객체 지향 시뮬레이션의 응용에 관한 연구

Review of Object-Oriented Simulation

김경섭, 서석주°
연세대학교 산업시스템공학과

Kyung Sup Kim, Seok-Joo Seo°
Dept. of Industrial Systems Engineering Yonsei Univ.

요 약

본 논문에서는 객체지향개념을 도입한 시뮬레이션, 즉 객체지향시뮬레이션(Object-Oriented Simulation)에 대하여 살펴보았다. 기존의 시뮬레이션언어가 가지고 있는 한계점을 통해서 객체지향시뮬레이션의 필요성과 장점에 대해서 알아보았다. 객체지향 프로그램인 Smalltalk를 이용해서 객체지향시뮬레이션의 모델링 방법론에 대해서 논의하였으며, 향후 이 방법론을 적용하여 객체지향 시뮬레이터 개발을 하고자 한다.

1. 서 론

오늘날의 시스템의 특징은 복잡성과 역동성이라는 두 단어로 표현될 수 있다. 이러한 시스템을 현실적이고 세밀하게 묘사하고 시스템의 변화에 적극적으로 대처하기 위해서 시뮬레이션 기법이 사용되어 왔다. 지금까지 많은 시뮬레이션 언어들이 개발 되었지만은 아직까지는 사용하기가 어렵고 그 응용범위가 한정되어 있다는 맹점을 가지고 있는 것이 사실이다.

이에 좀더 사용하기 쉬우며 수많은 다른 환경에 쉽게 응용할 수 있는 유연성을 가
진 새로운 시뮬레이션 기법에 대한 요구가 대두 되었으며 객체지향 시뮬레이션(Object-Oriented Simulation : OOS)이 이에 대한 해답으로 각광을 받고 있다.

OOS는 그 이름에서 쉽게 알 수 있듯이 시

뮬레이션을 객체지향적인 방법으로 접근함을 말한다. 현실세계의 모든 것을 객체(Object)로 보는 객체지향접근을 이용한 OOS는 강력한 재사용성, 확장성 그리고 사용하기 쉽다는 장점으로 기존 시뮬레이션 언어의 대안으로 훌륭히 자리를 잡아가고 있다.

OOS의 개념이 등장한지 꽤 많은 시간이 흘렀지만 아직까지 국내에서의 연구는 다소 미비하다. DEVS연구[15]가 괄목한 만한 성과를 이루어 내고 있지만 아직까지 국내에서의 OOS 에 대한 연구는 활발하다고 볼 수는 없을 것 같다. 본 연구에서는 지금까지 OOS와 관련해 이루어진 여러 연구들을 분석, 정리해 보고 Smalltalk[16]를 이용한 모델링에 대해 살펴보겠다.

2. 배 경

시뮬레이션은 복잡한 공정이나 시스템을 분석하고 모델링하는데 있어 가장 강력한 분석툴 중의 하나이며 지난 수십년동안 시뮬레이션을 지원하는 많은 언어들이 개발되어 왔다[5].

이들 언어 중에는 GPSS, SIMAN, SLAM등과 같은 시뮬레이션 프로그래밍 언어들과 비프로그램언어 패키지인 WITNESS, SIMFACTORY, ARENA등이 포함된다[1]. GASP나 Simscript같은 초기 언어들은 시뮬레이션의 모델링을 좀더 쉽게하기 위해서 무작위변수추출, 통계결과분석, 사건 카렌더와 같은 프로그래밍 facility를 제공했다. GPSS는 분산 "운송단위(units of traffic)"에 기반한 모델링 구조를 제시했으며, 1980년대의 SIMAN이나 SLAM은 생산시스템이나 자재운반시스템(MHS)을위한 별도의 모델링 모듈을 개발했으며 결과치 분석을 위한 통합환경을 제시했다. WITNESS나 PROMOD, ARENA와 같은 패키지

시뮬레이션 툴은 모델 개발에 있어 비프로그램적 접근을 보여 주어 사용자들이 프로그램에 관한 특별한 기술 없이도 이 툴을 사용할 수 있게 되었다.

