# 웹 기반 시뮬레이션을 위한 시각적 모델 개발 환경*

# A Visual Modeling Environment for Web-based Simulation

김기형**, 남영환**,
Kihyung Kim, Yeonghwan Nam

**Abstract**

The Web-based simulation was introduced for conducting simulation experiments in the Internet and the Web. Due to the use of the Java language, the Web-based simulation can have such characteristics as reusability, portability, and the capability of execution on the Web. Most of existing Web-based simulation tools have focused mainly on the development of the runtime simulation libraries and mechanisms on the Web. Thus, the model development work in such Web-based simulation tools still requires hand-written coding of model developers. This paper presents a visual model development environment for the Web-based simulation. The proposed environment provides a framework for model development and animation. To show the effectiveness of the proposed environment, we perform simulation experiments for transaction routing algorithms in a distributed transaction processing system.

* 본 논문은 1998년도 정보통신부 초고속정보통신 응용기술개발사업에 의하여 연구되었음.

** 영남대학교 컴퓨터공학과

## 1. 서론

기존의 이산 사건 시뮬레이션의 개발을 위한 환경은 언어나 라이브러리에 종속적인 경우가 대부분이었다. SIMSCRIPT II[1], MODSIM III[2], CSIM[3] 등과 같이 특정 언어와 라이브러리에 종속된 시뮬레이션의 개발과 수행의 경우에 시뮬레이션 모델은 이기종 환경에서 새로 컴파일 되어야 하기 때문에 개발된 시뮬레이션의 구성 요소들은 이기종 환경에서의 수행을 기대하기 어렵다. 또한 인터넷과 웹을 통한 실행 능력을 제공할 수 없다.

웹 기반 시뮬레이션은 인터넷과 웹의 환경에서 시뮬레이션을 수행하고자 하는 목적에 의해 제안되었다. 즉 자바[4]를 기반으로 모델을 개발하므로서 모델의 재사용성과 이기종 환경에서의 수행 능력을 제공하고 웹에서의 실행이 가능하도록 할 수 있다. 또한 자바의 그래픽라이브러리를 사용함으로써 웹 환경을 통하여 애니메이션 기능을 제공받을 수 있다.

최근까지 웹 기반 시뮬레이션에 관한 연구는 주로 Simjava[5], JavaSim[6], Silk[7], Multi -Verse[8] 등과 같은 시뮬레이션 라이브러리의 구현과 HLA[9], JavaRMI[10], CORBA[11] 등을 이용한 분산 시뮬레이션 환경의 구현에 초점이 맞추어져 왔으며, 모델 개발에 관한 연구는 발표되지 않았다. 즉, 모델 개발자는 시뮬레이션 언어에 대해 능숙해야 하며, 이는 새로운 모델을 개발하는 경우에 개발자의 모델링 수행을 저해하는 요인이 된다. 따라서 새로운 모델 개발 환경에 대한 필요성이 대두되었으며, OPNET[12], Digital Workshop[13] 등의 시뮬레이션 도구는 시각적인 개발 환경을 제공해 이러한 문제를 해결하고 있다. 그러나 OPNET은 웹 기반 시뮬레이션 능력을 제공하지 못하며, Digital Workshop의 경우에는 웹 기반의 도구이지만 디지털 회로 설계에만 사용될 수 있다.

본 논문에서는 위와 같은 기존 웹 기반 시뮬레이션 모델 개발의 문제점을 해결하기 위해 시각적 모델 개발 환경, VMAC(Visual Model and Applet Composer)을 제안한다. VMAC은 시각적 모델 작성 도구를 제공해 개발자의 모델 개발에 대한 부담을 줄여준다. VMAC은 라이브러리를 이용해 모델 코드를 생성해주며, 개발자는 자동으로 생성되는 클래스에 자신의 시뮬레이션 코드를 삽입해 목적한 모델을 간단하게 구성할 수 있다. 개발된 클래스 코드는 자바 애플릿 형태의 코드와 통합되어 웹 환경에서의 실행이 가능한 형태로 변환된다.

VMAC을 이용한 효과적인 시뮬레이션 수행 방법을 설명하기 위해 분산 트랜잭션 처리 시스템에서의 부하 평준화(load balancing) 기법들에 대한 시뮬레이션을 수행한다. 부하 평준화를 위해 분산된 각각의 노드에 부하를 균등하게 분산하는 것을 부하 분산이라 하며, 효과적인 부하 분산을 실현하기 위해 트랜잭션의 처리를 요청할 적절한 노드를 결정하고 분배하는 기능을 수행하는 것을 트랜잭션 분배(transaction routing)라 한다. 부하 평준화를 위한 기존의 트랜잭션 분배 기법을 설명하고, VMAC을 이용한 모델 개발과 이를 이용한 시뮬레이션 실행에 대해 설명한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 웹 기반 시뮬레이션에 대해 설명하며, 3장에서는 기존 환경의 문제점을 극복하기 위해 제안되는 VMAC 을 설명하고, 4장에서는 VMAC을 이용한 시뮬레이션 수행 방법을 트랜잭션 분배의 예를 통해 설명한다. 끝으로 5장에서 본 논문의 결론을 맺는다.

