객체지향 기법을 이용한 공장운용 환경 하에서의 혼합시뮬레이션 구현*

An Implementation of Hybrid-Simulation in Manufacturing Environments using Object-Oriented Methodology

김성식,** 홍윤기,*** 전 진,** 강동우****
Sung Shick Kim, Yoon Gee Hong, Jin Jun, Dong Woo Kang

Abstract

In building a shell-based FMS, which is known as one of the top-down approaches in the field of factory automation, we may take a hybrid simulation into consideration. The modeling of a hybrid simulation consists of real physical entities, virtual simulation, and central clock algorithm, etc. to carry out the whole system operation. In this paper, we show a way to construct a hybrid simulation software system in manufacturing environments. We bring in the object-oriented methodology in system design and it can contribute in dealing with a wide variety of production types and configurations. Some classes such as project, product, process, order, schedule, stage are defined. These are used and tested by implementing a specific LSI circuit assembly line process.

* 본 연구는 한국과학재단의 목적 기초연구의 일환으로 수행되었음.
** 고려대학교 산업공학과
*** 한성대학교 산업시스템공학부
**** 삼성전자 전략기획총괄 미디어서비스 사업팀

1. 서론

현대의 기업환경에서는 수요자가 원하는 제품을 보다 빠른 시간에, 보다 싼 비용으로, 보다 좋은 품질로 제공할 수 있는 기업만이 살아남을 수 있다. 기업들은 이러한 환경 하에서 자신의 경쟁력을 재고하기 위한 자구책의 일환으로 유연 생산 시스템(FMS; Flexible Manufacturing System), 컴퓨터 통합 생산 시스템 (CIMs; Computer Integrated Manufacturing system) 등 자동화 시스템을 구축하기 위하여 많은 시간과 비용을 투자한다. 자동화를 통한 생산비용의 절감, 제품품질의 향상, 기업 정보의 통합관리 등을 목표로 한 공장자동화는 특히, 고가의 자동화 장비를 설치해야 하기 때문에 투자비용을 분산하고 투자의 안전성을 높이기 위한 방법으로 부분적으로 자동화를 이루어 나가는 bottom-up 접근방법과 체계적인 구축을 하기 위한 top-down 접근방법이 제안되고 있다. Bottom-up 접근방법의 경우, 머시닝 센터와 같은 단위기계나 FMC(Flexible Manufacturing Cell)로 구성된 자동화시스템을 부분적으로 도입하여 구축하게 된다. 이러한 시스템들은 기존에 구성되어 있는 장비들의 수준과는 동떨어진 자동화섬[8]이라는 문제점을 가지게 된다. 그와는 반대로 top-down 접근방법은 자동화 시스템을 체계적으로 구성할 수 있는 장점을 가지고 있다. 이러한 top-down 접근방법의 일종인 shell을 이용한 FMS 구축 방법[2]에서는 장비를 연결하지 않은 상태에서 운영 S/W를 구축하고 실험할 수 있도록 지원하고 있다. Shell을 이용하여 자동화 시스템을 구축할 때, 실제 기계가 작동하는 물리적 시간진행과 아직 자동화 장비를 설치하지 않은 부분을 대신하는 시뮬레이션이 처리되는 논리적 시간진행이 혼합되어 나타난다. 물리적 시간진행과 논리적 시간진행이 혼합된 시뮬레이션이라고 볼 수 있는 혼합시뮬레이션에서의 시간진행을 처리하기 위하여 중앙시계 알고리즘이 제안되었다[2][10]. 또한 분산환경에서 구현 시 네트워크상의 메시지를 처리할 때 발생할 수 있는 인과성 오류의 처리방법이 제안되었다[3].

앞에서 설명한 상황이 일어나는 여러 공장환경 하에서의 혼합시뮬레이션을 처리할 수 있는 시뮬레이션 시스템 구축이 본 연구의 목적이다. 시뮬레이션 시스템을 구축하기 위해서는 특정 공장의 환경에 맞도록 대상 공장 시스템을 분석하고 적절한 구조로 설계하여 프로그램으로 구현해야 한다. 소프트웨어 구축의 일반적 단계인 '분석-설계-구현'은, 시뮬레이션 시스템의 경우, 대상 시스템의 특성에 따라서 상이하므로 이미 구축해놓은 시스템이 있다 하더라도, 타 공장에 시뮬레이션 시스템을 구축하기 위해서는 분석 단계에서부터 다시 시작해야 하는 단점이 있다. 즉 다른 공장의 환경에 맞게 시뮬레이션 시스템을 구축하기 위해서는 기존 프로그램 코드의 많은 부분을 수정하거나 또는 프로그램을 처음부터 다시 작성해야 하는 경우까지 발생한다.

이러한 단점을 극복하기 위하여 본 연구에서는 소프트웨어 공학의 새로운 패러다임인 객체지향 기법을 적용한다. 대상 공장의 설정(configure)이 각 공장마다 다르므로 객체지향 기법을 사용하여 기존 프로그램의 재사용성을 높이고 대상 공장에 맞는 맞춤형 프로그램을 가능하도록 한다[13]. 즉, 여러 형태의 공장에 대하여 시뮬레이션 하기 위해서는 일반적인 공장 구성 요소들에 대한 추상화 과정을 거쳐 필요한 클래스들을 추출/구성하고, 특정 공장 환경에 적용할 때는 이미 생성된 클래스를 재사용하거나 특정 공장 환경에 맞도록 기존의 클래스를 상속받아서 수정하여 사용한다. 이때 수정되는 클래스는 변경되는 부분이 다른 클래스들에 영향을 미치지 않기 때문에 소프트웨어의 생산성을 높일 수 있다.

본 논문의 내용은 이러한 객체지향 기법을 이용하여 여러 형태의 공장에 적용 가능한 혼합시뮬레이션 시스템의 구현에 관한 내용이다. 논문의 구성을 보면, 2장에서는 기존 관련 연구와의 비교를, 3장에서는 혼합시뮬레이션 클래스의 구조를 각각 서술하고, 4장에서는 구현된 프로그램에 대한 설명을 하며, 마지막으로 5장에서는 결론과 추후 연구사항을 살펴본다.

