□ 研 究 論 文 □

韓國軍事運營分析學會誌
第 23 卷, 第 1 號, 1997. 6. 30

MODSIM II 환경에서 객체지향 시뮬레이션개발 방법인 SMG 방법의 적용성에 관한연구 - 전차 교전 시뮬레이션 소프트웨어 프로토타입 개발 예를 중심으로-

(Applicability of SMG method for the development of object-orient simulation with MODSIM II)

최 상 영*

Abstract

This paper is aimed to investigate the applicability of SMG method to the development of an object-oriented simulation software in the MODSIM II environment. SMG method is an object-oriented simulation software development method proposed by System Modeling Group(SMG), National Defense University of Korea. Through this study, we concluded that SMG method can be a powerful method for the development of object-oriented simulation software in the MODSIM II environment. However, SMG method lacks in depicting some of messages in the MODSIM II, for example, ASK method, TELL method, WAITFOR method, interrupt. In the future, SMG method is expected to extend for incorporating those things presumably by referring to Professor Baileys pictures, OOSPICs.

* 국방대학원 조교수

1. 연구 배경 및 목적

시뮬레이션 기술은 정보화 사회의 중요한 핵심 기술 중의 하나로서, 국방 분야에서는 군사 교육 훈련, 교리 및 전술 발전, 전장 임무 수행 연습, 전력 소요 분석 및 평가, 국방 물자 획득 관리, 연구 개발 및 시험 평가 등 다양한 분야에 적용하고 있다[11].

오늘날 시뮬레이션 기술은 시뮬레이션 대상 시스템이 복잡하고 대형화 됨으로써, 시뮬레이션 소프트웨어 개발과 유지 보수 및 확장을 용이하게 하기 위하여 객체지향 기술을 적용하고 있으며, 실시간 처리 기술, 분산 및 병렬 처리 기술, 그래픽 인터페이스 기술, 에니메이션 기술[3] 등을 접목하여 가상 현실의 구현 뿐만 아니라 분산 대화식 시뮬레이션을 가능하게 하고 있다. 특히, 분산 대화식 시뮬레이션 기술은 군사 워게임 개발에 활용하여 공간적으로 떨어진 곳에서 실시간으로 상호 대화식 게임을 할 수 있게 함으로서 현실적인 전쟁 모의를 가능하게 하고 있으며, SIMNET[7]은 그 좋은 예가 된다.

이러한 시뮬레이션 소프트웨어를 개발하는 데에는 개발 언어와 환경에 제한을 받게 되는데, 시뮬레이션 언어는 일반적으로 사용하고 있는 고급 컴퓨터 언어와 특수 컴퓨터 언어로 구분된다. 고급 언어에는 기존의 BASIC, FORTRAN, PASCAL, C 등이 있고, 객체 지향 개념을 지원하는 SMALLTALK, C++, Ada 등이 있다. 그런데 고급 언어들은 시뮬레이션 루틴, 예를 들어, 시간 루틴(timing routine), 사건 루틴(event routine) 등을 제공하지 않기 때문에 시뮬레이션 응용 소프트웨어를 개발할 때에는 그 루틴들을 추가적으로 만들어야 하는 어려움이 있다. 반면에 특수 언어는 시뮬레이션을 위한 루틴들를 지원하기 때문에 시뮬레이션 소프트웨어 개발를 상대적으로 용이하게 한다. 특수 언어에는 GPSS, SIMSCRIPT, GASP, DYNAMO, SIMAN 등이 있다. 그런데 시뮬레이션 특수 언어도 시뮬레이션 소프트웨어를 개발함에 있어서 규모가 방대해 질 때에는 소프트웨어의 확장성, 유지 보수 및 재사용에 제한점이 있다.

