▩ 연구논문

Smalltalk 패러다임을 이용한 객체지향 시뮬레이션기반 전문가시스템*
- Object-Oriented Simulation-Based Expert System Using a Smalltalk Paradigm -

김 선 욱**
Kim, Sun-Uk
양 문 희**
Yang, Moon Hee

Abstract

Simulation-Based Expert System(SIMBES) is a very effective tool to solve complex and hard problems. The SIMBES model includes a simulator, a feature extractor, a machine learning system, a performance evaluator, and a Knowledge-Based Expert System(KBES). Since SIMBES depends on problem domains, a schedule-based material requirements planning problem, which is NP-hard, was selected to exemplify the SIMBES model.

To implement the SIMBES application in Smalltalk paradigm, a system class hierarchy was constructed. The hierarchy consists of five large classes such as Job Generator, Job Scheduler, Job Evaluator, Inference Engine, and Executive System. Several classes inside these classes were identified. Additionally, instance protocols about all classes have been described in terms of messages and pseudo methods. These protocols can be implemented easily by any other object-oriented languages. Furthermore, these results may be used as a skeletal system to develop a new SIMBES efficiently, especially when the application is related to other scheduling problems.

1. 서론

복잡하고 어려운 문제를 풀기 위한 현실적인 방법으로 전문가시스템과 시뮬레이션이 널리 이용되고 있다. 문제 해결을 좌우하는 요소로 전문가시스템은 대개 전문지식을 갖고 있는 전문가에 의존하는 반면 시뮬레이션은 적절한 시뮬레이션 모델링 능력이 대단히 중요하다. 결국 전문가의 확보와 모델링 능력은 시스템 성능에 지대한 영향을 미친다. 전문가시스템과 시뮬레이션 방법은 각각 장단점을 갖는다. 전자의 방법은 상대적으로 문제해결을 신속하게 할 수 있으나 지식획득이 애로공정으로 널리 알려져 있다. 반면에 후자의 방법은 시간이 과다하게 소요되며 유도되는 많은 자료들은 일회용으로 버려진다.

결국 이들 방법은 각각 지식획득의 어려움과 시뮬레이션중 축적되는 많은 정보들의 비효율적 이용으로 어려움을 겪고 있다. 이러한 어려움을 해소하기 위한 하나의 방법으로 시뮬레이션을 통한 또는 시뮬레이션기반 전문가시스템(SIMBES: Simulation-Based Expert System) 패러다임이 이용될 수 있다[1].

* 이 연구는 2000학년도 단국대학교 대학연구비의 지원으로 연구되었음

** 단국대학교 산업공학전공

SIMBES를 구축하기 위해서는 여러 구성요소(다음장 언급)가 반드시 구비되어야 한다. 그러나 이 요소들의 구현은 시간과 노력을 요하므로 가능한 일회용의 사용을 지양하고 코드의 재활용성을 높여야 한다. 더욱 중요한 것은 시뮬레이션을 통한 전문가시스템 패러다임의 속성상 일단 전 과정을 통하여 전문가시스템이 구축되었다면 그 이전의 결과물들은 거의 의미를 지니지 않는다는 점이다. 뿐만 아니라 시뮬레이션을 통한 전문가시스템이 대상으로 하는 문제의 영역은 쉽게 변화할 수 있다. 예를들면, 주문과 공장의 자원에 입각하여 일정계획을 수립할 때 이들의 내역과 구성이 변경될 가능성은 대단히 높다. 이것은 자연스럽게 기존 방법의 재검토를 불가피하게 만들며 많은 비용의 수반을 요구한다. 따라서 코드의 재활용성을 극대화 하는 것은 시뮬레이션을 통한 전문가시스템구축의 생산성에 바로 직결된다.

코드의 생산성을 도모하는 방안으로 객체지향기술이 최근 각광을 받고 있다. 객체지향기술은 객체지향 프로그래밍 언어에 기반하여 소프트웨어를 개발하는 기술이다. 객체지향 프로그래밍은 클래스간의 메시지 교환에 의한 프로그래밍의 특성 외에도 뛰어난 모듈화를 갖춘 프로그램이 가능하다. Ledbetter와 Cox[7]는 객체지향 프로그래밍 언어로 개발한 소프트웨어 모듈을 일종의 "소프트웨어 IC"라 불렀다. 하드웨어 제품의 부품처럼 객체지향 프로그래밍도 사용자가 언제든지 필요한 모듈을 수정 없이 그대로 사용할 수 있는 특성을 가지고 있음을 시사하였다.

객체지향 기술로 개발된 제품은 재사용성과 보증된 모듈화로 소프트웨어의 신뢰성을 크게 향상시킬 수 있다. 뿐만아니라 재사용성, 확장성이 뛰어나고, 실제세계와 객체간의 높은 유사성으로 이해도를 높일 수 있어 시스템의 보수및 유지하는데 소요되는 노력을 크게 줄일 수 있다.

대표적 객체지향기법들로 OOSE(Object-Oriented Software Engineering)[11], Coad and Yourdon[4,5,6] 방법론, Booch[3]방법론, OMT[17](Object Modeling Technique)와 Smalltalk[9] 방법론들이 있다. 특히 Smalltalk 방법론은 객체지향기술의 선구자 역할을 한 방법으로 최근에도 널리 이용되고 있다.