2. 기존 연구

서론에서도 언급했듯이 본 연구는 소프트웨어의 확장, 보수, 그리고 변경 등의 요구가 있을 때 빠르

고 효율적으로 대처할 수 있도록 객체지향 방법을 적용하여 공장환경 하에서의 혼합시뮬레이션을 구축하는 것이다. 제조시스템에 대한 객체지향적 접근은 여러 논문에서 언급되었는데[1][4] 이들 논문에서는 제조시스템 전반에 대한 분석을 통한 객체지향적 설계에 중점을 두고 있다.

객체지향 시뮬레이션에 대한 논문들은 다수 있으며 DEVS-C++, Simple++과 같은 많은 제품들이 존재한다. 하지만 이들 시뮬레이터들은 기존 시뮬레이션 프로그램 또는 패키지를 객체지향 기법을 사용하여 설계, 구축한 것이며, 본 연구의 주요한 목적인 혼합시뮬레이션의 구현에는 적합하지 않다.

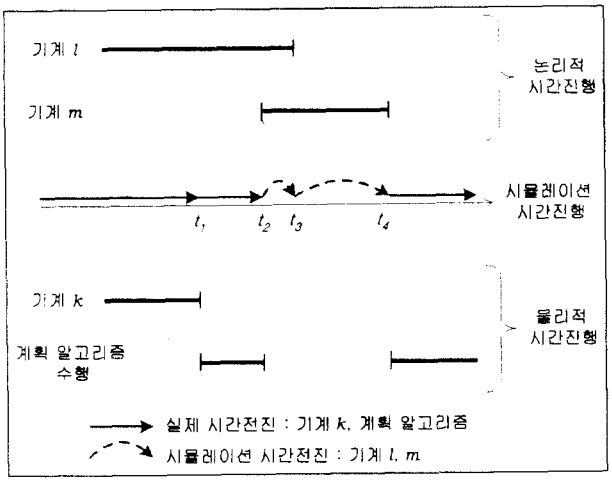
혼합시뮬레이션에 관하여 배경한의 논문[2]에서 혼합분산 시뮬레이션이라는 시뮬레이션 상황을 처음으로 정의하였으며, 혼합분산 시뮬레이션이 실제 시스템과 가상 시스템이 공존하는 시뮬레이션 환경 하에서 발생하고 또한 이 두 시스템을 동일한 시간축으로 관리해야 하는 문제점을 가지고 있음을 밝혔다. 배경한의 논문에서는 이러한 혼합분산 시뮬레이션의 시간진행을 위하여 중앙시계를 이용한 방법을 제안하였다. 제안된 방법에서는 분산된 각 프로세스들의 상태 즉, 물리적 시간전진 상태와 논리적 시간전진(시뮬레이션) 상태를 중앙시계에 등록하여, 중앙시계가 프로세스들의 상태에 적절하게 시간을 진행시킨다. 또한 서동욱 등[3]은 혼합분산 시뮬레이션이 분산 시뮬레이션의 일종이므로, 분산 환경 하에서 인과오류(causality error)가 발생할 수 있음을 증명하였으며 이에 대하여 수정된 시간진행 방법을 제시하였다. 일반적으로 분산 시뮬레이션 연구에서는 시간진행 방법을 크게 보수적인 접근방법(conservative approach)과 낙관적인 접근방법(optimistic approach)으로 분류하는데[11] 서동욱 등의 연구는 보수적인 접근방법에 그 원리를 둔다. 낙관적인 접근방법을 사용하려면 시간을 되돌려야하며(rollback) 되돌리려는 시간만큼의 변경된 정보가 보관되어 있어야 시간을 되돌려야 할 경우에 보관해 놓은 정보로 복구할 수 있다. 그러나 공장 운용 시뮬레이션과 같이 많은 정보를 갱신하고, 생성하는 시스템의 경우에는 낙관적인 접근방법을 사용하기 어렵다. 특히 shell을 이용한 FMS 구축 방법에서와 같이 실제 기계가 연결되어 있는 경우에는 시간을 되돌리지 못하는 경우가 발생한다.

본 연구에서 특정 공장에 국한되지 않고 일반적인 공장 환경에 맞는 혼합시뮬레이션 시스템을 구현하고 특정 공장에 적용할 수 있도록 재사용성(reusable)과 맞춤형(tailorability)의 특성을 가진 객체지향기법을 사용하였다. 또한 혼합시뮬레이션 시간진행 방법으로 서동욱 등에 의해서 제안된 중앙시계를 이용한 혼합시뮬레이션 운용 방법[3]을 사용하여 '객체지향 혼합시뮬레이션 시스템'을 구현하였다.

