웹 기반 자동창고시스템 시뮬레이터의 개발 : Web-SAW

임대진*, 박양병**

Development of Web-based Simulator for an Automated Warehouse System : Web-SAW

Dae-Jin Lim, Yang-Byung Park

Abstract

Simulation has been recognized as one of major application areas of world wide web. Over the last several years, most research has been focused on the development of web-based simulation languages mostly using JAVA and validation of simulation modeling on web with the newly developed languages. In this paper, we develop a tentative web-based simulator for a simple unit-load automated warehouse system, named Web-SAW. In constructing Web-SAW, we program a user interface program, a simulation logic program, and an animation program with JAVA while utilizing the class libray functions and embodiment environment of existing web-based simulation languages. Web-SAW simulates the automated warehouse system with the user input about system operation, displays the animation on a static background representing the system, and generates the various textual outputs after simulation.

Key Words: world wide web, simulation, automated warehouse system, JAVA

* B&B(주) BI 사업부

** 경희대학교 테크노공학대학 기계·산업시스템공학부

1. 서론

웹상에서 애플리케이션(application)을 구현하는 새로운 형식은 웹의 사용과 그에 따른 기술의 발전으로 애플리케이션의 모든 분야에 영향을 미치고 있으며, 이에 따라 기업과 사회의 관심이 점점 높아지고 있다. 시뮬레이션 분야도 역시 이러한 영향으로 인해 웹 환경에서의 실행을 위한 연구가 활발히 진행되고 있으며, 대부분의 경우 웹 프로그래밍 언어인 자바(JAVA)를 기반으로 이루어지고 있다. 1990년대 중반까지의 웹 기반 시뮬레이션에 대한 연구는 주로 자바를 이용하여 간단한 대기시스템의 시뮬레이션을 웹상에서 구현하는 개념과 설계기술에 제한되었다(Kilgore *et al.*, 1998). 그러나 그 후에 웹의 급격한 확산과 웹 구현 환경의 기술이 비약적으로 발전됨에 따라 웹 기반 시뮬레이션을 위한 언어 및 구현환경에 대한 연구가 본격적으로 진행되었다.

자바는 애플리케이션을 웹상에서 구현할 수 있는 대표적인 객체지향 프로그래밍 언어이다. 특히 자바는 애플리케이션을 웹상에 설계, 구현하는데 플랫폼(platform)의 독립성을 가지고 있어 어떤 운영체계 및 사용환경에서도 호환될 수 있는 장점을 가지고 있다. 이것은 자바가 분산 객체모델을 다루는데 있어 클라이언트-서버 모델에서 클라이언트와 서버를 미리 구분하지 않고 양쪽 다 모듈들을 객체단위로 구현하여 서버가 클라이언트에게 미리 구축되어 있는 환경을 통해 분산 서비스를 제공케 함으로써 객체들 사이에서 쉽게 호환이 이루어지도록 해 주기 때문이다. 즉, 특정 하드웨어나 소프트웨어 환경에 관계없이 웹상에서 인터페이스를 통하여 원격으로 객체를 엑세스하여 애플리케이션을 직접 실행 및 제어할 수 있게 해 준다. 이러한 이유로 자바는 애플리케이션을 웹상에 구현시키는데 사용되는 표준언어로 자리잡고 있다.

지금까지의 웹 기반 시뮬레이션에 관한 연구는 대부분의 경우 자바를 기본으로 한 웹 기반 시뮬레이션 언어를 개발하고, 웹에 시뮬레이션 모델을 구현할 수 있는 가능성을 확인해 주는 내용이었다. 웹 기반 시뮬레이션 언어의 개발은 크게 두 가지 방식을 따르고 있다. 첫 번째는 자바로 만들어 진 시뮬레이션용 클래스를 추가하여 새로운 시뮬레이션 언어를 개발하는 방식이다. 대표적으로, JAVASIM (Little, 2002), SIMJAVA(Howell and McNab, 1998), JSIM(Miller, 1997), SILK(Herly and Kilgore, 1997) 등이 있다. 두 번째로는 기존의 범용 시뮬레이션 언어를 웹상에서 실행할 수 있도록 자바로 변환시키는 방식이다. 여기에는 JAVAGPSS(Klein *et al.*, 1998)가 있다. 하지만 새로운 웹 기반 시뮬레이션 언어의 계속되는 소개에도 불구하고 이를 사용한 시뮬레이션 프로그램의 개발은 극히 미미한 실정이다. 이러한 현상은 지금까지 연구되어 온 웹 기반 시뮬레이션 언어들이 주로 자바의 구현방법 개발에 치우쳐 인터넷의 중요한 특징인 사용의 용이성과 범용성을 거의 무시한데 기인한 것으로 보인다.

본 논문에서는 지금까지 연구되어 온 주요 웹 기반 시뮬레이션 언어의 구현환경과 특성을 조사하고, 기존의 웹 기반 시뮬레이션 언어와 자바를 이용하여 저자들이 개발한 실험용 웹 기반 자동창고시스템 시뮬레이터, Web-SAW (Web-based simulator for an automated warehouse system)를 소개한다. 서론에 이어, 제 2장에서는 기존의 웹 기반 시뮬레이션 언어와 구현환경을 고찰한다. 제 3장에서 Web-SAW의 설계와 개발, 그리고 제 4장에서 Web-SAW의 실행을 기술한다. 끝으로, 결론과 향후과제를 제 5장에서 정리한다.

2. 기존의 웹 기반 시뮬레이션 언어

웹상에 구현할 수 있는 기존의 이산사건 시뮬레이션 언어는 많지 않다. 이들 언어는 시뮬레이션의 주요 기능을 수행하는 메소드(method)와 클래스 라이브러리(class library)

를 거의 모두 자바로 구현하였으며, 작성된 시뮬레이션 모델을 그래픽 형태로 보여 준다.

2.1 SIMJAVA

SIMJAVA(Howell and McNab, 1998)는 복잡한 시스템을 모델링 할 수 있는 프로세스 기반의 이산사건 시뮬레이션 언어로서, 자바로 만들어져 있다. SIMJAVA는 화면상에 제공되어 있는 아이콘을 이용하여 사건 객체 프로그램을 작성하고, 구축된 시뮬레이션 모델을 웹 문서에서 다이어그램으로 나타낼 수 있는 기능을 보유하고 있다. 시뮬레이션 과정에서 생성된 개체들은 쓰레드(thread)를 통하여 개별적으로 실행되고, 이들 개체는 포트(port)에 의해 함께 연결되어져 사건 객체와 교류함으로써 모든 개체들의 정보는 공유된다. 중앙의 시스템 클래스는 모든 쓰레드를 제어하고, 시뮬레이션 시간을 진행하여 사건들을 연결시킨다. 시뮬레이션의 진행과정에서 개체는 추적(trace) 메시지를 발생하여 정보를 파일에 저장시킨다.
(http:www.dcs.ed.ac.uk/home/hase/simjava, 2002.)