본 연구에서는 Smalltalk 방법론을 이용하여 시뮬레이션을 통한 전문가시스템을 구현함으로써 이 결과들이 다른 영역의 시뮬레이션을 통한 전문가시스템 구축시에도 용이하게 확대 응용될 수 있는 기반을 제공하고자 한다. 부연하면, 우선적으로 시뮬레이션을 통한 전문가시스템의 객체지향 모델이 제시된다. 이 모델에 기초하여 문제 형태에 따른 다양한 시뮬레이션에 기반한 전문가시스템을 구축할 수 있다. 본 연구에서는 선정된 문제에 대하여 시뮬레이션을 통한 전문가시스템을 구축하고 평가함으로써 다른 문제에서도 SIMBES를 용이하게 구축할 수 있는 하나의 골격을 제공한다.

2. 시뮬레이션기반 전문가시스템모형

SIMBES의 행태를 보여주는 핵심 요소들이 그림 1에 보여지고 있다. 이 모형은 문제의 상황이나 난이도에 따라 크게 두가지 부류로 나눌 수 있다. 첫번째 가능한 구조도는 모의실험기(Simulator), 지식추출관과 전문가시스템으로 구성될 수 있다. 지식추출관은 대개 많은 시뮬레이션을 통해서 지식을 축적한 작업자 또는 지식공학자가 된다.

이 구조도는 학습 훈련을 통한 지식획득 방법으로서 비교적 덜 복잡한 문제에 적용할 수 있으며, 그렇지 않고 난이도가 높은 문제에 대하여 전문가 시스템을 구축하기에는 적합하지 않다. 그 주된 이유는 작업자가 복잡한 문제와 시뮬레이션 결과를 분석하여 지식획득을 수행하기에는 인간의 능력을 크게 벗어나기 때문이다. 예를들면, 어떤 객체를 규정하는 속성(Attribute)의 수가 대단히 많고 그 속성이 이산치가 아닌 연속치 값을 갖는다면, 그 속성들의 상호작용이 커짐으로 인해 연속치 값에 대한 명확한 구분을 정하여 객체들을 분류하는 일은 인간으로 하여금 불가능한 일이다. 그러나 대강의 지식만을 추출하여 전문가시스템을 구축하여도 기존시스템에 비해 성능을 크게 향상시킬 수 있는 경우는 쉽게 찾아 볼 수 있다[2].

두번째 가능한 구조도는 모의실험기, 속성 추출기(Feature Extractor), 기계학습시스템(Machine Learning System), 성능 평가기(Performance Evaluator), 전문가시스템으로 이루어진다. 이 구조도는 전자가 갖는 한계성을 극복하여 복잡하고 어려운 문제나 인간 전문가를 확보하기가 어려울때 전문가 시스템을 구축할 수 있도록 하여 준다. 이러한 의미에서 후자의 구조도가 더욱 일반성을 갖는다고 볼 수 있다.

시뮬레이션을 통한 전문가시스템의 주요 구성요소인 모의실험기는 Task Generator, Task Executer와 Evaluator로 구성된다. 본 연구에서는 일정계획 Task의 예를들어 객체지향시스템을 구현한다. 이 때 많은 경우의 작업을 발생시키는 Job Generator, 이 작업들을 일정계획하는 Job Scheduler와 그 유도된 일정계획을 특정기준에 의거 평가하는 Schedule Evaluator가 필요하게 된다.

시뮬레이션을 통한 전문가시스템의 실제 구현은 Task의 난이도나 시뮬레이션 가능 여부에 따라 좌우될 수 있다. 이와같이 필수적인 Task의 정의를 위해 본고에서는 NP hard 문제로 널리 알려져 있는 일정계획에 기반한 자재소요계획 문제가 다루어진다. 적절한 Task의 선정기준은 명확하지는 않으나 대개 실시간 또는 준실시간을 필요로 하는 시뮬레이션 또는 NP hard 문제 등이 대상이다.

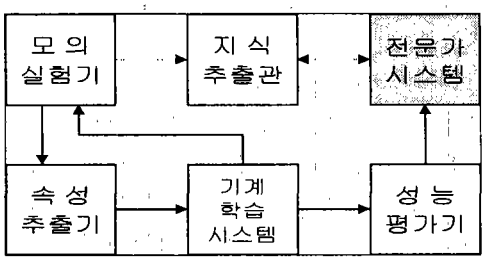

그림 1. 시뮬레이션기반 전문가시스템 모형

이외에도 속성추출기, 기계학습시스템, 성능평가기와 전문가시스템 등이 조심스럽게 설계되어야 한다. 정의된 문제에 대하여 속성이 정의되면, 기계학습시스템과 성능평가기의 도움으로 이들 속성과 결과에 대한 관계가 분석되어 지식이 추출된다. 이렇게 추출된 지식이 성공적임이 판명되면 실제 문제에 적용하기 위하여 이들은 전문가시스템에 장착된다.