생산시스템과 자재운반 시스템은 이러한 시뮬레이션 언어들을 이용한 모델링이 가장 빈번하게 적용된 분야이다. General-purpose simulation언어인 SIMAN과 SLAM은 이들 시스템을 모델링하기 위해 애용되어 왔다[8]. 이러한 언어를 사용함으로써 시뮬레이션 코드의 크기가 현저하게 줄어드는 장점이 있기 했지만 여러 다양한 시스템의 상이성을 표현하기에는 부족한 점이 많았다[10].

특히 오늘날과 같이 시스템과 프로세스가 새로운 기술의 보조를 맞추기 위해 끊임없이 변화하는 환경에서 효율적인 시뮬레이션 언어 는 다음의 네 가지 조건이 요구된다[1].

1) 사용하기 쉬울 것
2)모델링 하고자 하는 시스템을 세부적인 면까지 정확하게 표현할 수 있을 것
3) 모델 개발을 빨리 할 수 있을것
4) 시스템 변화에 빠르게 대응할 수 있을 것

이들 네 가지 요구조건은 결국 기존의 시뮬레이션 언어가 가지고 있는 한계를 말해주는 것이다. 이 분야에 대한 연구는 많은 학자들에 의해 이루어 졌으며 다음의 두 분야로 다시 요약 할 수 있겠다[1,3,4,6,12].

a. Flexibility의 부족

오늘날의 생산 시스템 환경은 그 규모가 매우 복잡할 뿐만 아니라 빠른 속도로 변화하고 있다. 따라서 이들 환경을 시뮬레이션 하는 일도 더욱 더 많은 노력과 시간을 요구한다. 기존의 시뮬레이션 언어들은 단순하고 정적인 환경을 모델링 하는 데는 많은 이점을 제공하나 일단 응용범위를 벗어나면 쉽게 적용할 수가 없다. 기존의 모델로 시스템의 변화나 다른 시스템 환경으로의 확장이 용이하지가 않은 것이다. 가령 생산시스템에 새로운 장비가 추가되어 이를 모델링에 반영해야 할 때 극단적인 경우에는 소스 코드를 모두 살펴보고 고쳐야 하거나 아니면 시뮬레이션 소프트웨어 회사가 해당 장비와 관계되는 새로운 모델링 block을 제공할 때 까지 기다려야만 하는 경우가 발생할 수 있다.

b. 사용이 용이하지 않음

사용자가 기존의 시뮬레이션 언어를 이용해 모델링을 하기 위해서는 특정 프로그램의 전문적인 자질을 가지고 있어야만 한다. 많은 시뮬레이션 프로그램들은 사용자에게 능숙한 프로그래머가 되길 요구하고 있다. 따라서 프로그램에 대한 전문적인 소질이 있는 사람만이 시뮬레이션을 수행 할 수 있는 것으로 인식되었다.

기존의 시뮬레이션이 가지고 있는 이러한 한계를 극복하고 복잡하고 끊임없이 변화하는 시스템과 프로세스를 현실적으로 묘사하기 위해서 객체지향 시뮬레이션(Object- Oriented Simulation)이 대두되었다,

3. Object-Oriented Simulation

3.1 OOS 개요

객체지향시뮬레이션(Object-Oriented Simulation:OOS)는 세상의 모든 것을 객체(Object)로 보는 객체지향접근방법을 시뮬레이션에 응용한 것이다. 객체지향접근법이 프로그램에 응용된 것은 아주 오래전 부터이다. 전문가 시스템(expert system)이나 지식 기반 시스템(Knowledge-based System)을 위한 개발 도구로 객체 지향 접근법은 애용되어 왔다[9].

OOS는 보통 C++과 같은 객체지향프로그램을 이용해 구현되어 왔다. OOS이전에도 물론 시뮬레이션을 지원하는 여러 언어들이 존재했었다. 우리는 이들 언어를 2장에서 대략적으로 살펴보았다. 최초의 객체지향 시뮬레이션 언어로 Simula가 개발된 이후에도 MODSIM, Sim++, Smalltalk [6]같은 OOS언어들이 개발되었다. SIMAN, SLAM, GPSS와 같은 언어의 경우 QUEUE, RESOURCE, TRANSACTION과 같은 객체를 제공해 객체지향적접근을 보여주었지만 사용자가 자신만의 객체를 만들어 낼 수 없다는 점에서 진정한 OOS언어라고 할 수는 없다[6].