그래서 최근에 시뮬레이션 특수 언어에 객체지향 개념을 구현한 객체지향 시뮬레이션 언어와 환경이 개발되고 있는데, MODSIM II[13,14]가 그 중의 하나이다. MODSIM II 는 객체지향 개념을 지원할 뿐만 아니라 분산 처리, 그래픽 사용자 인터페이스, 그리고 에니메이션 기능을 지원하기 때문에 시뮬레이션 소프트웨어를 개발하는데 있어서 확장성, 유지 보수성, 그리고 소프트웨어의 재 사용성을 증가시킬 수 있으며, 에니메이션을 통해 보다 현실적인 모의를 가능하게 해 주는 장점을 지니고 있다. 그런데 MODSIM II는 객체지향 환경을 제공하기 때문에 시뮬레이션 소프트웨어를 개발함에 있어서 객체지향 소프트웨어 개발 방법에 의존하게 된다. 일반적인 객체지향 소프트웨어 개발 방법은 많은 연구자에 의해 제안되었다[6,8,9,12]. 그러나 이들 방법은 데이타베이스 소프트웨어 개발, 실시간 시스템 소프트웨어 개발등에는 중점을 두고 있다. 한편, 국대원 시스템 모델링 그룹에서 제안된 방법인 SMG(System Modeling Group) 방법[2]이 있는데, 이는 객체지향 시뮬레이션 소프트웨어를 개발할 때 적합하도록 고안되었다. 그외 Bailey 교수에 의해 제안된 OOSPICs 방법[5]이 있는데, 이는 MODSIM II 언어에 종속된 방법이다.

따라서 본 연구는 MODSIM II 환경에서 객체지향

시뮬레이션을 개발하는데 있어서 SMG 방법의 적용성에 대하여 연구하는데 그 목적이 있다. 이를 위하여 군사 시뮬레이션 소프트웨어 중의 하나인 전차 교전 시뮬레이션 소프트웨어 프로토타입을 개발하는데 SMG 방법을 적용해 봄으로서 그 적용성을 보고자 한다.

그래서 연구의 구성은 두 번째 부분과 세 번째 부분에서는 MODSIM II의 특징과 MODSIM II에서 객체지향 시뮬레이션 구도에 대하여 설명한다. 네 번째 부분에서는 본 연구에서 적용할 객체지향 시뮬레이션 소프트웨어 방법을 소개한다. 다섯 번째 부분에서는 MODSIM II 환경에서 SMG 방법에 의한 전차 교전 시뮬레이션 소프트웨어 프로토타입 개발 예를 기술하고, 여섯 번째 부분에서 결론을 맺도록 한다.

2. MODSIM II 환경 일반

MODSIM II 는 Modula-2, Ada 와 유사한 구문과 구조를 가지며, 일반적인 시뮬레이션 언어와 몇가지의 다른 특징을 가지고 있다.

첫 번째, MODSIM II는 객체지향 특성을 제공한다. 객체지향 특성에는 객체(object), 클라스(class), 상속성(inheritance), 조립성(aggregation), 다형성(polymorphism) 등이 있다. 객체는 데이타 구조(data structure)와 그 데이타 위에서 수행할 수 있는 절차(procedure)들을 가진 소프트웨어의 가장 기본적인 단위를 말한다[2]. 이는 현실 세계에서 개체와 대응될 수 있다. 클라스는 유사한 성질을 가진 객체들의 모임이고 동시에 객체를 생성하는 형틀(template)이라고 할 수 있다. 상속성은 이미 정의된 하나 또는 그 이상의 클라스로 부터 그 속성을 상속받는 것인데, 여기서 속성이라는 것은 데이타 구조와 절차를 의미한다. 조립성은 클라스 내에 또다른 종류의 클라스를 내포하는 것인데, 상속성이 실 세계의 개체와 개체간의 관계에 있어서 IS-A 관계를 나타내는 것이라면 조립성은 IS-A-PART 관계를 나타내는 것으로 이해될 수 있다. 다형성은 여러가지 종류의 객체가 하나의 메세지에 대하여 그들의 해당 클라스에서 정의된 방식대로 응답하게 하는 특성이다. 그래서 객체지향 소프트웨어를 개발하는데 이러한 특성을 적절히 사용하여 클라스를 설계하는 것이 무엇보다 중요하다.