3. 객체지향 혼합시뮬레이션 모형

3.1 혼합시뮬레이션 모형

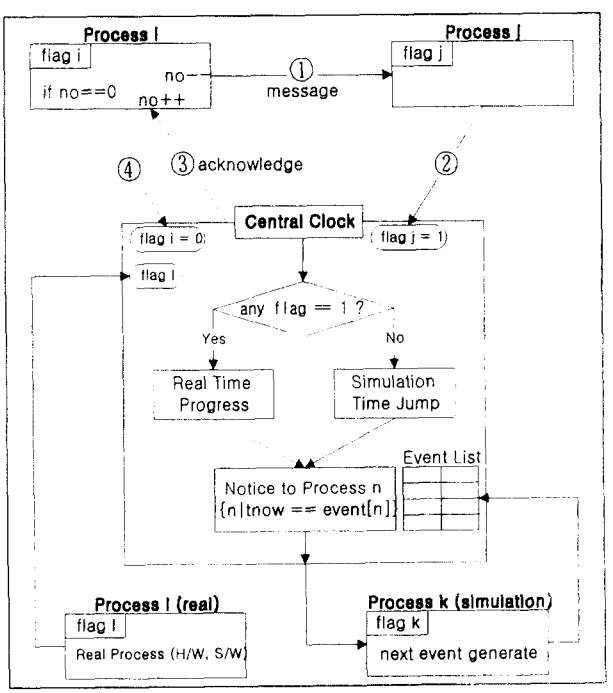
혼합시뮬레이션은 공장 자동화 구축 과정상, 설치된 장비와 설치되지 않은 시뮬레이션간의 혼합 시간진행을 위하여 제기되었다. 이와 같이 물리적 기계와 연결된 시뮬레이션을 하는 경우뿐만 아니라 공장 자동화 소프트웨어 자체의 효율성을 검증하기 위하여 혼합시뮬레이션이 사용되기도 한다. 후자의 경우, 일반 시뮬레이션과 같이 소프트웨어 시간이 시뮬레이션 시간에 포함되지 않는 경우에 공장 운영 알고리즘의 실제 수행속도를 반영하기가 어렵다. 즉, 생산계획 알고리즘의 수행완료가 다른 사건의 발생시간보다 늦어지면 알고리즘 수행 결과로서의 최적해가 더 이상 최적해일 수가 없는 경우가 발생한다. 그런데 이러한 상황은 시뮬레이션이 아닌 실제 상황에서도 일어날 수 있으므로 공장 운용 소프트웨어의 실시간 처리 능력을 판단하기 위해서는 이러한 상황이 시뮬레이션 수행 시에도 나타나야 한다. 이와 같이 알고리즘 수행 시간이 시뮬레이션 시간에 물리적 시간진행시간으로 포함되어야 하는 경우가 논리적 시간진행과 물리적 시간진행이 혼합된 혼합시뮬레이션의 또 다른 대표적인 예라고 할 수 있다. 이러한 경우에는 생산계획 수립과 같은 알고리즘에 해당하는 부분은 시뮬레이션 시스템과는 별도의 프로세스로 진행되어 물리적 시간진행을 이루어야 한다[2]. 또한 시뮬레이션 도중 임의의 사건을 발생하기 위하여 사용자 화면이 호출되는 경우도 이러한 경우에



<그림 1> 혼합시뮬레이션 모형

해당된다. 예를 들어 시뮬레이션이 진행되는 도중, 임의로 특정 주문을 취소하는 사건을 생성하려고 사용자 화면을 호출하면 그때부터는 시뮬레이션 시간으로 진행되는 것이 아니라 물리적인 시간으로 진행되어야 한다. 이러한 혼합시뮬레이션 모형을 그림 1과 같이 표현할 수 있다. 그림 1에서 기계 l과 m은 실세 장비가 연결되어 있지 않은 시뮬레이션 상태이고, 기계 k는 실제 기계가 연결되어 있는 상태이다. 초기상태는 기계 l과 기계 k가 작업중이며, 각각 t1과 t3에서 작업이 완료된다. 기계 k의 작업이 완료되면 다음 작업에 대한 계획 알고리즘이 수행되고 그 결과 기계 m에서의 작업이 결정되었다. 기계 k는 실제 기계이므로 기계 l이 시뮬레이션 작업을 수행하고 있다하더라고 t1 시점까지는 실제의 시간으로 전진한다. 마찬가지로 t1-t2 구간에서도 계획 알고리즘이 수행되고 있는 상황이므로 알고리즘 수행시간을 그대로 시뮬레이션에 반영한다. 그러나, t2-t3, t3-t4 구간과 같이 더 이상 물리적 시간진행이 존재하지 않는 경우에는 일반적인 이산형 시뮬레이션 진행방식과 동일하게 시간이 진행된다. 즉, t3과 t4의 사건 시간으로 시뮬레이션 시간이 도약하게 된다. t4 시점 이후에는 다시 계획 알고리즘이 수행되고 있으므로 실제 시간으로 시뮬레이션이 진행된다.

그림 1에서 각 프로세스의 상태, 즉 물리적 시간진행 상태, 또는 논리적 시간진행(시뮬레이션) 상태는 프로세스간 통신 방법에 의하여 중앙시계가 있는 시뮬레이션 시스템으로 전달된다. 이때 사용할 수 있는 방법은 내부통신 방법 또는 외부통신 방법 등 통신에 의존하는 방법이 있고, 분산 데이터베이스를 사용하는 방법이 있다. 두 가지 방법 모두 데이터의 일치성(consistency)을 유지할 수 있는 방안이 필요하다. 통신 메시지를 이용하여 중앙시계를 운영할 때 발생할 수 있는 인과성 오류를 해결하기 위하여 제안된 방법은 그림 2와 같다[3]. 기본적으로 각 프로세스의 상태를 중앙시계로 전달해야 하는데, 이때 전달하려고 보낸 통신의 순서가 수신 프로세스에서 그 순서대로 유지되지 않는 경우(causality error)가 있을 수 있으며 혼합시뮬레이션에서 이러한 경우의 발생은 절대적으로 방지해야한다. 이러한 경우의 발생을 방지하기 위하여 그림 2에서 ③,④와 같은 확인절차를 필요로 하는 것이다[3].



<그림 2> 혼합시뮬레이션 운용방법

이러한 혼합시뮬레이션 모형이 필요한 대표적인 예는, 이론상으로 좋은 해를 산출하는 알고리즘이 실제 사용에 있어서 그 수행시간 때문에 적용되지 않는 경우이다. 일반적인 시뮬레이션으로는 알고리즘의 구현이 시뮬레이션 프로그램 안으로 들어가고 시간진행에도 전혀 영향을 미치지 않는, 즉 시간진행이 0으로 처리되지만, 혼합시뮬레이션의 경우, 알고리즘을 시뮬레이션 프로세스와는 다른 프로세스로 구성하여 그 수행시간을 물리적 시간(그림 1의 물리적 시간진행)으로 처리하기 때문에 고려할 수 있게 되는 것이다.