OOS에서는 모델링하고자 하는 시스템의 모든 구성요소를 객체(Object)로서 취급하며 이들 객체들 사이의 상호작용을 기술하는데 초점을 맞추고 있다. 객체의 기술과 생성을 지원한다는 점에서 OOS는 다른 시뮬레이션 언어와 차별화된다.

객체는 자신만의 데이터 구조와 procedure를 가진 entity이며 encapsulation은 모듈속에 데이터를 묶음으로서 객체들이 상호의존적으로 되는 것을 방지한다. 즉 객체들은 메시지 전송을 통해서만 접근할 수가 있는 것이다. 메시지 전송(sending message)은 객체들 사이의 상호작용을 가능하게 한다. 메시지를 전달받은 객체는 그 메시지에 상응하는 메소드(method)를 생성시킨다. 이러한 현상은 생성된 procedure가 컴파일 단계에서가 아니라 실행단계에서 결정이 된다는 점에서 기존의 절차지향 언어가 함수를 호출하는 것과 차이가 있다. 이를 dynamic binding이라 부른다. 객체들은 클래스를 통해 계층적으로 조직되어 있으며 하위 클래스의 객체들은 상위 클래스의 속성을 그대로 물려받는다. 이를 상속성(inheritance)라 부른다. 이 특성은 프

로그램의 재사용성(reusability)을 보장해 주어 프로그램의 확장을 쉽게 만들어 준다[1].

3.2 Smalltalk를 이용한 모델링

Diane P.Bischak와 Stephen D. Roberts는 "Object-Oriented Simulation"[6]이라는 논문에서 C++를 이용해 객체지향 시뮬레이션을 모델링하는 기초적인 방법을 제시하며 객체지향 시뮬레이션의 여러 특징들에 대해서 설명했다. 여기서는 이 논문에서의 예제를 참고로 하여 또다른 객체지향 언어인 Smalltalk를 이용해서 설명해 보고자 한다. 소프트웨어는 ObjectShare사의 VisualWorks를 사용했다[16].

3.2.1 Smalltalk Environment

Smalltalk는 message/object개념을 이해하는 것이 중요하다. 메시지가 객체에게 보내어지면 해당 메시지에 대응하는 메소드를 생성시킨다. 예를 들어 theta라는 수의 sin값을 알기 위해서는 다음과 같이 표현한다[9].

```
theta sin
```

여기서 sin은 메시지이고 theta는 sin이라는 메시지를 받는 리시버이다. 메시지를 받은 리시버는 적당한 메소드(여기서는 sin값을 계산하는 메소드)를 발생시켜 theta에 대한 sin값을 얻어낸다.

3.2.2 Modeling

1) 클래스 정의

다음은 Smalltalk에서 Vehicle이라는 클래스를 정의하는 문구이다. 이 클래스는 세개의 instanceVariableNames를 가지고 있다. 각각은 Vehicle의 속도, 용량, 형태를 나타낸다.

```
Object subclass: #Vehicle
    instanceVariableNames:'speed
        capacity make'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Example'
```

2) 객체생성

Vehicle이라는 클래스안의 객체(instance)를 생성시키기 위해서는 new라는 메시지를 사용한다. 다음은 Vehicle이라는 클래스안 의 Fork_truck이라는 객체를 생성시키는 문구이다[2].

```
Fork_truck := Vehicle new
```

후에 Fork_truck이라는 객체는 instance variable인 speed, capacity, make의 값을 할당 받을 것이다.

3) 상속성을 이용한 클래스 정의

만약 사용자가 새로운 클래스생성이 필요하고 그 클래스 속성들이 이미 만들어진 클래스와 비슷하다면 사용자는 미리 만들어진 클래스를 이용해 쉽게 새로운 클래스를 만들어 낼 수 있다.

복도와 같은 특정 통로에서만 작동하는 또 다른 Vehicle에 대한 클래스를 만들어야 된다고 생각해 보자

```
Object subclass: #Fixed_path
        instanceVariableNames:'speed
capacity make xloc yloc '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'Example'
```

여기서 xolc, yloc는 통로에서의 위치를 나타내는 변수이다. 이 클래스는 통로에서의 직각거리를 계산하는 distance xloc: yloc:라는 메소드를 가지고 있다. 그런데 이 클래스는 우리가 먼저 기술했던 Vehilce의 모든 속성(speed, capacity, make)을 가지고 있다. 따라서 새로운 클래스를 기술하는 대신에 미리 만들어진 클래스인 Vehicle로부터 속성들을 상속받을 수 있다.

```
Vehicle subclass:#Fixed_path
    instanceVariableNames:'xloc, yloc'
    classVariableNames:''
    poolDictionaries:''
    category:'Example'
```

위의 Fixed_path 클래스에 대한 새로운 정의를 보면 Fixed_path는 Vehicle의 하위 클래스로서 Vehicle의 모든 속성(speed, capacity, make)을 계승하고 있다.

