INSIMS: 시뮬레이션 코드 자동생성이 가능한 제조공정 전용 시뮬레이터 개발에 관한 연구

INSIMS: An Intelligent Simulation Generator for Manufacturing Systems

이건창* 이양규**
Kon-Chang Lee, Yang-Kyu Lee

Abstract

This paper proposes an intelligent simulation generator, called INSIMS, for manufacturing systems. The INSIMS provides an interactive dialogue interface and code generation tools for modeling and code generation of manufacturing systems The interactive dialogue interface helps non-expert simulation modeler specify manufacturing systems with ease. After completing the model construction processes, the modeling specifications are automatically converted into SLAM II codes by code generation tools. To validate the usefulness of INSIMS and to illustrate the modeling processes, an example system has been presented.

1. 서 론

최근 제조공정의 효율화를 위하여 시뮬레이션이 크게 활용되고 있다. 이러한 배경에는 보다 저렴한 비용으로 생산성을 제고시키고자 하는 기업의 노력이 있었음은 물론이다. 제조공정은 수많은 구성요소가 하나의 유기적인 시스템으로서 작동을 하고 있기 때문에, 생산성을 제고하기 위해서는 많은 노력을 기울여야 가능하다. 그러나, 제조공정의 생산성 제고를 위하여 시뮬레이션 기법을 적용하면 각 구성요소의 운용 메카니즘을 효과적으로 모델링할 수 있고, 구성요소간의 관계를 모형화 함으로써 전체 제조공정의 작동 메카니즘을 손쉽게 파악할 수가 있다. 더욱이 상황의 변화를 파라미터의 간단한 조작만으로 반영할 수 있으므로, 제조공정에 관련된 여러 응용분야에 있어서 시뮬레이션 기법은 매우 실용성이 높은 기법이라고 할 수 있다. 그러나, 시뮬레이션의 유용성이 아무리 크다고 하더라도 이를 적용하는 과정에 있어서 실무자들이 쉽게 접근할 수 없는 기술적인 어려움이 있다면 실무적인 응용분야에 있어서 그 활용도는 떨어질 수밖에 없을 것이다.

지금까지 시뮬레이션 분야에서는 시뮬레이션 전용 컴퓨터 언어가 개발되어 많은 실무자들이나 학자들에게 큰 도움을 주었다. 예를 들어 GPSS, SIMSCRIPT, SLAM, SIMAN등이 대표적인 시뮬레이션 전용 컴퓨터 언어이다. 이러한 언어들은 시뮬레이션에 필요한 프로그램 블록 등

* 성균관대학교 경영학부
** 서원대학교 경영정보학과

을 기본적으로 제공하고 있기 때문에 범용컴퓨터 언어인 FORTRAN, PASCAL, C 보다 모델링에 소요되는 수고와 노력이 훨씬 감소된다. 특히 최근에는 사용자가 보다 쉽게 주어진 문제를 모형화할 수 있도록, 정교한 그래픽이나 애니메이션 (animation) 및 비쥬얼 모델링 기능을 추가한 시뮬레이션 개발환경이 제공됨으로 인하여 종전의 시뮬레이션 개발환경보다 사용자의 개발편의성이 크게 증가되었다.

그러나 아직 시뮬레이션 기법이 실무에 널리 사용되는 데에 따른 모든 난관이 완전히 해결된 것은 아니다. 일반적으로 시뮬레이션을 보다 효과적으로 실무에 적용하기 위해서는 다음과 같은 사항이 고려되어야 한다. 첫째, 시뮬레이션 언어에 대한 전문적인 지식과 경험이 요구된다는 것이다. 시뮬레이션 프로그래밍을 하려면 시뮬레이션에 확률 및 통계 등과 관련된 여러 가지의 복잡한 이론들과 시뮬레이션 언어를 익혀야 한다. 둘째, 주어진 문제를 시뮬레이션에 맞게 모델링 하기 위해서는 많은 시간과 노력이 소요된다는 것이다. 이러한 노력은 제조공정과 같이 방대한 구성요소가 복잡한 관계를 맺고 있는 시스템을 시뮬레이션 하는 경우에 더욱 커질 것이다.

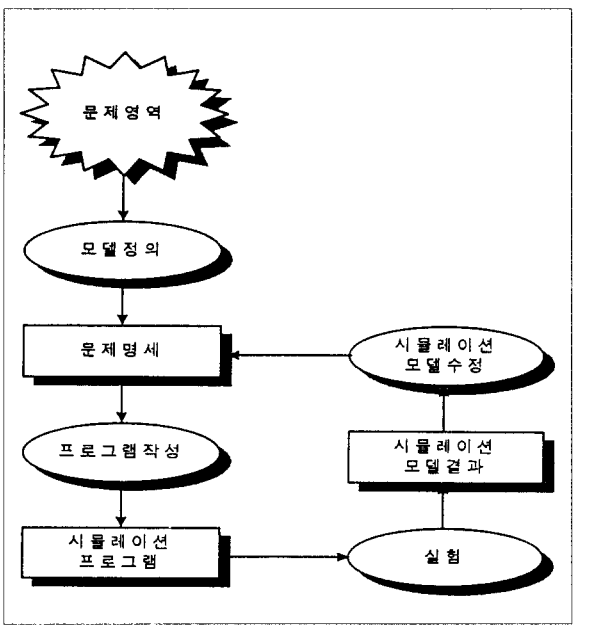
이러한 배경 하에 본 연구는 제조공정에 관한 시뮬레이션 적용을 보다 지능화 하여 개발자의 노력을 감소하기 위한 것이다. 본 연구에서는 시뮬레이션 문제를 명세화하고 (specification), 이러한 명세화에 적합한 시뮬레이션 코드를 자동으로 생성하는 (automatic code generation) 시뮬레이터 (simulator)를 제시하고자 한다. 본 연구에서 제시하는 지능적 시뮬레이터는 INSIMS (INtelligent SImulator for Manufacturing System)라 명명되었으며 대상으로 하는 시뮬레이션 전용언어는 SLAM-II이다. INSIMS는 퍼스널 컴퓨터에서 작동이 되며 사용자가 손쉽게 운용을 할 수 있도록 설계되어 있다.

2. 관련 연구 (Related Researches)

우리의 목적을 수행하기 위하여, 제안된 시스템은 두 부분－사용자의 시스템 명세 지원 (system-specification support)과 자동 시뮬레이션 코드 생성 (automatic simulation code generation)－으로 나뉘어져 있다. 이 장에서 우리는 두 주제에 관한 몇 개의 관련된 작업들을 알아보고자 한다. 먼저, 이 두 부분에 대한 전통적 접근법 (approach) 들을 살펴보겠다.

2.1 전통적인 시뮬레이션 접근법

시뮬레이션 모델링에 대한 전통적 접근법 (traditional approach)의 윤곽이 〈그림 1〉에 제시되어 있다. 전통적 접근법은 먼저 시뮬레이션니스트가 문제나 모델화될 시스템에 대한 철저한 이해를 가지고 있어야 한다. 시스템에 대한 지식이 부여된 상태에서 시뮬레이션니스트는 시스템의 모델을 정의하여 문제 명세 (problem specification)를 얻어낸다.