일반적으로 MODSIM II에서는 객체를 객체 인스탄스(Object Instance)라고 부르고, 데이타 구조를 필드(Field)라고 부르며, 절차를 메소드(Method)라고 부른다. 그리고 클라스를 객체형(Object Type)이라고 부른다. 그래서 MODSIM II 프로그램은 객체 인스탄스(Object Instance)라는 단위로 이루어져 있다. 객체 인스탄스는 시뮬레이션 대상 시스템의 개체(Entity)와 대응되는 개념으로 객체지향 시뮬레이션 소프트웨어의 기본 단위가 된다. 그리고 프로그램의 실행은 객체 인스탄스에 메세지를 전달하고 객체 인스탄스가 그 메세지에 대하여 적절히 응답함으로써 이루어진다. 여기서 메세지에 대한 응답 방식은 자신의 객체형의 메소드에서 정의된다.

객체지향 시뮬레이션 소프트웨어 개발은 무엇보다도 먼저 객체 인스탄스를 식별하여 이를 생성할 수 있는 객체형을 만들고 그 객체형을 사용하여 필요한 객체 인스탄스를 생성한 후에 객체간의 메세지 전달을 정의하여 코드화함으로써 이루어 진다.

MODSIM II는 객체지향 시뮬레이션을 위한 여러

가지의 객체형을 내장하고 있다. 그래서 시뮬레이션 프로그램을 개발할 때는 내장 객체형을 활용하면 개발을 용이하게 할 수 있다.

두 번째, 시뮬레이션 방법에는 사건 계획 방법(Event Scheduling), 활동 주사 방법(Activity Scanning), 과정 상호작용 방법(Process Interaction)이 있는데, MODSIM II는 과정 상호작용 시뮬레이션을 지원한다. 과정(Process)은 여러 가지의 활동(Activity)으로 이루어져 있고, 과정의 실행 시에는 몇 가지의 활동 수행 및 대기 단계가 필요하고 때로는 자원(Resource)이 필요할 수도 있다. 시뮬레이션 실행간에 각 객체 인스탄스는 활동 목록(Activity List)을 지니고 시스템은 객체 인스탄스의 미결 목록(Pending List)을 지닌다. 객체 인스탄스의 활동 목록에서는 활동 및 활동 발생 시간을 관리하고 시스템 미결 목록에서는 이들 객체 인스탄스를 관리한다. 그래서 시뮬레이션은 시스템 미결 목록에서 가장 최근에 발생할 활동을 가진 객체 인스탄스를 찾아 그 활동을 실행하고, 실행한 후에 그 활동을 제거한 다음 해당 객체의 활동 목록을 최신화(update)시키면서 계속 진행한다.

세 번째, MODSIM II는 그래픽 사용자 인터페이스와 시뮬레이션을 시각화 할 수 있도록 에니메이션(animation) 기능을 제공한다. 이를 위해서 그래픽 인터페이스 및 에니메이션을 위한 화상 객체형(Visual Object Type)을 내장하고 있다. 사용자는 내장된 화상 객체형을 상속 받아 재 사용함으로써 MODSIM II 프로그램을 그래픽 인터페이스와 에니메이션을 위한 프로그램으로 확장할 수 있다. 그래픽 인터페이스를 위하여 윈도우, 대화상자, 제어기 등의 화상 객체형을 사용하며, 에니메이션을 위해서는 비트맵, 이메지 등의 화상 객체형을 사용한다. 윈도우는 시뮬레이션 응용 프로그램의 주화면(main window)를 나타낼 때 사용할 수 있고 대화상자, 제어기 등은 입출력시에 주로 사용한다. 특히, 화상 객체는 MODSIM II의 SIMDRAW라는 Workbench에서 사용자가 직접 제작할 수 있다. 그리고 프로그램 실행간에 사용자와 인터페이스를 유지할 수 있는데 이는 Event Driven 구도로 이루어져 있다.

네 번째, MODSIM II 는 병렬 컴퓨터에서 분산 시뮬레이션을 지원한다.순차적 시뮬레이션(sequential simulation)의 경우 global clock을 사용해서 단계적으로 시뮬레이션 시계를 전진시키기 때문에 시뮬레이션 사건의 수가 많아지면 사건 목록(event list)에 대기 해야 할 사건들이 엄청나게 증가하므로 시뮬레이션을 수행하는데 많은 계산 시간이 필요하다[1]. 분산 시뮬레이션은 순차적 시뮬레이션과는 달리 하나의 시뮬레이션 프로그램을 여러 가지 프로세서로 나누고 프로세서간 공유 변수 들을 제거한 후 프로세서 들이 각기 다른 프로세시에서 동시에 수행할 수 있도록 함으로써 시뮬레이션 처리 속도를 향상시킨다.

