GK-DEVS: 3차원 인간제작 시스템의 시뮬레이션을 위한 형상 기구학 DEVS

GK-DEVS: Geometric and Kinematic DEVS for Simulation of 3 Dimensional Man-Made Systems

황문호*, 천상욱**, 최병규***
Moon-Ho Hwang, Sang-Uk Choen, Byoung-Kyu Choi

Abstract

Presented in this paper is a modeling and simulation methodology for 3 dimensional man-made systems. Based on DEVS(discrete event system specification) formalism[13], we propose GK-DEVS (geometrical and kinematic DEVS) formalism to describe the geometrical and kinematic structure and continuous state dynamics. To represent geometry and kinematics, we add a hierarchical structure to the conventional atomic model. In addition, we employ the "empty event" and its external event function for continuous state changing. In terms of abstract simulation algorithm[13], the simulation method of GK-DEVS, named GK-Simulator, is proposed for combined discrete-continuous simulation. Using GK-DEVS, the simulation of an FMS(flexible manufacturing system) consisting of a turing machine, a 3-axis machine and a RGV-mounted robot has been performed.

Key Words: DEVS, Geometry, Kinematics, Virtual Environment, 3 Dimensional Man-Made System

* (주) 큐빅테크 기술연구소
** 포항공과대학 산업공학과
*** 한국과학기술원(KAIST) 산업공학과

1. 서 론

3차원 공간상에 존재하며 여러 구성요소의 유기적인 관계로 구성된 시스템을 모델링 하고 시뮬레이션 하는 방법론은 주 관점에 따라 다양하게 접근되었던 분야이다. 3차원 공간상에 존재하는 시스템을 형상(geometry)과 기구학(kinematics)적 관점에서 공간상의 연속적인 운동에 주 관심을 가진 연구는 로보틱스 분야에서 활발히 진행되었다[1]. 또 다른 관점에서는, 이벤트(event)의 발생에 따라 시스템의 구성요소들이 상호 작용하는 이산사건 시스템(discrete event system) 모델링 방법론이 발전하였는데, 사건 발생에 따른 상태의 변화로 모델링이 되어야 하는 인간제작시스템(man-made system)의 특성을 표현하는 데 유용하게 사용되고 있다[4, 5].

그러나 3차원 공간상에서 작업을 하는 가상제조시스템[7][11], 가상 시술시스템[10]등의 가상환경에 대한 모델링 및 시뮬레이션의 필요성이 증가됨에 따라, 형상/기구학 모델과 이산사건 시스템 모델의 통합방법론에 필요가 지적되고 있다[9]. 이들 모델링 기법의 통합에 관한 연구는 크게 두 개의 요구사항을 반영해야 하는데

① 형상/기구학적 정보 모델링 기능,

② 연속상태/이산사건 시스템을 모델링 및 시뮬레이션 기능으로 정리된다.

본 연구에서는 DEVS[13][14]을 기초로하여 상기의 두 요구조건을 반영한 GK-DEVS (geometric and kinematic DEVS)을 제안한다. 먼저, 형상/기구학 정보의 모델링을 위해, 기존 원소모델(atomic model)의 상태변수 집합에서 '형상'을 표현하고, 모델간의 연결계층이 표현될 수 있도록 하여 '기구학' 정보를 표현하였다. 두 번째 요구조건인 연속상태/이산사건 시스템의 모델링을 위해 DEVS의 특성함수에, 공이벤트(empty event)의 처리가 추가되도록 수정하였다. 더 정확히 설명하면, 이벤트 발생 없이 연속적 시간흐름 상에서 상태변화가 가능할 수 있도록, 기존의 '외부 변위함수(external transition function)'에서 공이벤트를 처리 가능하도록 하였다. 이러한 모델을 이용하여 연속상태/이산사건 시뮬레이션이 지원되는 추상화 시뮬레이션 방법론인 GK-Simulator의 개발도 함께 이루어졌다. 제안된 GK-DEVS 방법론은 가상 제조시스템, 가상 시술시스템 등 3차원 공간상에서 존재하는 시스템을 모델링 하고 시뮬레이션 하는 방법론으로 활용할 것이라 기대된다.

본 논문은 다음과 같이 구성되어 있다. 제 2장에서는 우리가 표현하고자 하는 모델의 특징을 소개하고 기존 방법론의 한계를 논한다. 제 3장에서는 GK-DEVS 형식론과 간단한 모델링 예를 소개한다. GK-DEVS 형식론에 대한 계층적 시뮬레이션 방법론인 GK-Simulator는 제 4장에서 소개되며, 제 5장에서는 이것을 이용한 유연생산시스템(FMS)의 적용을 소개한다. 제 6장에선 결론을 맺는다.

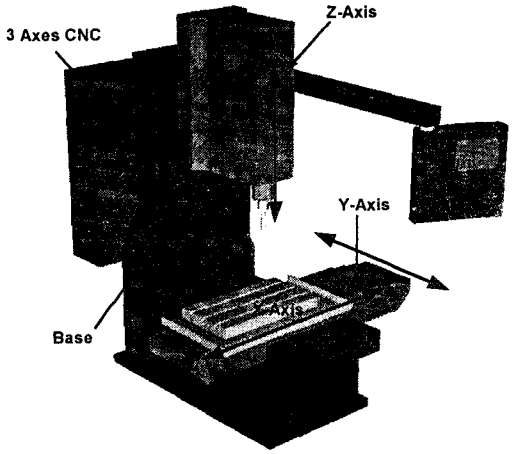
2. 연구의 배경

본 장에서는 고려하고 있는 대상시스템의 특징을 소개하고, 모델링 방법론과 관련한 기존의 연구와 이것의 한계를 소개하고자 한다.

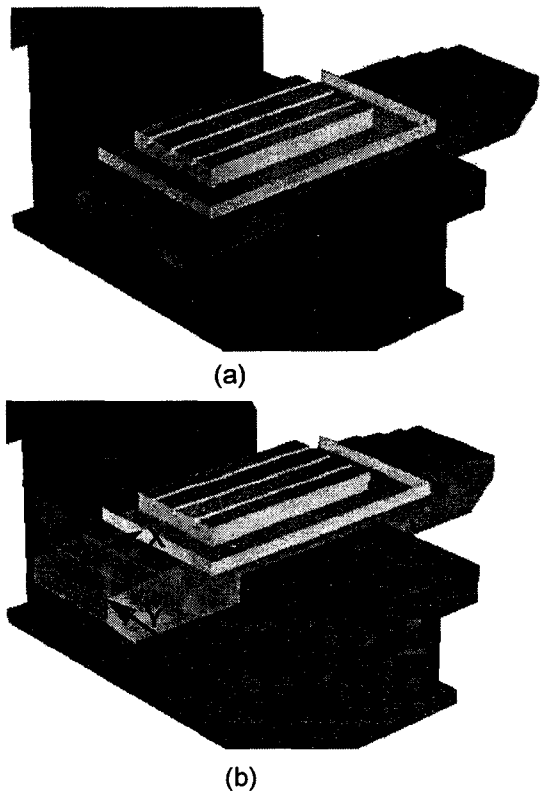
2.1 대상 시스템의 이해

대상 시스템의 특징을 설명하기 위해 <그림 1>의 자동화된 설비를 예로 들어보자. 본 예는 CNC(Computerized Numerical Controller)의 제어를 받아 X, Y, Z축의 이동과 공구회전을 통한 절삭(cutting)을 하는 3축 CNC 밀링(milling)을 대상으로 한다. Base의 후면에 제어기 3 Axes CNC가 장착되어있고, Z-Axis와 Y-Axis가 각각 직선운동을 하며, Y-Axis상에 Y-Axis의 이동방향과 수직으로 X-Axis가 운동을 하는 기구이다.