## 2. 웹 기반 시뮬레이션

웹 기반 시뮬레이션은 인터넷과 웹 환경에서의 시뮬레이션 수행을 위해서 연구되어 왔으며, 주로 시뮬레이션 라이브러리 개발에 대한 연구가 진행되어 왔다.

웹 기반 시뮬레이션 라이브러리는 기존의 시뮬레이션 라이브러리를 자바 기반으로 재작성해 웹과 자바가 지닌 특징을 이용하도록 한다. 특히 기존 시뮬레이션 라이브러리가 지원하지 못했던 애니메이션 기능을 제공할 수 있다. Simjava[5]는 시뮬레이션 지원 기능 외에 애니메이션과 분석 기능을 클래스 라이브러리 형태로 제공한다. 이는 웹의 특징을 이용해 기존의 시뮬레이션 환경에 비해 적은 노력으로 시뮬레이션 과정을 애니메이션 가능하게 한다. Silk

의 경우에는 시뮬레이션 애니메이션을 위해서 JavaBeans[14] 형태의 컴포넌트를 제공한다. 시뮬레이션 모델을 구성할 때 개발자는 원하는 컴포넌트를 모델에 삽입해 애니메이션 기능을 제공받을 수 있게 구현되었다. 또한 MultiVerse[8]는 시뮬레이션 수행 단계에서 사용자의 요구를 반영하기 위해 애니메이션 형태를 조절할 수 있는 시뮬레이션 수행 환경을 제공한다.

모델 개발의 관점에서 웹 기반 시뮬레이션 라이브러리는 기존의 시뮬레이션 툴들에 비해 장점을 제공하지 못한다. 즉 CASE툴의 도움이 없이 개발자의 코딩에 의존하기 때문에 프로그래밍 전문가들만이 사용하게 된다. Digital Workshop[13]의 경우에는 사용자의 모델 개발 노력을 줄여주기 위해 시각적 모델 개발 도구가 제공된다. 그러나 이는 디지털 회로 설계를 위한 전용 도구이며 일반적인 시뮬레이션 응용에 사용할 수 없다. 따라서 다양한 분야에 적용 가능한 시뮬레이션 모델 개발 도구의 개발에 대한 필요성이 대두되었다.

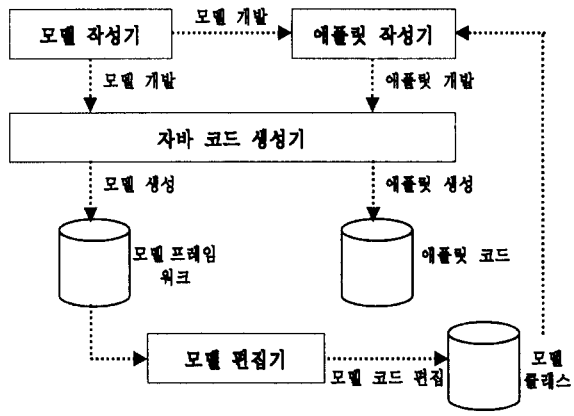
## 3. 시각적 모델 개발 환경

효율적인 웹 기반 시뮬레이션 수행을 위해 요구되는 요소들은 다음과 같다.

1) 모델 개발자는 언어나 라이브러리에 대해 능숙하지 않더라도 원하는 형태의 모델을 간단하게 작성할 수 있어야 한다.
2) 개발된 모델을 이용한 시뮬레이션의 수행이 간편해야 한다.
3) 인터넷과 웹 환경에서 시뮬레이션 수행이 가능해야 한다.

이러한 요구 조건하에서 시뮬레이션이나 개발환경에 능숙하지 않은 사용자들이 간단하게 원하는 시뮬레이션 모델을 개발하고 웹을 통해 실행할 수 있도록 하기 위해 시각적 모델 개발환경인 VMAC을 제안한다.

VMAC은 <그림 1>과 같이 크게 네 부분, 즉, 시각적 모델 작성기, 모델 편집기, 애플릿 작성기, 그리고 코드 생성기로 구성된다. 모델 작성기는 모델 엔




<그림 1> VMAC의 전체 구조

터티들의 연결 구조를 시각적으로 개발하는 GUI에 기반한 모델개발 환경이다. 모델 엔터티 각각의 행동(behavior)은 모델 편집기를 통해서 기술하게 된다. 애플릿 작성기는 모델작성기에서 개발된 모델을 웹브라우저나 애플릿뷰어에서 에니메이션하기 위한 사용자인터페이스를 작성하는 환경이다. 모델 및 애플릿작성기에서 개발한 모델 정보는 코드 생성기를 통해서 자바 코드로 변환된다. 이는 컴파일을 통해 바이트 코드로 변환되고 웹 브라우저를 이용해 바로 실행할 수 있다.