3.2 객체지향 기법

객체지향 기법은 분석단계, 설계단계, 구현단계에서 각각 적용 가능하다. 분석단계에서는 일반적인 공장 구성 요소들을 추상화하여 클래스(abstract class)로 추출하고 추출된 클래스들 간의 관계를 구축한다. 설계단계에서는 공장 자동화 시스템의 특징과 혼합시뮬레이션의 특징을 반영한 시뮬레이션 시스템 모형을 구축하는데, 이때 주로 사용되는 기법이 Pattern을 이용한 설계[7]와 프레임웍 기법[12]이다. 이때 각 클래스 관계도에 해당하는 데이터베이스 설계 또한 병행된다. 구현단계에서는 분석/설계 단계를 거치면서 생성된 설계사양에 적절한 객체지향 언어와 개발 툴 등을 선정하여 구현한다. 이러한 객체지향적 분석, 설계, 구현 등 전체 개발단계에 대한 방법론으로 Rumbaugh의 OMT[13]나 Booch의 Booch Method[5], Jacobson의 Usecase-driven approach[9] 등이 대표적이다. 최근에 이들의 방법론과 여타의 객체지향 모델링 방법론을 통합한 UML(Unified Modeling Language) 1.1[14]이 소개되었으며, 특히 UML의 경우 OMG(Object Manage Group)의 모델링 표준으로 채택되었다. 본 연구에서 표기법은 UML 1.1의 모델링 표기법을 따랐다.

(1) 객체지향 분석 단계

우선 분석단계에 대하여 알아보자. 분석단계에서 행하는 작업은 대상 공장의 설정(configure)이 각 공장마다 다르므로 객체지향 기법을 사용하여 재사용성과 맞춤형을 가능하게 하는 작업이다. 즉, 여러 형태의 공장을 시뮬레이션 하기 위하여 공장 구성에 대한 추상화 과정을 거친 후 이를 바탕으로 일반적인 공장의 구성 요소들을 추상화된 클래스로 추출하는 것이다. 본 연구에서는 주문, 제품, 프로세스, 스케줄이라는 클래스가 추출되었다.

프로세스(Process) : 시뮬레이션 환경에서 작동하는 기계 또는 공정을 의미한다. 변압기 조립공장의 예에서 조립공정 중 절연과정이 있다면 이 절연과정을 하나의 프로세스로 표현한다. 프로세스는 처리시간 분포 등의 가공정보를 포함하고 있다.

제품(Product) : 단일 혹은 복수의 프로세스를 거쳐 생산되는 생산품을 의미한다. 변압기 조립공장의 경우에는 변압기가 곧 제품을 의미하는 것이다. 제품은 생산 시 거쳐야할 프로세스의 종류와 그 순서를 포함하고 있다.

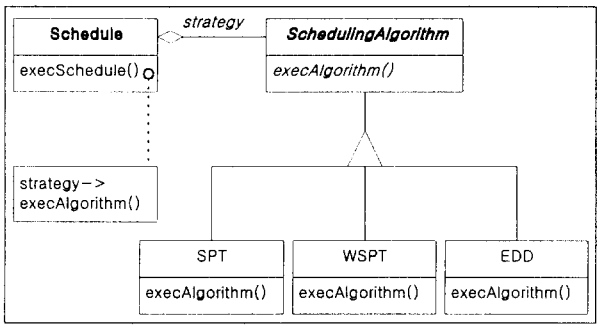
주문(Order) : 고객이 요구하는 제품의 목록서이다. 주문은 고객명과 납기 요구일, 우선 순위 등의 부가적인 요구를 지닐 수 있다.

스케줄(Schedule) : 현재 정의된 프로세스, 제품, 주문 정보를 바탕으로 생산 스케줄을 정의한다. 스케줄 방법으로 사용되는 알고리즘은 여러 가지 방법을 사용할 수 있어야 한다.

(2) 객체지향 설계 단계

다음은 설계단계에 대한 내용이다. 여기서 사용하게 되는 Design Pattern은 설계에 있어서 자주 발생하는 상황에 대처할 수 있는 설계 기법을 pattern화하여 기록하였다가 다음에 비슷한 상황이 발생한 경우 적용할 수 있도록 하는 방법이다[7]. 예를 들어,

"제조시스템의 계획, 통제 부분은 알고리즘에 해당하는 부분이며 대상 시스템이 변하면 계획, 통제 알고리즘도 변할 수 있다. 이러한 변화를 시스템의 다른 부분에 영향을 미치지 않게 하고 싶다."