사용자는 기존의 클래스를 재사용 함으로써 프로그램을 쉽게 확장시킬 수 있다.

4) 변수

Smalltalk에서 사용하는 변수의 종류는 temporary, instance, class instance, class, pool, global의 여섯가지이다. Temporary와 Instance 변수는 각각 메소드와 객체(instance)에 그 적용범위가 한정되기 때문에 private 변수라고 하며 나머지는 global변수라고 부른다.

Instance 변수는 클래스 안에 포함된 객체의 데이터를 저장하기 위한 것이며 오직 객체(instance)에 의해서만 접근이 가능하다.

Class 변수는 주클래스 객체와 그 하위클래스의 객체를 위한 것이며 모든 객체에서 접근이 가능하다. 여러 객체들에 의해서 공유되기 때문에 Class 변수의 첫 문자는 대문자로 표기해 준다. Class 변수의 초기값들은 일반적으로 클래스 메소드(보통 initialize)에서 할당되어 진다.

Float class의 Pi 변수는 수학에서의 상수값을 나타내는 Class변수의 예가 될 수 있다.

우리가 위에서 살펴본 Vehicle과 Fixed_truck Class의 경우는 모두 instnace 변수만 가지고 있는 경우이다.

5) Polymorphism

Polymorphism이란 동일한 메시지가 서로 다른 리시버에 의해 서로 다른 방법으로 해석될 수 있는 성질을 말한다. 즉 실제 method나 procedure는 메시지가 특정한 리시버에게 보내지기 전까지는 발생하지 않는다.

위의 Vehicle 클래스에 fork_truck과 stacker_cranes라는 객체가 있다고 하자. 이 둘은 모두 단위 운송물을 선적하고 수송하는 기능을 가지고 있다. 따라서 pickup이라는 메세지가 fork_truck과 stacker_cranes두 객체에 모두 적용될 수가 있을 것이다.

fork_truck pickup

stacker_crane pickup

pickup이라는 동일한 메시지가 서로 다른 두 객체에 이용되었다..

Polymorphism의 장점중 하나는 메시지 이름들의 중복을 가능하게 한다는 것이다. 그래서 똑같은 이름을 가진 메시지가 자주 사용되는 기능을 위해 시스템안에서 여러번 사용될 수 있다.

Smalltalk에서 많이 사용되는 new:, =, +, do:, copy와 같은 메시지들은 20번도 넘게 새롭게 정의된다[14].

Dynamic Binding

Polymorphism을 가능하게 하는 것은 dynamic binding이라는 개념이다. 기존의 절차지향 언어에서는 함수와 파라미터의 행동이 컴파일 타임때 결정되었다. 그러나 Smalltalk에서는 메소드(기존언어에서의 함수)와 파라미터의 행동이 컴파일 타임에서가 아니라 실행될때 결정이 난다. 이를 Dynamic binding이라 부른다[14]. 예를 들어 우리가 여러 가지 도형을 화면상에 나타내려는 표현을 원한다고 하자.

C나 Pascal과 같은 언어에서는 도형을 화면상에 나타내기 위해서 각 도형의 속성 기술과 display라는 명령을 위한 긴 문장을 필요로 했다. 그러나 Smalltalk에서는 display라는 메시지(메소드 포함)를 이용해 다음과 같이 표현해 주기만 하면 된다.

aFigure display

리시버인 aFigure는 Triangle, Rectangle, Square, Circle 등이 될 수 가 있을 것이다. 이는 display라는 메소드와 aFigure라는 파라미터의 행동이 컴파일타임 에서가 아니라 실행될때 결정되기 때문이다. (즉 display라는 메시지를 받은 리시버인 aFigure가 어떤 메소드를 발생시킬지 결정하는 것이다.)