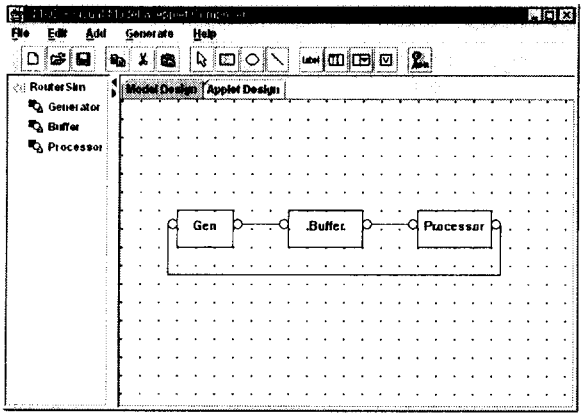
### 3.1 모델 작성기

VMAC은 개발자의 시뮬레이션 모델 구성을 위해 엔터티(entity), 포트(port), 포트간의 연결(link)을 시각적으로 표현할 수 있도록 해준다. 엔터티는 모델을 표현하는 기본 요소로서 독립적인 행위와 변수로 구성되며, 포트는 엔터티간의 사건(event) 정보 교환에 이용된다. 또한 포트들은 목표한 엔터티에 정확한 사건 정보 전달을 위해 개별적으로 연결된다. <그림 2>는 VMAC의 모델 작성기를 이용해 프로세서-버퍼 모델을 표현한 예를 나타낸다.

작성된 모델을 이용해 시뮬레이션 코드를 생성하기 위해 Simjava 시뮬레이션 라이브러리를 사용한다. Simjava는 시뮬레이션 수행에 필요한 클래스들을 제공한다. 따라서 Simjava 라이브러리를 이용하면 시뮬레이션에 필요한 기능들을 직접적인 구현

없이 비교적 간단하게 제공받을 수 있다. Simjava가 제공하는 시뮬레이션 지원 기능에는 메시지 교환을 이용한 이벤트의 전달과 시뮬레이션 시간 관리 기능, 일정한 분포에 따른 임의값(random value) 생성 기능 등이 있다. 다음은 Simjava가 제공하는 시뮬레이션 지원 기능들이다.

- sim_schedule(): 포트를 통해 다른 요소에게 사건을 전달함
- sim_hold(): 시뮬레이션 시간을 잠시 멈춤
- sim_wait(): 사건 객체가 도착할 때까지 대기
- sim_select(): 큐로부터 사건들을 선택함
- sim_trace(): 시간 정보가 포함된 메시지를 추적 파일에 기록



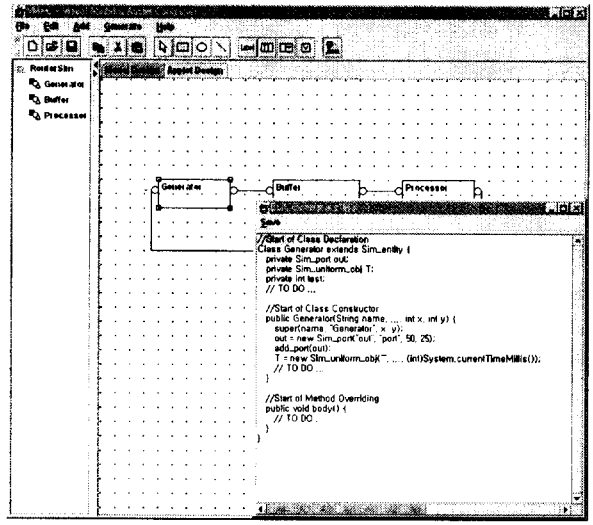

<그림 2> VMAC을 이용한 모델 작성

개별적인 개체의 결합을 위해서 Simjava는 sim_schedule()를 이용하며 포트를 통해 서로간의 사건의 교환한다. 전송된 사건들은 큐에 대기하게 되며, 필요에 따라 sim_select()나 sim_wait()를 사용해 수신측의 개체에 전달된다. 이때 sim_select()는 이미 도착한 사건들을 선택하는데 사용되며, sim_wait()는 도달될 사건을 기다리게 된다. 또한 Simjava는 메시지의 정확한 순서를 보장하기 위해 모든 사건을 시간 순서에 의해 전역적으로 순서화한다.

### 3.2 모델 편집기

모델 편집기는 모델 작성기에서 개발된 모델 엔터티 각각에 모델의 행동(behavior)를 기술한 시뮬레이션 코드를 추가해 모델에 대한 완전한 클래스 생성을 가능하게 한다.

모델 편집을 수행하기 위해 모델 개발자는 모델 작성기에서 하나의 엔터티를 선택하게 된다. 선택된 엔터티는 편집 가능한 상태가 되며, 이때 저장되어 있던 모델 프레임워크가 모델 편집기에 나타나게 된



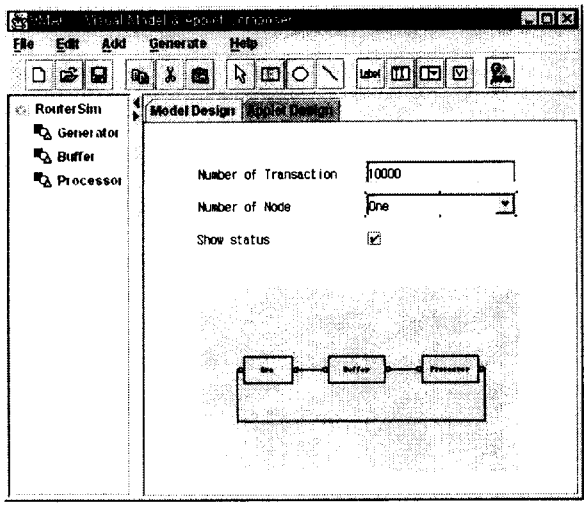

<그림 3> VMAC을 이용한 모델 편집

다. 개발자는 Simjava 라이브러리를 이용해 시뮬레이션 코드를 삽입하거나 수정할 수 있으며, 편집이 종료되면 독립적인 클래스로 저장된다. <그림 3>은 모델 편집기를 이용한 코드 추가 작업을 보여준다.