성공적으로 이루어진 시뮬레이션 결과는 속성 추출기에 의해 속성치들이 추출된다. 이 때 물론 주요 속성들이 우선적으로 정의되어져야 한다. 이러한 속성을 결정하는 방법은 대개 많은 시뮬레이션을 통해 수동으로 행해지며, 많은 시간과 노력을 필요로 한다. 그러나 일단 이 과정이 끝나면 그 이후의 과정은 거의 자동적으로 행하여 진다는 점에서 속성의 정의 과정을 애로공정으로 볼 수 있다. 이렇게 하여 기계학습시스템에 필요한 많은 수의 훈련용 예제가 확보될 수 있으며, 이것들은 전문가 시스템의 지식획득을 위한 주요한 지식원이 된다.

인간 두뇌에서 행해지는 배움 현상을 여하히 컴퓨터 프로그램으로 재현시킬 수 있는가를 기본 목표로 하는 기계학습에는 성능 향상과 지식획득의 두 관점에서 설명되어 진다. 전자의 관점으로는 SOAR[8]나 LEX[8]등을, 후자의 관점으로는 Induction[8]과 Neural Net[13]등을 대표적인 것으로 볼 수 있다. SIMBES에서 주로 이용되는 기계학습시스템은 후자의 범주에 속한다. 특히 Induction은 Neural Net과는 달리 적은 갯수의 훈련예제와 짧은 훈련기간이 소요된다는 잇점때문에 본 연구에서 이용 되었다. Induction에 대한 연구는 초창기 CLS[10]에서 출발하여 최근 C4[16]에 이르기까지 많은 연구가 진행되어 왔다.

Induction시스템 중 하나인 C4.5는 C4의 개정판으로 유도된 훈련용 예제를 통하여 일련의 규칙을 출력시킨다. 성능평가기의 역할은 학습되지 않은 자료에 대하여 이 출력된 규칙을 적용하여 분별하는 성능을 관측한다. 훈련용 예제의 갯수가 증가함에 따라 출력된 규칙의 성능이 변화하기 때문에 평가관 또는 시스템 개발자는 만족스러운 수준에 도달할 때까지 예제 발생과 규칙 출력 과정을 반복적으로 수행하여야 한다. 이렇게하여 최종적으로 결정된 규칙은 전문가 시스템의 지식베이스를 형성한다.

선정된 문제에 대하여 시뮬레이션을 통한 전문가 시스템을 구축하기 위한 방법은 여러 가지가 있으나 재현성, 유지 및 보수 등에서 뛰어난 객체 지향기술을 이용하는 것이 바람직하다.

OOSE, Coad and Yourdon 방법론, Booch 방법론, OMT, Smalltalk방법 등 많은 객체지향기법들이 독자적으로 시스템 분석에서 부터 설계와 구현에 이르는 과정들을 지원한다. 그러나 다음 장에서는 특히 객체지향기술의 모태이며 지금도 주요 방법중의 하나로 이용되고 있는 Smalltalk 방법론으로 이들 구성요소들을 설계하고 구현한다. 시뮬레이션을 통한 전문가시스템의 사례에서 이용되는 클래스, 관련 메시지와 메소드, 클래스 계층도 등이 설계되고 구현된다. 이렇게 함으로써 객체지향시스템과 시뮬레이션을 통한 전문가시스템의 최종제품인 전문가시스템과의 연계가 자연스럽게 이루어질 수 있다. 또한 이를 기반으로 하여 다른 문제의 시뮬레이션을 통한 전문가시스템 구축시 변화 가능한 요소를 신속하게 검토할 수 있다.

3. 객체지향 SIMBES 구현 및 평가

1) Smalltalk 패러다임

문제를 해결하는 과정을 살펴보면 Smalltalk세계와 실세계는 너무나 많은 유사성을 갖고 있다. Smalltalk 패러다임은 두 세계의 이렇게 뛰어난 Compatibility외에도 친숙한 사용자 인터페이스 제공, 풍부한 자료구조 등을 제공한다. 이들에 대한 상세한 설명이 Digitalk사가 개발한 Smalltalk/V 메뉴얼[9]에 수록되어 있다. 물론 이 패러다임은 객체지향시스템이 강조하는 코드의 재사용성으로 인한 시스템 개발시간 단축, 시스템 유지보수의 용이성과 같은 특징들을 갖고 있다.

객체지향 프로세스를 완성하기 위해서 Smalltalk시스템은 객체, 메시지, 메소드, 클래스, 인스탄스와 같은 5개의 기본적인 요소를 필요로 한다. Smalltalk의 기초가 되는 객체는 어떤 행태를 보여주는 유형의 실체[3,4], 행태들과 그 행태를 규정하는 상태를 갖는 실체[11], 또는 관련 데이터와 절차를 포함하는 팩키지[18] 등으로 정의되고 있다. 이 객체들은 특별한 속성과 행태(Behavior)를 갖는 실체로 간주할 수 있다.

전통적인 시스템이나 수동적인 데이터 아이템과는 달리, 객체들은 행동할 수 있는 능력을 지닌다. 이 행동은 객체가 어떤 방법으로 행동하는지를 요청하는 메시지를 받았을 때 나타난다. 모든 메시지들은 어떤 구조를 갖는다. Smalltalk에서는 한 메시지 3개부분, 즉 수신객체, 메시지 실렉터와 아규먼트로 구성되어 있다. 객체지향형 프로그램들이 실행될 때, 객체들은 다른 객체들로부터 메시지를 받고, 처리한 후 반응한다. 그러나, 메시지를 전송하는 객체는 수신 객체가 요청을 어떻게 수행하는지에 대해서 전혀 인지할 필요가 없다.