〈그림 1〉 전통적 시뮬레이션 접근법

다음 단계는 시뮬레이션니스트가 선택한 목적언어로 정의된 문제명세에 대한 시뮬레이션 프로그램을 작성하는 것이다. 작성된 시뮬레이션 프로그램에 대한 모든 문법적 에러들은 조사되고 제거된다. 프로그램이 디버깅 (debugging)되면서 모델은 입증 (verified) 되고 확인 (validated) 되어져야 한다. 모델의 입증은 시뮬레이션 모델이 시뮬레이션 코드에 의해 정확히 반영되어졌는가 하는 것이다. 바꾸어 말하면, 모델의 입증과정은 코드가 적절히 수행되어지는가를 확인하는 과정이다. 모델 확인은 현실의 시스템

의 행동과 모델의 행동을 비교함으로써 현실의 시스템을 모델이 얼마나 모방했는가를 확인하는 것이다 [3].

실험조정에 있어, 모델러 (modeller)는 시작조건, 정지조건, 균형 또는 안정된 상태, 견본크기를 결정해야 한다. 시뮬레이션은 보통 시스템이 비어 있고 유휴 상태일 때 시작된다. 그러므로 시스템은 시스템이 균형상태에 있기 전 기간까지 실행되어야 한다. 문제는 균형상태의 중대한 시스템에 의해 생긴 에러를 모델러가 무시하려는 시점을 찾아내는 것이다. Conway [5]에 의한 기술은 종종 균형상태를 결정하는 데 사용되어진다. 이 기술은 주기적으로 측정치를 수집하기 위해 시뮬레이션을 조정하는 것이다. Conway의 기술은 하나의 수치가 무시되어지는 측정치집단중 최대치이거나 최소치가 아닐 때까지 모든 측정치들을 무시하는 것이다.

이 무시되어진 측정치들의 집단은 만약 측정치가 수집된 자료로부터 얻어진 것이라면 표준집단으로 사용되어진다. 시뮬레이션의 실행을 중지시키는 조건은 시작조건과 유사하다. 대부분의 제조에 관한 시뮬레이션들은 부분의 주어진 숫자가 생성된 후 그친다. 이러한 연구법을 사용함으로써 만약 상호작용 (transaction) 이 시뮬레이션 모델로 들어온다면 수치에 대한 한계가 없다. 그러므로 실행의 마지막에 상호작용은 여전히 시스템에 존재하고 다시 사용될 수 있는 자원이 된다.

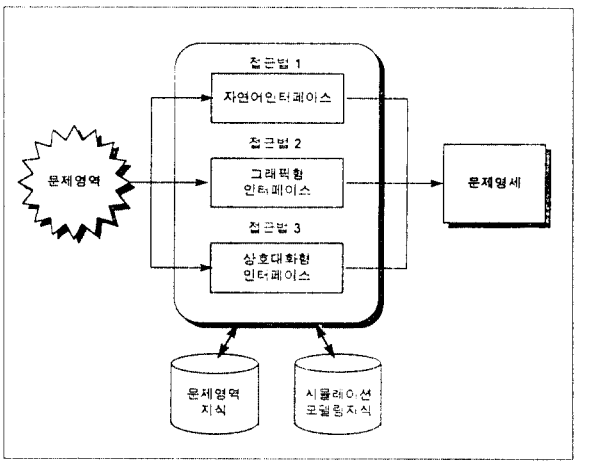
2.2 인공지능과 시뮬레이션

인공지능 (AI)과 전통적인 시뮬레이션 접근법 (approach)을 결부시켜 생각하는 현대의 연구는 두개의 분명한 영역으로 집중된다. 첫 번째 영역은 문제 명세 과정을 자동화하기 위한 인공지능의 사용이다. 두 번째는 더 어려운 영역으로써 목적 시뮬레이션 언어에서 수행 가능한 코드의 생성 자동화를 위한 인공지능의 사용이다.

가. 자동 문제 명세 (automatic problem specification)

〈그림 2〉는 인공지능과 시뮬레이션을 연관시키는 것에 대한 자동 문제 명세 접근법의 개요이다. 자동 문제 명세는 시뮬레이션 모델구축을 정의함에 있어 사용자에게 지능적인 지원기 (assistant)로 여겨질 수 있다 [14].

지식의 두 가지 형태를 포함하는 지식베이스 (knowledge base)는 자동 문제 명세를 지원하곤 한다. 이러한 지식의 형태는 문제 영역 지식 (problem domain knowledge) 과 시뮬레이션 모델링 지식 (simulation modelling knowledge)이다. 영역지식은 분명한 문제에 관련된 지식이다. 따라서, 각 문제 영역은 자기만의 지식베이스를 필요로 한다. 예로써, 전자 제조 시뮬레이션 시스템 (Electronics Manufacturing Simulation System)]은 전자공학 부품에 대한 명세된 영역 (domain-specific)이다 [7]. 시뮬레이션 모델링 지식은 목적언어에서 마지막 시뮬레이션 코드를 만들어내기 위해 필수적인 지식이다.



〈그림 2〉 자동문제 명세에 대한 세 가지 접근법

그 다음 이러한 지식베이스는 사용자와 시스템 사이의 상호작용적인 대화를 결정하는 일련의 규칙들을 정의하기 위해 사용되어진다. 시스템은 정확한 질문에 응답하는 사용자에 대해 다른 룰 베이스 (rule base)를 제공함으로써 다른 논리구조를 수행한다. 이러한 상호작용적 대화의 종결시에 시스템은 문제나 모델의 내부 명세를 정의할 것이다.

시뮬레이션 모델이나 문제 명세를 정의하는데 있어 사용자를 지원하기 위하여 〈그림 2〉에서 세 가지 접근법의 윤곽을 잡았다. 세 가지 접근법은 다음과 같다.

- 자연어 인터페이스 (Natural language interface)
- 상호 그래픽형 인터페이스 (Interactive graphical interface)
- 상호 대화형 인터페이스 (Interactive dialogue interface)

① 자연어 인터페이스 (Natural language interface)

자연어 인터페이스 (Natural language interface : NLI)는 사용자로 하여금 키보드를 거쳐 컴퓨터로 입력되는 자유로운 텍스트방식으로 문제를 명시하도록 한다. 이러한 텍스트 방식의 명세로부터 NLI는 목적언어로 시뮬레이션코드를 자동적으로 발생시키고 텍스트의 분석을 시도한다. 대부분의 NLI는 흘린 정보와 가능한 모순을 설명하기 위해 사용자와 상호작용적으로 작동한다.

가장 초창기의 NLI중의 하나는 대기행렬 시뮬레이션을 위한 자연적 언어 프로그래밍 (Natural Language Programming for Queueing Simulation : NLPQ) 이었 다 [9]. 영어로 상호작용적 대화를 통해 NLPQ시스템은 내부적인 기술을 하고, 사용자의 질문에 응답하고, 추가적인 정보를 사용자에게 묻는다. 필수적인 정보가 선택되어지면, NLPQ는 목적언어인 GPSS에서 시뮬레이션 코드를 생성시킨다.