2.2 JSIM

JSIM(Miller, 1997; Miller *et al.*, 1998)은 자바를 기반으로 한 시뮬레이션 언어이고, GUI와 애니메이션 환경을 프로그래밍 할 수 있는 기능과 JavaBeans를 이용한 컴포넌트(component) 기반의 기술을 제공한다. 개체의 대기행렬을 보여줄 수 있을 정도의 아주 좋은 그래픽 환경을 구축할 수 있다. 컴포넌트 기반 기술을 이용함으로써 컴포넌트들의 조합과 재사용을 통하여 시뮬레이션 모델을 구축한다. JSIM에서 시뮬레이션 모델은 사건 및 프로세스 패키지를 사용하여 구축할 수 있다. 이들 패키지는 객체의 활동과 기능을 포함하고 있는 컴포넌트들의 집합이다. 시뮬레이션 과정에서 얻어진 결과 값을 저장하기 위해 자바 데이터베이스를 사용한다. JSIM의 장점은 시스템의 모델링 작업이 유연하고, 또한 특정 하드웨어나 소프트웨어 환경에 구애받지 않는 독립적인 플랫폼을 제공하는 것이다. 그리고 사용자가 웹상에서 시뮬레이션 모델을 개발하고 이를 실행하는데 있어 친숙한 그래픽 환경을 제공한다. JSIM은 개방 소스 소프트웨어이므로 웹에서 누구나 자유롭게 소스를 이용할 수 있다.
(http://orion.cs.uga,edu:5080/~jam/home/, 2002)

2.3 JAVASIM

JAVASIM(Little, 2002)은 원래 C++SIM 시뮬레이션 툴키트(toolkit)를 자바로 변환한 언어이다. 디자인과 기능 면에서는 SIMULA로부터 상속을 받아 이를 더 확장하였다. 이를테면, I/O facilities, 변수생성기, 분산객체들이 클래스의 상속에 의존하여 구현될 수 있도록 완전한 통합을 이루었다. 이에 따라 JAVASIM은 SIMULA 보다 더 유연하고 확장성이 높은 언어로 인식되고 있으며, 학습과 사용이 비교적 용이하다.
(http://javasim,ncl,ac.uk/, 2002)

2.4 JAVAGPSS

JAVAGPSS(Klein *et al.*, 1998)는 GPSS 범용 시뮬레이션 언어로 구축된 시뮬레이션 모델을 웹상에서 구현하기 위해 만들어 진 컴파일러 프로그램이다. 즉, JAVAGPSS는 GPSS 소스코드 파일을 자바 소스코드로 변환하여 GPSS 프로그램이 웹상에서 실행되도록 해 주는 기능을 수행한다. JAVAGPSS에 의해 변환된 GPSS 시뮬레이션 모델은 일반 웹 기반 시뮬레이션 언어처럼 어떠한 인터넷 브라우저에서도 애플릿처럼 실행될 수 있으며, 실행결과는 사용자에게 제공된다.

2.5 SILK

SILK(Herly and Kilgore, 1997; Kilgore and Burke, 2000)는 프로세스 상호작용을 기반으로 실행되는 웹 기반 시뮬레이션 언어이다. 모델을 설계하기 위한 도구로서의 역할을 충분히 수행할 수 있도록 JavaBeans를 기반으로 만들어져 있다. SILK는 시뮬레이션 모델의 구축에서 컴포넌트를 재사용 할 수 있으며, 제한된 적용분야에 대해서는 아주 간단히 모델링 할 수 있는 일종의 시뮬레이터 기능도 가지고 있다. 특히, SILK는 자바 프로그램 응용 개발 도구인 Symantecs, VisualCafe, IBMs VisualAge, MS J++ 의 환경에서 시뮬레이션을 구현할 수 있도록 그래픽 인터페이스를 제공한다.

2.6 WSE

WSE(Iazeolla and D'Ambrogio, 1998)는 자바와 CORBA(common object request broker architecture)라는 두 개의 웹 기술이 결합된 웹 기반 시뮬레이션 언어이다. 이 두 기술의 결합으로 사용자는 시뮬레이션 모델에 곧바로 엑세스 할 수 있고, 모델을 추적 및 변경할 수 있고, 또 이들과 인터넷상에서 교호작용을 할 수 있다. 이것은 초기에 시뮬레이션 모델을 사용자 환경에 포함할 수 있는 구조(architecture)에 플러그 인 함으로써 가능하다.

일부 기존의 웹 기반 시뮬레이션 언어들에 있어 XML, JavaBeans와 같은 기술의 활용은 전혀 새로운 언어의 개발을 의미하는 것은 아니다(Bray *et al.*, 2002). 웹 기반 시뮬레이션 언어 역시 기존의 범용 시뮬레이션 언어와 크게 다르지 않게 플랫폼, 유연성, 재사용, 확장성 등의 면에서 보수성이 그대로 존재하고 있다. 인터넷을 이용하여 웹 기반 시뮬레이션 언어를 조사하는 과정에서 실제로 존재하지 않은 웹사이트가 확인되었고, 대부분의 언어들은 초기버전 이후 전혀 업그레이드가 이루어지지 않았다. 이것은 시스템의 시뮬레이션을 위해 웹 기반 시뮬레이션 언어를 사용하는 사람이 많지 않다는 것을 단편적으로 보여준다. 이들 언어는 단순히 교육 및 연구 목적으로 사용되고 있는 것으로 판단된다.

이와 같은 현상은 웹에서 모델링 하는데 필요한 함수나 기능들을 "어떻게 하면 쉽게 구현할 수 있을까?"가 아니라, '어떻게 구현할 수 있을까?' 에 초점이 맞추어져 개발되었기 때문으로 보인다. 이에 따라 개발된 언어는 배우고 사용하기가 쉽지 않을 뿐만 아니라 웹의 주요 특성(장점)들과 부합되지 않아서, 결국 기존의 범용 시뮬레이션 언어와 비교하여 사용환경에 커다란 이점이 없게 된 것이다.

앞서 소개한 웹 기반 시뮬레이션 언어들의 일반적 비교를 <표 1>에 정리한다. 표에 나타난 바와 같이, 웹 기반 시뮬레이션 언어는 주로 90년대 후반에 개발되었는데, 웹 기술의 발전과 보급이 활발했던 시기와 일치한다. 이 후에는 새로운 웹 기반 시뮬레이션 언어가 소개되지 않고 있으며, 기존의 언어들에 새로운 웹 기술을 부가하는 연구만이 이루어지고 있다. 인터넷 조사 과정에서, <표 1>의 웹 기반 시뮬레이션 언어 중에서 비교적 활발히 사용되고 있는 언어는 SIMJAVA와 JSIM으로 나타났다. JAVAGPSS와 WSE는 순수 자바로 만들어져 있지 않은 이유인지 별로 사용되지 않고 있다.