유사한 활동을 하는 동일한 성격의 객체 집합을 클래스라 한다. 또는 클래스가 유사한 객체를 위한 탬플리트라고도 한다. 클래스는 그에 속하는 객체들의 공통된 특성을 요약하는 속성들과 메소드들을 지니고 있다. 실제로 객체들은 부모의 클래스가 생성을 요청하는 메시지를 받았을 때 생성된다. 이때, 생성된 객체는 속성과 메소드를 그 부모가 되는 클래스로부터 상속받는다. Smalltalk 패러다임에서 속성은 클래스 변수와 인스탄스 변수로 구분된다. 여기서 클래스 변수들은 클래스에 저장되지만 인스탄스 변수들은 클래스로부터 생성된 인스탄스에 따라 독립적으로 연계된 값들을 갖는다. 마찬가지로 절차들도 클래스 메소드와 인스탄스 메소드로 구분되는데, 클래스 메소드는 오직 클래스 변수만을 조작하고, 이를 수행하기 위하여 인스탄스를 갖지 않는다. 반면에 메소드의 대부분을 차지하는 인스탄스 메소드들은 두 가지 모두의 변수에 대하여 작동되어지며 이를 활성화시키기 위하여 인스탄스를 갖는다.

Smalltalk 패러다임에서 시스템을 구축할 때 가장 중요한 작업은 객체/클래스와 메시지의 정의로 집약된다. 그것은 적절한 객체/클래스와 메시지의 정의는 시스템 행태를 결정하는 가장 중요한 요소이기 때문이다. 따라서 이들에 대하여 다음 절에서 상세하게 서술된다. 이 때 클래스의 계층적인 구조는 클래스들 간의 공통적인 특징을 공유하게 만든다는 의미에서 중요성을

갖는다. 계층구조의 최하위에 있는 클래스들은 보다 상위 계층에 있는 클래스의 변수와 메소드를 상속받는다.

2) 시스템 구성 요소

본 절의 중요성은 두 측면에서 강조된다. 첫째, 주어진 문제에서 생산성을 제고하기 위하여 객체지향 SIMBES를 구현하는 방법을 제시한다. 둘째, 이렇게 구현된 시스템은 그 특성상 여타 SIMBES와 많은 부분의 중복이 불가피하므로 주요 골격을 빌려 쓸 수 있다. 특히 일정계획에 관련된 문제에 대해서는 거의 동일 골격으로 역할을 수행할 수 있다. 이런 의미에서 객체/클래스 계층도와 메시지를 포괄하는 인스탄스 프로토콜 중심으로 시스템 요소가 기술된다.

가. 시스템 계층도

많은 문제들이 SIMBES로 모형화되었더라도 실질적인 효과를 얻기 위해서는 구현되어져야 한다. 이 구축된 시스템을 위하여 정의된 한 형태의 문제가 필요하며, 기존의 논문[1]에서 이를 인용하였다. 이를 간략하게 설명하면, MRP(Material Requirements Planning)시스템과 생산 일정계획시스템은 각각 독립적으로 세부 자재 계획과 일정계획 수립 능력을 갖고 있으나 통합되지 못함으로 인한 시너지 효과를 상실하고 있다. 설령 통합되었더라도 그 소요시간은 이용에 어려움이 되기도 한다. 이러한 어려움에 효과적으로 대처하기 위해서 제안된 Schedule-Based Material Requirements Planning(SBMRP)시스템[12]은 최적은 아니더라도 실행가능한 생산 일정계획 뿐만 아니라 실행가능한 자재 소요계획을 빠르게 유도할 수 있다는 점에서 대단히 유용하다.

이 시스템에서 이용하는 주요 방법은 MPS내의 작업을 주어진 자원의 용량내에서 전진일정계획(Forward Scheduling)방법으로 로딩(Loading)시킨다. 이때 물론 작업의 선후관계가 지켜질 수 있도록 작업의 일정계획은 진행된다. 작업 로딩 순서는 암시적으로 FIFO(First In First Out)를 이용하였다. 이렇게 함으로써 최적안은 아닐지라도 실행가능한(Feasible) 일정계획안을 대단히 용이하게 도출할 수 있다. 또한 자제 소요계획은 유도된 작업 일정계획에 따라 자동적으로 시간별 계획이 수립된다. 여기에서 중요한 것은 일정계획의 실행가능성이 이미 검증되었기 때문에 이 계획안을 기본으로 유도된 시간별 자재 소요 계획도 물론 실행가능한 점이다.

그러나 전진일정계획 방법을 이용하는 SBMRP시스템은 많은 재고비용을 수반하는 단점이 있다. 이러한 문제점을 해결하기 위해 White[19]는 후진일정계획(Backward Scheduling)을 도입하였다. 이 후진일정계획 방법은 재고비용을 감소시킬 수 있는 반면 마감일을 지키지 못함으로 인한 지연비용을 증가시킬 수 있는 문제점이 있다. 이러한 이유로 본 연구에서는 재고비용과 지연비용의 합을 최소화하는 일정계획을 수립하고자 한다.