좀더 최근의 NLI는 전자 제조 시뮬레이션 시스템 (Electronic Manufacturing Simulation System: EMSS) [7]이다. 그 시스템은 전자공학적 조합어와 표현의 사전을 바탕으로 구축되었다. EMSS는 제조공정에 대해 쓰여진 기술을 입력으로 받아들인다. 그런 다음 EMSS는 텍스트를 분석하고 텍스트를 개념적 종속표현으로 바꾼다. 그 다음 이러한 표현은 목적언어인 SIMAN [16]에서 시뮬레이션코드를 자동으로 작성하기 위해 EMSS에 의해 사용되어진다.

② 상호 그래픽형 인터페이스 (Interactive graphical interface)

사용자가 문제를 명확히 할 수 있도록 지원하는 두 번째 접근법은 NLI보다 좀 쉬운 상호 그래픽형 인터페이스 (Interactive Graphical Interface :IGI) 이다. IGI는 시뮬레이션 되어지는 시스템의 그래픽 표현을 만드는 데 있어 사용자에 의해 마우스로 선택할 수 있는 아이콘들의 메뉴로 구성된다. 한번 시스템이 만들어지면, 사용자는 키보드를 통해 아이콘에 대응하는 속성을 입력한다.

Hoshnevis[11]는 시스템을 그래픽 모델링하기 위한 목적중심적 접근법 (object-oriented approach)을 개발했다. 아이콘 라이브러리는 모델을 만드는데 있어 사용자에게 유용하다. 이러한 아이콘은 창조(create)하고 파악(seize)하고 서비스(service)하고 전이(transfer)하며 그리고 게이트 블럭들을 포함하고 있다. 시스템은 분실되고, 풍부한 정보를 체크하고 바로 사용자에게 알려준다. 시스템은 규칙을 바탕으로 하고 IBM PC에서 common LISP으로 쓰여진다. 일단 모델의 그래픽 기술이 끝나면, 시스템은 자동적으로 동등한 SLAM 시뮬레이션코드를 생성시킨다 [18].

③ 상호 대화형 인터페이스 (Interactive dialogue interface)

문제 명세를 정의하는데 있어 사용자를 지원하는 세 번째 접근법(approach)은 상호 대화형 인터페이스 (Interactive dialogue Interface :IDI) 이다. IDI는 사용자에게 묻는 일련의 질문들로 구성된다. 질문의 순서는 사용자의 응답에 의존한다. 〈그림 2〉에서 문제 명세를 정의하기 위한 세 가지 접근법 중에서 IDI는 가장 보편적으로 개발자들에 의해 사용되어진다.

몇몇 시스템은 상호 대화형 접근법을 사용하여 개발되어져 왔다. Haddock [8]은 융통성 있는 제조시스템 (Flexible manufacturing system : FMS) 시뮬레이션 생성기를 개발했다. 그 시스템은 시뮬레이션 모델로 FMS의 특성을 변환시키는데 있어 사용자를 보조하는 일련의 메뉴 중심의 (menu-driven) 화면으로 구성된다. 이러한 입력특성은 숫자와 FMS에서 배정 위치의 형태, 재고 이전 장치, 도착율과 설치, 그리고 진행시간을 포함한다. 시스템은 IBM PC에서 BASIC으로 작성되고 SIMAN 시뮬레이션 코드를 생성시킨다.

자동 운송수단 안내시스템 (Automatic Programming system for modeling automated guided vehicle systems : AGVS) 을 모델링하기 위한 자동프로그래밍 시스템[4]은 AGVS를 정의하는데 사용자를 지원하기 위해 상호작용적 대화를 사용한다. 일단 AGVS가 정의되어지면 시스템은 대응하는 SIMAN코드를 작성한다. 그 시스템은 IBM PC를 위한 Turbo Prolog로 작성되어 있다. 더욱 최근에 모델의 정의와 코드생성을 자동화하기 위해 지식 기초 모델 구축시스템 (Knowledge-based model construction : KBMC)이 개발되어졌다 [14]. 그 시스템은 사용자가 문제 명세를 정의하도록 구조적인 상호대화를 사용한다. 문제 명세는 자동적으로 대응하는 수행가능 SIMAN 시뮬레이션코드를 작성하기 위해서 KBMC 시스템에 의해 사용되어진다.

나. 자동적인 시뮬레이션 작성

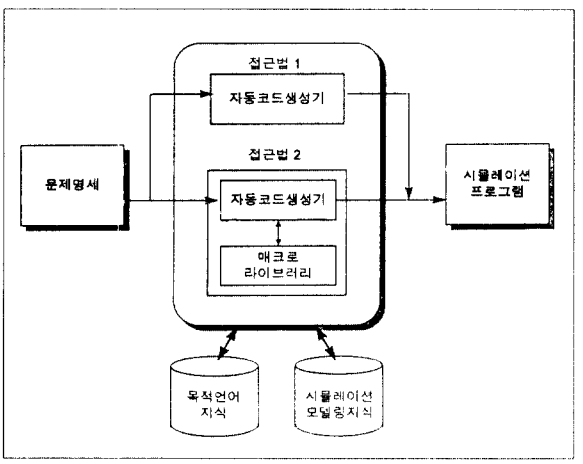
〈그림 3〉은 인공지능과 시뮬레이션을 결부시키기 위한 자동적 코드 생성 접근법의 개요이다. 기본적으로, 두 접근법은 내부적인 문제 명세를 취하고 자동적으로 목적 시뮬레이션 언어에서 수행 가능한 시뮬레이션코드를 생성시

키기 위해 존재한다. 첫 번째 접근법은 문제 명세의 내부적 표현으로부터 직접적으로 시뮬레이션 코드를 생성시키는 것이다. Sakthivel [SAKT 91]은 문제 명세를 Petri net와 로직언어로 기술하고 이를 GPSS코드로 추출하는 방법을 제시하고 있다. 이 방법은 Petri net가 갖는 모델링 능력과 분석기능을 이용하여 문제 명세를 검증하고 이를 실행코드로 변환하는 방법이다. 본 논문에서 논의되고 있는 인공지능을 기초로한 시스템(INSIMS 시스템)도 이러한 접근법을 사용하여 시뮬레이션 코드를 작성하고 있다.

두 번째 접근법은 시뮬레이션 코드의 자동 생성을 지원해주기 위해 먼저 정의된 매크로의 라이브러리를 사용하는 것이다. Pegden[16]은 SIMAN에서의 매크로를 구현하였으나 네스팅, 속성을 공유하는 것, 그리고 코드의 연결 등에서 문제점을 보여주었다. Ahmad [1]는 시뮬레이션 인터페이스로서의 매크로를 제시하고 있다. 이 매크로를 통하여 사용자는 시뮬레이션 모델을 쉽게 구축할 수 있다. 이러한 연구법의 장점은 문학에서 주로 논의된 것들보다 더욱 전문화된 문제를 해결하는 능력이다. 매크로의 단점은 첫째, 대부분의 매크로가 영역 명시적 (domain-specific)이라는 것이다. 즉, 다른 문제영역을 해결하기 위해서는 추가적인 매크로가 필요하다는 것이다. 둘째, 매크로의 수가 많아질 경우 모델링의 유연성과 매크로와 데이터 인터페이스간의 연계성을 유지하기 어려워진다는 점이다.