<그림 1>에서 볼 수 있듯이, 각각 X, Y, Z축은 직선운동이라는 동적행태(behavior)를 갖는 단위이며, 3 Axes CNC 제어기 또한 링크들의 위치를 파악하고 목적지로 이동케 하는 기능을 하는 단위이다



<그림 1> 3축 밀링 조립 예



<그림 2> 계층적이고 연속적인 위상변화

에니메이션을 위해서는 시간 흐름에 따라 연속적으로 상태(예를 들면, 공간상의 위상)가 변하는 모델에 관심을 가지게 된다. 즉, 두 이산사건 발생시점 사이의 상태가 구간동일(piecewise constant)가 아닐 수도 있는 모델이다. 뿐만 아니라, 상위링크의 위상이 변했을 경우에 연속적으로 하위링크에 전달되는 특성을 가지고 있음을 알 수 있다.

<그림 2(a)>와 같은 상태 <그림 2(b)>로 변하는 경우를 예를 들어 살펴보자. Y_Axis가 Y만큼 위상에 변하는 동안 X_Axis도 동시에 X만큼 이동하는 예인데, X_Axis의 위상은 결과적으로 자신의 위상변화뿐만 아니라 Y_Axis의 상위변화도 반영된 것이어야 한다.

2.2 기존 관련연구

앞서 서론에서 지적하였듯이, 3차원 인간제작 시스템의 모델링 및 시뮬레이션을 위해서는 ① 형상/기구학적 정보의 표현과 ② 연속상태/이산사건 시스템의 통합이 필요하다.

연속상태/이산사건 시스템의 통합에 관한 연구로는 Fishwick의 멀티모델(Multi-model)과 Praehofer의 DEVS기반 연속상태/이산사건 모델러인 DEV/DESS을 들 수 있다. [2][3]는 FSM (finite state machine), Petri-net 그리고 미분방정식(differential equation)이 공존하는 멀티모델의 사용을 제안하였고, [12]는 실수(real number)를 연속적으로 입/출력하고, 내부에는 미분방정식 형태가 내재된 DEVS를 확장한 DEV/DESS라는 형식론을 제안하였다.

그러나 멀티모델의 다각적 상태변화 기술법이나, DEV/DESS의 연속적인 실수 값 입출력 기능 모두 3차원 인간제작 모델에서 지원해야 할 첫 번째 요구 사항인 "형상/기구학적 특징"를 표현하기에 적합하지 않다. 즉, 2.1의 예에서 볼 수 있듯이 운동의 단위모델이 계층성을 가지고 있어야 하며 그 계층을 따라서 형상의 위치 즉, 위상변화가 연속적으로 전달되어야 하는 특징을 갖는다. 따라서 본 연구에서는 DEVS를 기초로 하여

형상/기구학의 계층구조를 반영하는 모델을 소개하고 이것의 연속상태/이산사건 시뮬레이션 방법론까지를 제안하고자 한다.

3. GK-DEVS: Geometric and Kinematic DEVS

3.1 정의(Definition)

형상과 기구학 정보를 갖고 내부에 회귀적(recursive)인 구조를 갖으며 이산사건 시스템 기술이 가능하도록 제안된 모델은 다음과 같이 정의된다.

GK-DEVS =

-(식 1)

$\langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta, M, Z, SELECT\rangle$ where

X: 입력 이벤트 집합; Y: 출력 이벤트 집합;

S: 상태변수 집합, S= $\langle GK, S^r\rangle$,

$GK=\langle G, type, F, {}^UF, A, v\rangle$ G: 볼록 다각형 집합; $F=\langle R, P\rangle$; 상위 GK.F의 상대적 프레임, R(3x3 matrix): 상대 회전정보, $P(\in R^3)$: 상대 위치 정보; ${}^UF=\langle {}^UR, {}^UP\rangle$: 절대 좌표계상의 프레임, ${}^UR$(3x3 matrix): 절대 회전정보, ${}^UP(\in R^3)$: 절대위치정보; $type\in\{fixed, prismatic, revolute\}$ 조인트(joint) type; $A\in R^3$: 링크의 축벡터; $v\in R$: 축벡터 상의 값 및 이동범위; S^r: 링크정보를 제외한 자신의 상태변수 집합, w.r.t $GK\bigcap S^r=\{\}$; M: 연결된 링크 GK-DEVS의 집합; $\delta_{int}: S\rightarrow S$: 내부 상태변이 함수 ; $\delta_{ext}: Q\times(X\bigcup\{\varnothing\})\rightarrow S$: 외부 상태변이 함수, 단, $Q=\{(s,e)|s\in S, 0\le e\le ta(s)\}$, 총체적 상태, $\varnothing$: 공(공) 입력 이벤트(non-event); $\lambda: S\rightarrow Y$, 외부 출력함수; $ta: S\rightarrow R_0^\infty$, 시간전진함수; $Z: Y\rightarrow X$: 출력전달함수;

$SELECT$: $2^{M\bigcup\{self\}}-\{\}\rightarrow M\bigcup\{self\}$, 타이 해결함수;

여기서 프레임(F)를 4x4 동차변환행렬(homogeneous transformation matrix)로 표현하면 다음과 같다.

$$T=\begin{cases}\begin{bmatrix} R & | & P\\ 0\ 0\ 0 & | & 1\end{bmatrix} & \text{if type= fixed}\\ \begin{bmatrix} R & | & P+vA\\ 0\ 0\ 0 & | & 1\end{bmatrix} & \text{if type= prismatic}\\ \begin{bmatrix} RR^{\ v}_{A} & | & P\\ 0\ \ 0\ \ 0 & | & 1\end{bmatrix} & \text{if type= revolute}\end{cases}$$

where

$$R^{\ v}_{a}=\begin{bmatrix} a_xa_xkv+cv & a_xa_ykv-a_zsv & a_xa_zkv+a_ysv\\ a_xa_ykv+a_xsv & a_ya_ykv+cv & a_ya_zkv-a_xsv\\ a_xa_zkv-a_ysv & a_ya_zkv+a_zsv & a_za_zkv-cv\end{bmatrix}$$

where $cv=\cos v$, $sv=\sin v$, $kv=1-\cos v$

$A=[a_x, a_y, a_z]$. 따라서, 어떤 프레임 관점에서 본 형상정보를 ${}^bG$라고 한다면,

${}^bG={}^bT\cdot G=\bigcup_{g\in G}{}^bT\cdot g$로 계산 가능하다. 이와 유사하게, 전체 시스템의 프레임 관점에서 본 GK 모델의 형상부 ${}^UG$는, 전체 좌표계에서본 i 번째의 링크의 변환 행렬 ${}^U_{Li}T$ 을 이용하여