<표 1> 웹 기반 시뮬레이션 언어들의 비교

		SIM JAVA	JSIM	JAVA SIM	JAVA GPSS	SILK	WSE
개발자		Howell & McNab	Miller *et al.*	Little	Klein *et al.*	Herly & Kilgore	Iazeolla & D'Ambrogio
개발시기		1998	1998	1996	1998	1997	1998
특성	Java-Beans	미사용	사용	미사용	미사용	사용	미사용
	재사용	난이	용이	난이	난이	용이	난이
	편의성	낮음	높음	낮음	높음	높음	낮음
	유연성	보통	높음	보통	낮음	높음	낮음
	XML	사용	미사용	미사용	미사용	미사용	미사용

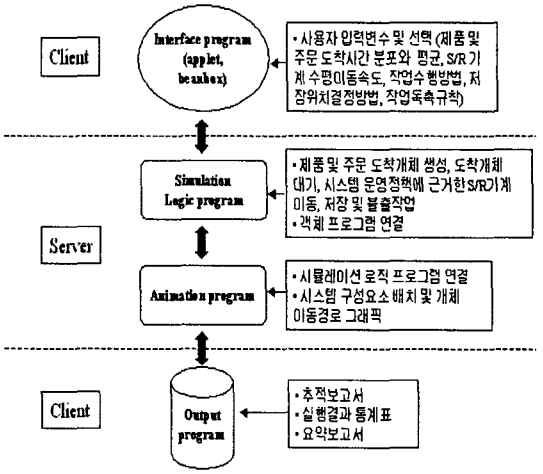
3. Web-SAW의 개발

3.1 개요

웹 기반 시뮬레이션 언어로써 시스템을 모델링 하기 위해서는 해당 언어, 자바, 시뮬레이션 모델링에 관한 전문적인 지식을 필요로 한다. 따라서 일반인은 쉽게 사용할 수 없게 되고, 웹 기반 시뮬레이션의 장점에도 불구하고 그 활용수준은 기대에 미치지 못하고 있다. 이러한 현실을 극복하는 한 가지 대안으로 웹 기반 전용 시뮬레이터의 보급을 고려해 볼 수 있다. 웹 기반 전용 시뮬레이터는 사용자가 인터넷상에서 GUI를 통해 입력값과 메뉴선택만을 간단히 제공하면 되기 때문에, 일반인도 전문적인 지식 없이 장소와 시간에 구애받지 않고 쉽게 시뮬레이션을 수행할 수 있게 된다.

Web-SAW의 사용상 특징을 정리하면 다음과 같다: (i) 사용자는 시뮬레이션 모델링과 자바 등의 시뮬레이션 관련 전문지식을 필요로 하지 않는다. (ii) 사용자는 시간과 장소에 제약 없이 시뮬레이션을 수행할 수 있다. (iii) 사용자의 플랫폼 환경에 구애를 받지 않는다. (iv) 사용자는 간단한 입력과 메뉴 선택만으로 실행결과를 제공받는다. (v) 먼 거리에 분산되어 있는 여러 사용자를 공동으로 훈련 및 교육할 수 있다. 특히, 시뮬레이션의 웹상 구현은 서로 다른 장소에 위치한 여러 사용자가 웹을 통하여 공동으로 작업을 수행함을 중요한 목적으로 하고 있기 때문에, 일반적인 애플리케이션의 공간 제약성을 극복하여 시간과 비용을 크게 절감할 수 있는 이점을 기대할 수 있다(Kilgore *et al.*, 1998).

본 논문에서는 저자들이 자동창고시스템을 대상으로 개발한 실험용 웹 기반 전용 시뮬레이터(Web-SAW)를 소개한다. Web-SAW를 개발하는데 있어 기존의 웹 기반 시뮬레이션 언어로부터 시뮬레이션 전용 클래스 라이브러리 함수의 소스 프로그램을 지원 받으면서, 사용자와의 인터페이스 프로그램, 자동창고시스템의 운영에 따른 시뮬레이션 로직 프로그램, 시뮬레이션 동안 시스템의 동적 상황을 보여주는 애니메이션 프로그램을 자바로 작성하였다. <그림 1>은 Web-SAW 프로그램의 구성을 보여준다.

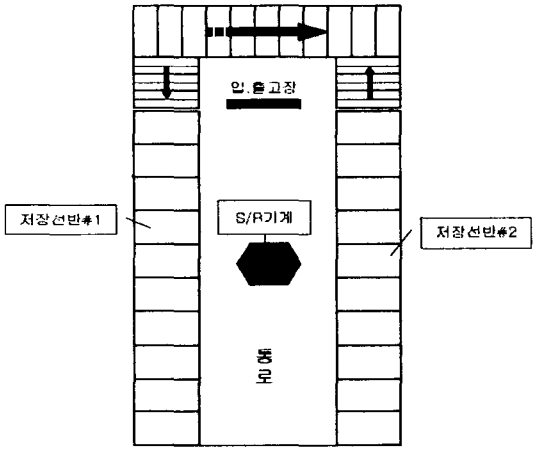


<그림 1> Web-SAW 프로그램 구성

자동창고시스템은 자재의 저장 및 불출과 관련된 장비와 운영이 자동화된 첨단의 창고시스템이다(Groover, 2001). 자동창고시스템은 컴퓨터의 통제로 재래의 창고보다 더욱 신속 안전하게 그리고 효율적으로 저장 및 불출기능을 수행한다. 특히, 최근의 공장자동화 추세에 따라 창고시스템의 자동화는 제조와 분배기능을 연결시켜 주는 필수적인 기능으로 매우 중요하다. 자동창고시스템의 운영정책은 시스템의 수행도에 크게 영향을 미친다. 즉, S/R 기계의 작업방법(command cycle), 제품의 저장위치 결정방법, 작업독촉규칙 등에 따라 시스템의 일일 작업량(throughput), 작업소요시간, 작업대기물량, S/R 기계 이용률 등이 달라진다. 따라서 자동창고시스템의 운영에서 최적의 운영정책 결정은 매우 중요하다.

Web-SAW는 간단한 단일품목 단위적재(unit load) 자동창고시스템을 대상으로 하며, 시스템의 구조는 <그림 2>와 같다. 시스템은

통로 양옆에 저장 칸이 각각 10개인 1단 저장 선반 두 개, 통로에서 저장선반의 시작점에 위치한 통합 입.출고장, 수평방향으로 왕복 이동하는 S/R 기계 한 대로 구성되어 있다. 자동창고시스템을 이와 같이 단순하게 구성한 이유는 전적으로 프로그램 작성의 어려움에 기인한 것이다.



<그림 2> Web-SAW의 대상 단위적재 자동창고시스템

3.2 인터페이스 프로그램

웹상에서 사용자에게 그래픽 인터페이스를 제공하기 위해 자바의 애플릿(applet)과 빈박스(beanbox)를 이용하여 Web-SAW의 인터페이스 프로그램을 작성하였다. Web-SAW의 인터페이스는 크게 시뮬레이션에 필요한 데이터 입력, 시뮬레이션 컨트롤 선택, 애니메이션 컨트롤 선택의 세 부문으로 나누어진다.

데이터 입력 부분은 자동창고시스템의 시뮬레이션에 필요한 외생변수 및 운영정책 선택을 사용자로부터 직접 입력받기 위함이다. <그림 9>의 하단 입력 창에 나타나 있는 바와 같이, 화면상에는 최대 세 종류 제품의 도착 및 주문분포의 선택과 해당 분포의 평균값, S/R 기계의 수평 및 수직 이동시간, 세 가지 운영정책 각각에 대한 2~3개 대안의 선택을 입력할 수 있도록 설계되어 있다.