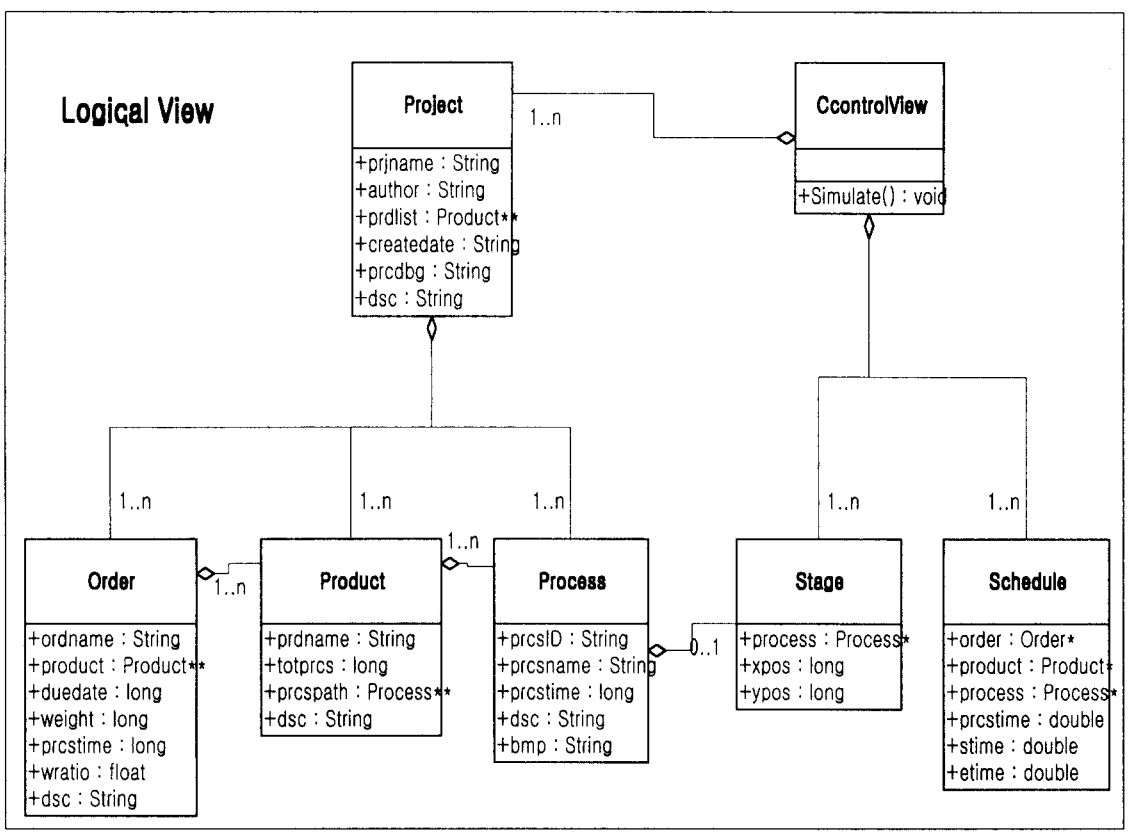


<그림 3> Strategy Pattern의 예

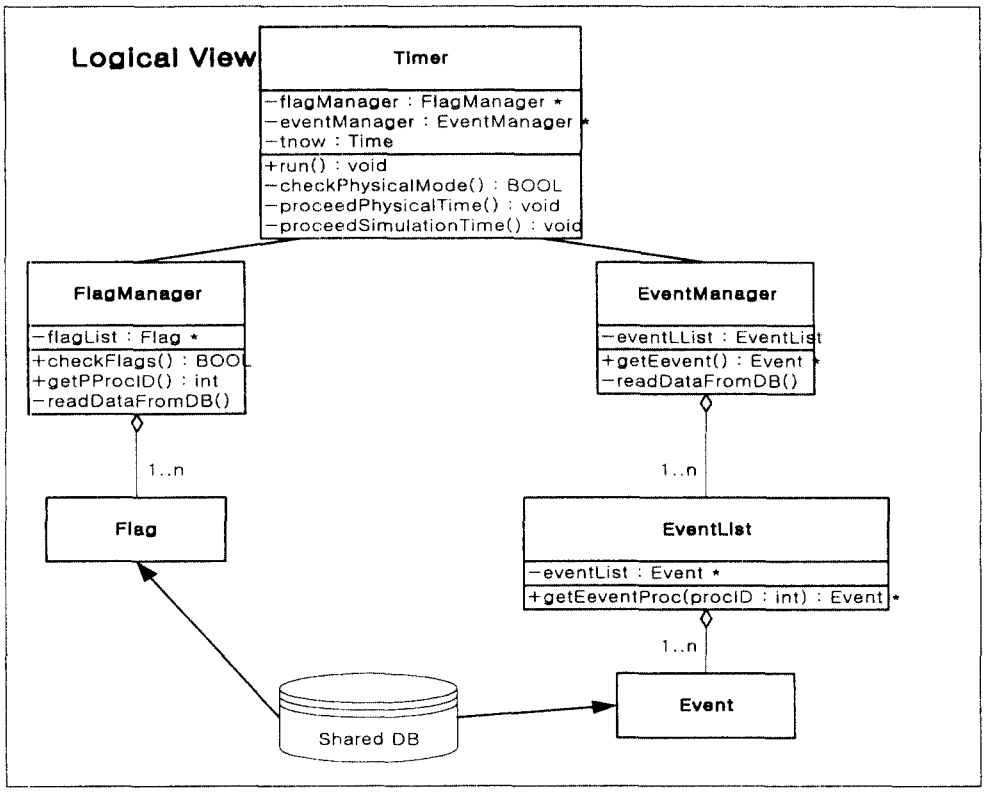
제조시스템이 사용하는 알고리즘의 변화가 다양하기 때문에 새로운 알고리즘이 개발되어 추가되는 경우에 시뮬레이션 시스템의 변화가 없도록 설계를 해야 한다. 이와 같이 객체의 인터페이스(여기서는 Schedule 객체의 execSchedule())의 실제 구현이 달라지는 경우에 그 변화의 파급을 줄일 수 있도록 제안된 패턴이 Strategy Pattern이다[7].

그림 3을 보면 Schedule 객체의 execSchedule() 함수의 실제 구현 부분은 SchedulingAlgorithm이라는 추상 클래스의 execAlgorithm()라는 가상함수를 호출하도록 되어 있다. 만약, 특정 공장에서 사용하는 알고리즘이 기존의 알고리즘이 아니라 새로운 것이라면 사용자가 SchedulingAlgorithm을 상속받아서 새로운 계획 알고리즘 객체를 만들고 이것을 Schedule객체의 구현 부분에 연결시켜 놓으면 Schedule객체와 그 이외의 객체들은 아무런 변화 없이 계속 사용할 수 있다.

본 연구에서는 각 process에서의 스케줄링 알고리즘의 구현 부분이 그림 3과 같은 Strategy Pattern으로 구성되어 기존 알고리즘의 선택은 물론, 새로운 알고리즘의 추가 시에도 시뮬레이션 시스템의 다



<그림 4> 시뮬레이션 시스템 관련 클래스 관계도



<그림 5> Timer 클래스 관계도

른 부분에는 전혀 영향을 미치지 않도록 하였다. 객체지향 기법을 사용하여 설계한 결과는 그림 4와 같다. 분석단계에서 추출된 제품, 프로세스, 주문, 스케줄 클래스와 더불어 사용자 화면을 통한 공장 구성 정보를 저장하는 스테이지, 각 공장관련 전체 정보를 포함하고 있는 프로젝트 클래스 등이 그림 4에서 클래스 관계도로 보여진다.