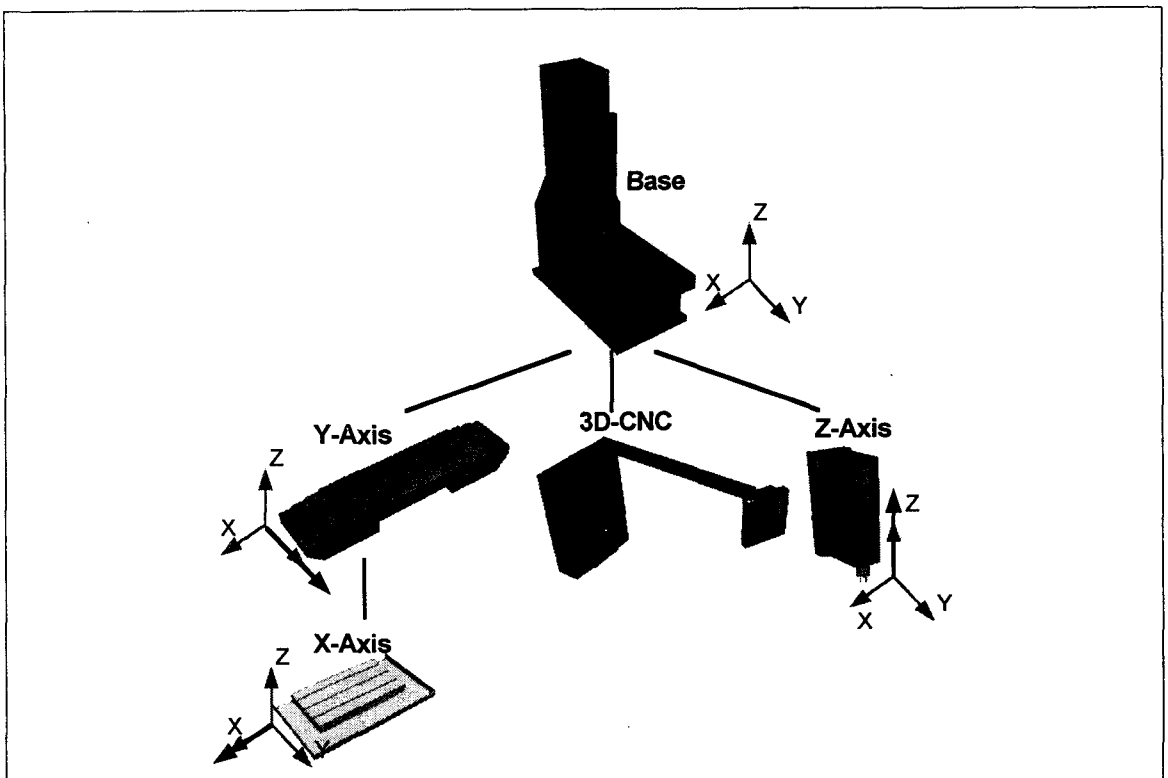
$$\begin{aligned}{}^U_{Li}T&={}_UT\,{}^U_{L1}T\,{}^{L1}_{L2}T\ldots{}^{Li-1}_{Li}T\\&=\begin{bmatrix} {}^UR & | & {}^UP\\ 0\ \ 0\ \ 0 & | & 1\end{bmatrix}\\ {}^UG&={}^U_{Li}TG=\bigcup_{g\in G}{}^U_{Li}T\,g\end{aligned}$$

와 같이 표현된다.

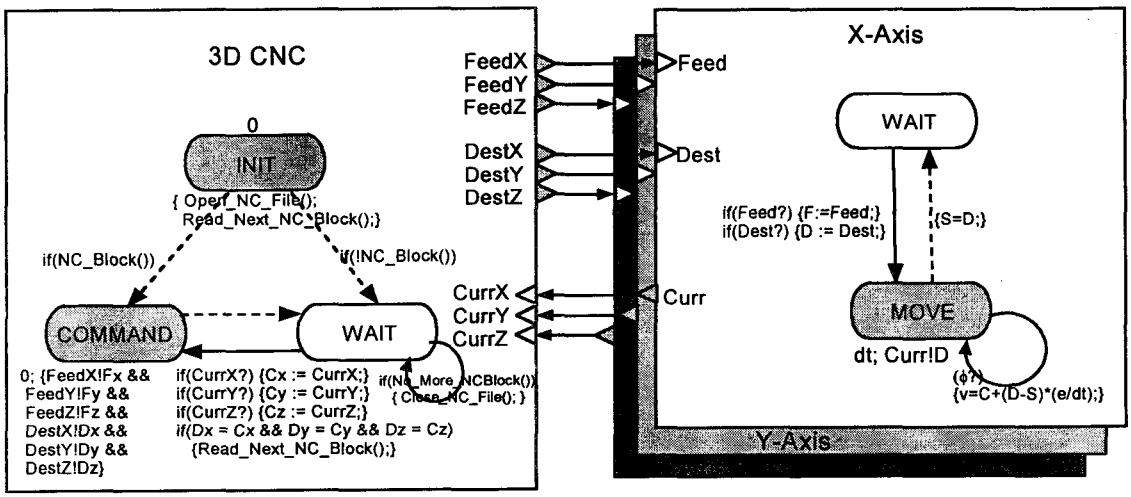
3.2 GK-DEVS를 이용한 모델링 예

GK-DEVS를 이용하여 3축 CNC(computer numerical control) 밀링(milling) 시스템의 모델링 예에 대하여 <그림 3.a>는 형상 기구학 계층관계를 보여주고 있고 <그림 3.b>는 모델들간의 연결관계와 동적 행태를 도식화한 것이다.

<그림 3.b> 에서 알 수 있듯이, 3축 제어기(3D-CNC)는 이동 목적지가 기록되어있는 NC (Numerical Control) 데이타 파일을 읽어 가면서 X,Y,Z축이 이동 목적지와 속도를 각 축에 보낸



a. 형상 및 기구학 모델



b. 연결관계 및 상태변이도

<그림 3> CNC Milling의 형상/기구학 및 연결관계/상태 변이도

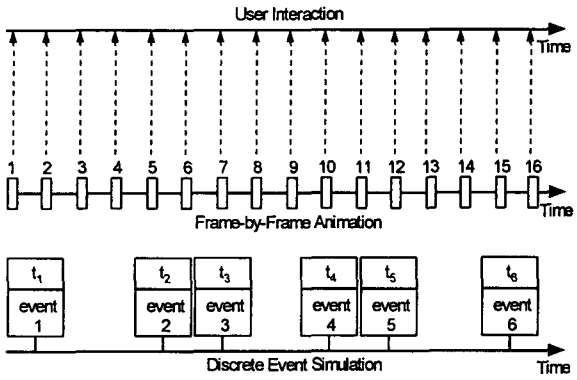
다. 이때 각 축으로부터 목적지에 도달했음을 알리는 신호(Curr)를 제어기는 받게 된다. Base, X,Y,Z-Axes, 그리고 3D-CNC의 집합론 적인 기술은 부록을 참조 바란다.

4. 추상화 시뮬레이션

본 장에서는 이산사건들의 모델들 간 입출력과 기구학적인 계층관계를 주 내용으로 하고있는 GK-DEVS 모델에 대한 시뮬레이션 방법론을 소개한다. <그림 4>는 본 논문이 고려하고 있는 시뮬레이션의 범위를 보여준다. 즉,

① 이산사건 시뮬레이션,

② 애니메이션을 고려한 연속상태 시뮬레이션이 포함되어 있다.