그러나 Web-SAW의 실제 사용 시에는 지수분포를 따르는 제품 1의 평균 도착 및 주문시간 간격과 S/R 기계의 수평이동시간, 운영정책으로는 단지 단일명령 작업방법(single command cycle), 임의위치 저장방법(randomized storage location method), 선입선출 독촉규칙(FIFO dispatching rule) 대안의 선택만을 입력받도록 프로그램을 작성하였다. 여기서 S/R 기계의 수평이동시간은 한 개 저장 칸 폭의 수평이동시간을 의미한다. 예를 들어, S/R 기계의 수평이동시간이 20이면, 입고장으로부터 첫 번째와 두 번째 저장 칸까지의 이동시간은 각각 20과 40이다. 현재 허용되지 않는 입력의 인터페이스 객체 실행 프로그램은 향후 지속적인 연구를 통해서 추가될 수 있기를 기대한다.

데이터 입력을 위한 인터페이스 화면 구현은 자바의 AWT 클래스 중에서 PANEL 클래스를 사용하여 데이터 입력 창에 들어가는 각종 컴포넌트를 패널에 포함시킨 다음, 이 패널을 컨테이너에 부착시키는 방법을 적용하였다. 라벨과 수치 입력을 위해서 Label과 TextField 컴포넌트, 분포선택을 위해서 Choice 컴포넌트, 운영정책 대안 선택을 위해서 Checkbox 컴포넌트 프로그램을 작성하였다. 사용자의 입력이 시뮬레이션에 적용될 수 있도록 이들 프로그램 모두에 이벤트 핸들러를 포함하였다.

시뮬레이션 컨트롤 인터페이스는 Web-SAW의 기본화면 상단에서 'Start Simulation', 'Stop Simulation', 'Toggle Trace'의 세 가지 선택항목으로 구성된 'Simulation Control' 메뉴를 선택함으로써 이루어진다. 이를 구현하기 위해 자바의 AWT 클래스 중 Menu 컴포넌트를 사용하였다. 'Toggle Trace'는 시뮬레이션 과정에서 제품 및 주문의 도착, 이동, 저장, 불출 사건을 지속

적으로 추적하여 그 내용을 모두 도스화면에서 보여주는 기능으로서, 시뮬레이션이 오류없이 제대로 진행되었는지를 확인하는데 사용된다. 기본화면에서 한 항목을 선택하면 이벤트 핸들러 'ActionListener'에 의해 해당 기능이 실행된다.

애니메이션 컨트롤 인터페이스는 시뮬레이션의 시작과 함께 화면에서 실행되는 애니메이션의 속도를 사용자가 직접 조절하기 위한 기능으로서, 기본화면 상단에서 'Speed Up', 'Speed Down'의 두 가지 선택항목으로 구성된 'Animation Speed' 메뉴를 선택함으로써 이루어진다. 'Speed Up'을 선택하면 애니메이션 속도가 1 증가하고, 'Speed Down'을 선택하면 속도가 1 감소한다. 프로그램 작성 방식은 시뮬레이션 콘트롤과 유사하다.

3.3 시뮬레이션 로직 프로그램

시뮬레이션 로직 프로그램은 자동창고시스템의 운영을 시뮬레이션 하는 프로그램으로서, 데이터 입력, 시뮬레이션과 애니메이션의 컨트롤 선택과 연동된다. 시뮬레이션 로직 프로그램의 작성에는 시뮬레이션 클래스 라이브러리 함수와 그에 해당되는 메소드의 이해, 자바에 대한 지식, 시뮬레이션 모델링에 대한 경험이 필요하다.

자동창고시스템에서 제품 및 주문의 도착, 이동, 대기, 불출, 통계분석, 운영정책 등의 기능과 관련된 기존의 패키지, 클래스, 메소드의 일부를 <그림 3>에 정리한다. 패키지 JSIM.VARIATE는 제품 및 주문의 도착시간 분포와 관련된 기능, JSIM.EVENT는 이산사건 방식의 시뮬레이션 모델링 기능, JSIM.STATISTICS는 수행결과의 통계분석 기능, JSIM.PROCESS는 프로세스 기반의 시뮬레이션 모델링 기능, JSIM.QUEUE는 개체의 대기와 서비스 순서 등과 관련된 기능을 포함한다.

저자들은 이러한 기존 클래스 라이브러리 함수에서 시뮬레이션 메소드의 필요한 부분을 재 정의하여 Web-SAW의 개발에 사용하였다. 즉, 시뮬레이션 모델을 프로그래밍 하는 과정에서 간단히 클래스를 선언해 두고 필요한 경우 메소드 문을 호출함으로써 별도의 프로그램 작성 없이 필요한 시뮬레이션 기능을 수행할 수 있게 하였다. 이 점이 바로 자바의 객체 지향적 특성으로서, 어떤 기능을 객체화 시켜 클래스로 만들어 두면 언제든지 재 사용할 수 있게 된다.

시뮬레이션 로직 프로그램의 작성은 모두 네 부분으로 나누어 객체화 하였다: 제품 및 주문의 도착으로 인한 개체 생성, 도착개체의 대기와 S/R 기계의 이동, 제품의 저장 및 불출작업, 마지막으로 이 세 부분의 통합을 위한 연결 부분이다.

<그림 3> 시뮬레이션 로직 프로그램 작성에 사용된 패키지, 클래스, 메소드

도착개체 생성 부분은 사용자가 입력한 제품의 도착분포와 주문분포에 따라 개체를 생성하는 내용이다. JSIM.PROCESS 패키지의 ENTITYMAKER와 MODEL 클래스를 상속받아 프로그램을 작성하였다. 도착개체 생성

프로그램의 일부를 <그림 4>에 정리한다.

```
public Model (String modelName, ModelBean mBean)
{
    this (modelName, mBean, 0.0);
} ; // Model
public Model (String modelName, ModelBean mBean, double slowness)
{
    super (modelName);
    trc = new Trace ("Model", modelName);
    clock = new Clock (slowness);
    trc.show ("Model", "constructor started", clock.time ());
    this.mBean = mBean;
    stopTime = 600000000.0;
    startSim = false;
    stopSim = false;
                :
public abstract class EntityMaker
{
    public final Model env;
    public EntityMaker (Model env)
    {
        this.env = env;
    }; // EntityMaker
    public abstract SimObject makeEntity ();
}; // class
                :
public class ProductMaker extends EntityMaker
{
    public ProductMaker (WebSaw env)
    {
        super (env);
    };
    public SimObject makeEntity ()
    {
        return new Product1 (WebSaw.this);
    };
};
```

<그림 4> 도착개체 생성 프로그램의 일부

도착개체의 대기와 S/R 기계의 이동 부분은 S/R 기계가 작업 중일 때 도착한 제품과 주문이 기다리고, S/R 기계가 작업을 위해 이동하는 내용이다. 20개의 저장 칸이 모두 채워져 있을 때 도착한 제품은 저장 불능으로 간주하여 시스템에서 자동 소멸되도록 처리하였다. 그러나 저장되어 있는 제품이 하나도 없을 때 도착한 주문은 제품이 이용 가능할 때까지 기다린다. 저장 및 불출작업의 대기 열은 각각 무한이며 대기 중인 개체는 작업종류를 망라하여 도착순서에 따라 S/R 기계에 의해 서비스를 받는다. S/R 기계의 이동시간은 입고장으로부터 저장 칸까지의 거리에 비례한 시간으로 정해진다. 불출작업도 저장작업과 동일한 로직이 적용된다. 작업을 마친 S/R 기계는 항상 입고장으로 돌아온다. JSIM.PROCESS 패키지의 FACILITY 클래스와 JSIM.QUEUE 패키지의 FIFO_QUEUE 클래스를 상속받아 프로그램을 작성하였다.