프로젝트(Project) : 사용자가 정의한 시뮬레이션 환경의 이름이다. 만약 변압기 조립공장을 구성한다면 그 프로젝트명은 변압기 조립공장이다. 프로젝트 안에는 그 외에 작성자 및 전체 설명에 관한 정보를 저장할 수 있다.

스테이지(Stage) : 사용자가 프로세스를 배치하여 만든 시뮬레이션 환경이다. 사용자는 미리 정의한 프로세스를 가지고 화면상에 끌어다 놓기(Drag & Drop)를 구사하여 가상 공장을 구현한다.

(3) 객체지향 구현 단계

본 연구에서는 구현에 적합한 언어로 C++을 사용하였으며 통합 개발환경으로는 MS studio를, 사용자 화면 구현에는 MFC를 이용한 Visual C++4.0을 사용하였다. 그밖에 객체들의 저장소로는 데이터베이스 MS Access를 사용하였으며, 프로세스간 통신은 MS Access를 이용하여 각 프로세스의 상태정보를 접근하였다.

4. 객체지향 혼합시뮬레이션 구현

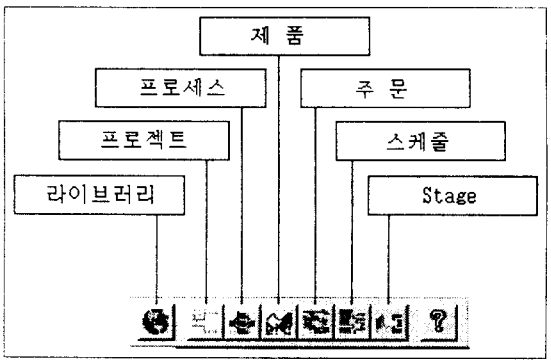
4.1 타이머 구성

구현된 시스템 중 혼합시뮬레이션의 핵심이라고 할 수 있는 타이머를 객체지향 기법을 적용하여 구성한 결과는 그림 5와 같다. 그림에서 보듯이 각 프

로세스의 상태를 나타내는 flag를 관리하는 FlagManager와 각 프로세스의 사건들의 list를 관리하는 EventManager, 그리고 혼합시뮬레이션 시간진행을 담당하는 Timer의 클래스들로 이루어져 있다. 전장에서 설명한 혼합시뮬레이션 시간진행 알고리즘은 Timer 객체에 구현되어 있다.

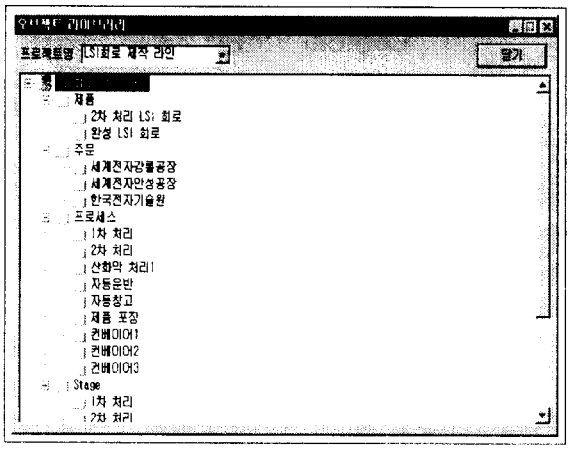
4.2 시스템 구성

구현된 시스템은 시뮬레이션에 필요한 정보를 입력하거나 선택할 수 있는 화면들과 실제 시뮬레이션을 수행하는 화면으로 구성되어 있다. 화면의 메인 툴바(그림 6)에 속해 있는 각각의 버튼들이 이러한 화면들과 시뮬레이션 수행의 역할을 담당하고 있다.



<그림 6> 메인 툴바의 기능

이러한 기능 중 라이브러리는 현재 저장된 모든 객체의 리스트를 출력해주어 사용자로 하여금 현재 사용 가능한 프로젝트들의 세부정보를 검색할 수 있도록 한다. 본 시스템은 그 구조상 각각의 객체를 보기 위해서는 해당 객체 정의 다이얼로그를 열어서 볼 수 있지만 전체적인 구조를 보기에 때로는 불편한 점이 없지 않아서 전체 프로젝트의 구조를 한눈에 볼 수 있도록 설계된 객체 브라우저와 같은 것이다. 그림 7은 'LSI회로 제작 라인'이란 시뮬레이션 프로젝트내의 객체들을 출력하고 있다. 해당 프로젝트 내에는 제품, 주문, 프로세스, 스테이지 등의 객체들이 분류되어 있으며, 이와 같은 정보는 객체저장소인 데이터베이스에 저장되어 있어서 사용자가 원하는 시점에 원하는 프로젝트를 선택할 수 있도록 되어있다..