<그림 4> 시뮬레이션 종류와 관계

본 연구에서는 기존의 DEVS 모델의 추상화 시뮬레이션 기법[14]과 유사한 방법론을 적용하였다. 즉, GK-DEVS를 데이터로 하고 이것을 콘트롤 하는 프로세서인 GK-Simulator를 설계하여 데이터와 콘트롤의 분리의 장점인 모듈라한 설계[8]의 장점을 꾀하였다. 여기서, GK-Simulator는 기존의 Coordinator에서 수행하던 계층적인 스케줄링의 기능과 기존의 Simulator에서 수행하던 특성함수 호출의 역할을 수행한다.

본 시뮬레이션에서 고려하고 있는 메시지의 종류는 ① (*,t): 내부변이 이벤트 메시지; ② (x,t): 외부변이 이벤트 메시지; ③ (done,t): 재스케줄 메시지; ④ (Ø,t): t 시점에서의 연속상태 변화를 알리는 공이벤트 메시지 등이다.

4.1 Root-Coordinator

시뮬레이터의 계층구조상에서 가장 상위에 존재하는 Root-Coordinator는 (1)계획된 이산사건 이벤트(*,t), (2) 외부 입력이벤트(x,t) (3) 연속상태 메시지(Ø,t)중에 하나를 선택하여 계층구조를 따라서 시뮬레이터에 전달하게 된다. <그림 5>은 Root-Coordinator의 이러한 과정을 설명하고 있다.

Root-Coordinator는 다음이벤트 스케줄 시각 t_{ne}과 다음 애니메이션 시각 t_{nv} 중 최소 값이 시뮬레이션 종료시각 t_f 보다 작은 경우에 계속 진행된다(<그림 5(a)>의 3~12번째줄 참조). 만약 다음 이벤트 발생 시각 t_{ne}이 다음 애니메이션 시각t_{nv} 보다 작으면, (*, t_{ne}) 메시지를 자식 GK-Simulator에게 전달하고, 새롭게 t_{ne}를 계산한다(<그림 5(a)>의 4~6번째 줄). 만약, 다음 애니메이션 시각 t_{nv}가 다음 이벤트 발생 시각 t_{ne} 보다 작으면, (Ø, t_{ne}) 메시지를 자식 GK-Simulator에게 전달하고, 애니메이션 화면을 갱신하며, 다음 애니메이션 시각 t_{nv}를 계산한다(<그림 5(a)>의 7~11번째 줄). 시뮬레이션이 종료되면 (Ø, t_f)메시지를 자식 GK-Simulator에 전달한다(<그림 5(a)> 13번째 줄). 뿐만 아니라, 하위의 스케줄이 바뀌었음을 알리는 done message가 t시각에 전달되었을 경우에, Root-Coordinator의 t_{ne}는 t시각으로 갱신된다(<그림 5(b)>)

4.2 GK-Simulator

GK-DEVS의 시뮬레이터인 GK-Simulator는 자신의 마지막 이벤트 발생시각(t_{Lself}), 모델의 상태변수집합(S), 상태지속시간(e), 그리고 자신의

```
Procedure Root-Coordinator::main(TimeType t_f,
                                 TimeType t_vi)
 1 TimeType t_ne = its child simulator's t_N;
 2 TimeType t_nv := t_vi;
 4 while (MIN(t_ne,t_nv) < t_f) begin
 5    if t_ne <= t_nv then
 6        send (*, t_ne) to its child simulator;
 7        t_ne = its child simulator's t_N;
 8    else
 9        send (ϕ, t_ne) to its child simulator;
10        update_screen(); /* draw windows */
11        t_nv  := t_nv + t_vi;
12    end if
13 end_of_while
14 send (ϕ, t_f) to its child simulator;
```

(a) Main Loop Procedure

```
Procedure Root-Coordinator::when_receive_(done,t)
 1 if t < INFINITE then
 2    t_ne = t;
 3 else
 4    PASSIVE
```

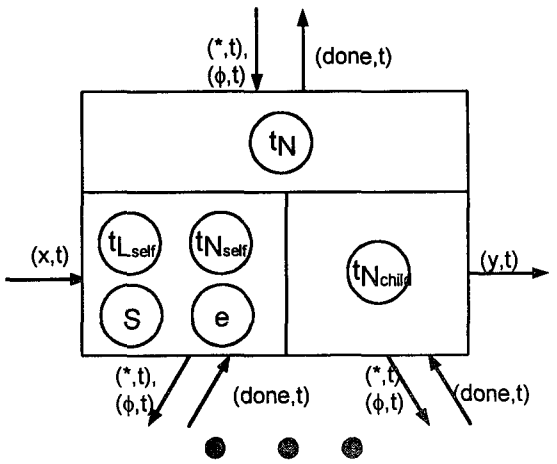
(b) Procedure for a done message

<그림 5> Root-Coordinator의 메시지 처리 과정

다음 이벤트 계획시각(t_{Nself})의 관리와 자식들의 다음 이벤트 계획시각(t_{Nchild})의 관리를 위해 <그림 6>과 같은 관련된 기억변수들을 가지고 있다.

GK-Simulator가 스케줄 된 이벤트를 알리는 (*,t)를 받았을 경우 자신의 스케줄에 의한 것인지 또는 하위(자식)의 GK-DEVS에 스케줄인 지를 판단하여 자신의 것인 경우(<그림 7>의 2번째 줄) 출력함수(λ)를 이용하여 출력 이벤트를 발생하여 전달하고 내부 상태변이 함수 (δ_{int})를 이용하여 상태 변수를 변화시키며 자신의 마지막 이벤트 발생시각(t_{Lself})과 자신의 다음 이벤트 스케줄시각(t_{Nself})을 계산한다(<그림 7>의 3~7번째 줄).

만약 자신의 스케줄이 아니라 하위의 스케줄인 경우 하위의 모델 중에 하나를 선택하여 그것에 (*,t) 메시지를 전달하고 후에 하위의 스케줄을 재 정렬하며 자식 중에 가장 빠른 스케줄을



<그림 6> GK-Simulator의 구조와 메세지들

t_{Nchild}에 기억한다(<그림 7>의 8~13번째 줄). 다음 이벤트 스케줄시각(t_N)은 t_{Nself}와 t_{Nchild}의 최

```
Procedure GK-Simulator::when_receive_(*,t)
 1 if t = t_N then
 2    if t_Nself < t_Nchild or t_Nself = t_Nchild and this is
                                selected by SELECT then
 3        y := λ(s);
 4        send to (y,t) to influencee simulators;
 5        s := δ_int(s);
 6        t_Lself := t;
 7        t_Nself := t_Lself + ta(s)
 8    else
 9        find the imminent simulators;
10        select one, i*, and send the (*,t)to it;
11        resort children by their t_N ;
12        t_Nchild := minimum of component t_N^s;
13    end if
14    t_N := MIN(t_Nself,t_Nchild);
15 else
16    ERROR;
17 end if
```

<그림 7> GK-Simulator의 (*,t) 메시지 처리과정

소값으로 설정한다(<그림 7> 14번째 줄).