지식의 두 가지 형태가 자동적 시뮬레이션 생성을 지원하기 위해 사용되어진다. 이러한 지식의 형태들은 시뮬레이션 모델링 지식 (simulation modelling knowledge)과 목적 언어 지식 (target language knowledge)이다.

시뮬레이션 모델링 지식은 〈그림 3〉에서 자동 문제 명세를 위해 사용되어진 것과 같은 기초적인 지식이다. 목적 언어 지식베이스는 내부적인 문제 명세로부터 적당한 시뮬레이션코드의 라인을 구조화시키기 위한 일련의 규칙을 정의하는데 사용되어진다.



〈그림 3〉 자동 시뮬레이션 작성에 대한 두 가지 접근법

3. 자동 명세 및 프로그래밍 시스템

자동 명세와 프로그래밍 시스템은 앞의 섹션에서 윤곽을 잡은 개념적 토대를 사용하여 구조화되어온 전형적인 시뮬레이션 도구이다. 특히, INSIMS시스템은 상호 사용자 대화를 통해 제조시스템 모델러들의 문제정의를 지원하기 위한 시뮬레이션 지원기 (simulation assistant) 이다. INSIMS시스템은 문제명세를 정의하기 위해 〈그림 2〉에서 나타낸 접근법들 중에서 상호대화형 인터페이스를 사용한다. 사용자들은 INSIMS 시스템이 요구하는 내용을 입력하기만 하면 문제명세를 얻을 수 있다. 문제명세가 정의된 후에 INSIMS시스템은 자동적으로 이에 대응되는 SLAM II 시뮬레이션 코드를 생성시킨다.

INSIMS시스템의 개발은 다음과 같은 단계로 이루어졌다.

- 제조영역의 선택과 선택된 영역에 대한 보통의 제조 기능의 정의
- 사용자 인터페이스 프로그램의 구현
- 제조기능의 SLAM 매크로의 설계
- 자동 코드 생성 프로그램의 구현

3.1 제조영역의 선택과 선택된 영역에 대한 보통의 제조 기능의 정의

선택된 제조영역은 다음의 특징을 갖는 제조시스템으로 구성된다. 이러한 제조영역의 특징은 매우 일반적이다. 이를 좀더 세분화하여 특정영역에 보다 더 적합한 제조영역을 만들 수도 있겠지만, 본 연구에서는 보다 넓은 응용분야를 갖게 하기 위하여 포괄적이고 일반화하였다.

가. 제조부문과 작동의 특징

① 작업센타 (Work Centers : WCs)

작업센타는 실질적인 제조작업이 이루어지는 부분으로, 각각의 작업센타(work center)는 동일한 서비스를 하는 몇 개의 기계를 갖는다. 1회분 크기의 작동 후에 각각의 기계는 지속적인 시간에 대한 조정을 필요로 한다. 각 작업센타는 대기중인 부품에 대한 신속한 처리규칙을 가질 수 있다. 각 작업센타에는 공간의 크기가 한정된 대기행렬을 갖는다.

② 검사기 (Inspector)

검사기는 가공된 부품에 대한 재작업 여부나 스크랩이 필요한 몇 개의 항목을 검사한다. 재작업을 필요로 하는 부품은 선택된 WC에서 신속하게 처리됨에 있어 최우선권을 갖는다. 재 작업된 부품은 시스템을 빠져나간다.

③ 수송기 (Transporter)

수송기는 현재 WCs로 부터 다음 WCs로 옮겨준다. 이 시스템에서 고려될 수 있는 수송기의 형태는 컨베이어(Conveyor) 이다. 각 컨베이어는 일관된 수송속도를 갖는다.

나. 부품 특징 (Item Characteristics)

한 형태의 부품이 이 공장에 도착하고 가공시간을 갖는다. 모든 가공 후에, 각 부품에는 결함이 발생할 수 있다. 발생된 결함은 재작업(reworking)이나 불필요한 것을 버리는 작업(scrapping)을 필요로 한다.

3.2 상호 대화형 인터페이스

상호 대화형 인터페이스 프로그램은 C언어로 작성되었다. 〈표 1〉은 INSIMS에서 사용되는 분포함수와 신속처리규칙을 나타낸다. 이것들은 모델 명세에 쓰여진다. 사용자는 메뉴형태로 제공되는 사용자 인터페이스에서 분포함수와 신속처리규칙을 선택할 수 있다.

사용자 인터페이스 프로그램은 지적 명세 취득 (intelligent specification acquisition) 후 모델 파일을 만들고 다음과 같은 변수들과 자료들로 분류한다. 모델 파일은 상호

〈표 1〉 확률분포 및 신속처리규칙

Probability Distribution Type	Dispatching Rule Type
1. exponential	1. EDD : Earlist Due Date
2. uniform	2. SPT : Shortest Processing Time
3. normal	3. FIFO : First in First Out
4. triangular	4. LIFO : Last In First Out
5. constant	

〈표 2〉 모델 명세를 위한 변수 정의

```
FINAME : /* 모델 화일 네임 */
MRNAME : /* 모델러 네임 */
MONAME : /* 모델 네임 */
MDATE : /* 모델링 날짜 */
WNUM : /* 작업센타 수 */
MNUM[k] : /* K는 1부터 WNUM까지 각 작업센타의 기계수 */
ST[k] : /* K는 1부터 WNUM까지 k번째 작업센타의 set-up 횟수 */
BUF[k,1] : /* K는 1부터 WNUM까지 k번째 작업센타의 대기행렬의 크기 */
BUF[k,2] : /* K는 1부터 WNUM까지k번째 작업센타의 초기재고 크기 */
DISP[k] : /* K는 1부터 WNUM까지 k번째 작업센타의 신속히 처리하는 규칙 */
INSRATE : /* 사항의 검사율*/
INBUFF[1] : /* 검사센타의 버퍼의 크기 */
INBUFF[2] : /* 검사센타의 초기 재고크기 */
NRENUM : /* 검사센타의 서비스 자원의 수 */
RERATE : /* 검사후 재작업 비율 */
SRRATE : /* 검사후 스크랩 비율 */
RECENTER : /* 재작업 센타 위치 */
BATCH : /* 부품의 1회분 크기 (batch size) */
ARRDIS : /* 도착 분포 형태 */
ARP[0] : /* 도착분포의 인자 */
ARP[1] :
ARP[2] :
INSDIS : /* 검사분포의 형태 */
INP[0] : /* 검사분포의 인자 */
INP[1] :
INP[2] :
SERDIS[k] : /* K는 1부터 WNUM까지 k번째 작업센타의 서비스 분포형태 */SERP[K,0] : /* K번째 WCs 서비스분포의 인자 */
SERP[k,1] :
SERP[k,2] :
CONVEY[k+1] : /* 각 간격의 수송시간 */
STIME : /* 시뮬레이션 시간 */
```

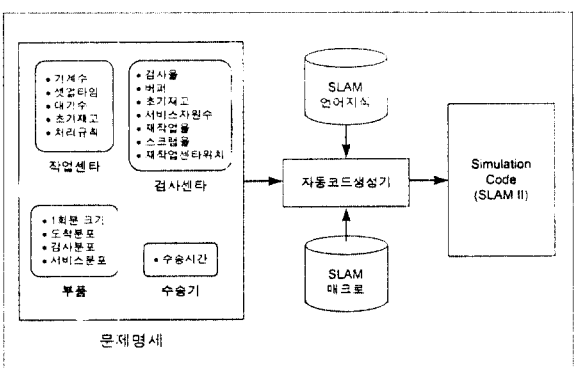
적이고 지능적인 대화 사용자 인터페이스를 통해 만들어진다. 이 모델 파일은 코드생성 이전에 수정되어질 수 있다.