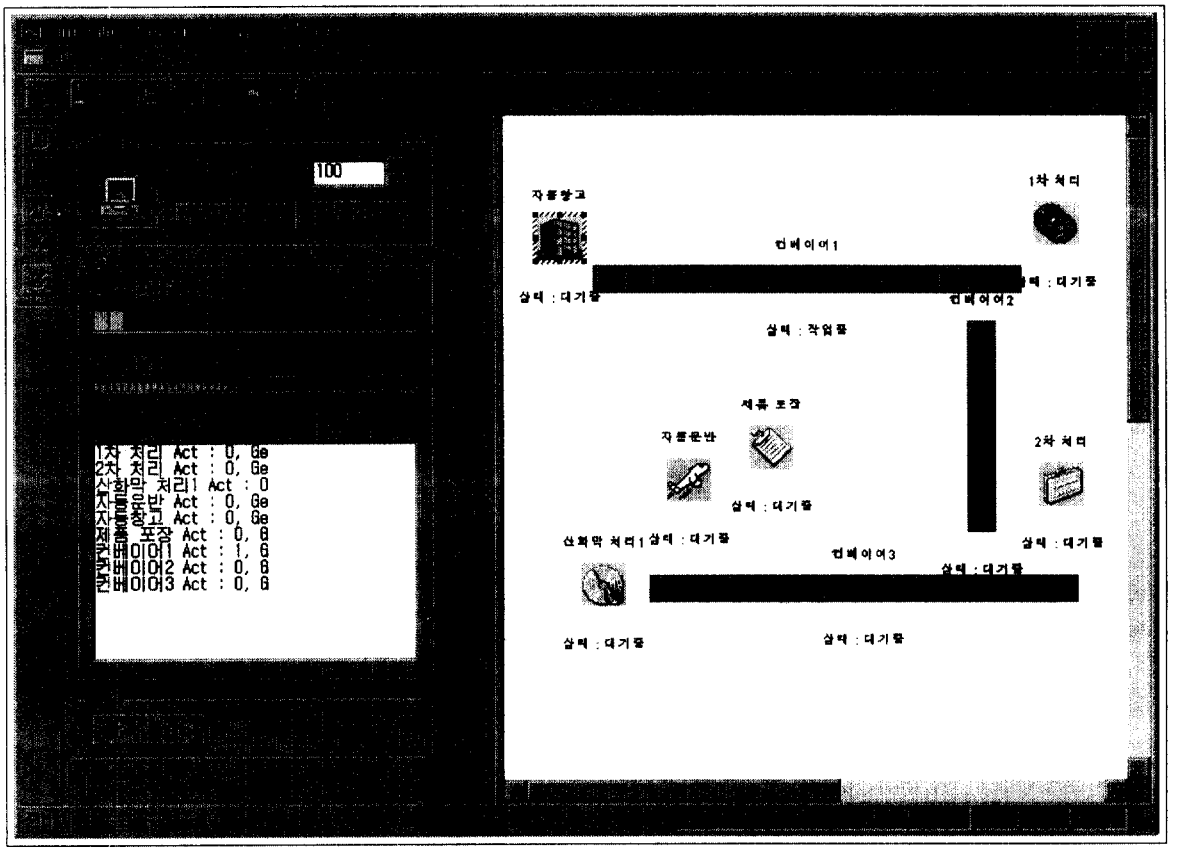
<그림 7> 라이브러리 윈도우

시뮬레이션을 수행하기 위해서 해야 할 작업은 1) 프로젝트의 설정, 2) 프로세스, 제품의 선택 및 정의, 3) 주문의 선택 및 정의이다. 이러한 작업이 완료되면 Stage정의 윈도우를 통해서 앞에서 정의된 프로세스 객체를 이용하여 구현하고자 했던 대상 공장을 만드는 작업을 한다. 그 조작은 Stage 조작 툴바를 이용하여 간단히 이루어 질 수 있다. 그림 8은 스테이지를 정의한 예이다.

그림 8의 좌측에 나타나 있는 시뮬레이션 조정 윈도우는 정의된 Stage를 시작하고 정지시키는 기능과 함께 현재 중앙시계의 상태를 담은 정보가 나타내어 시뮬레이션 상황을 종합적으로 표현해주는 부분이다. 전체 진행도는 사용자가 설정한 시뮬레이션 시간에서 현재 진행시간의 진척정도를 보여준다. 현재 시뮬레이션 시간은 현재 중앙시계의 시간이 나타난다. 서버 환경 변수는 중앙시계에 저장되어 있는 Active Flag와 Generation Flag의 값을 보여준다. 그림 8은 구현된 혼합시뮬레이션 시스템의 작업 화면이다.

4.3 결과 및 적용 예

본 연구에서 개발된 시뮬레이션 시스템을 이용하여 가상 전자부품공장을 설정한 후 LSI회로 조립공정의 생산 모의실험을 실시하기로 하자. 다음은 LSI회로 공장의 각 객체들을 정의한 내용이다.



<그림 8> 구현된 객체지향 혼합시뮬레이션 시스템

프로젝트명 : LSI회로 조립 라인

등록된 프로세스 : 등록된 프로세스는 총 아홉 가지이며 각각의 특성은 <표 1>과 같다.

<표 1> LSI회로 프로젝트에 등록된 프로세스 종류

프로세스 명	처리시간	최소	최대
산화막 처리1	10	5	15
컨베이어1	8	3	13
1차 처리	10	8	12
컨베이어2	6	4	8
2차 처리	6	4	8
컨베이어3	10	7	13
자동운반	6	3	9
제품 포장	10	9	13
자동창고	2	1	3

등록된 제품 : 등록된 제품은 모두 두 가지이며 각각의 특성은 <표 2>와 같다.

<표 2> LSI 회로 프로젝트에 등록된 제품

제품명	개수	선택한 프로세스 및 경로
완성 LSI 회로	9	자동창고 : 컨베이어1 : 1차 처리 : 컨베이어2 : 2차 처리 : 컨베이어3 : 산화막 처리1 : 자동운반 : 제품 포장 :
2차 처리 LSI 회로	7	자동창고 : 컨베이어1 : 1차 처리 : 컨베이어2 : 2차 처리 : 자동운반 : 제품 포장 :

등록된 주문 : 등록된 주문은 총 3가지이며 이들의 내용을 간략하게 정리하면 <표 3>과 같다. 비율의 계산은 '처리시간/가중치' 수식을 따른다.

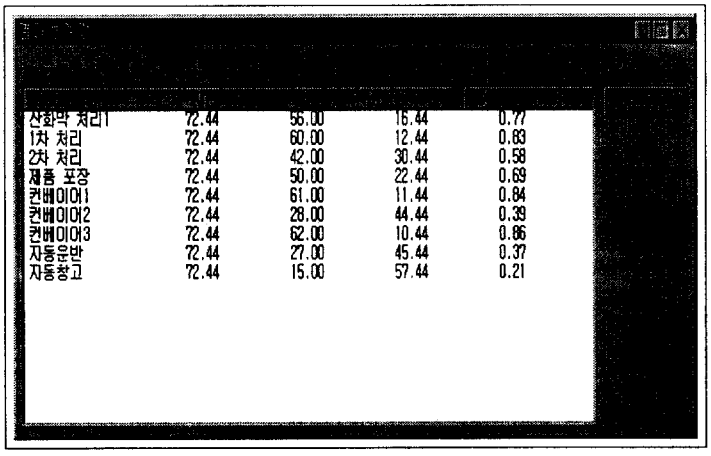
<표 3> LSI회로 프로젝트에 등록된 주문

주문명	제품명	납기일	가중치	시간	비율
세계전자안성공장	완성 LSI 회로	30	2	68	34
한국전자기술원	2차 처리 LSI 회로	60	1	48	48
세계전자강릉공장	완성 LSI 회로	80	1	68	68

스케줄 시행 : 위의 기술한 세 가지 정보를 이용하여 스케줄링을 했을 때 적용하는 스케줄링 방법에 따라 아래와 같은 결과를 얻었다.