GK-Simulator의 (x,t)메시지 처리과정은 <그림 8>에 정리되었다. 단, (x,t) 메시지는 (*,t)메시지와 달리 계층적으로 전달하지 않고 (수식 1)의 출력전달함수 Z를 이용하여 output 모델에서부터 input 모델로 직접 전달되는 것으로 설계하였다. 따라서 (x,t)를 받게되면 소요시간(e)를 갱신하고 외부상태변이 함수(δ_{ext})를 이용하여 상태변수를 갱신하게되고(<그림 8>의 2~3번째 줄), 자신의 최근 이산 상태 변화시각 t_{Lself}은 현재 시각으로 갱신한 후 다음 이벤트 스케줄 t_{Nself} 을 현재시각에 현 상태 유지시간 ta(s)를 더해 갱신한다(<그림 8>의 4~5번째 줄). 그리고 다음 이벤트 스케줄시각은 t_N을 t_{Nself}와 t_{Nchild}의 최소값으로 설정한 뒤 상위의 GK-Simulator에게 (done, t_N)을 보내어 자신이 속한 스케줄 리스트의 재 정렬을 알린다(<그림 8>의 6~7번째 줄).

```
Procedure GK-Simulator::when_receive_(x,t)
 1 if t_L <= t <= t_N then
 2    e := t-t_L;
 3    s := δ_ext(s,e,x);
 4    t_Lself := t;
 5    t_Nself := t_L + ta(s);
 6    t_N := MIN(t_Nself,t_Nchild);
 7    send (done, t_N) to the parent Simulator;
 8 else
 9    ERROR;
10 end if
```

<그림 8> GK-Simulator의 (x,t) 메시지 처리과정

GK-Simulator가 (done,t)를 받는 경우를 <그림 9>는 보여주고 있다. 이 메시지를 받았을 때에는 하위의 GK-Simulator의 스케줄을 재 정렬하고(<그림 9> 2번째 줄), 가장 빠른 스케줄을 t_{Nchild}에 기억하며(<그림 9> 3번째 줄), 다음 이벤트 스케줄시각(t_N)을 t_{Nself}와 t_{Nchild}의 최소 값으로 설정한다(<그림 9> 4번째 줄). 그리고 또 다시 상위에 (done, t_N)를 보내어 알린다(<그림 9> 5번째 줄).

```
Procedure GK-Simulator::when_receive_(done,t)
 1 if t_Lself <= t then
 2    resort children by their t_N ;
 3    t_Nchild := minimum of children's t_N;
 4    t_N := MIN(t_Nself, t_Nchild);
 5    send (done, t_N) to the parent Simulator;
 6 else
 7    ERROR;
 8 end if
```

<그림 9> GK-Simulator의 (done, t) 메시지 처리과정

GK-Simulator가 (∅,t)를 받았을 경우에는 소요시간(e)를 갱신하고 외부 상태변이함수 δ_{ext}를 이용하여 공이벤트(∅)의 도착을 알린다(<그림 10(a)>의 2~3번째 줄). 추상화 객체 GK-DEVS의 $\delta_{ext}(s, e, \varnothing)$의 처리과정은 <그림 10(b)>와 같은데, t시점에서의 위상상태 계산을 위해 ${}^{U}T_{p}$, ${}^{U}T$, 그리고 위상정보를 고려한 형상정보인 ${}^{U}G$를 갱신하게된다. 물론, GK-DEVS의 파생 객체들은 <그림 10(b)>의 외부 상태변이 함수를 재정의(overriding)할 수 있다. GK-Simulator는 자신의 상태를 위상 상태 뿐 아니라, 자식들에게도 (∅,t)를 전달하여 시점 t에서의 자신의 위상상태가 반영된 자식들의 연속적인 위상 상태를 계산하도록 한다(<그림 10(a)>의 4~6번째 줄).

```
Procedure GK-Simulator::when_receive_(ϕ,t)
 1 if t_L <= t <= t_N then
 2    e := t-t_L;
 3    s := δ_ext(s,e,ϕ);
 4    for-each m in child simulators
 5       send (ϕ,t) to m;
 6    end_of_for-each
 7 else
 8    ERROR;
 9 end if
```

(a)

```
Procedure GK-DEVS::δ_ext(s,e,ϕ)
 1 uT_p := get pareant uT;
 2 uT  := uT_p s.T;
 3 uG  := G uT;
```

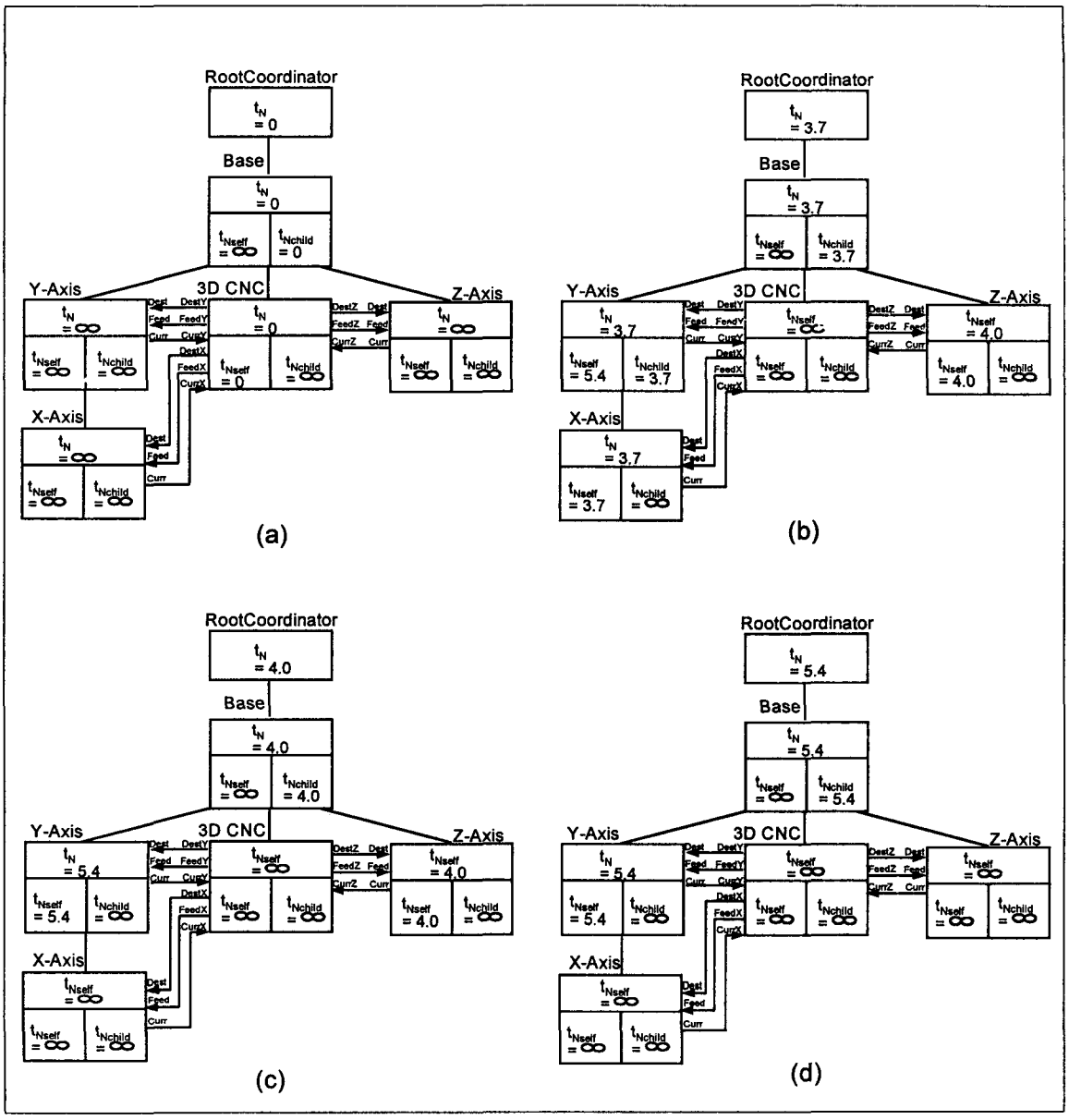
(b)

<그림 10> GK-Simulator의 (∅,t) 메시지 처리과정

4.3 계층적 스케줄링 예

본 절에서는 3.2절에서 예제로 소개했던 CNC Milling을 이용하여 계층적인 스케줄링 방법을 소개하겠다. <그림 11>는 3.2절에서 소개한 3축 CNC 밀링기의 GK-DEVS 모델에 대한 GK-Simulator의 계층적인 스케줄링의 변화를 보여주고 있다.