3.3 제조기능의 페항 SLAM 매크로의 설계

일단 제조기능이 정의되면, 확률분포, 작업센타, 검사센타, 컨베이어, 출력통계 등의 시뮬레이션 함수와 루틴 등을 SLAM II 매크로를 이용하여 만든다. 우리는 자동명세의 영역을 위에서 설명한 제조영역으로 한정하였기 때문에 이러한 특성에 맞는 함수들만을 우선적으로 포함하였다. 상세한 설명은 자동 코드 생성기 프로그램에 쓰여 있다.

3.4 자동 코드 생성기

모든 영역을 지원하는 자동코드 생성기를 개발하는 것은 대단히 어렵고 효용성면에서도 문제가 있기 때문에, 본 연구에서는 자동코드 생성의 영역을 〈표 2〉와 같은 특정 제조 영역에 한정하였다. 이러한 접근법은 실제 문제에 대한 업무 영역 지식을 반영하여 보다 실제적인 코드를 얻어낼 수 있다. 이러한 제조영역의 명세를 SLAM II 코드로 변환하는 방법은 〈그림 4〉와 같다.



〈그림 4〉 자동코드 생성기 개요

자동 코드 생성기 프로그램은 상호 사용자 대화로부터 나온 결과인 문제 명세를 SLAM II 시뮬레이션 매크로와 결합시키고, SLAM II 언어의 지식을 이용하여 상응하는 제조공정의 SLAM II 코드로 변환한다. 코드 생성기 프로그램은 IBM PC 호환기종에서 C언어로 작성되어졌다.

자동 코드 생성기의 변환 알고리즘은 아래와 같다.

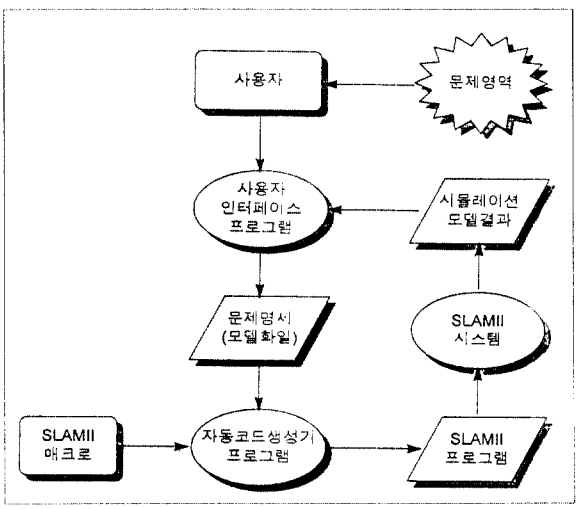
SLAM II 에서 다음의 특별한 변수는 코드 생성을 확인하기 위해 사용되어졌다.

〈표 3〉 자동 프로그래밍을 위한 특수 변수

1. XX(K) : K는 1부터 WNUM까지 K번째 작업센타의 기계의 조정을 위해 뱃치사이즈를 세는 전역 (global) SLAM II 변수
2. ATRIB(1) : 도착시간을 표시하는데 사용되어진다.
3. ATRIB(2) : 계산된 의무일 (due date) 를 보유하는 데 사용되어진다. 의무일은 다음과 같이 계산되어진다. 도착시간 + N * 예상되는 진행시간 (N의 값은 4로 사용한다)
4. ATRIB(2+K) : K번째 작업센타에서 진행시간을 표시하는 데 사용되어진다.

3.5 INSIMS 시스템 작동

〈그림 5〉는 INSIMS 시스템 작동의 개요이다. 일단 사용자가 문제영역에 들어가면, 사용자는 IBM PC에 앉아 인터페이스 프로그램에 의해 표현되는 질문에 대답한다. 그 대답을 기초로, 인터페이스 프로그램은 내부적인 문제명세 화일(모델화일)을 만든다. 이 화일은 제조 공정망흐름과 모든 작업센타에 대한 속성과 기능, 그리고 재고가 있는 시점을 포함한다. 모델 화일은 목적 언어인 SLAM



〈그림 5〉 INSIMS 시스템 개요

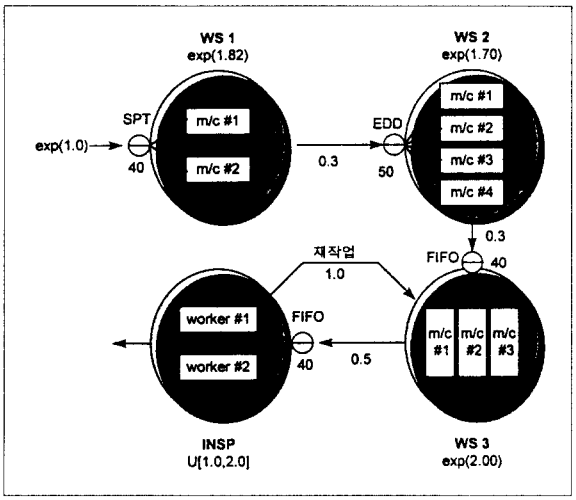
II에서 시뮬레이션 프로그램을 생성시키는 자동 코드 생성기 프로그램으로 입력할 때 사용되어 진다. 자동 코드 생성기 프로그램의 출력은 SLAM II 시뮬레이션 시스템 위치로 다운로드 되어지는 하나의 SLAM II 프로그램 화일이다. 그리고 나서 사용자는 그 프로그램을 수행한다. 출력화일은 PC에서 프린트되거나 디스켓에 저장된다.

3.6 실 례 (Example)

가. 선택된 제조시스템의 설명

개발된 실례모델은 〈그림 6〉와 같다. 세개의 작업센타(work centers)와 하나의 검사센타(inspect center)로 공장의 흐름을 나타낸다. 첫 번째 작업센타는 두대의 기계를 갖고 있고 두 번째 작업센타는 네대의 기계를 갖고 있다. 세 번째 작업센타는 세대의 기계를 갖고 있고 검사센타는 두 명의 근로자가 있다. 각 작업센타의 기계는 똑같은 작동을 하는 능력을 갖고 있다. 각 작업센타의 작동은 각각의 다른 작업센타와는 다르다. 각 작동은 다음과 같은 순서로 수행되어져야 한다. 작업센타1이 첫 번째, 작업센타2가 두 번째, 작업센타3이 세 번째, 검사센타가 네 번째이다. 공장의 도착(도착의 형태는 도착평균비율 1.0인 지수분포이다) 에 따른 각각의 작업은 작업센타1에서 첫 번째 비사용기계로 간다. 만약 모든 기계가 사용중일때는 작업은 기계가 사용가능할 때까지 대기한다. 작업센타1에서 과정을 거친 후 작업은 작업센타2로 신속히 넘어간다. 그리고 비사용중인 첫 번째 기계로 작업되어진다. 작업센타2에서 처리되어진 후 작업은 곧바로 작업센타3로 넘어가고 비사용중인 첫 번째 기계로 작업이 시작된다. 작업센타3에서 처리가 끝난 후 작업은 검사센타로 넘어간다.