### 3.3 애플릿 작성기

VMAC은 개발된 모델을 이용한 웹 기반의 시뮬레이션 수행을 위해 자바 애플릿을 이용한다. 애플릿 작성기는 시뮬레이션 실행과 인터페이스에 대한 추가적인 코드를 개발하기 위한 노력을 최소화하기 위해서 애플릿 클래스를 자동으로 생성한다. 모델 작성기에서 개발된 모델의 연결구조는 애플릿 작성

창에 자동적으로 표시되며 개발자는 시뮬레이션 실행 시 웹 브라우저를 통해 사용자의 입력을 반영할 수 있는 인터페이스만을 배치하면 된다. 이때 사용할 수 있는 인터페이스 형태로는 텍스트 입력 상자(Text Field), 콤보 상자(Combo Box), 체크 상자(Check Box)가 제공된다. <그림 4>는 프로세서-버퍼 모델의 실행을 위한 애플릿 작성을 나타낸다. 모델 클래스와 애플릿 클래스를 분리함으로써 사용자의 목적에 맞는 다양한 인터페이스를 손쉽게 제공할 수 있다. 즉 애플릿 클래스만을 변화시키면 다양한 실행형태를 제공할 수 있다. 실제 이런 구조는 MultiVerse의 모델 계층과 인터페이스 계층의 분리에서도 사용되었다.




<그림 4> VMAC을 이용한 애플릿 작성

애플릿을 실행할 때의 시뮬레이션 수행 과정을 보다 효과적으로 시각화하기 위해 VMAC은 Simjava의 애니메이션 클래스를 사용한다. Simjava 애니메이션 클래스는 각 모델을 표현하며 이들간의 연결과 이벤트 전달 등을 시각화한다. 또한 추적 파일(trace file)을 이용해 모델의 변화를 감시한다.

### 3.4 코드 생성기

코드 생성기는 개발된 모델에 대한 자바 애플릿 코드 생성한다. <표 1>은 자바 코드 생성을 위해

<표 1> 모델과 애플릿 정보 테이블 구조

| 모델 정보 테이블 | | |
|---|---|---|
| 개체 이름 | | Entity_name[] |
| 개체 위치, 크기 | 시작점 좌표 | Entity_X1_Y1[] |
| | 끝점 좌표 | Entity_X2_Y2[] |
| 포트 정보 | 포트 이름 | Port_name[] |
| | 포트 좌표 | Port_X_Y[] |
| 감시 속성 정보 | 속성 이름 | Trace_name[] |
| | 추적 타입 | Trace_type[] |
| 랜덤 변수 | 변수 이름 | Random_name[] |
| | 분포 타입 | Random_type[] |
| 애플릿 정보 테이블 | | |
| 개체 연결 정보 | | Con_Entity[] |
| 포트 연결 정보 | | Con_Port[] |
| 변수 정보 | 변수 이름 | Var_name[] |
| | 변수 타입 | Var_type[] |
| | 컨트롤 타입 | Control_type[] |
| | 초기값 | Control_def[] |

관리하는 정보 테이블 구조를 보여준다. 코드 생성기는 이들 테이블 정보 파일을 읽어 들여 개체 정보와 애플릿 정보를 구별하며 기술된 각각의 속성에 대해 코드를 생성한다.

생성된 자바코드에는 모델 엔터티들간의 연결구조와 애플릿 실행코드가 포함되는데 이 코드는 다시 코드 편집기로 입력되어 모델의 행동에 대한 기술을 하는데 사용된다.

## 4. 구현

VMAC은 자바를 이용해 구현되었는데 이기종 환경에서의 자유로운 시뮬레이션 모델 개발과 코드 생성이 가능하기 때문이다. VMAC은 자바 애플리케이션의 형태를 가지는데, 이는 자바 애플릿이 보안상의 이유로 지역 디스크에 쓰기를 할 수 없기 때

문이다. 모델과 애플릿 작성 인터페이스를 구현하기 위해서는 자바 스윙(swing)[15] 패키지를 사용했다.

애플릿 코드 생성을 위해 모델 정보와 애플릿 정보에 대한 클래스를 구현하였다. 또한 이들 정보를 효과적으로 관리하기 위해 전체 모델과 애플릿에 대한 클래스 정보는 벡터화(vectorization)된다. 다음은 모델과 애플릿에 대한 정보를 관리하기 위해 구현된 클래스들과 이들이 관리하는 정보를 나타낸다.