시점 0에서, 3D-CNC 제어기가 X, Y, Z축에 명령어를 동시에 보내는 상태를 <그림 11(a)>은 보여주고 있는데, 3D-CNC의 다음 자신 스케줄 시각 t_{Nself} 는 0설정될 것이고 3D-CNC는 자식이 없는 관계로 항상 t_{Nchild}가 무한대의 시간 ∞으로 설정되어 3D-CNC의 최종적인 t_N은 0으로 설정되게 된다. 아직까지 제어기의 명령을 받지 않은 수동적인 각 축들 X, Y, Z-Axis는 t_{Nself} 가 모두 ∞이다. 또 이 상황에서 X, Z-Axis의 t_{Nchild}는 자식이 없는 관계로 계속 ∞를 유지할 것이며 Y-Axis의 t_{Nchild}는 자식인 X-Axis의 t_N



<그림 11> 3D Milling 예제에서의 계층적 스케줄링

값을 갖게 되어 ∞을 갖는다. 이와 같은 논리로 Y-Axis, 3D-CNC, 그리고 Z-Axis를 직접적인 자식으로 갖는 Base는 t_{Nchild}는 자식의 스케줄 중 가장 빠른 3D-CNC의 스케줄 시각인 0을 갖게 된다. Base 자신의 스케줄 관점에서 보면 이 모델은 항상 Passive 한 모델이어서 자신의 스케줄이 없는 즉, t_{Nself} 는 ∞이 된다. 그러나 Base 모델 자식의 스케줄 0과 비교하여 작은 값이 채택되므로 Base의 다음 스케줄 시각 t_N은 0으로 설정되며 이것은 가장 상위에 존재하는 Root

-Coordinator의 다음 스케줄 시각은 0에 전달된다.

만약, 0 시각에서 제어기 3D-CNC로부터 목적지와 속도를 알리는 신호가 각각 DestX, FeedX, DestY, FeedY, DestZ, FeedZ를 거쳐서 X,Y,Z-Axis의 Dest, Feed로 전달되었다고 가정해 보자. 현재의 위치와 목적지(Dest)의 차를 속도(Feed)로 나누어 목적지 도달 소요시각을 각각의 이동 축들이 계획하고 그 결과가 X축은 3.7단위시각, Y축은 5.4, Z축은 4.0으로 스케줄 되었다고 가정해 보자. 이 상황에 대한 계층적인 스케줄 상황을 <그림 11(b)>는 보여주고 있다. 주목할 것은 Y-Axis의 t_N인데, X-Axies의 스케줄 시간을 담은 t_{Nchild} 3.7이 자신의 스케줄을 기억하는 t_{Nself} 5.4보다 작기 때문에 다음 스케줄 시각 t_N은 3.7로 기억되고 이 시각을 상위 계층의 GK-Simulator에게 알린다. 상위의 Base에 대한 GK-Simulator는 자식들 중 가장 빠른 스케줄인 3.7을 갖는 Y-Axis의 스케줄을 기억하고 이것을 자신의 t_{Nchild} 및 t_N은에 반영하여 Root-Coordinator에 보고한다.

같은 방법으로, 3.7 단위시각 이후에는 X-Axis의 스케줄은 ∞으로 변하고, 따라서 Y-Axis의 다음 스케줄은 자신의 스케줄 시간으로 5.4로 <그림 11(c)>와 같이 갱신된다. Base와 Root-Coordinator의 다음 스케줄 시각은 Y-Axis의 5.4와 Z-Axis의 4.0중에 작은 4.0으로 계산된다. 단위 시각 4.0이 되어 Z-Axis의 이동이 종료되고 Z-Axis의 상태지속시간이 ∞로 되면, <그림 11(d)>와 같이 전체 시스템에서 Active한 스케줄을 갖는 모델은 Y-Axis이며 이때 전체의 스케줄 시각, 즉 Base와 Root-Coordinator의 스케줄 시각은 Y-Axis의 시각인 5.4로 설정된다.

물론, 시뮬레이션이이 진행되는 동안, 애니메이션 간격으로 (∅,t)의 전달이 계층구주를 따라 전달되어지며 이것은 기구학적인 이동상황을 보여주게된다(4.1 절 Root-Coordinator 및 4.2절 GK-Simulato의 <그림 10> 참조).

5. 유연생산 시스템의 시뮬레이션 예

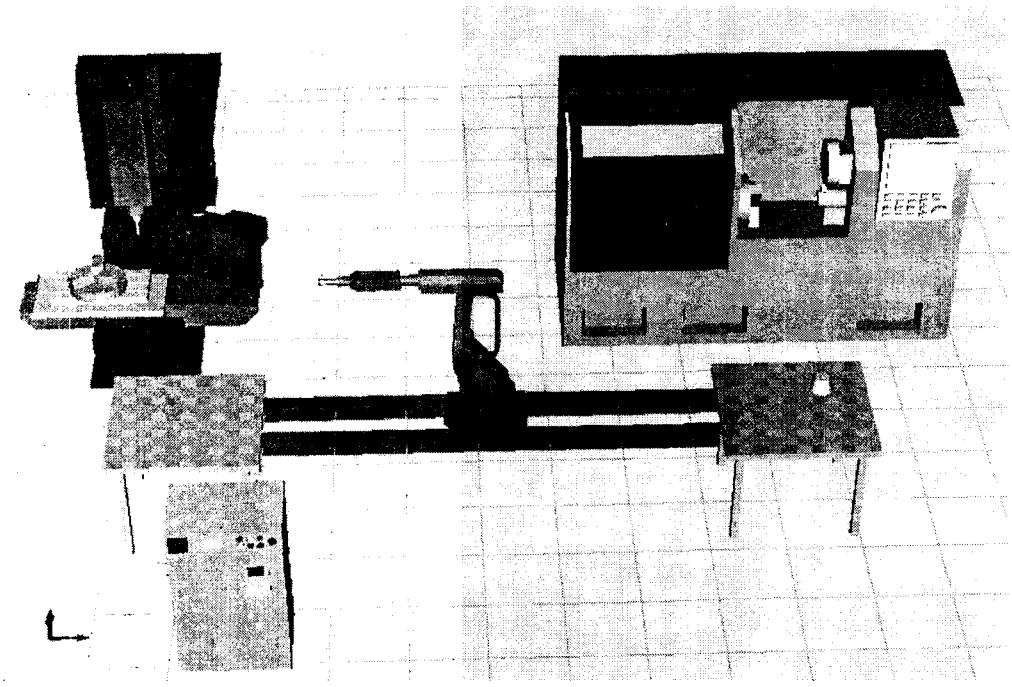
본 장에서는 3 장에서 소개한 GK-DEVS를 이용하여 유연생산 시스템(Flexible Manufacturing System)을 모델링 하고, 4장에서 소개한 GK-Simulator를 이용하여 시뮬레이션을 수행한 예를 소개한다.

