                    ROUTE : Process_i_j_Next;
                    j++;        }
            i++;                }
단계 5 : 전체 시스템으로부터의 출력 실체 정의
    i = 0
    while( i 〈 SysOutNO )
    {       STATION : SysOutput_i ;
            TALLY : ;
            COUNT : : DISPOSE ;
            i++;        }
```

A2. 부시스템의 유형별 generic code

알고리즘에 의해 생성되는 부시스템(sub-system)별로 generic SIMAN code는 〈표 A1〉과 같다. 여기서 Axx는 PROWD 모형에서 기능/활동을 나타내는 노드 번호이고, { }는 해당 코드가 identifier의 수만큼 반복됨을 의미한다.

〈표 A1〉 부시스템별 generic SIMAN code

전체 시스템의 입력 처리 부분에 대한 Generic Code		
{	CREATE : IntervalIC_s : MARK(SysIn);	시스템에 투입될 실체 생성 Interval은 사용자 정의
	ASSIGN : EntityId = { IC_s } ;	모든 시스템 입력 s에 입력 실체 id 부여
	ASSIGN : TriggeringType = {NTE|TE};	격발/비격발 흐름 지정
	ROUTE : ,AxxInput; }	해당 단위 시스템 Axx의 입력 모듈에 투입
단위시스템 Axx의 입력 처리 부분에 대한 Generic Code		
	STATION, AxxInput ;	단위 시스템 Axx에 대한 입력 모듈 선언
	BRANCH, 1 : { IF, EntityId==IC_i, AxxIC_i : } ;	입력 실체 집합(i)에 대해 실체 id별로 대기행렬로 보낸다.
{ AxxIC_j AxxNTEdefined	QQQUEUE, AxxIC_j_Q : DETACH; } MATCH : {AxxIC_j : } AxxQNEdefined ;	비격발 흐름으로 정의된 실체(j)가 모두 도착하면 Axx에 대한 QNE로 보낸다.
{ AxxIC_k AxxTEdefined	QUEUE, AxxIC_k_Q : DETACH; } MATCH : {AxxIC_k : } AxxQTEdefined;	격발 흐름으로 정의된 실체(k)가 모두 도착하면 Axx에 대한 QTE로 보낸다.
AxxQNEdefined	QUEUE, AxxQNE_Q : DETACH;	Axx-QNE가 정의되고 TE를 대기.
AxxQTEdefined	QUEUE, AxxQTE_Q : DETACH; MATCH : AxxQNEdefined : AxxQTEdefined : AxxActivation;	AXX-QTE가 정의되고, TE가 도착하기를 기다렸다가 activation 상태로 들어감. QTE가 없이면 delayed 상태.
AxxActivation ROUTE : ,AxxProcessing ;		단위시스템 Axx의 활동 수행 모듈로 보냄.
단위시스템 Axx의 활동 수행 부분에 대한 Generic Code		
	STATION, AxxProcessing ;	단위시스템 Axx에서의 활동 수행 모듈 선언
	QUEUE, AxxM_Q : SEIZE : { AxxM_n: } ;	메커니즘 가용 상태에 대한 대기
	DELAY : AxxTime ; RELEASE : { AxxM_n: } ;	사용자 정의 시간 AxxTime 만큼 활성화
	ROUTE : ,AxxOutput;	단위 시스템 Axx의 출력 모듈로 보냄
단위시스템 Axx의 출력 처리 부분에 대한 Generic Code		
	STATION, AxxOutput ;	단위 시스템 Axx의 출력 모듈 선언
	DUPLICATE : {1, AxxOutToUS_1 :} {1, AxxOutToSys_m :} DIPOSE ;	단위 시스템 Axx에 정의된 다른 단위 시스템으로의 출력 실체(l)와 시스템으로의 출력 실체(m) 수만큼 실체를 준비
{AxxOutToUS_l	ASSIGN : EntityId = { IC_s }; ASSIGN : TriggeringType={NTE|TE}; ROUTE : ,AyyInput; }	다른 단위 시스템으로의 출력 실체(l)는 새로운 입력 실체 id를 부여하고 해당 단위 시스템의 입력부분으로 보냄.
{AxxOutToSys_m DELAY : 0 : NEXT(SysOut_m);}		시스템으로의 출력 실체(m)는 시스템 출력 모듈로 보냄
전체 시스템의 출력 처리 부분에 대한 Generic Code		
{ SysOut_m	COUNT : OutCount_m; TALLY : OutTally_m, INT(SysIn) : DISPOSE ; }	시스템 출력에 대한 통계량을 수집하고 실체 제거

● 저자소개 ●

박찬권

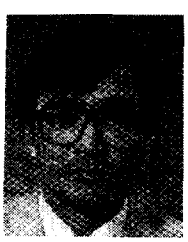
1987년	서울대학교 산업공학과 졸업(공학사)
1989년	서울대학교 산업공학과 석사
1989～1991년	LG 정보통신
1991～1992년	LG 전자
1996년	서울대학교 산업공학과 박사
1997년～현재	영산 국제산업대학교 경영정보학과 전임강사
관심분야	CIM, Enterprise Integration, Simulation, ERP

김기태

1992년	서울대학교 산업공학과 졸업(공학사)
1994년	서울대학교 산업공학과 석사
1996년	서울대학교 산업공학과 대학원 박사과정 수료
관심분야	Simulation & Simulator, Knowledge Acqusition, AI, Scheduling, FA

장성용

1980년	서울대학교 산업공학과 졸업(공학사)
1982년	서울대학교 산업공학과 석사
1991년	서울대학교 산업공학과 박사
	미국 University of Michigan 연구교수
	한국 해양연구소 연구원
	해운산업연구원 연구원
	현재 서울산업대학교 산업공학과 부교수
관심분야	CIM, CAM, 시뮬레이션, 교통 및 물류 시스템



박진우

1974년	서울대학교 산업공학과 졸업(공학사)
1976년	한국 과학 기술원 석사
1976～1979년	한국 중공업 생산관리 과장
1985년	Univ. of California, Berkeley, Industrial Engineering, Ph.D.
1982～1985년	Univ. of California, Berkeley 연구원
1985년～현재	서울대학교 산업공학과 교수
관심분야	MRP, ERP, Simulation, Scheduling, CAM