- Entity 클래스: 엔터티 이름, 위치, 크기 정보
- Port 클래스: 포트 이름, 위치 정보
- RdNumber클래스: 임의값(random value) 이름, 분포 정보
- Trace클래스: 감시 변수 이름, 타입 정보
- Ent_Connect 클래스: 엔터티와 포트 연결 정보
- Ent_Var 클래스: 사용자 입력 변수 이름, 타입, 컨트롤 타입, 초기값 정보

## 5. 모델개발과 수행

본 장에서는 구현된 개발 도구를 이용해 시뮬레이션 모델을 개발하고 이를 애플릿으로 변환해 실행하는 방법을 설명한다. 시뮬레이션 대상은 트랜잭션 라우터이며, 이를 위해 트랜잭션 라우터를 설계하고 이를 VMAC을 이용해 자바 코드가 생성되도록 한 후 실행한다.

### 5.1 다중 트랜잭션 처리 시스템

트랜잭션 분배(transaction routing)란 다중 트랜잭션 처리 시스템에서 사용자로부터 요청되는 트랜잭션 처리를 적절한 노드에 할당하는 역할을 수행하는 것을 뜻하며, 이런 기능을 수행하는 노드를 트랜잭션 분배기(router)라 한다[16]. 효율적인 트랜잭션 분배를 통해 전체 노드의 부하를 평준화(load balancing)할 수 있으며, 이는 한정된 자원의 이용 효율을 극대화할 수 있도록 해준다. 이때 일정 기간 동안의 트랜잭션 처리율은 트랜잭션 처리 시스템에 참여하는 노드의 수와 연동되는 디스크의 수, 네트워크의 대역폭, 동시에 동작하는 쓰레드의 수 등에 의해 변화된다. 이를 고려해 트랜잭션 처리 시스템을 모델링한 후 트랜잭션 분배 전략이 상이한 경우를 시뮬레이션한다.

가정된 트랜잭션 처리 시스템은 데이터를 공유하는 다중 시스템 환경의 고성능 트랜잭션 처리 시스템[17]이며, 수퍼 컴퓨터나 MPP에 비해 저렴한 워크스테이션을 다수 개 연동해 고속의 트랜잭션 처리 능력을 실현할 수 있다. 또한 이기종 환경을 고려하고 있으므로 다양한 형태의 연동이 가능하며, 각 시스템의 성능과 데이터의 저장 형태에 따라 전체 시스템의 트랜잭션 처리 성능이 유동적일 수 있다. 따라서 트랜잭션의 적절한 분배를 통해 자원의 효율적인 활용이 가능해진다.

요청되는 트랜잭션 수가 일정 범위 이상이 되면 디스크 IO 횟수가 계속적으로 증가하며, 이에 따라 실제 트랜잭션의 처리에 소요되는 시간보다 디스크 IO에 소요되는 시간이 많아지게 된다. 때문에 트랜잭션 처리율은 더 이상 증가하지 못하며, 오히려 감소하게 된다. 따라서 동일한 트랜잭션 타입들을 동일한 노드에서 실행하도록 함으로써 노드의 참조 지역성을 높이고 캐시 효율성을 극대화하기 위해 각 노드의 주사본 권한(Primary Copy Authority) 정보[18]를 이용한다. 캐시의 이용은 디스크 IO 횟수를 감소시키기 위한 것이므로 이에 따른 트랜잭션 처리율의 감소를 해결할 수 있다. 또한 PCA 정보를 이용하는 경우에 로킹에 소요되는 메시지 오버헤드를 최소화할 수도 있으므로 전체적인 노드들의 성능의 향상을 기대할 수 있다.

### 5.2 트랜잭션 라우팅 전략

본 실험에서는 다음과 같은 두 가지의 트랜잭션 라우팅 방법을 비교한다.

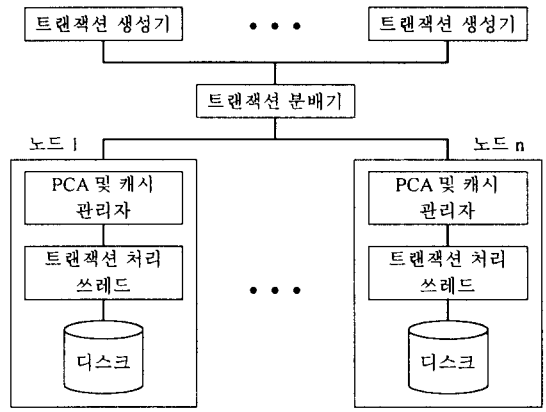
1) 랜덤 라우팅(random routing)
2) PCA 기반 라우팅(PCA-based routing)

랜덤 라우팅은 각각의 노드에 일정 수의 트랜잭션을 고정적으로 할당하는 방법이다. 이는 동적인 부하량의 변화를 전혀 고려하지 않는다. PCA 기반 라우팅은 각 노드가 보유하고 있는 데이터에 대한 주사본 권한을 이용해 트랜잭션을 라우팅한다. 즉

요청되는 트랜잭션이 참조할 데이터에 대한 PCA 정보를 보유하고 있는 노드에 트랜잭션을 요청하는 것이다. 이 방법은 캐시 효율성을 극대화할 수 있으나 PCA 보유 노드에 부하가 폭주하는 경우에는 오히려 성능이 저하될 수 있다는 문제점이 있다.