SIMBES구현을 위한 Job Shop은 편의상 5대의 기계로 구성되며, 각제품의 BOM은 최대 6개의 제조 부품과 많은 수의 구매 부품으로 구성된다고 가정한다. 이 문제는 NP-Complete에 속하며 수학적인 방법이나 시뮬레이션 방법을 적용할 때 앞서 언급했던 한계를 갖는다. 따라서 이를 극복하기 위한 하나의 방법으로 SIMBES가 구현된다. 이 객체지향 SIMBES를 구현하기 위해 우선적으로 객체/클래스가 주어진 문제에 대하여 정의되었다. 이들 클래스에 대하여 상속성과 연관성을 고려하여 그림 2와 같은 클래스 계층도가 유도되었다. 이들 클래스에 대한 내용들은 좌측에 명시된 절에서 순서적으로 기술된다.

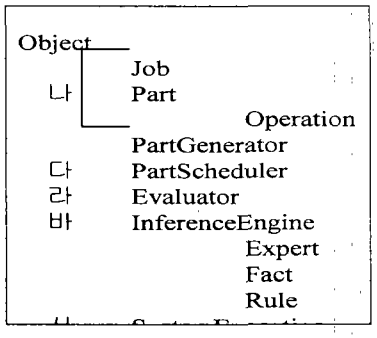

그림 2. 클래스 계층도

나. Job Generator

Job Generator는 Rule Extractor가 필요로 하는 다수의 예제를 제공하며, 생성된 작업은 Job Scheduler의 입력으로 이용된다. 각 작업은 파라메터로 작업의 수, 파트의 수, 처리시간 분포, 납기일 분포 등을 갖는다. 예를들면, 각 작업은 6개의 파트를 가공하며, 각 파트는 확률분포에 따라 기계별 처리경로와 작업시간, 납기일이 설정된다. 이 때 발생된 작업은 총가용시간/총처리시간, 마감일의 분산, 여유시간의 분산, 처리시간의 분산, 마감일의 tightness비율, 기계별 작업갯수의 분산, 공장부하의 분산, Job별 작업갯수의 분산 등과 같은 특성이 계산된다.

Job Generator는 그림 2에서 보여지는 바와 같이 "Job", "Part", "Operation"과 "PartGenerator" 클래스에 의해 구현된다. 클래스 "Job"은 "PartGenerator"가 생성한 작업들에 대하여 작업의 특성을 계산하며 이와 관련된 메시지들이 표1에 제시된다. 클래스 Part는 파트 파라메터를 설정하는 것에 관련되며 그 메시지들이 표2에 보여진다. 클래스 Operation은 각 파트의 오퍼레이션을 위한 시작과 마침시간을 정하는 것에 관련되며 그 메시지들은 표3에 보여진다. 마지막으로 PartGenerator는 확률분포에 입각하여 각 파트의 마감일, 기계별 처리시간 등을 결정하며, 그 관련 메시지들은 표4에 보여진다.

표 1. Job 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
getJobChars: aJob	Job의 주요 특성을 계산한다.
initialize	Job 파라메터를 초기화 한다.
jobId: aSymbol	Job ID를 특정 심볼로 설정한다.
makeOutputFile: aJob	특정 수의 Job DB를 만든다.
noOfParts	각 Job의 파트 수를 반환한다.
taOVERtp	Job의 총여유시간/총처리시간의 비를 반환한다.
tightness	Job 마감 tightness수준 반환한다.
varOfDues	다른 마감일의 분산을 반환한다.
VarOfProcTimes	Job 처리시간의 분산을 반환한다.
varOfSlacks	Job slacktime 분산을 반환한다.

표 2. Part 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
dueDate:	각 파트의 마감일을 설정한다.
initialize	파트 속성을 초기화 한다.
partID: aString	파트의 ID를 설정한다.
processTime: aString	각 파트의 일련의 처리시간을 초기화한다.
seqOfOperation: aString	각 파트의 일련의 가공순서를 초기화한다.
slack	파트의 여유시간을 계산한다.
totalTime	파트의 처리시간을 계산한다.

표 3. Operation 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
finishDate	각 operation 완료일을 반환한다.
finishStart: aString	완료일자에서 시작일자를 계산한다.
operationID: aString ofPart: aPart	특정 파트의 각 operation ID를 설정한다.
startDate	각 operation의 시작일을 반환한다.
startFinish: aString	시작일에서 완료일을 계산한다

표 4. PartGenerator 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
setDueDate: aPart for: index	Uniform분포에 의거 파트의 마감일을 발생한다.
setPartID: aPart with: ascii	ascii로 파트 ID를 설정한다.
setProcessTime: aPart for: index	지수분포에 의거 일련의 처리시간을 발생한다.
setSeqOfOperation: aPart	각 파트의 가공경로를 임의로 설정한다.