제품의 저장과 불출 부분은 S/R 기계가 입고장에서 가장 가까운 빈 저장 칸에 단위적재를 저장하고 가장 오래 전에 저장된 단위적재를 불출하는 내용이다. 프로그램 과정에서 S/R 기계를 서버로 간주하였다. JSIM.PROCESS 패키지의 FACILITY 클래스와 TRANSPORT 클래스, JSIM.QUEUE 패키지의 FIFO_QUEUE 클래스를 상속받아 프로그램을 작성하였다.

마지막 부분은 이상 세 부분의 객체화된 프로그램을 통합하여 한 모델이 되도록 연결하는 내용이다. 이에 따라 시뮬레이션 과정에서 개체의 도착, 이동, 저장, 불출 등의 활동이 시간의 경과에 따라 발생하면서 Web-SAW의 실행이 이루어진다. 연결 프로그램에서 사용한 주요 클래스와 메소드는 각 부분에서 사용한 프로그램 메소드를 재 호출하여 사용하였다. 1000번째 도착한 단위적재의 저장작업이 완료될 때 시뮬레이션을 종료하도록 프로그램을 작성하였다. 프로그램에서 시뮬레이션 종료조건의 변경은 간단하다.

3.4 애니메이션 프로그램

시뮬레이션이 시작됨과 동시에 사용자가 자동창고시스템에서 제품 및 주문의 도착, 운반, 저장, 불출 등의 진행상황을 화면에서 볼 수 있도록 애니메이션 프로그램을 작성하였다. 시뮬레이션 로직 프로그램과 애니메이션 프로그램의 연계가 필요하며, 이를 위해 시뮬레이션 메소드와 그래픽 메소드를 결합하였다. <그림 5>는 시뮬레이션 메소드와 그래픽 메소드를 결합하는 프로그램이다. Web-SAW의 초기화면을 둘로 나누어 상단은 애니메이션, 하단은 데이터 입력을 위한 창으로 할당하였다.

자동창고시스템의 구성요소를 화면에 배치하기 위해서 <표 2>에서와 같이 자동창고시스템의 구성요소를 시뮬레이션 클래스 함수에 맞추어 노드를 정의하였으며, 정의된 노드들의 배치를 화면상 고정된 위치에 그래픽으로 나타내기 위해 자바의 AWT 클래스에서 그래픽

처리를 할 수 있는 Campus 컴포넌트와 자바의 그래픽 메소드 중에서 Paint, Repaint, Update, Point 메소드를 사용하여 프로그램을 작성하였다. <그림 6>은 노드들의 배치를 나타내는 그래픽 프로그램이다.

```
public WebSaw (Prop[] pList, ModelBean mBean)
{
    super ("WebSaw", mBean);

    Product1 = new Source (pList [0], new Point (110, 240), this,
        new ProductMaker (this), Product1Es);

    Machine1 = new Facility (pList [1], new Point (250, 168), this,
        new FIFO_Queue (500), Machine1Es);
    Machine2 = new Facility (pList [2], new Point (760, 235), this,
        new FIFO_Queue (500), Machine2Es);

    Rack1 = new Facility (pList [3], new Point (500, 135), this,
        new FIFO_Queue (1), Rack1Es);
    Rack2 = new Facility (pList [4], new Point (500, 305), this,
        new FIFO_Queue (1), Rack2Es);

    Exit = new Sink (pList [5], new Point (900, 225), this,
        null);

}; //
```

<그림 5> 시뮬레이션 메소드와 그래픽 메소드의 결합 프로그램

<표 2> 애니메이션에 사용한 클래스 함수와 배치에서의 노드 명

자동창고구성요소	시뮬레이션 클래스 함수	배치에서의 노드 명	노드 설명
입고장	SOURCE	Product1	단위적재가 시스템에 도착하는 노드
S/R기계	SERVER	Machine1: 저장작업 S/R기계 Machine2: 불출작업 S/R기계	단위적재를 운반하는 서버 노드
저장선반	FACILITY	Rack1: 저장선반 #1 Rack2: 저장선반 #2	단위적재를 저장하는 설비 노드
출고장	SINK	Exit	단위적재가 시스템을 떠나는 노드

```
private void paintDynamic (Node n, int i)
{
    DynamicNode dn      = env.getDynNode (i);
    Point       token   = new Point ();
    int         qLength = 0;
    int         tCount  = 0;
    int         eCount  = 0;

    switch (n.nodeType) {

    case Node.SERVER:
        token.x = dn.location.x + Node.T_SERVER.x;
        token.y = dn.location.y + Node.T_SERVER.y;
        tCount  = ((Server) dn).tokenV.getNumTokens ();
        break;

    case Node.FACILITY:
        token.x = dn.location.x + Node.T_FACILITY.x;
        token.y = dn.location.y + Node.T_FACILITY.y;
        tCount  = ((Facility) dn).tokenV.getNumTokens ();
        qLength = ((Facility) dn).queueLength ();
        break;

    case Node.SOURCE:
        token.x = dn.location.x + Node.TOK_RADIUS;
        token.y = dn.location.y + 0 * Node.TOK_RADIUS;
        eCount  = ((Source) dn).getEntitiesCreated ();
        break;

    case Node.SINK:
        token.x = dn.location.x + 2 * Node.TOK_RADIUS;
        token.y = dn.location.y + 4 * Node.TOK_RADIUS;
        eCount  = ((Sink) dn).getEntitiesConsumed ();
        break;
```

<그림 6> 노드들의 배치 그래픽 프로그램

애니메이션에서 개체의 흐름을 쉽게 나타내기 위해 화면에 입고장과 출고장을 분리 배치하였으며, 저장 및 불출작업에 각각 한 대씩의 전용 가상 S/R 기계를 할당하였다. 두 저장선반의 저장 칸은 각각 일렬로 10개의 청색 굵은 점으로 표현하였다. 그리고 시스템 내의 제품과 주문 개체는 각각 적색과 흑색 굵은 점으로 표현하였으며, 단위적재가 저장 칸에 저장되어 있는 경우 해당 저장 칸 점 앞에 제품 개체 점을 위치시켰다. <그림 9>를 참고하시오.