<그림 12(a)> 레이아웃과 같은 유연생산시스템에서는 자동화 설비가 크게 3개가 운영되고 있다. 수치제어기(Numerical Controller)에 의해 각각의 구동부(manipulator)가 제어되는 설비로 3축 밀링 절삭기, 2축 선반(lath), 그리고 이들 두 설비에 물류 공급을 담당하는 6축 다관절 로봇도 운용된다. 6축 다관절 로봇은 레일 따라 이동하는 대차(Vehicle) 상에 탑재되어 있고, 이것은 로봇 콘트롤러의 7축의 관점에서 제어된다.

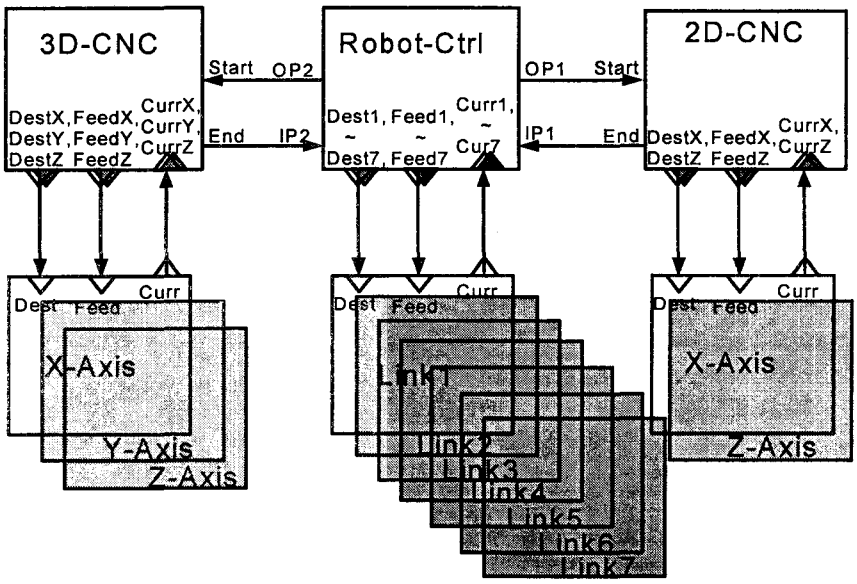
<그림 12(b)>는 대상 유연 생산 시스템 모델의 이벤트 입출력 관계를 보여주고 있다. 이 시스템에서는 각각의 수치제어기는 목적지 Dest, 속도 Feed를 구동부에 명령하고 이들의 도달상태를 Curr로 받는다. 설비들 간의 소재(material) Loading과 Unloading의 처리를 위해서 로봇 제어기 Robot_Ctrl과 2D-CNC, 3D-CNC는 이벤트를 입출력한다.

먼저 <그림 13(a)>와 같이 로봇이 2축 선반에 소재를 하고, 복귀 자세를 <그림 13(b)>와 같이 취한 후 Loading 완료를 선반에 알리면 <그림 13(b)와 같이 선반의 절삭이 진행된다. 선반의 절삭 작업이 완료되면 선반은 로봇에 작업완료를 알리고, 작업완료를 기다리고 있던 로봇은 <그림 13(c)>와 같이 unloading작업을 시작한다.

선반에서 반 가공소재를 꺼낸 로봇은 레일상의 다차를 이용하여 <그림 13(d)>와 같이 밀링 설비 쪽으로 이동하고, <그림 13(e)>에서 보여주는 것과 같이 로봇은 밀링에 소재를 공급한다. <그림 13(f)>와 같이 로봇이 복귀자세로 복귀하면 밀링 절삭 작업은 시작된다.