등록된 Stage : 등록된 Stage는 앞에서 등록된 프로세스를 모두 이용하였다.

시뮬레이션의 시행 : 스케줄링 방법으로는 SPT (Shortest Processing Time)를 사용하여 우선 작업 순위를 결정하였고, 각 프로세스의 작업시간 발생은 균일 분포를 사용하였다. 시뮬레이션은 총 시간은 69 시간이었다.



<그림 9> 시뮬레이션 실행 결과

5. 결론 및 추후 연구사항

연구에서는 대상 생산 시스템 시뮬레이션에서 고려되는 다수의 요소들을 객체화하여 공장환경 하에서의 혼합시뮬레이션을 구현하고 실제 공장에 적용하였다. 연구의 전체적인 중점 사항은 동적(Dynamic), 분산(Distributd), 혼합(Mixed) 시뮬레이션의 모델을 통합하는데 용이한 방법의 하나로 상황 또는 환경 변화에 재사용이 용이 할 수 있도록 요소들을 객체화하여 구현하는데 있다. 위의 세 가지 요소는 본 연구의 선행연구에 해당하는 연구 기간 중에 정의된 바 있고 그를 바탕으로 클래스들을 기반으로 하는 시뮬레이션 시스템을 완성하였다.

현재 본 시뮬레이션 시스템에서 정의된 핵심 객체의 종류는 모두 여섯 가지이다. 한편 이들을 이용하여 여러 상황이 전개될 수 있는 대다수의 실제 제조 생산 시스템의 경우를 한결같이 적용하는 데에는 한계가 뒤따른다고 볼 수 있다. 보다 더 실제 상황에 가깝고 현실적인 시뮬레이션 모델의 구축 및 이를 시행하여 이용하기 위해서는 기존에 설정된 클래스를 계승(Inheritance)하여 각각의 특성에 맞는 객체를 정의하는 것이 요구된다.

추가적인 요구사항으로 시뮬레이션 시행 속도의 조정 기능, 시각적인 효과를 감안한 애니메이션 기능, 결과 값에 대한 리포트 기능, Stage 작성 템플릿(Template) 기능, 다양한 통계적 분석 방법의 제공 등은 이 연구와 관련하여 고려할 수 있는 향후 과제로 꼽을 수 있겠다.

참고문헌

[1] 강용혁, 서동욱, 김성식, "다공장구조의 생산계획 및 통제시스템에 대한 객체지향적 분석 및 설계", *시뮬레이션학회지*, 1996.

[2] 배경한, "Shell을 이용한 FMS 구축방법에 관한 연구", 고려대학교 산업공학과 박사논문, December 1992.

[3] 서동욱, 전진, 김성식, "혼합시뮬레이션에서의 causality error 해결방안", *시뮬레이션학회지*, Vol.4, No.2, pp.31-40, 1996.

[4] Adiga, S., *Object-Oriented Software for Manufacturing Systems*, Chapman & Hall, 1993.

[5] Booch, G., *Object-Oriented Analysis and Design*, Addison-Wesley, 1994.

[6] Fujimoto, R. M., "Parallel Discrete Event Simulation", *Comm. of the ACM*, Vol.33, No.10, pp.31-53, October 1990.

[7] Gamma, E. et al, *Design Patterns*, Addison-Wesley, 1994.

[8] Greenwood, N. R., *Implementing Flexible Manufacturing Systems*, Maxmillian Education Ltd., 1988.

[9] Jacobson, I., *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison Wesley, 1992.

[10] Kim, S. S., "A Synchronization Scheme For Real And Simulated Events", *Computers & Industrial Engineering*, Vol.27, No.1-4, pp.181-184, 1994.

[11] Misra, J., "Distributed Discrete-Event Simulation", *Computing Surveys*, Vol.18, No.1, pp.39-65, March 1986.

[12] Rogers, G. F., *Framework-Based Software Development in C++*, Prentice-Hall, 1997.

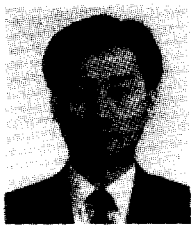
[13] Rumbaugh, J. et al., *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[14] OMG, *Unified Modeling Language ver 1.1*, www.omg.org

● 저자소개 ●

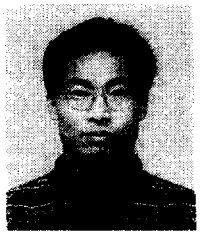
김성식

1972년 고려대학교 기계공학과 학사
1974년 고려대학교 산업공학과 석사
1976년 미국 S.M.U산업공학과 석사
1979년 미국 S.M.U산업공학과 박사
1979~현재 고려대학교 산업공학과 교수



홍윤기

1977~78년 해군본부 정책발전실
1980년 고려대학교 산업공학과 학사
1980~81년 (주)한국건업(현 벽산건설)기획관리
1985년 Univ. of Southern California O.R 석사
1989년 Univ. of Southern California 산업공학 박사
1989~91년 Calfornia state Univ. Northridge 조교수
현 재 한성대학교 산업공학과 교수



전 진

1992년 고려대학교 산업공학과 학사
1994년 고려대학교 산업공학과 석사
1994~현재 고려대학교 산업공학과 박사과정

강동우

1996년 한성대학교 정보전산학부 전산통계학과 학사
현 재 삼성전자 미디어 서비스 사업팀 비즈니스 S/W그룹 연구원으로 재직중