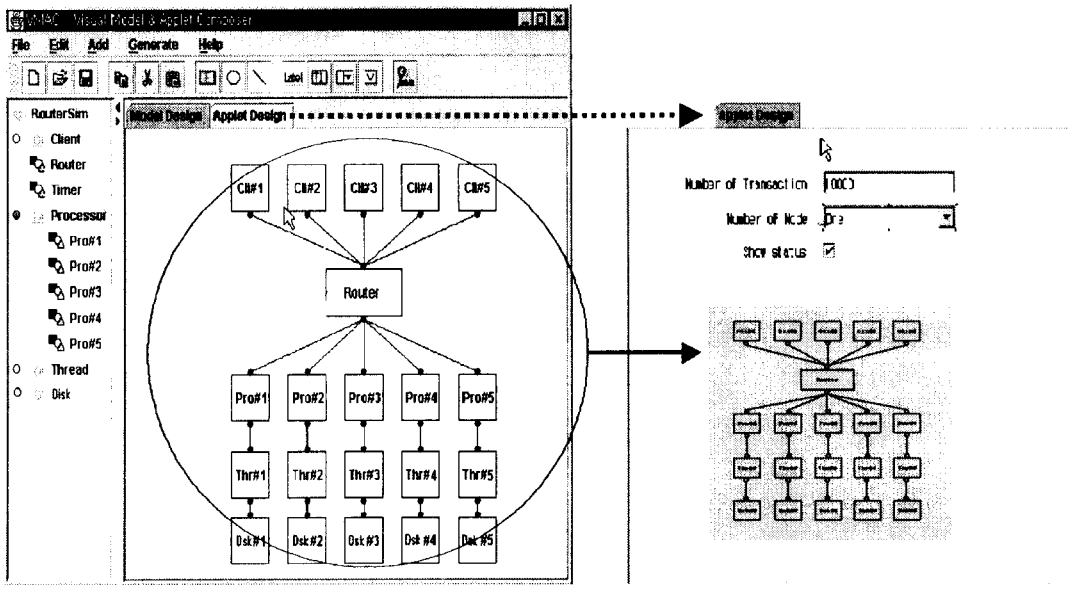
### 5.3 모델 설계

시뮬레이션 모델은 <그림 5>와 같이 트랜잭션 생성기, 트랜잭션 분배기, 트랜잭션 처리 노드로 구성된다. 트랜잭션 생성기는 트랜잭션을 요청하는 클라이언트를 표현한다. 트랜잭션 분배기는 요청된 트랜잭션을 2가지 분배 전략에 의해 트랜잭션 처리 노드로 분배하며, PCA에 기반한 트랜잭션 분배를 위해 각 노드에 저장된 레코드의 PCA 정보를 관리한다. 트랜잭션 처리 노드는 트랜잭션처리의 기본 단위로서 분산 트랜잭션에서의 동기화를 위한 PCA 및 캐시관리자, 트랜잭션 처리 쓰레드, 그리고 디스크로 구성된다. PCA 및 캐시관리자는 자신이 저장하고 있는 레코드에 대한 PCA 정보와 캐쉬 정보를 관리한다. 트랜잭션 처리 쓰레드는 트랜잭션처리의 병렬성을 위해 다수개로 구성된다. 시뮬레이션을 위한 변수들은 <표 2>와 같다[19].




<그림 5> 분산 트랜잭션 처리 시스템의 시뮬레이션 모델

<그림 6>은 VMAC에서의 모델 및 애플릿 작성 과정을 보여준다. 모델 작성이 끝나면 코드 생성을 하게 되는데 <그림 7>은 코드생성에 사용되는 모델 및 애플릿 정보를 보여준다. 결과적으로 생성된 코드는 <그림 8>과 같다.




<그림 6> VMAC을 이용한 모델과 애플릿 작성

<표 2> 시뮬레이션 변수

| 시스템 구성 변수 | | |
|---|---|---|
| LCPUSpeed | 처리 노드의 CPU 속도 | 10 MIPS |
| NetBandWidth | 네트워크의 데이터 전송 속도 | 10 Mbps |
| NumOfNodes | 처리 노드의 수 | 5 |
| NumTerms | 시스템 전체의 터미널 수 | 10 ~ 700 |
| NumDisk | 공유 디스크의 수 | 5 |
| MinDiskTime | 최소 디스크 액세스 시간 | 10 ms |
| MaxDiskTime | 최대 디스크 액세스 시간 | 30 ms |
| DBSizePerNode | 각 노드에 저장된 DB 크기 | 20000 record |
| PageSize | 각 페이지의 크기 | 4096 bytes |
| RecPerPage | 한 페이지에 포함된 레코드 개수 | 20 record |
| CacheSize | 캐시 버퍼의 크기 | 20% of DB size |
| 트랜잭션 변수 | | |
| FixedMsgInst | 메시지 처리를 위한 고정 명령수 | 20,000 |
| PerPageMsgInst | 메시지 길이 당 추가되는 명령수 | 10,000 / page |
| ControlMsgSize | 제어 메시지의 길이 | 256 bytes |
| LockInst | 로크 등록/해제를 위한 명령수 | 300 |
| PerIOInst | 디스크 IO를 위한 명령수 | 5000 |
| CreationDelay | 트랜잭션 생성 시 평균 대기 시간 | 0.1 |

```
Entity
Entity_name Generator
Entity_X1_Y1 10 10
Entity_X2_Y2 50 40
Port_name out
Port_X_Y 50 25
Random_name T
Random_type 2
Trace_name test
Trace_type 2
End

생략...

Entity
Entity_name TRProcessc
Entity_X1_Y1 200 200
Entity_X2_Y2 250 240
Port_name in
Port_X_Y 230 210
Random_name IA
Random_type 1
End

생략...
```

```
Connected_Entity Generator F
Connected_Port out in
Connected_Entity Router TRP
Connected_Port out in
Variable_name numTR
Variable_type 2
Control_type 2
Control_default 10
Variable_name numNode
Variable_type 2
Control_type 3
Control_default 5

생략...

End
```