시뮬레이션 진행과정에서 노드간 단위적재의 이동경로를 표현하기 위해 JSIM. PROCESS 클래스의 TRANSPORT 메소드를 사용하였다. TRANSPORT 메소드는 Transport(x, y, x, y, x, y)로 표현되는데, 첫 번째 좌표 x와 y는 이동의 시작점, 두 번째 x와 y는 중간점, 마지막 x와 y는 끝점을 나타낸다. <그림 7>은 노드를 연결하여 이동경로를 그려 주는 그래픽 프로그램이다.

```
Transport    Product1EO = new Transport (150, 380, 200, 280, 250, 260);
Transport [] Product1Es = {Product1EO};

Transport    Machine1EO = new Transport (340, 350, 450, 190, 500, 160);
Transport    Machine1E1 = new Transport (340, 430, 450, 310, 500, 280);
Transport [] Machine1Es = {Machine1EO, Machine1E1};

Transport    Machine2EO = new Transport (640, 250, 800, 235, 820, 255);
Transport [] Machine2Es = {Machine2EO};

Transport    Rack1EO = new Transport (610, 160, 680, 192, 730, 260);
Transport [] Rack1Es = {Rack1EO};
Transport    Rack2EO = new Transport (610, 330, 680, 313, 730, 260);
Transport [] Rack2Es = {Rack2EO};
```

<그림 7> 이동경로 그래픽 프로그램

3.5 시뮬레이션 출력 프로그램

시뮬레이션 출력은 진행과정 추적 보고서(Trace Report), 실행결과 통계표(Statistical Results Window), 요약 보고서(Summary Report)로 구성된다. 시뮬레이션의 추적은 시뮬레이션 프로그램의 여러 위치에 TRACE 메소드를 삽입함으로써 실현된다. 그 결과, 사용자는 시뮬레이션 과정에서의 모든 진행상황을 보고서 형태로 도스화면에서 볼 수 있다.

시뮬레이션 로직 프로그램에 MATH 클래스를 이용한 통계자료 수집 및 계산 프로그램을 작성함으로써, 시뮬레이션이 종료된 후에 저장 및 불출작업 대기시간과 대기수의 평균, 표준편차, 최대치, 최소치, 작업수행횟수, 작업불능횟수, S/R 기계와 저장선반의 평균 이용률 등에 대한 다양한 정보를 통계표 형태로 보여주도록 하였다. S/R 기계의 이용률은 S/R 기계가 한 작업을 마칠 때마다 그리고 저장선반의 이용률은 저장 또는 불출이 발생할 때마다 각각 계산하여 시뮬레이션의 종료 시에 평균치를 구하도록 하였다. 실행결과 통계표는 많은 정보를 하나의 표에 모두 포함하기 때문에 사용자가 이해하기 쉽지 않다. 예로서, <그림 8>은 저장선반과 S/R 기계의 이용률 계산을 위한 프로그램의 일부이다.

요약 보고서는 실행결과 통계표로부터 정보를 발췌하여 요약한 보고서이다. 이를 위해 서버의 데이터베이스를 이용하여 통계표로부터 자료를 추출, 처리한 결과를 양식에 맞추어 출력하는 프로그램을 작성하였다.

```
    public double confidence (double level)
{
    double df = nobs - 1.0;      // degrees of freedom
    if (df <= 0.0) {
        trc.tell ("confidence", "must have at least 2 observations");
        return 0.0;
    };
    double p = (1.0 - level) / 2.0;          // e.g., .95 -> .025
    double t = InverseT.tValue (p, df);
    trc.show ("confidence", "p = " + p + " df = " + df + " t = " + t);
    if (t == MAX_DOUBLE) {
        trc.tell ("confidence", "essentially infinite interval");
        return t;
    }; // if
    return (t * stdDev ()) / Math.sqrt (df);
}; // confidence
public double confidence ()
{
    return confidence (CONF_LEVEL);
};
public double precision (double level)
{
    return (nobs > 1) ? confidence (level) / mean () : 0.0;
}; // precision
public double precision ()
{
    return precision (CONF_LEVEL);
}; // precision
```

<그림 8> 저장선반과 S/R 기계의 이용률 계산 프로그램 일부

4. Web-SAW의 실행

4.1 실행 환경

Web-SAW는 플랫폼의 독립성을 가지고 있어 운영체제나 하드웨어의 사용환경에 제약을 받지 않고 사용할 수 있다. 이것은 자바로 작성된 프로그램은 자바가상머신(Java Virtual Machine)이 해당 플랫폼에 설치되면 하부 플랫폼과는 상관없이 동일한 실행환경을 제공받기 때문이다. 따라서 Web-SAW는 사용자의 컴퓨터에 자바가상머신과 웹 브라우저가 설치되어 인터넷이 가능한 환경만 구축되어 있으면 언제 어디서든 실행할 수 있게 된다.

현재, Web-SAW는 인터넷에 연결하지 않은 상태에서 저자들의 PC에서 실행된다. 이것은 저자들의 컴퓨터에 시뮬레이션 기능의 클래스와 메소드가 포함된 컴퓨터 환경변수와 클래스 패스를 설정해 두었기 때문에 가능하다. 물론 다른 컴퓨터에도 환경변수와 클래스

패스를 설정해 두면 실행될 수 있다. 하지만 인터넷을 통하여 누구나 Web-SAW를 이용하기 위해서는 웹상에서 Web-SAW 프로그램을 실행할 수 서버를 구축하여야 한다. 서버를 위해서는 표준 SQL(Structure Query Language)을 지원하는 자바의 JDBC(Java Data Base Connectivity) 클래스를 사용하여 DBMS(Data Base Management System)과 분산환경 시스템을 구축하여야 한다.

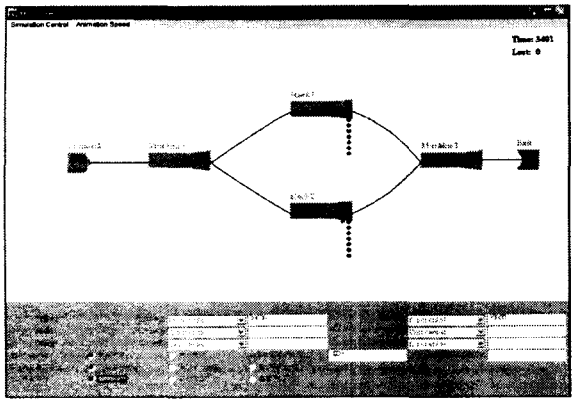
표준 SQL을 이용하면 어떤 종류(벤더)의 DBMS와도 간단히 연결이 이루어져 자료의 검색, 저장, 관리 등을 할 수 있게 된다. JDBC 클래스를 사용하기 위해서는 자바와 SQL에 대한 이해가 모두 필요하다. 자바에서는 분산환경 시스템을 구축할 수 있도록 RMI(Remote Method Invecation) 기능을 제공한다. 분산환경 시스템이란 네트워크로 연결된 여러 대의 컴퓨터에 하나의 애플리케이션이 실질적으로 분산 실행되면서, 논리적으로는 마치 하나의 프로그램처럼 움직이는 시스템을 의미한다. 서버구축 작업에서 RMI 관련 클래스만으로 프로그램을 작성함으로써 원격지 객체의 메소드를 마치 같은 자바가상머신 내 객체의 메소드 처럼 호출할 수 있게 된다. 자바는 RMI 기능을 위해 Java.rmi 패키지를 지원하다. 그러나 이 패키지의 매뉴얼만 가지고는 프로그램을 작성하기가 어려운데, 이것은 로컬과 원격 객체를 동시에 만들어야 하고, 일반 프로그램 작성과 달리 설정해 주어야 하는 것이 매우 많기 때문이다(이현우 외, 2000).