다. Job Scheduler

기존의 방법들이 암시적으로 FIFO를 사용하였다는 점에 착안하여 많은 작업선택 규칙(Dispatching Rules)중 전형적으로 이용되는 다섯개를 선정하였다. 이들은 FIFO, EDD(Earliest Due Date), SPT(Shortest Processing Time), LPT(Longest Processing Time)와, LS(Least Slack)로 Penwaker[14]는 상세하게 설명하고 있다. 여기에 전진및 후진 일정계획 방법까지 고려하면 총 10개의 조정된 규칙(CLM: Combined Loading Method)을 얻을 수

있다. 예를들면, Forward EDD는 전진 일정계획에 의거 작업 일정계획을 수립하되, 작업선택 규칙은 EDD로 납기일이 가장 이른 작업을 우선적으로 일정계획함을 의미한다.

Job Scheduler는 생성된 Job에 대하여 10개의 CLM을 적용하여 각 Job 당 10개의 일정을 산출하는 클래스이다. 이 일정들을 저장하는 것은 지나친 메모리 사용이 불가피하므로 즉시 Job Evaluator에 보내진다. 이러한 일정계획을 담당하는 클래스에 이용되는 메시지는 표5와 같다.

표 5. PartScheduler 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
backwardBlocksInCharts	후진CLM에 의거 파트의 operation 시작과 완료일자 를 결정한다.
forwardBlocksInCharts	전진CLM에 의거 파트의 operation 시작과 완료일을 정한다.
initialize	loading 규칙에 따른 큐를 정한다.
noConflict:anOperation with: bOperation	두 operation간의 충돌여부를 반환한다.
possibleSlot: aDict for: aPart at: index	기존일정을 고려, operation의 시작일자 결정
schedule: aSymbol	loading 규칙으로 스케쥴을 행한다.
scheduleRulesFor:aJob	스케쥴 위한 loading 규칙을 정한다.
undo: aDictionary for: aPart	작업의 시작일이 불가하면 생성된 일정을 무시하고, 후진에서 전진으로 변경한다.

표 6. Evaluator 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
answerString: aJob	10CLM을 총비용으로 평가, 최적CLM 반환한다.
days: aSymbol for: aString	일정계획방법 의거 조기 및 지연일자 계산한다.
initialize: aJob	조기/지연비용을 단위비용으로 초기화 한다.
makeCostFile: aSelectedCLM	CLM에 대한 총비용 데이터베이스를 만든다.
makeOutputFile: rule	각 Job의 모든 특성과 최적 CLM을 포함하는 데이터베이스를 만든다.

라. Job Evaluator

Job Evaluator는 모의실험기에서 만들어진 각 Job에 대하여 10개의 스케쥴링된 일정들을 평가하는 클래스이다. 이러한 과정은 모든 Job이 평가될 때까지 반복된다. 평가기준은 재고비용과 지연비용의 가중치 합이 이용된다. 여기에서 지연과 조기날짜에 따른 지연단위비용과 재고단위비용에 대한 가중치는 조정할 수 있다.

따라서 이 Job Evaluator의 결과는 생성된 각각의 job에 대하여 정의된 8개의 job 특성과 10개의 CLM중 최저비용을 보장하는 CLM을 포함한다. 물론 이때 job 특성들은 계산된 수치로 이루어져 있다. Job Evaluator의 임무를 수행하는데 필요한 메시지는 표 6에 보여지고 있다.

마. Rule Generator

이 요소는 객체지향 SIMBES를 구성하는 클래스이기 보다는 독립적인 기계학습시스템으로 SIMBES를 지원하는 주요 요소이다. 본 고에서는 많은 기계학습시스템 중 예제를 통한 지식획득을 지원하는 C4.5[15]를 이용한다. 이 시스템은 ID3의 후속판 중 하나로 노이즈를 포함하는 예제에서도 효과적으로 분류하는 규칙을 유추하는 능력을 갖고 있다. 시뮬레이션과 평가를 거쳐 만들어진 예제들은 C4.5에 입력되어 표 7과 같은 규칙으로 출력된다.

이 때 예제의 수를 결정하는 문제는 다른 기계학습의 경우 처럼 대단히 어려운 문제로 반복적인 실험 및 분석이 요구된다. 최적의 조합으로 이루어진 규칙을 찾기 위해서 예제 수의 증가에 따른 분류성능이 비교되었다. 기존 방법에서는 10개의 CLM 방법중 임의로 택일하여 적용하므로 기대성능은 10%로 간주할 수 있다. 반면에 예제에 기반하여 획득된 지식은 모든 경우에서 기존방법보다 우월한 결과를 보여주고 있다. 그러나 이들에 대한 성능의 증가는 예제수의 증가에 따라 증가하다가 480개의 예제에서 성능이 감소하였다. 이들을 통해 도출된 규칙들이 최적의 조합으로 이루어진 규칙으로 정의되었다. 이 규칙들은 전문가시스템에 장착될 지식으로 표 7에서 보여지는 것과 같은 30개의 규칙으로 구성된다.

표 7. C4.5의 출력 규칙 예제

규칙번호	규칙
Rule 1	taOVERtp <= 1.17 varOfDues > 30.14 tightness <= 50 --> class forwardSPT
Rule 2	taOVERtp >1.46 taOVERtp <=1.66 varOfSlacks > 12.54 varOfSlacks<= 46.8 varOfProcTimes > 7.94 varOfProcTimes <= 27.1 varOfLoad > 35.13 --> class backwardSPT

표 8. Expert 인스탄스 프로토콜

메시지	설명(Pseudo메소드)
add: aRule	Expert에게 규칙을 추가한다.
initialize	규칙을 OrderedCollection으로 초기화 한다.
rules	production rule의 모임을 반환한다.
solve	Expert의 규칙이 firing될 때까지 반복수행한다.