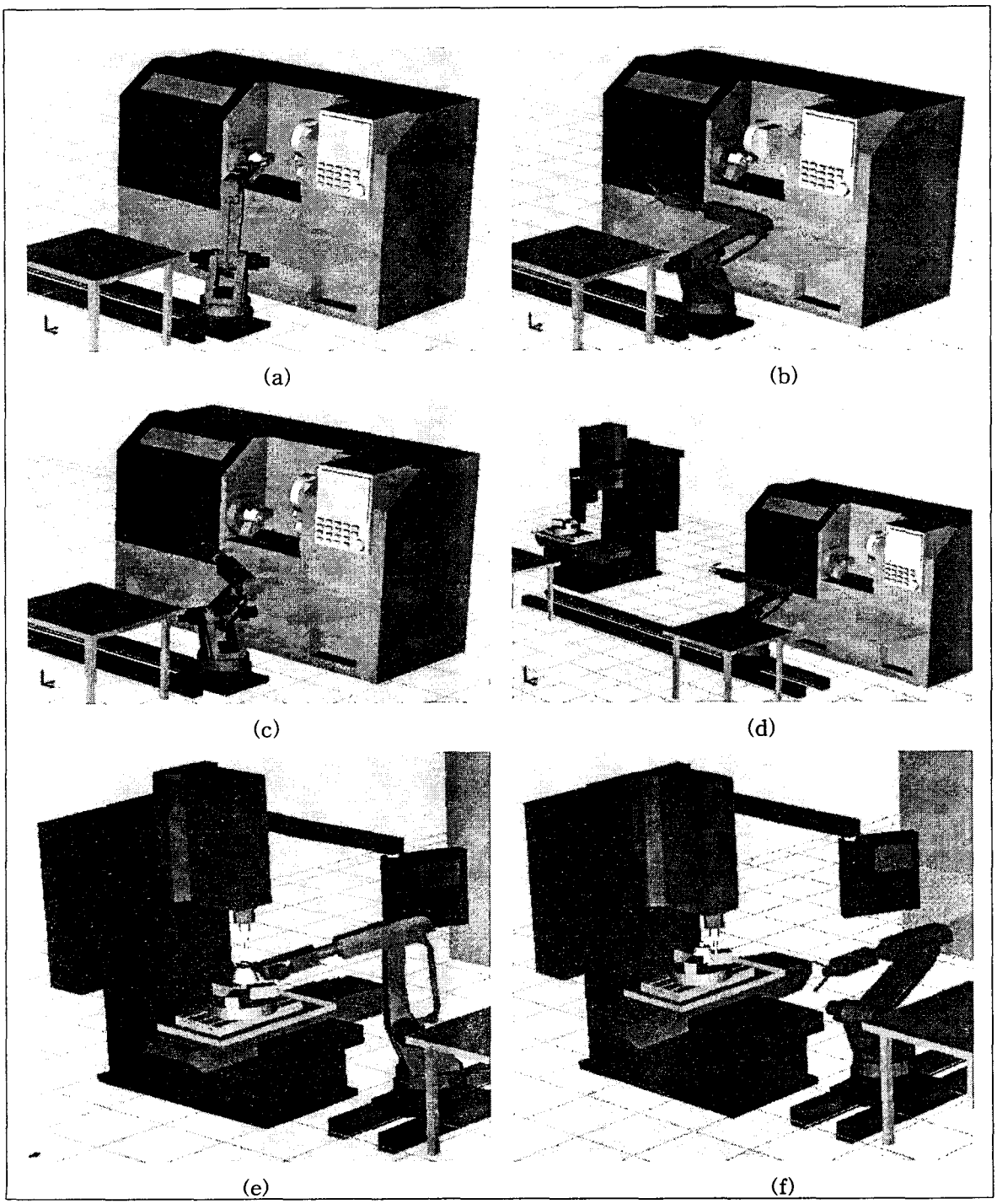


(a) FMS를 구성하는 자동화 설비의 배치도



(b) 구성 자동화 설비간의 이벤트 연결관계

<그림 12> 유연 생산 시스템 모델 예



<그림 13> FMS 구성 설비들의 협동 작업

6. 결론

본 논문에서는 3차원 공간상에서 동작하는 인간 제작시스템(Man-Mad System)의 시뮬레이션을 위한 새로운 모델링 및 시뮬레이션 방법론을 제안하였다. 제안된 GK-DEVS는 이산사건시스템의 특성과 모듈라(modular)한 개발을 가능하게 하는 DEVS 형식론에 기반을 두고 있어 인간제조시스템의 특징인 구성요소들 사이에 발생하는 사건의 전달에 따른 동적변화 특성을 잘 반영하고있다. 뿐만 아니라, 3차원 공간상에서 작동하는 기구학적 정보, 즉 계층적인 기구학적 연결관계의 표현을 위해 각 연결단위인 GK-DEVS는 계층적인 관계에 따른 연속적인 위상변화관계의 표현도 추가적으로 고려되었다.

GK-DEVS의 시뮬레이션을 위해서 GK-Simulator라는 객체의 설계와 구현을 하였는데 이 관계는 기존 DEVS의 추상화 시뮬레이션 방법을 다룬 Simulator와 Coordinator의 대비된다. 특기할 만한 차이점은, 각 GK-DEVS의 상태 변이 스케줄과 또 계층적인 하위 모델의 스케줄을 함께 고려해서 다음 이벤트 발생 시각이 계산되어 나간다는 것이 하나이다. 또 다른 하나는 이산 사건이 발생하지 않는 동안에도 계층구조를 따라서 연속적인 위상정보의 변화가 이뤄지는 3차원 공간의 인간제작시스템 시뮬레이션을 위해, [13]가 소개한 공이벤트 ∅를 하나의 메시지 형태 (∅,t)으로 구분하여, 연속상태의 계산이 필요한 시점 - 본 논문에서는 애니메이션 시각- 에서 메시지를 발생시켜 전체 모델에 전달하였다.

연속상태 시뮬레이션과 관련하여, [2][3][12]에서 소개된 것과 같이 시간의 흐름에 대하여 상태변화가 미분방정식등으로 표현되고 이것의 해가 발생하는 시점이 이벤트의 발생시점으로 계산되는 경우, GK-DEVS의 δ_{ext}(s, e, ∅)를 이용하여 미분방정식을 모델링하면 된다. 미분방정식뿐만 아니라, 공간상에서 구성 요소들의 이동시 발생하는 충돌- 예를 들면 앞서 소개한 유연생산시스템에서 로봇, 밀링, 선반 등의 loading, unloading작업시의 충돌[6]과 같이- 연속상태의 변화상에서 발생하는 수 있는 미리 스케줄 하기 힘든 이벤트(unschedulable event)의 혼합 시뮬레이션을 위해, 이산사건 및 연속상태 시뮬레이션의 혼합이 필요할 수 있다. 이때에도, 연속상태의 계산을 δ_{ext}(s, e, ∅)통해 수행하게 되며 unschedulable event의 발생이 검출되면 이산사건 스케줄의 재스케줄링이이 수행되면서 시뮬레이션이 진행되도록 GK-DEVS와 GK-Simulator의 응용이 가능하다.

참고문헌

[1] John J. Craig, Introduction to Robotics, Mechanics and Control, 2nd Edition, Addison Wesley, 1989

[2] Paul A. Fishwick and Bernard P. Zeigler, "A Multimodeling Methodology for Qualitative Modeling Engineering", *ACM Transactions on Modeling and Computer Simulation*, V2, N1, January, 1992, P52-81

[3] Paul A. Fishwick, "A Simulation Environment for Multimodeling", *Discrete Event Dynamic Systems: Theory and Applications*, V3, N2/3, July, 1993, p151-172

[4] Yu-Chi Ho, "Scanning the Issue," *Proceedings of the IEEE*, V77, 1989, p3-6

[5] Yu-Chi Ho, "Forward to the Special Issue," *Discrete Event Dynamic Systems: Theory and Applications*, V3, 1993, p111

[6] T.C. Hudson, M.C. Lin, J. Cohen, S. Gottschalk, D. Manocha, "V-Collide: Accelerated Collision Detection for VRML," *In Proc. of VRML Conference*, p119-125, 1997

[7] K. Iwata, M. Onosato, K. Teramoto, S. Osaki, "A Modelling and Simulation Architecture for Virtual Manufacturing Systems," *Annals of the CIRP*, V44, January, 1995, p399-402

[8] Tag G. Kim, Lecture Note: EE612, EE

Dept, KAIST,
(http://smslab.kaist.ac.kr/Course/EE612/)

[9] p. Klingstam and P. Gullander, "Overview of Simulation Tools for Computer-Aided Production Engineering," *Computer in Industry*, V38, 1999, p173-186

[10] U. Kuhnapfer, H.K. Cakmak, H. Maaß, "3D Modeling for Endoscopic Surgery", *Proc. IEEE Symposium on Simulation*, Delft University, Delft, NL, Oct. 13, 1999, pp22-32, ISBN: 90-804551-7-2 (1999)

[11] M. Onosato and K. Iwata, "Development of a Virtual Manufacturing System by Integrating Product Models and Factory Models," Annals of the CIRP, V42, 1993, p475-478

[12] H. Praehofer, System Theoretic Foundation for Combined Discrete-Continuous System Simulation, Ph.D. Thesis , Department of Systems Theory, University of Linz, Austria, 1991

[13] Bernard P. Zeigler, *Theory of Modelling and Simulation*, John Wiley & Sons, New York, 1976,

[14] Bernard P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, Orlando, FL, 1984,

저자소개

황문호
1990년 홍익대학교 산업공학과 학사
1992년 한국과학기술원(KAIST) 산업공학과 석사
1999년 한국과학기술원 산업공학과 박사
1999년~현재 (주) 큐빅테크 기술연구소 시뮬레이션팀장
관심분야: 연속상태/이산사건 시뮬레이션, 실시간 렌더링,
가상제조시스템, 가상 수술시스템
학회활동: 한국시뮬레이션학회 정회원, 한국공작기계학회 정회원
ACM professional member

천상욱
1994년 한국과학기술원 산업공학과 학사
1999년 (주) 큐빅테크 기술연구소 주임 연구원
2000~현재 포항공과대학 산업공학과 석사과정
관심분야: 형상 모델링, 곡면가공, NC 검증, STEP-NC

최병규
1973년 서울대 산업공학과 학사
1975년 한국과학기술원 산업공학과 석사
1982년 미국 Purdue대 산업공학과 박사
1982년~현재 한국과학기술원 산업공학과 교수
및 KAIST CIM연구센터장
관심분야:
국제 학술지 편집위원 활동
Computer Aided Design (CAD)
International Journal of Computer Integrated Manufacturing
International Journal of Manufacturing Engineering
학회 활동
American Society of Manufacturing Engineers (ASME)
Society for Industrial and Applied Mathematics (SIAM)
Society of Manufacturing Engineers (SME)
Institute of Industrial Engineers (IIE)
Society of CAD/CAM Engineers (SCCE)