본 연구에서 Web-SAW의 서버를 구축하지 못한 이유는 DBMS와 분산환경시스템을 구축하는 일이 고도의 전문적인 지식을 필요로 하기 때문이다. 만일 앞으로 전문가의 도움으로 DBMS와 분산환경시스템을 갖춘 서버를 구축하게 된다면 Web-SAW 프로그램을 서버에 올려놓아 사용자가 인터넷을 통해 서버에 로그인해서 언제 어디서나 프로그램을 실행할 수 있게 될 것이다.

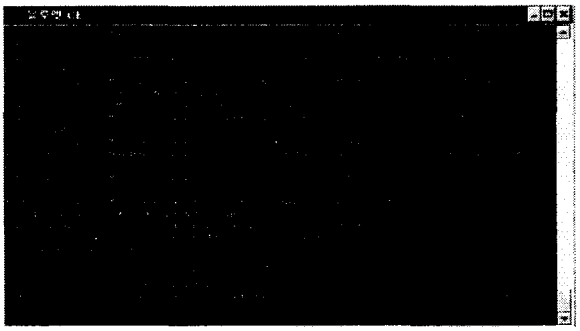
4.2 실행 과정

Web-SAW의 초기화면은 주 메뉴, 자동창고의 배치그림을 배경으로 하는 애니메이션 창, 사용자의 입력 창으로 구성되어 있다. <그림 9>는 Web-SAW의 사용자 입력과 시뮬레이션 실행 시 애니메이션 순간그림(snapshot)을 보여준다. Web-SAW를 실행하기 위해서 사용자는 맨 먼저 시뮬레이션 수행에 필요한 데이터를 입력하여야 한다. 이 때 3.2절에서 기술한 데로 허용된 항목에 대해서만 입력을 하여야 한다. 자바의 쓰레드 기능에 의해 시간은 0.001초를 기본단위로 한다. 제품 1의 도착 및 주문 시간분포로서 모두 지수분포를 선택하고, 분포의 평균값으로 각각 30000을 입력한다. S/R 기계의 수평이동시간으로 3000을 입력한다. 그리고 세 가지 운영정책으로는 각각 단일명령, 임의저장방법, 선입선출을 선택한다.

모든 입력이 끝나면 화면 최 상단에 위치한 주 메뉴를 선택하여 애니메이션 속도를 조정한다. 애니메이션 속도의 초기값(default)은 'Speed Up'으로 설정되어 있으며, 애니메이션 속도를 낮추고 싶으면 'Animation Speed' 메뉴에서 'Speed Down'을 클릭하면 된다. 애니메이션 속도를 더욱 빠르게 하고 싶으면 'Speed Up'을 반복 클릭하면 된다. 애니메이션 속도는 시뮬레이션 진행 중에도 조절할 수 있다. 'Simulation Control' 메뉴에서 'Simulation Start'를 클릭하면 시뮬레이션이 시작된다. 시뮬레이션 도중에 시뮬레이션을 멈추고 싶으면 'Simulation Stop'을 클릭하면 된다. 'Toggle Trace'를 클릭하면 시뮬레이션 종료 후 <그림 10>과 같은 시뮬레이션 추적 보고서가 출력된다.



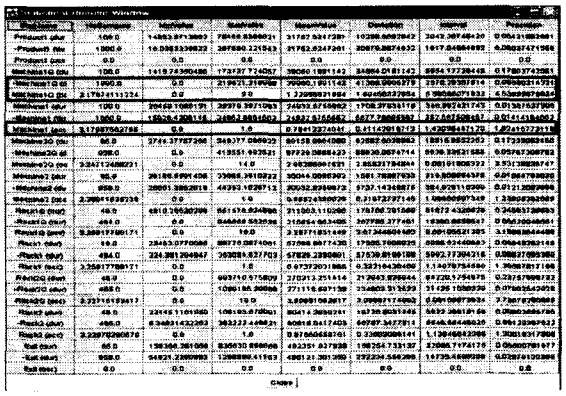
<그림 9> Web-SAW의 초기화면에서 사용자 입력과 애니메이션 순간그림



<그림 10> Web-SAW의 추적 보고서

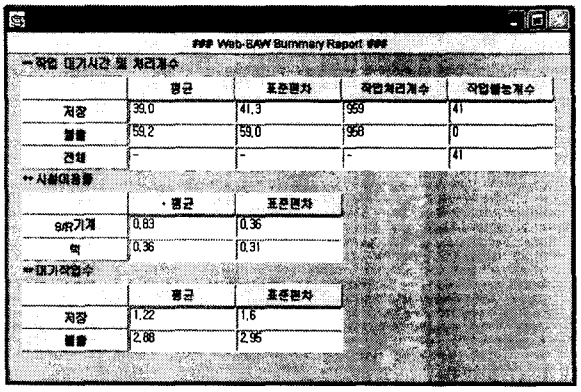
시뮬레이션이 진행되면 애니메이션 창에 자동창고시스템에서의 작업진행 상황이 실시간 동화상으로 전개된다. Product1 노드에서 생성된 단위적재는 Machine1 서버를 거쳐 Rack1 또는 Rack2 설비에 저장되고, 주문지시가 발생하면 Rack1 또는 Rack2에 저장되어 있는 단위적재는 불출되어 Machine2 서버를 거쳐 Exit 노드로 이동하여 시스템에서 소멸된다. <그림 9>는 시뮬레이션 시간 3401 때의 시스템 상황을 보여주는 순간그림이다. 순간그림을 통해 지금까지 총 129개의 단위적재가 도착하였고, 총 120개의 단위적재가 불출되었음을 알 수 있다. 그리고 현재 2개의 저장작업과 1개의 불출작업이 대기 중이며, 저장선반 1번과 2번에 각각 3개와 4개의 단위적재가 저장되어 있음을 알 수 있다.

시뮬레이션이 종료되면 곧 바로 <그림 11>과 같은 실행결과 통계표가 화면에 출력된다. 이 통계표는 전문적인 해석을 필요로 하여 일반 사용자가 이해하기 매우 어렵다. 예로서, 진한 선으로 그려진 첫 번째 박스에서 -Machine1Q(d)의 MinValue는 저장작업 최소대기시간, MaxValue는 저장작업 최대대기시간, MeanValue는 저장작업 평균대기시간, 그리고 Machine1Q(oc)의 MeanValue는 저장작업 평균대기수를 의미한다. 두 번째 박스에서 Machine1(occ)의 MeanValue는 S/R 기계의 저장작업 평균이용률을 나타낸다.



<그림 11> Web-SAW의 실행결과 통계표

실행결과 통계표 화면에 이어 <그림 12>와 같은 요약 보고서가 출력된다. 요약 보고서는 실행결과 통계표를 사용자가 이해하기 쉽게 요약 정리한 시뮬레이션 결과 보고서이다. 요약 보고서로부터, 시뮬레이션 동안 총 1000개의 단위적재가 시스템에 도착하여 이중 959개는 저장선반에 저장되고 나머지 41개는 도착 시점에 빈 저장 칸이 없어 시스템 밖으로 보내진 것을 알 수 있다. 그리고 시뮬레이션 종료시점에 저장선반에는 1개의 단위적재가 저장되어 있음을 알 수 있다. 각종 척도의 표준편차가 큰 이유는 짧은 시뮬레이션 시간과 시스템 변동성 때문인 것으로 판단된다.