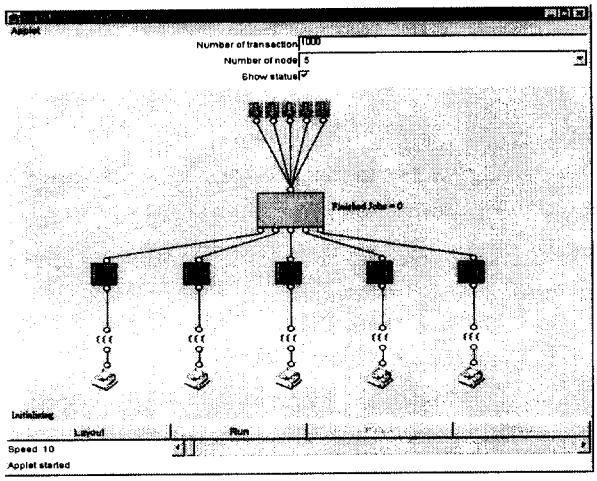
<그림 7> 저장된 모델과 애플릿 정보

```
// Import : You can append some other classes
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import eduni.simjava.*;
import eduni.simanim.*;

// Start of Class Declaration
class Generator extends Sim_entity {
  private Sim_port out;
  private Sim_uniform_obj T;
  private int test;
  // TO DO ...

// Start of Class Constructor
  public Generator(String name, ..., int x, int y) {
    super(name, "Generator", x, y);
    out = new Sim_port("out", "port", 50, 25);
    add_port(out);
    T = new Sim_uniform_obj("", ..., (int)System.currentTimeMillis());
    // TO DO ...
  }

// Start of Method Overriding
  public void body() {
    // TO DO ...
  }
}

생략...

// Start of Applet code
public class RouterSim extends Anim_applet implements ItemListener {
  Panel inputs;
  Label numTRLabel, numNodeLabel;
  Choice numNodeChoice;

// Initialise the Layout
  public void anim_init() {
    // TO DO
  }

// Setup the animation
  public void anim_layout() {
    // TO DO
  }

  // TO DO
}
```

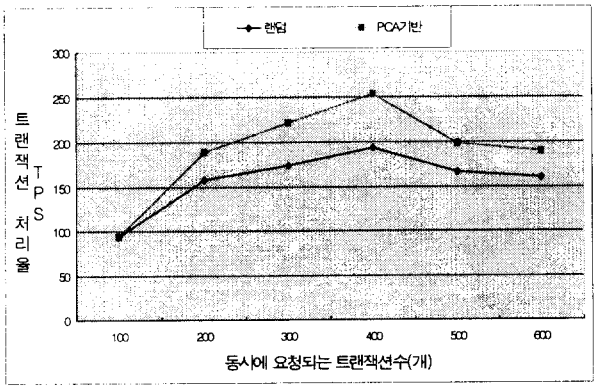
<그림 8> 생성된 자바 코드 예

### 5.4 시뮬레이션 실행 및 결과

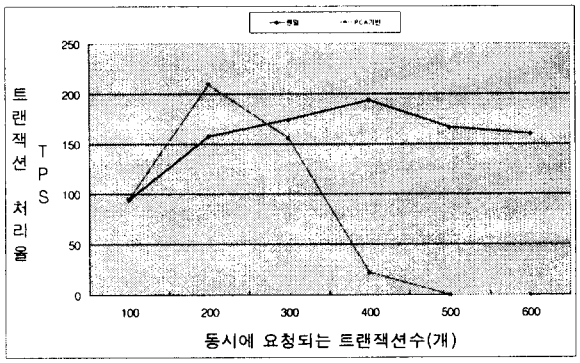
모델 개발이 끝난후 애플릿 뷰어(appletviewer)에서 실행시킨 모습은 <그림 9>와 같다. 시뮬레이션 변수로는 동시에 요청되는 트랜잭션의 개수를 사용하였다. 이 값을 변화시키며 트랜잭션 시스템의 트랜잭션 처리율을 측정했으며, 측정은 2가지 트랜잭션 분배 방법에 대해 개별적으로 수행되었다. 실제 전체 트랜잭션 처리 시스템의 각 노드에 요청되는 트랜잭션이 균등한 경우에 랜덤 분배 기법은 캐시 이용률이 좋지 못하므로 PCA 기반 분배 기법에 비해 트랜잭션 처리율이 떨어진다. 그러나 PCA 기반 분배 기법은 특정 노드에 트랜잭션 요청이 많아지는 경우에 일부 노드의 과부하 상태가 발생할 수 있으며, 이는 전체 시스템의 성능을 저하시킬 수 있다. <그림 10>은 전체 노드에 균등한 트랜잭션이 요청되었을 때의 실험 결과를 나타내며, <그림 11>은 일부 노드들(3개 노드)에만 편중되게 트랜잭션이 요청되는 경우의 실험결과를 나타낸다.