바. KBES

Rule Generator를 통해 유도된 지식은 Smalltalk환경에서 전문가시스템(KBES)으로 구현된다. 물론 이 지식은 다른 전문가시스템 쉘에 장착되어 이용될 수 있으나 모의실험기와 동일한 환경에서 구현됨으로써 시스템의 평가시에 대단히 편리하다. 획득된 지식은 전문가시스템에서 널리 이용되고 있는 If-Then 규칙으로 간단하게 표현될 수 있다. 이 KBES는 각 작업의 특성에 의거하여 최상의 CLM을 추천하여 주는 시스템이다.

객체지향 전문가시스템은 세 개의 클래스인 Expert, Fact와 Rule에 의해 구현된다. 이들 클래스는 메시지가 없는 추상클래스인 Inference Engine의 하위클래스로 구성된다. 클래스 Expert는 지식베이스로 일련의 규칙을 포함하고 있고, 각 규칙은 클래스 Rule의 한 인스탄스이다. Fact는 주어진 사실 또는 추론된 사실인가에 관계없이 Inference Engine에서 참으로 알려진 모든 것들을 포함한다. 표 8, 표 9와 표 10은 이들 클래스에서 이용하는 메시지를 각각 보여주고 있다.

표 9. Fact 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
initialize	Fact클래스를 초기화 한다.
is: aSymbol gt: aNumber	Job특성치가 특정 수보다 크면 참을 반환한다.
is: aSymbol lt: aNumber	Job특성치가 특정 수보다 작으면 참을 반환한다.
value: aNumber at: aSymbol	Job특성치의 값을 설정한다.

표 10. Rule 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
action	규칙에서 then 부분을 반환한다.
condition	규칙에서 조건 부분 값을 반환한다.
fire	규칙의 조건이 참이면 CLM을 추천한다.
number:ruleNumber condition:cBlock action:actBlock	규칙의 내용을 초기화 한다.

사. Executive System

객체지향 지식기반 전문가시스템은 시스템시작, 통제 및 전반적인 관리를 담당하는 객체가 필요하다. 뿐만 아니라 구성 요소별 세부적인 관리도 필요에 따라 수행되어야 한다. 이러한 임무를 수행하는 클래스인 SystemExecutive는 표 11과 같은 메시지가 필요하다.

3) 시스템 평가

구축된 SIMBES시스템의 효과성을 증명하기 위해 전절의 시스템계층도에서 보여준 시스템에 대한 평가와 분석이 수행되었다. 즉 5대의 기계로 구성된 Static Job Shop은 Job Generator에

서 임의로 만들어지는 작업시간, 작업순서, 납기등이 지정된 다수의 제품들을 처리한다. 각 제품의 BOM은 최대 6개의 제조부품과 다수의 구매부품으로 구성된다. 이렇게 480개의 작업을 발생시킨 뒤 기존의 두 개 방법과 KBES로 일정계획이 수립된다.

표 11. SystemExecutive 인스탄스 프로토콜

메시지	설명(Pseudo 메소드)
consultKBES: aJob	특정 Job에 대하여 KBES가 추천하는 CLM을 반환한다.
initAnalysis	일정방법별 통계량과 비용을 정렬하여 반환한다.
initFact: aJob	시스템 파라메터를 초기화 한다.
rank: aNumber in: anArray	가장 적은 비용을 갖는 위치를 반환한다.
revisedJobScheduler	일정을 수립하기 위해 KBES와 기존 Scheduler를 부른다.
startJobGenerator	Job Generator를 부른다.
startJobScheduler	Job Scheduler를 시작한 후 Job Evaluator를 요청한다.

표 12. 시스템 성능 평가 결과

반복 회수	Forward FIFO	Backward FIFO	KBES
1	21088	18551	13754
2	21302	18605	13788
3	21432	18515	14074
4	20628	17494	13139
5	20717	17368	13154
6	21262	18024	14090
7	21364	18345	13863
8	20531	16947	13026
9	21370	18258	13837
10	21017	17885	13400
평균	21071	17999	13612

KBES는 작업을 분석하여 일정계획을 위한 최적 CLM을 Job Scheduler에게 제시한다. 이때 Job Scheduler는 기존의 방법뿐만 아니라 추천된 CLM에 대하여 일정계획을 수행한다. 이렇게 만들어지는 일정계획안이 총비용에 의거하여 상호 비교되었다. 본 예제에서는 지연과 조기작업일에 대한 지연단위비용과 재고단위비용이 동일하다고 가정하였다. 상기와 같은 평가작업을 10회 반복하였으나 기존시스템과의 차이는 통계적으로 안정된 결과를 얻었다. 이에 대한 결과가 표 12에 제시되고 있다. 결국 SIMBES를 통하여 구축된 KBES의 성능이 기존시스템을 크게 능가하고 있음을 보여주고 있다.