<그림 12> Web-SAW의 요약 보고서

5. 결론 및 향후과제

웹 환경에서 시뮬레이션을 수행하기 위해서는 범용 웹 기반 시뮬레이션 언어로 시스템의 시뮬레이션 모델을 구축하여야 하는데, 이를 위해서는 사용자가 자바와 같은 웹 프로그래밍 언어와 시뮬레이션에 관한 전문적인 지식을 필요로 한다. 이러한 이유로 웹 기반 시뮬레이션은 많은 기대효과에도 불구하고 그 사용과 보급이 매우 미진한 현실이다.

본 논문에서는 Web-SAW 이름의 실험용 웹 기반 자동창고시스템 전용 시뮬레이터의 개발을 소개하였다. Web-SAW는 일반 사용자가 시뮬레이션과 웹 프로그래밍 언어에 대한 전문적인 지식 없이 웹상에서 GUI 기능을 통해 자동창고의 운영에 대한 간단한 입력만으로 자동창고시스템의 시뮬레이션을 손 쉽게 수행할 수 있게 해 준다. Web-SAW 구축을 위해 기존의 웹 기반 시뮬레이션 언어들로부터 클래스 라이브러리 함수와 구현환경 등을 지원 받으면서, 사용자 인터페이스 프로그램, 시뮬레이션 로직 프로그램, 애니메이션 프로그램은 자바로 직접 작성하였다.

Web-SAW는 플랫폼의 독립성으로 하드웨어나 운영체계에 상관없이 실행된다. 비록 Web-SAW가 지금은 저자들의 PC에서 실행되도록 설치되어 있지만, DBMS과 분산환경 시스템을 갖춘 서버의 구축이 완료되면 사용자들이 인터넷을 통하여 시간과 장소에 제약없이 이용할 수 있게 될 것이다. Web-SAW의 개발과정에서 저자들이 개발한 자동창고 시뮬레이션 로직프로그램은 유사한 시스템의 시뮬레이션 모델을 구축할 때 객체 클래스를 상속받는 것만으로 쉽게 재 사용할 수 있다.

Web-SAW는 실험용 웹 기반 시뮬레이터로서, 개발과정에서 프로그램 작성능력의 한계로 처음 계획했던 모델에 대한 단순화와 생략화가 불가피하였다. 향후 과제로서 제품종류, 제품 및 주문의 도착시간 분포, 저장선반 구조와 수, S/R 기계 이동능력과 대수, 시스템과 S/R 기계 운영정책 등에 대한 입력의 확장과 다양성을 통한 시뮬레이터의 유연성 증대가 필수적이다. 이를 위해서는 좀 더 전문적이고 높은 수준의 웹 기반 시뮬레이션 프로그램 작성 능력이 요구된다. 또한 자동창고의 시뮬레이션을 보다 정확히 수행하기 위해서는 S/R 기계의 고장, 품질검사, 재고나 저장공간이 부족한 경우의 처리 등의 여러 현실적 사항들이 모델에 반영되어야 한다.

앞으로, 지금의 Web-SAW 설계와 프로그램을 토대로 보다 복잡한 자동창고시스템에서 여러 실제적 요소가 추가된 웹 기반 전용 시뮬레이터가 완성되면, 기업에서는 이를 경영정보시스템과 연결하여 자동창고시스템의 설계 및 운영을 위한 의사결정도구로서 귀중하게 사용할 수 있을 것이다.

참고문헌

[1] 김윤명, *XML을 위한 JAVA Programing*, 가남사, 2002.

[2] 이현우, 김형국, 홍성미, *Java Programing Bible*, 영진출판사, 2000.

[3] Bray, T., Paoli, J., and Sperberg-McQueen, C. M., *Extensible Markup Language(XML) 1.0 Specification*, http://www.w3.0 rg/tr/rec-xml, 2002.

[4] Groover, M. P., *Automation, Production Systems, and Computer Integrated Manufacturing*, 2nd Ed., Prentice Hall, 336-342, 2001.

[5] Herly, K. J. and Kilgore, R. A., SILK: A Java Based Process Simulation Language, *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 475-482, 1997.

[6] Howell, F. and McNab R., SIMJAVA: A Discrete Event Simulation Library for Java, *Proceedings of the 1998 International Conference on Web-based Modelling and Simulation*, Society for Computer Simulation International, San Diego CA, 51-56, 1998.

[7] Iazeolla, G., and D'Ambrogio, A., A Web-based Environment for the Reuse of Simulation Models, *Proceedings of the 1998 International Conference on Web-based Modeling & Simulation*, San Diego, CA, 37-42, 1998.

[8] Kilgore, R. A. and Burke, E., Object-oriented Simulation of Distributed Systems Using Java and Silk, *Proceedings of the 2000 Winter Simulation Conference*, Orlando, FL, 1802-1909, 2000.

[9] Kilgore, R. A., Healy, K. J., and Kleindorfer, G. B., The Future of Java-based Simulation, *Proceedings of the 1998 Winter Simulation Conference*, Washington D. C., 1707-1712, 1998.

[10] Klein, U., Strabburger, S., and Beikirch, J., Distributed Simulation with JavaGPSS Based on the High Level Architecture, *Proceedings of the 1998 International Conference on Web-based Modeling and Simulation*, The Society for Computer Simulation International, San Diege, CA, 85-90, 1998.

[11] Little, M. C., *JAVASIM Home page*, http://javasim,ncl,ac.uk/, 2002.

[12] Miller, J. A., JSIM: A Java-based Simulation and Animation Environment, *Prodeedings of the 30th Annual Simulation Symposium*, Atlanta, Georgia, 31-42, 1997.

[13] Miller, J. A., Ge, Y., and Tao, J., Component-based Simulation Environments: JSIM As a Case Study Using Javabeans, *Proceedings of the 1998 Winter Simulation Conference*, Washington D. C., 373-381, 1998.

주 작 성 자 : 임 대 진
논문투고일 : 2003. 07. 24
논문심사일 : 2003. 08. 04(1차), 2003. 08. 21(2차), 2003. 10. 24(3차), 2003. 12. 09(4차)
심사판정일 : 2003. 12. 15

저자소개

임대진

경희대학교 산업공학과 학사
경희대학교 테크노공학대학 산업공학과 석사
현재: BNB(주) BI 사업부/ 선임컨설턴트
관심분야: SCM, ASP, 컴퓨터시뮬레이션

박양병
한양대학교 산업공학과 학사
Pennsylvania State University 산업공학과 석사
Oklahoma State University 산업공학과 박사
Northeastern University 산업 및 정보공학과 교수
VPI, UBC 객원교수
현재: 경희대학교 테크노공학대학 산업공학과 교수
관심분야: SCM, 물류/생산관리, 컴퓨터시뮬레이션