다섯번째, MODSIM II는 모듈성(modularity)을 지닌다. 그래서 MODSIM II 프로그램은 주 프로그램(Main Program)과 모듈(Module)로 구성되어 있다. 모듈은 정의 부분(Definition Part)과 구현 부분(Implementation Part)으로 나누어 진다. 각 모듈은 서로 다른 화일에 저장되며, 각 화일은 독립적으로 컴파일이 가능하여 임의 프로그램 수정시 전체 프로그램의 재 컴파일을 하지 않아도 됨으로써 컴파일 시간을 절약할 수 있다. 모듈간의 인터페이스는 모듈의 정의 부분을 포함시키므로써 가능하다.

MODSIM II 프로그램에서는 화일 명을 사용하는데 제한하고 있다. 주 프로그램은 Mfilename.Mod와 같이 화일명 앞에 M이라는 첨자로 시작해야 하고 확장자는 *.Mod를 사용한다. 모듈의 정의 부분을 나타내는 화일은 Dmodulename.Mod와 같이 D라는 첨자로 시작한다. 그리고 모듈의 구현 부분을 나타내는 화일은 Imodulename.Mod와 같이 I라는 첨자로 시작한다.

3. MODSIM II에서 객체지향 시뮬레이션 구도

3.1 객체 인스탄스의 활동과 메소드

MODSIM II에서 객체 인스탄스는 다중 활동, 동시 활동 들을 가질 수 있다. 여기서 활동이란 활동 개시와 활동 완료를 이루는 두 가지 사건을 동반하고 활동 개시부터 완료까지 시간 경과가 요구된다. 활동은 조건 활동(conditional activity)과 묶인 활동(bounded)으로 구분된다. 조건 활동은 그 활동의 실행을 위해서 자원이 필요한데 아직까지 자원이 충족되지 않은 활동을 의미하고, 묶인 활동은 그 실행을 위하여 자원이 충족되어 스케줄링이 가능한 활동을 의미한다. 그런데 자원이 필요하지 않는 활동은 자원의 가용 여부에 관계 없이 스케줄링이 가능하게 된다.

객체 인스탄스의 활동과 활동 스케줄링은 그 객체형의 메소드에서 정의된다. 일반적으로 객체형의 메소드는 3가지 종류 즉, ASK 메소드, TELL 메소드, 그리고 WAITFOR 메소드가 있다. 이들은 시뮬레이션 시간 경과 처리와 관련하여 중요한 차이가 있다.

ASK 메소드는 호출시에는 절차(PROCEDURE)에서 시행하는 것과 같은 방식으로 시행한다. 즉, ASK 메소드가 호출되어 시행될 때는 ASK 메소드에서 정의된 방식대로 수행을 완료한 후에 다음 문장을 수행하게 된다. 이때, 호출후 값을 반환할 수 있다. 그런데 시뮬레이션과 관련하여 ASK 메소드는 시뮬레이션 시간의 경과(예를 들어 활동에 필요한 시간 경과)를 구현하지 못하고 임의 순간에 발생하는 행위를 처리하게 된다. 그래서 ASK 메소드에서는 동기(synchronous) 호출에 대하여 응답할 수 있는 방식만을 정의할 수 있다.

TELL 메소드는 ASK 메소드와 달리 시뮬레이션 시간의 경과(예를 들어 활동에 필요한 시간 경과)를 구현할 수 있다. 그래서 TELL 메소드를 통하여 임의 시간 간격 동안에 발생하는 활동과 그 스케줄링 방식을 정의한다. TELL 메소드가 호출될 때 ASK 메소드와는 달리 TELL 메소드에 정의된 방식대로 수행이 완료될 때까지 기다리지 않고 곧 다음 문장을 수행하게 된다. 이는 TELL 메소드에 정의된 활동 시간에 따라 스케줄링하고 곧장 다음 문장을 수행하며, 시뮬레이션 시각이 스케줄된 시간으로 전진하면 그 활동이 완료되어 TELL 메소드 호출도 완료된다는 의미이다. 그래서 TELL 메소드는 비동기(asynchronous) 호출에 대하여 응답할 수 있는 방식을 정의할 수 있다. 이러한 이유로 TELL 메소드는 호출후 값을 반환할 수 없다.