각 작업센타의 대기행렬의 크기는 작업센타1에서 40단위, 작업센타2에서 50단위, 작업센타3에서 40단위, 검사센타에서 40단위이다. 각 작업센타와 검사센타의 초기 재고 크기는 제로이다. 신속히 다루어지는 규칙은 작업센타1의 SPT, 작업센타2의 EDD, 작업센타3의 FIFO, 그리고 검사센타의 FIFO이다. 작업센타1의 작동은 평균1.82(시간단위)의 지수분포이다. 작업센타2는 평균1.70, 작업센타3은 평균2.00의 지수분포이다. 그리고 검사센타의 작동은 최하치 1.00, 최고치 2.00의 균일분포(uniform distribution) 이다. 모델은 각 작업이 작업센타1에서 작업센타2로 넘어가는데 0.3단위, 작업센타2에서 작업센타3으로 넘어가는데



〈그림 6〉 제조시스템 예제

0.3단위, 작업센타3에서 검사센타로 넘어가는데 0.5단위, 그리고 검사센타에서 지속적인 속도로 작업을 옮기는 컨베이어벨트로 여겨질 수 있는 재작업센타 (작업센타3)로 넘어가는데 1.0단위의 지속적 시간을 필요로 한다는 것을 추정한다. 공장에서 기계가 작동(1회분 크기 : batch size)의 특정 숫자를 수행한 후 새로운 도구를 필요로 한다. 각각의 기계가 50번의 작업을 수행했을 때 기계는 작업센타1과 2에서 0.5시간단위, 작업센타3에서 0.7시간단위의 수리지연 후에 다시 서비스를 할 수 있는 상태로 된다. 이러한 과정은 모든 기계에 있어 시뮬레이션기간을 통해 반복된다. 또한 모델은 종결된 작업의 10%가 검사되어지고 검사되어진 작업의 10%가 재작업을 필요로 하고 검사된 작업의 10%가 스크랩을 필요로 한다고 추정한다. 시뮬레이션 기간은 10000시간 단위이다.

나. 모델 명세 후 모델화일의 내용

```
FINAME : mddel.mod /* 모델 화일명 */
MRNAME : ChungNH /* 모델러 명 */
MONAME : Simulation Project /* 모델명 */
MDATE : 12/29/92 /* 모델링 날짜 */

WNUM : 3 /* 워크센터의 수 */
MNUM : 2 /* 기계의 수 */
```

```
ST: 0.500000 /* 조정 시간 */
BUF[0,1] : 40 /* 작업센터 1의 대기행렬의 크기 */
BUF[0,2] : 0 /* 작업센터 1의 초기재고크기 */
DISP : 2 /* 신속한 업무처리 룰 */
MNUM : 4 /* 기계의 수 */
ST : 0.500000 /* 조정시간 */
BUF[1,1] : 50 /* 작업센터 2의 대기행렬의 크기 */
BUF[1,2] : 0 /* 작업센터 2의 초기재고크기 */
DISP : 1 /* 신속한 업무처리 룰 */
MNUM : 3 /* 기계의 수 */
ST : 0.700000 /* 조정시간 */
BUF[2,1] : 40 /* 작업센터 3의 대기행렬의 크기 */
BUF[2,2] : 0 /* 작업센터 3의 초기재고크기 */
DISP : 3 /* 신속한업무처리 룰 */

INSRATE : 0.100000 /* 검사비율 */
INBUFF[1] : 40 /* 검사센터의 버퍼사이즈 */
INBUFF[2] : 0 /* 검사센터의 초기재고사이즈 */
INRENUM : 2 /* 서비스자원의 수 */
RERATE : 0.100000 /* 재작업 비율 */
SRRATE : 0.100000 /* 스크랩 비율 */
RECENTER : 3 /* 재작업 센타 */

BATCH : 50 /* 1회분 크기 */
ARRDIS : 1 /* 도착형태 */
ARP[0] : 1.000000 /* 인자 (parameter) 1 */
ARP[1] : 0.000000 /* 인자 2 */
ARP[2] : 0.000000 /* 인자 3 */
INSDIS : 2 /* 검사형태 */
INP[0] : 1.000000 /* 인자 1 */
INP[1] : 2.000000 /* 인자 2 */
INP[2] : 0.000000 /* 인자 3 */
SERDIS[0] : 1 /* 작업센타 1의 서비스 형태 */
SERP[0] : 1.820000 /* WCs (Work Centers) 1에 대한 인
          자 1 */
SERP[1] : 0.000000 /* WCs 1에 대한 인자 2 */
SERP[2] : 0.000000 /* WCs 1에 대한 인자 3 */
SERDIS[1] : 1 /* 작업센타 2의 서비스형태 */
SERP[0] : 1.700000 /* WCs 2에 대한 인자 1 */
SERP[1] : 0.000000 /* WCs 2에 대한 인자 2 */
SERP[2] : 0.000000 /* WCs 2에 대한 인자 3 */
SERDIS[2] : 1 /* 작업센타 3의 서비스형태 */
SERP[0] : 2.000000 /* WCs 3에 대한 인자 1 */
SERP[1] : 0.000000 /* WCs 3에 대한 인자 2 */
SERP[2] : 0.000000 /* WCs 3에 대한 인자 3 */

Convey module
0.300000 /* 1에서 2로 운반 */
0.300000 /* 2에서 3으로 */
0.500000 /* 3에서 검사(inspect)로 */
1.000000 /* 검사에서 3(재작업센타)으로 */
10000.000000 /* 시뮬레이션 시간 */
```