<그림 9> 시뮬레이션 실행 예




<그림 10> 시뮬레이션 결과(균등한 트랜잭션 요청)




<그림 11> 시뮬레이션 결과(편중된 트랜잭션 요청)

## 6. 결론

본 논문에서는 웹기반 시뮬레이션을 위한 시각적 모델 개발 환경 VMAC을 제안하였다. VMAC은 웹상에서의 시뮬레이션 수행 및 에니메이션을 위한 시뮬레이션 모델 코드를 자동 생성해주므로써 모델 개발을 쉽게 해준다. VMAC의 효용성을 보이기 위해 분산 트랜잭션 처리시스템에서의 부하 평준화기법에 대한 시뮬레이션을 수행하였다. 향후 과제는 모델의 행동에 대한 시각적 모델 개발 기능을 보완하는 것으로써 전문적인 지식을 보유하지 못한 일반인들의 시뮬레이션 활용에 큰 도움을 줄 수 있을 것이다.

향후과제로서 현재 VMAC에서는 모델의 행동에 대한 시뮬레이션 코드작성은 개발자가 직접 코딩을 해야하지만 이를 컴포넌트 기반(component- base)의 개발방법을 사용해 VMAC에서 시각적으로 설계할 수 있도록 확장해 나갈 예정이다.

## 참 고 문 헌


[1] Russell, E.C., G.W. Evans, M. Mollaghasemi, W.E. Biles, "SIMSCRIPT II.5 and SIMGRAPHICS Tutorial," Proc. of the Winter Simulation Conference, 1993, pp.223-227.

[2] CACI, *MODSIM III Tutorial*, CA, 1996.

[3] Mesquite Software, *User's Guide: CSIM18 Simulation Engine*, TX, 1994.

[4] Sun Microsystems, The Java Tutorial, http://java.sun.com/docs/books/tutorial/index.html, April. 1999.

[5] Howell, F.W., The simjava home page, http://www.dcs.ed.ac.uk/home/hase/simjava, April. 1999.

[6] University of Newcastle upon Tyne, JavaSim home page, http://javasim.ncl.ac.uk/, Apri. 1999.

[7] Healy, K.J. and R.A. Kilgore, "Introduction to Silk and Java-based Simulation," Proc. of the 1998 Winter Simulation Conference, 1998.

[8] Austin, W.J., J. Liddle, R.C. Thomas, P. McAndrew, Networked Educational Simulations in Java, http://www.ltc.hw.ac.uk/mverse, Oct. 1998.

[9] DMSO, HLA Homepage, http://hla.dmso.mil, April. 1999.

[10] Wutka, M., *Hacking Java*, Que Corp., IN, 1997.

[11] Orfali, R., D. Harkey, *Client/Server Programming with Java and CORBA*, Wiley Computer Publishing, New York, 1997.

[12] MIL 3, OPNET Tutorial Manual, Washington DC, 1997.

[13] Fishwick, P., Digital Circuit Simulation on the Web, http://www0.cise.ufl.edu/~fishwick/dig/dlesp.htm, Oct., 1998.

[14] DeSoto, A., Using the Beans Development Kit 1.0, Sun Microsystems, CA, 1997.

[15] Sun Microsystems, Using Swing Components, http://java.sun.com/docs/books/tutorial/uiswing/components/index.html, April. 1999.

[16] Rahm, E., "A Framework for Workload Allocation in Distributed Transaction Systems," System Software Journal, Vol.18, No.3, 1992, pp.171-190.

[17] 조행래 외 14명, 데이타를 공유하는 다중 시스템 환경에서 고성능 트랜잭션 처리 시스템 개발, 정보통신부 국책기술개발사업 2차년도 연차보고서, Jan. 1999.

[18] Reuter, A., "Load Control and Load Balancing in a Shared Database Management System," Proc. of 2nd International Conference on Data Engineering, 1986, pp.188-197.

[19] Zaharioudakis, M., M. Carey, M. Franklin, "Adaptive, Fine-Grained Sharing in a Client-Server OODBMS: A Callback-Based Approach," ACM Trans. on Database Systems, Vol.22, No.4, 1997, pp.570-627.

## 저자소개

김기형

1990년 2월 : 한양대학교 전자통신공학 학사
1992년 2월 : 한국과학기술원 전기 및 전자공학과 석사
1996년 8월 : 한국과학기술원 전기 및 전자공학과 박사
1996년 8월~1997년 2월: 한국과학기술원 위촉연구원
1997년 3월~현재 : 영남대학교 컴퓨터공학과 전임강사
관심분야 : 네트워크기반컴퓨팅, 멀티미디어운영체제, 시스템모델링 및 성능평가, ATM네트워크관리

남영환

1996년 2월 : 경북대학교 고분자공학과 학사.
1997년 9월~현재 : 영남대학교 컴퓨터공학과 석사과정.
관심분야 : 웹 기반 시뮬레이션, 자바 & 인터넷 보안