4. 결론

지금까지 객체지향 시뮬레이션에 대한 개괄적인 내용과 Smalltalk를 이용한 모델링에 관해 살펴보았다. 본 연구는 궁극적으로 OOS를 이용한 시뮬레이터의 개발에 그 목표를 두고 있다. OOS의 여러 장점들을 바탕으로 한 시뮬레이터의 개발은 많이 이루어져왔으며 생산시스템과 자재운반시스템의 경우는 자주 연구의 대상이 되어 왔다. 자재운반 시스템의 대표적인 예인 AGV 시스템을 위한 AgvTalk[10], 자재운반시스템 디자인을 위한 객체지향 시뮬레이터[7], Sawmill시뮬레이션을 위한 객체지향 모델링 프레임워크[11] 등은 OOS를 잘 활용한 시뮬레이터의 모습을 보여주고 있다. 추후의 연구에서는 본문에서 살펴본 객체지향 언어인 Smalltalk를 이용해서 생산시스템이나 자재운반시스템을 모델링하고 분석할 수 있는 시뮬레이터를 만들어 보고자 한다.

참고문헌

[1] Adeel Najmi and Susan J, Stein "Comparison of Conventional and Object-Oriented Approaches for Simulation of Manufacturing Systems" IIE Integrated Systems Conference & Society for Integrated Manufacturing Conference Proceedings. pp. 471-476, 1989.

[2] Adele Goldberg and David Robson "Smalltalk - 80" ADDISON WESLEY 1989.

[3] Ball Pete and Doug Love "Expanding the Capabilities of Manufacturing Simulators Through Application of Object-Oriented principles" Journal of Manufacturing System. Vol. 13, No. 6, 1994.

[4] Bernard P. Zeigler"A Knowledge-Based Simulation Environment for Hierarchical Flexible Manufacturing" IEEE Transaction on Systems Man and Cybergenetics-Part A:Simulation and Humans. Vol 26, No. 1, pp. 81-89, 1996.

[5] Dennis C. Pegden, Robert E. Shannon and Randall P. Sadowski "Introduction to Simulation Using SIMAN" McGraw Hill 1990.

[6] Diane P. Bischak and Stephen D. Roberts "Object-Oriented Simulation" Proceedings of the 1991 Winter Simulation Conference. pp. 194-203.

[7] Jocelyn R. Drolet and Marc Moreau "Development of an Object-Oriented Simulator for Material Handling System Design" Computers & Industrial Engineering. Vol. 23, Nos 1-4, pp. 249-252, 1992.

[8] John P. Shewchuk and Tien-Chien Chang "An Approach to Object-Oriented Discrete-Event Simulation of Manufacturing Systems" Proceedings of the 1991 Winter Simulation Conference. pp. 302-311.

[9] Onur M. Ulgen and Timothy Thomasma "Simulation Modeling in an Object-Oriented Environment Using Smalltalk-80" Proceedings of the 1986 Winter Simulation Conference. pp. 474-484.

[10] Russell E. King and Kyung Sup Kim "AgvTalk: An Object-Oriented Simulator for AGV Systems" Computers & Industrial Engineering. Vol. 28, No 3, pp.575-592, 1995.

[11] Sabah U. Randhawan, Charles C. Brunner, James W. Funck and Guangchao Zhang "Object-Oriented Modeling Framework for Sawmill Simulation" Computers & Industrial Engineering. Vol. 25, Nos. 1-4, pp 565-568, 1993.

[12] Sadashiv Adiga and Woo-Tsong Lin "Object-Oriented Simulation of Manufacturing Systems" IIE Integrated Systems Conference & Society for IntegratedManufacturing Conference Proceedings 1989. pp. 489-494.

[13] Timothy Thomasma and James Madsen "Object Oriented Programing Language for Developing Simulation Related Software" Proceedings of the 1990 Winter Simulation Conference. pp 482-485.

[14] Wilf R. Lalonde and John R. Pugh "Inside Smalltalk" Prentice Hall New Jersey 1990.

[15] Yeong Rak Seong, Tag Gon Kim and Kyu Ho Park "Performance Evaluation of a Parallel DEVS Simulation Environment P-DEVSIM++" 한국시뮬레이션 학회 논문지 제 2권, 제 1호, 1993.

[16] Objectshare "VisualWorks:Application Developer's Guide" ObjectShare 1998.