다. 자동시뮬레이션코드생성후 SLAM II 코드

```
GEN,ChungNH,Simulation Project, 12/29/92,1;
LIMITS,5,6,100;
INTLC,XX(1)=0,XX(2)=0,XX(3) = 0;
PRIORITY/1,LVF(3);
PRIORITY/2,LVF(2);
PRIORITY/3,FIFO;

NETWORK
    RESOURCE/MC1(2),1;
    RESOURCE/MC2(4),2;
    RESOURCE/MC3(3),5,3;

    CREATE,EXPON(1.000000),,1;

    ASSIGN,ATRIB(2)=ATRIB(1)+4*1.820000+4*
                    1.700000+4*2.000000;
    ASSIGN,ATRIB(3)=EXPON(1.820000);
    ASSIGN,ATRIB(4)=EXPON(1.700000);
    ASSIGN,ATRIB(5)=EXPON(2.000000);
    ASSIGN,ATRIB(6)=UNFRM(1.000000,2.000000);

WC1 AWAIT(1/40),MC1/1;
    ACT/1,ATRIB(3);
    FREE,MC1/1;
```

```
    ASSIGN,XX(1)=XX(1)+1;

GOON,2;
    ACT/2,0.300000,,WC2;
    ACT;
    GOON;
    ACT,,XX(1).GE.50.AND.XX(1).LE.51,SET1;
    ACT;
    TERM;

SETQ ALTER,MC1/-1;
    ACT/3,0.500000;
    ALTER,MC1/1;
    ASSIGN,XX(1)=0;
    TERM;

WC2 AWAIT(2/50),MC2/1,BLOCK;
    ACT/4,ATRIB(4);
    FREE,MC2/1;
    ASSIGN,XX(2)=XX(2)+1;

    GOON,2;
    ACT/5,0.300000,,WC3;
    ACT;
    GOON;
    ACT,,XX(2).GE.50.AND.XX(2).LE.51,SET2;
    ACT;
    TERM;

SET2 ALTER,MC2/-1;
    ACT/6,0.500000;
    ALTER,MC2/1;
    ASSIGN,XX(2)=0;
    TERM;

WC3 AWAIT(3/40),MC3/1,BLOCK;
    ACT/7,ATRIB(5);
    FREE,MC3/1;
    ASSIGN,XX(3)=XX(3)+1;

    GOON,2;
    ACT/8,0.500000,0.100000,INSP ;
    ACT/9,0.500000,0.900000,COL1 ;
    ACT;
    GOON;
    ACT,,XX(3).GE.50.AND.XX(3).LE.51,SET3;
    ACT;
    TERM;

SET3 ALTER,MC3/-1;
    ACT/10,0.700000;
    ALTER,MC3/1;
    ASSIGN,XX(3)=0;
    TERM;

INSP QUEUE(4),0,40,BLOCK;
    ACT(2)/11,ATRIB(6);
    GOON 1;
    ACT/12,,0.100000,SCRP;
    ACT/13,1.000000,0.100000,REW;
    ACT/14,0.800000,COL1;

REW AWAIT(5),MC3/1;
    ACT/15,EXPON(2.000000);
    FREE,MC3/1;
    ASSIGN,XX(3)=XX(3)+1;
    ACT,,COL1;

COL1 GOON;
    ACT,,TNOW.GT.500,COL2;
    ACT;
    TERM;
COL2 COLT,INT(1),SYETEM TIME;
    ACT,,ATRI(2).LT.TNOW,COL3;
    ACT;
    TERM;
COL3 COLT,INT(2),TARDINESS;
    TERM;
    SCRP COLT,BET,SCRAP INTERVAL;
    TERM;
ENDNETWORK;

INIT,0,10500.000000;
FIN;
```

라. 시뮬레이션 출력 : SLAM II 요약 보고서

S L A M I I S U M M A R Y R E P O R T

SIMULATION PROJECT Simulation Project — BY ChungNH
DATE 12/29/1992 — RUN NUMBER 1 OF 1

CURRENT TIME .1050E+05
STATISTICAL ARRAYS CLEARED AT TIME .0000E+00

****STATISTICS FOR VARIABLES BASED ON OBSERVATION****

	MEAN VALUE	STANDARD DEVIATION	COEFF. OF VARIATION	MINIMUM VALUE	MAXIMUM VALUE	NO.OF OBS
SYETEM TIME	.111E+02	.138E+02	.124E+01	.131E+01	.390E+03	9964
TARDINESS	.248E+02	.436E+02	.176E+01	.879E-02	.368E+03	533
SCRAP INTERVAL	.102E+03	.113E+03	.111E+01	.469E+00	.537E+03	99

****FILE STATISTICS****

FILE NUMBER	ASSOC NODE LABEL/TYPE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAIT TIME
1	WC1 AWAIT	3.374	3.017	19	1	3.355
2	WC2 AWAIT	.057	.336	6	1	.056
3	WC3 AWAIT	.940	2.055	20	0	.936
4	INSP QUEUE	.000	.020	1	0	.004
5	REW AWAIT	.003	.051	1	0	.273
6	CALENDAR	7.915	1.994	17	11	.350

****REGULAR ACTIVITY STATISTICS****

ACTIVITY INDEX/LABEL	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTIL	CURRENT UTIL	ENTITY COUNT
1	1.8349	.4699	2	2	10555
2	.3015	.5430	5	2	10553
3	.0136	.1297	2	0	286
4	1.7171	1.2103	4	4	10548
5	.3013	.5461	5	0	10548
6	.0133	.1282	2	0	279
7	2.0287	1.0357	3	2	10546
8	.0511	.2258	2	0	1074
9	.4510	.6704	6	0	9472
10	.0204	.1659	2	0	306
12	.0000	.0000	1	0	100
13	.0094	.0966	1	0	99
14	.0000	.0000	1	0	875
15	.0191	.1384	2	0	99

****SERVICE ACTIVITY STATISTICS****

ACT NUM	ACT START	LABEL OR NODE	SER CAP	AVERAGE UTIL	STD DEV	CUR UTIL	AVERAGE BLOCK	MAX TME/SER	IDL TME/SER	MAX BSY ENT CNT
11	INSP	QUEUE	2	.153	.39	0	.00	2.00	2.00	1074

****RESOURCE STATISTICS****

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTIL	STANDARD DEVIATION	MAXIMUN UTIL	CURRENT UTIL
1	MC1	2	1.83	.470	2	2
2	MC2	4	1.72	1.210	4	4
3	MC3	3	2.05	1.036	3	2

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	MC1	0	.1515	-1	2
2	MC2	0	2.2696	-1	4
3	MC3	1	.9317	-1	3

4. 결론과 심화 연구

INSIMS 시스템은 상호대화형 인터페이스를 통하여 문제 명세를 추출하고 이를 이용하여 SLAM II 코드를 자동 생산하는 지능형 시뮬레이션 환경이다. 여기에는 몇 가지 인공지능개념이 포함되어 있다. 이러한 개념 중에 하나는 자동 프로그래밍(AP)이다. 자동 프로그램은 컴퓨터 프로그래밍 과정(Barr and Feingenbaum 1982)의 자동적인 측면에서 인공지능의 적용으로써 정의되어진다. 이 자동화는 다른 프로그램 즉, 컴퓨터 프로그래밍 과정을 명시하는 수준을 끌어올린 AP시스템에 의해 이루어졌다. 바꿔 말하면, AP시스템은 프로그램(Barr and Feigenbaum 1982)을 정의하고 작성하기 위한 환경을 개선함으로써 프로그래머가 프로그램을 작성하는 것을 돕기 위한 프로그램이다. INSIMS는 시뮬레이션 하는 사람이 SLAM II코드를 작성하는 것을 돕는 AP시스템이다.

INSIMS시스템 내에서 또 다른 인공지능개념은 상호대화를 통한 인터페이스 프로그램에서 추출한 규칙(rule)과 자동 코드 생성기 프로그램에서 구축된 규칙의 사용이다. 이러한 룰은 기본적으로 IF-THEN 문장의 형태를 하고 C-코드에 끼워 넣어진다. 예를 들면, 추출규칙은 제조문제영역에 대한 지식의 사용과 시뮬레이션의 모델링으로 정의되어진다. 생성규칙은 시뮬레이션의 모델링과 목적 시뮬레이션언어(SLAM II)에 대한 지식을 사용함으로 정의되어진다.

문제명세 정의를 위한 지능적인 사용자 인터페이스를 통하여 얻을 수 있는 이점은 아래와 같다.