한편, WAITFOR 메소드는 ASK 메소드와 TELL 메소드의 특성을 모두 가지고 있다. 그래서 ASK 메소드처럼 호출후 값을 반환할 수 있고, TELL 메소드처럼 시뮬레이션 시간의 경과를 구현할 수 있다.

그러나 WAITFOR 메소드의 사용은 WAIT FOR 문에 제한되어 있다.

3.2 시뮬레이션 구성 요소

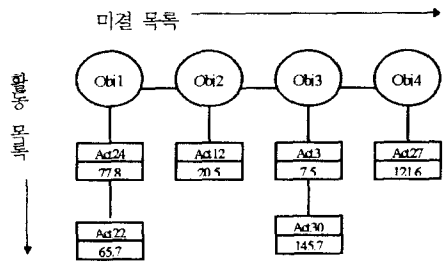
MODSIM II의 시뮬레이션 구성 요소로서 TELL 메소드, 미결 목록(pending list), 시간 루틴(time routine), 시뮬레이션 시각(simulation time)이 있다.

TELL 메소드는 객체형에서 그 객체형에 의해 생성될 객체 인스탄스의 활동들(프로세서)를 정의한다. 이를 위하여 WAIT 문장을 사용하는데, WAIT 문장을 통해 임의 시뮬레이션 시간이 경과하도록 하거나 임의 조건이 충족될 때까지 시뮬레이션 시간을 경과하도록 함으로써 무조건부 활동이나 조건부 활동을 구현한다. 그래서 TELL 메소드내의 WAIT 문장이 곧 활동이라고 할 수 있고, 다수의 WAIT 문장을 사용하여 여러개의 활동을 나타낼 수 있다. 시뮬레이션간에 TELL 메소드를 호출할 때 WAIT 문을 접하면 TELL 메소드는 실행을 중지(suspend)하고 WAIT문에 지정된 시간(시뮬레이션 시간)이 경과된 후에 TELL 메소드는 실행을 재개하게 된다. 그런데 WAIT 문에 의해 경과 처리가 되는 동안에 TELL 메소드의 실행이 재개되기 이전에 경과 처리를 중지(interrupt) 시키고자 할 때는 Interrupt(객체 인스탄스명, TELL 메소드 명)문장을 사용한다. 시뮬레이션간에 이 문장을 접하면 시뮬레이션 시간 경과 처리를 중지하게 되는데, 중지후 어떻게 할 것인가는 해당 WAIT 문의 ON INTERRUPT 부분에서 정의해 주어야 하는데, 일반적으로 TERMINATE 문장을 사용하여 TELL 메소드 호출을 중지시킨다. TELL 메소드의 호출은 그의 객체 인스탄스에게 관련 메세지를 보냄으로써 이루어 지는데, 일반적으로 메소드명이나 메세지 명은 같이 사용된다.

미결 목록은 자신의 활동을 스케줄링하고 있는 객체 인스탄스들의 순서 목록(ordered list)이다. 이는 이산 사건 시뮬레이션의 사건 목록과 비슷한데, 객체 인스탄스별로 스케줄링 시간을 관리하는 것이 다르며,<그림 1>과 같은 목록 형태를 지니고 있다.

예를 들어 객체형인 ObjType 과 그의 TELL 메소드 *move(x,y)*를 고려하고 *move(x,y)*는 임의 시간 경과후 (x,y)로 이동하도록 정의하며, 객체 인스탄스 *object1*이 ObjType 의 객체 인스탄스라고 한다면, *move(x,y)*는 *object1*의 활동이라고 할 수 있고 이 활동은 *object1*에*move(x,y)*라는 메세지를 보냄으로써 스케줄링 되게 된다.