4. 결론

본 연구에서는 시뮬레이션을 통한 전문가시스템 모형의 한 예시로서 일정계획기반 자재계획 문제에 대하여 실제로 SIMBES를 구현하고 평가하였다. 이 시스템은 모의실험기, 속성추출기, 기계학습시스템, 성능평가기와 전문가시스템으로 구성되어 있다. 모의실험기, 기계학습시스템과 전문가시스템이 주요한 요소임에 틀림없지만 전문가시스템의 지식의 질은 적절한 속성추출에 크게 의존한다. 따라서 시스템 구현에 이르는 전 과정은 성능평가기에 의해 반복적으로 피드백을 받는 시스템이어야 한다. 이 시스템은 특히 재사용성, 유지 및 보수 등에서 최근 각광을 받고 있는 객체지향기술의 선구자였던 Smalltalk에 기초하여 구축되었다.

객체지향시스템에서 적절한 객체/클래스와 메시지의 정의는 시스템 행태를 규정하는 가장 중요한 요소이다. 마찬가지로 Smalltalk에서도 객체/클래스, 메시지/메소드, 인스탄스와 같은 요소가 필수적이며, 이들의 정의는 대단히 중요한 작업이다. 따라서 본 고에서는 일정계획기반 자재계획 문제에 대하여 SIMBES를 구현함에 있어서 이들에 주안점을 두어 기술하였다. 즉, 해당문제의 SIMBES 클래스 계층도가 작성되었으며, 이 계층도의 요소별 클래스에 대한 인스탄스 프로토콜이 상세하게 서술되었다. 이들 프로토콜은 특정 객체지향언어에 구애받지 않고 쉽게 시스템을 구현할 수 있는 기반을 제공한다. 특히 클래스 "Inference Engine"을 이용하면 객체지향 전문가시스템을 쉽게 구현하고 운용할 수 있다.

또한 본 시스템의 각 구성요소는 객체지향시스템의 장점을 이용해 Skeletal 시스템으로의 역

할도 가능하다. SIMBES의 문제영역이 변화되더라도 본 연구에서 제시한 상당한 부분의 클래스와 메시지 등이 공유되리라 예상된다. 특히 일정계획과 관련된 문제에서는 대부분의 클래스, 메시지, 메소드 등이 차용될 수 있다. 실제 현장에서는 다양한 일정계획 문제가 존재하여 적용할 수 있는 기회도 대단히 많다. 그러나 대부분의 이들 문제는 수학적 또는 알고리즘으로 해결할 수 없는 NP-Hard 문제이고, 시뮬레이션을 적용하기에는 과다한 시간의 소요로 인해 실용성이 떨어짐을 감안할 때 본 연구의 큰 의의가 있다. 이것은 바로 SIMBES개발시 생산성과 직결되는 것으로 계층도와 프로토콜의 차용으로 인한 개발시간의 단축과 문제 해결로 인한 성능향상을 동시에 달성할 수 있다. _

참고문헌

[1] 김 선욱, "A Simulation-Based Expert System Paradigm", 대한산업공학회지, 18(2), 1992.

[2] 나태영, 김승권, 김선욱, "Knowledge Based Simulation for Production Scheduling", 대한산업공학회지, 23(1), 1997.

[3] Booch, G., Object-Oriented Analysis and Design with Applications, Benjamin / Cummings Publishing Company, Inc., 1994.

[4] Coad P., and Yourdon, E., Object-Oriented Analysis, Object International Inc., 1991.

[5] Coad, P., and Yourdon, E., Object-Oriented Design, Engleword Cliffs, NJ : Yourdon Press / Prentice Hall, 1991.

[6] Coad, P., and Nicola, J., Object-Oriented Programming, Prentice Hall Inc. 1993.

[7] Cox, B. J., Object-Oriented Programming: An Evolutionary Approach, Addison-Wesley, Massachusetts, 1986.

[8] Cohen, P. R., and Feigenbaum, E. A., The Handbook of Artificial Intelligence, Vol. 3, William Kaufmann Inc. 1982.

[9] Digitalk Inc., Smalltalk / V 286 : Tutorial and Programming Handbook, 1988.10

[10] Hunt, E. B., Marin, J., and Stone, P. J., Experiments in Induction, New York: Academic Press, 1966.

[11] Jacobson, I., Object-Oriented Software Engineering, ACM Press, 1992.

[12] Hastings et al., "Schedule-Based MRP: An Integrated Approach to Production Scheduling and Material Requirements Planning," J. Opl. Res. Soc., 33, 1021-1029, 1982.

[13] Lippman, R. P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April, 1987.

[14] Panwalker, S. S., and Iskander, W., "A Survey of Scheduling Rules", Operations Research, 25(1), Jan.-Feb, 1977.

[15] Quinlan, J. R, "Simplifying Decision Trees", Int. J. Man-Machine Studies, 27, 221-234, 1987.

[16] Quinlan, J. R., Compton, P. J., Horn, K. A., and Lazarus, L., "Inductive Knowledge Acquisition: A Case Study", Proceedings of the 2nd Australian Conf. on Applications of Expert Systems, Sydney, 1986.

[17] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., Object-Oriented Modeling and Design, Prentice-Hall Inc., 1991.

[18] Taylor, D., Object-Oriented Information Systems, John Wiley, 1994.

[19] White, C., Modelling and Design of Flexible Manufacturing Systems, edited by Kusiak, A., Elsevier Science, Amsterdam, The Netherlands, 1986.