- 모델화된 시스템에서 정보를 얻기 위해 구조화된 프로시저를 제공한다.
- 문제명세의 정의를 빠르게 처리한다.
- 문제명세의 완벽하고 상세한 정의를 확실케 한다.
- 자동코드발생기가 목적 시뮬레이션 언어에서 수행 가능한 코드를 작성하기 위해 할 수 있는 전체의 모델링의 질을 개선한다.
- 목적시뮬레이션언어의 모델러가 필요로 하는 지식을 줄여준다.
- 시뮬레이션코드를 바르고 정확하게 작성한다.
- 재조시스템의 신속한 원형을 가능케 한다.
- 모범적 에러에 걸리지 않는 수행 가능한 시뮬레이션 코드를 만든다.
- 읽고, 추적하고, 수정하기 쉬운 구조화된 시뮬레이션 코드를 낳는다.
- 시뮬레이션코드의 전반적인 명확성을 개선한다.

대부분의 시뮬레이션 지원기는 조건적 양도처럼 뻗어있는 복잡한 시뮬레이션코드를 생성시키는데 어려움이 있다. INSIMS시스템은 SLAM II 매크로를 만듦으로써 이러한 단점을 극복한다. 필수적인 분기한 로직은 이러한 매크로 내에 끼워져 있다. SLAM II 매크로의 라이브러리는 또한 INSIMS시스템의 약점이다. 매크로에 의해 정의되어진 제조기능은 INSIMS 시스템영역을 종속적으로 만든다. 추가적인 기능을 모델 하는 것은 다른 매크로의 작성을 필요로 한다. INSIMS시스템이 현실의 문제를 해결하는데 사용되어질 때, 더 많은 매크로 서브루틴이 라이브러리에 추가되고 완전히 새로운 라이브러리가 개발될 것으로 기대된다.

이 시스템에서 몇 개의 지식베이스와 상징적인 조작언어 (인공지능 언어: LISP, CLIPS 등)를 사용한 프로그램의 기록으로 이 시스템에서 사용되는 지식의 고립문제는 아직까지 더 해야할 작업으로 남아 있다.

참고문헌 (Reference)

[1] Ahmad, M.M. and F.A. Stam, "Macro Requirement Within a Simulation Interface", *Simulation*, 1993, vol. 60, no. 3, pp. 181-190.

[2] Bernard J.Schroer, "A Simulation Assistant for Modeling Manufacturing systems", *Simulation*, Nov 1989, pp. 201-206.

[3] W. E., "Introduction to Simulation", *In Proceedings of the 1987 Winter simulation Conference*, Atlanta, Georgia, 1987, pp. 7-15.

[4] "Automatic Programming of AGVS simulation model", *Proceedings of the 1987 Winter Simulation Conference*, Atlanta, Georgia, pp. 703-708.

[5] Conway, R. W., "Some Tactical Problems in Simulation Methods", *Memo RM-3244-PR*, Rand Corporation, October 1962.

[6] Donnie R. Ford and Bernard J.Schroer, "An Expert

manufacturing simulation system", *Simulation*, May 1987, pp. 193-200.

[7] Ford, D.R., and Schroer, B.J., "An Expert manufacturing simulation system ", *Simulation*, 48, 1987, pp. 193-200.

[8] Haddock, J., and Davis, R.P., "Building a simulation generator for manufacturing cell design and control", *Annual International Industrial Engineering Spring Conference Proceedings*, Los Angeles, California, pp. 237-244.

[9] Heidorn, G.E., "English as a very high level language for simulation programming", *SIGPLAN Notice*, 9, pp. 91-100.

[10] Joseph M. Mellichamp & Ahmed F.A. Wahab, "Process Planning simulation : An FMS modeling tool for engineers", *Simulation*, May 1987, pp. 186-192.

[11] Hoshnevis, B., and Chen, A.P., "An Expert simulation model builder", *Intelligent Simulation Enviornment*, 17, pp. 129-132.

[12] Mathewson, S.C., "Simulation support enviorments", *Computer Modeling for Discrete, Simulation*, M. Pidd (Ed.), John Willey & sons Ltd., 1989, pp. 57-100.

[13] Joseph M. Millichamp and Ahmed F.A.Wahab, "An Expert system for FMS Design", *Simulation*, May 1987, pp. 201-208.

[14] Murray, K.J., and Sheppard, S.V., "Knowledge-based simulation model specification.", *Simulation*, 50, pp. 112-119.

[15] Paul J. Nolan et al., "ISI-An Enviorment for the engineering use of general purpose simulation languages", *Simulation*, Jan 1991, pp. 41-47.

[16] Pegdn, C.D., *Introduction to SIMAN*, System Modeling Corporation, State College, Pennsylvania, November 1987.

[17] Prakash S. and R.E. Shannon, "Development of a goal directed simulation environment for discrete part manufacturing systems," Simulation, 1993, vol. 61, no. 2, pp. 102-115.

[18] Pritker, A. B., and Pegden, C.D., *Introduction to Simulation and SLAM*, Network: Johnson Wiley & Sons, 1979.

[19] Sakthivel, S. and Ritu Agarwal, "Knowledge-based model construction for simulating information systems", *Simulation*, 1992, vol. 59, no. 4, pp. 223-236.

[20] Schroer, Bernard J. and Fant T. Tseng, "An Intelligent assistant for manufacturing system simulation", *International Journal of Production Research*, 1991, Vol 27, No.10, 1665-1683.

[21] Rooks, Michael, "A User-Centered Paradigm for Interactive Simulation", *Simulation*, 1993, vol. 60, no. 3, pp. 168-177.

● 저자소개 ●

이건창

저자는 성균관대학교 경영학과를 졸업하고, 한국과학기술원(KAIST) 경영과학과에서 석사와 박사학위 (분야 : 경영정보)를 취득하였다. 1990년부터 1995년까지 경기대학교 경영정보학과 교수로 근무한 후 1995년 2학기부터 성균관대학 경영학부 교수로 재직중이다. 주요연구분야는 인공지능을 이용한 의사결정분야이며, 특히 인공신경망과 전문가시스템의 결합을 통한 보다 지능적인 의사결정지원시스템 개발에 관심이 있다. 국제학술지로는 Decision Support Systems, Fuzzy Sets and Systems, Expert Systems, Decision Science, Interligent Systems in Accounting, Finance & Management 등에 출간되었거나 출간될 예정이다. 국내학술지로는 경영학연구, 경영과학회지, 경영정보학연구, 시뮬레이

션지, 정보과학회, 전문가시스템연구지 등에 출간되었거나 출간될 예정이다. 현재 Fuzzy Sets and Systems의 심사위원으로 활동중이다.

이양규

저자는 고려대학교 경영학과를 졸업하고, 한국과학기술원 경영과학과에서 석사와 박사학위를 취득하였다. 1992년 10월부터 1995년 2월까지 국방정보체계연구소 의사결정지원기술실장으로 근무하였으며, 1995년 3월부터 서원대학교 경영정보학과에 재직중이다. 주요 관심 분야는 페트리네트를 이용한 생산시스템 모델링 및 분석, 소프트웨어 개발 방법론, 의사결정지원시스템, 객체지향 시스템 등이다.