객체 인스탄스 *object1*에 메시지 *move(x,y)*를 전달하고자 할 때는 TELL *object1* TO *move(x,y)*형태



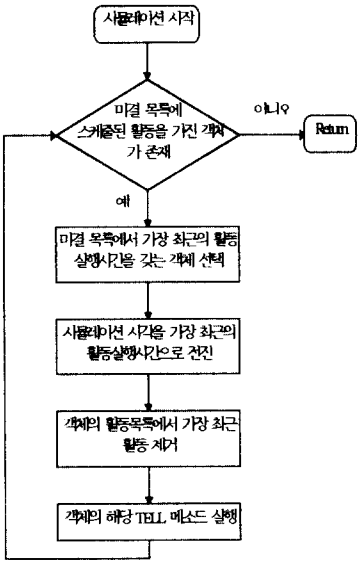
〈그림 1〉 MODSIM II의 미결목록

의 문장을 사용하는데, 이 문장을 접하면 객체 인스탄스 *object1*의 참조자(*reference*)를 미결 목록에 넣고, *move(x,y)*의 참조자를 객체 인스탄스의 활동 목록에 스케줄한 다음에 다음 문장을 수행하게 된다. 그러면 스케줄된 시간에 *object1*의 객체형의

TELL 메소드(여기서 메소드 명은 *move(x,y)*)에 정의된 활동을 완료하게 된다.

시간 루틴은 미결 목록과 활동 목록에서 가장 최신의 활동을 찾아내어 실행시키는 역할을 하는데, 시간 루틴의 처리 순서는 <그림 2>와 같다.

한편,시뮬레이션 시각은 MODSIM II에서 자동적으로 관리한다.현재 시뮬레이션 시각은 SimTime() 이라는 함수를 호출함으로서 얻을 수 있다. 시뮬레이션 시각은 그 시각에 스케줄된 활동이 완료된 후에 다음 스케줄된 시각으로 전진한다. 여기서 시뮬레이션 시각은 차원이 없다. 그래서 사용자에 따라 초, 분, 시간, 일 등 임의로 정의할 수 있다.



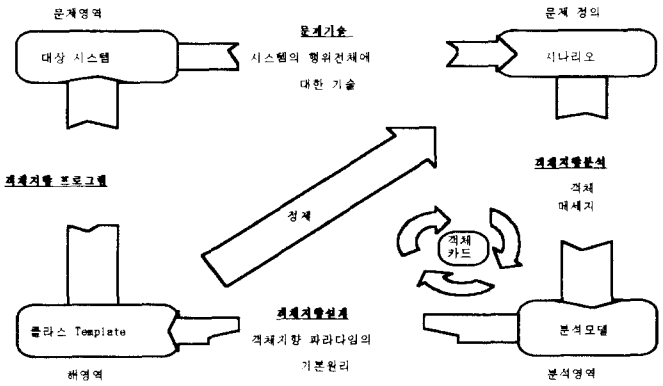
<그림 2> MODSIM II 시간 루틴

4. 객체지향 시뮬레이션 소프트웨어 개발을 위한 SMG 방법

객체지향 응용 소프트웨어 개발은 문제 영역에서부터 소프트웨어 구현에 이르기까지 객체지향 패러다임으로 일관되고, 소프트웨어 개발 과정에서 객체의 추출, 객체 관계 설정, 클라스 설계 등이 반복적으로 이루어져 정제화(refinement)되기 때문에 자연적으로 점진적 방법으로 개발하는 것이 바람직하다. SMG(System Modeling Group) 방법은 기존의 연구[6,8,9,12]를 바탕으로 객체지향 시뮬레이션 소프트웨어 개발에 적합할 수 있도록 국방대학원 무기체계과 시스템 모델링 그룹(System Modeling Group,SMG)에 의해 고안 발전된 방법[2]이다. SMG 방법은 시뮬레이션 소프트웨어 개발과정에서 객체, 메세지, 클라스를 도식화 하고, 이를 사용하여 개발하고자 하는 시뮬레이션 시스템의 정적 구조 및 동적 과정을 여러가지 분석 모형으로 나타내도록 하였다. 그래서 이를 바탕으로 시뮬레이션 클라스를 설계 구현하고 시뮬레이션 소프트웨어를 구성하도록 하였다.

특히, SMG 방법은 객체지향 소프트웨어 개발 절차를 <그림 3>과 같이 문제기술(problem description), 객체지향 분석(object-oriented analysis), 객체지향 설계 (object-oriented design), 객체지향 프로그램(object-oriented programming) 으로 나누어 진다.

문제기술 단계에서는 시뮬레이션 대상 시스템에 대한 행위를 기술하고 이를 시나리오 형태로 나타낸다. 그리고 객체지향 분석 단계에서는 객체, 메세지 식별과 그들의 관계성을 파악한다. 그래서 이를 객체/메시지 도표와 메세지 흐름도로 나타내어 시뮬레이션 대상 시스템에 대한 초보적인 정적 구조와 동적 구조를 분석하도록 한다. 객체지향 설계단계에서는 분석 단계에서 작성한 객체/메세지 도표와 메세지 흐름도를 각각 객체관계 모형과 상태전이 모형으



〈그림 3〉객체지향 소프트웨어 개발

로 발전시켜, 이 들을 근거로 객체형의 필드와 메소드를 추출하고 객체지향 파라다임의 기반요소를 이용해 객체형을 설계하고 구현한다. 객체지향 프로그램 단계에서는 설계된 객체형을 사용하여 객체 인스탄스를 생성하고 메세지 전달을 적절히 코드화 해줌으로써 소프트웨어가 완성된다. 그런데 객체지향 분석단계와 설계 단계는 거의 같은 단계로 이루어지며, 이들 단계를 반복적으로 수행하여 정제화(refinement)를 통해 최종적으로 객체형을 설계하게 된다. 한편, 설계 과정에서 객체 카드를 사용하여 분석과 설계 과정을 통해 파악한 객체 정보를 명시함으로써 객체형을 설계하고 구현하는데 도움을 주도록 한다. 그런데 소프트웨어가 크지 않을 때는 객체 카드를 사용하지 않아도 된다.

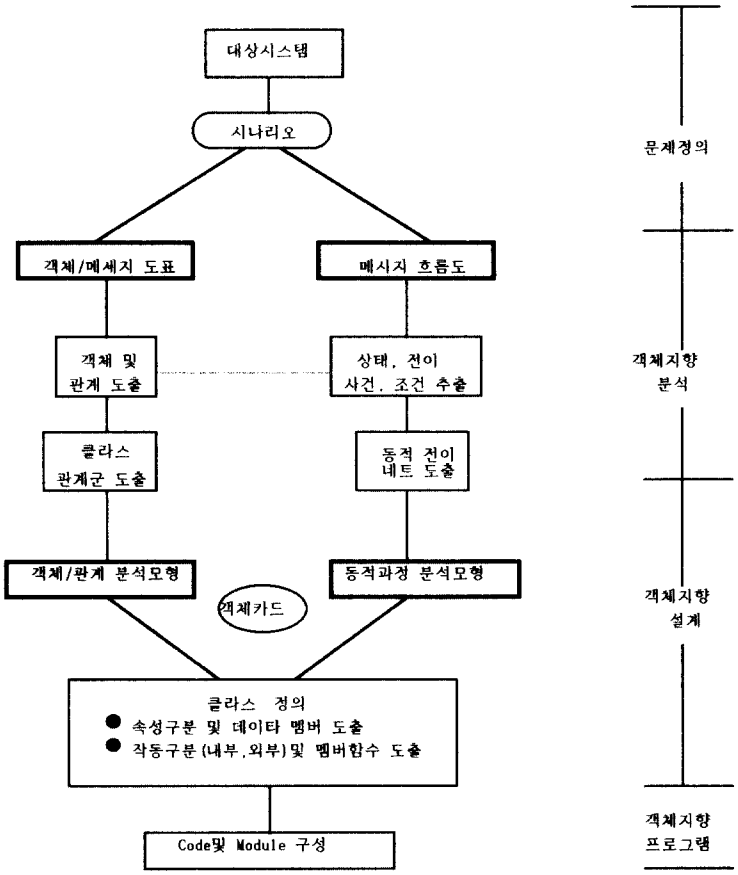
SMG 방법에의한 객체지향 시뮬레이션 소프트웨어 개발 절차는 <그림 4>와 같다.

5. 전차 교전 시뮬레이션 소프트웨어 프로토타입 개발 예

5.1 문제 기술

MODSIM II 언어를 사용한 군사 전차 교전 시뮬레이션 소프트웨어 프로토타입 개발 예를 보이기 위하여 지상 전차 교전을 문제 영역으로 설정하고 다음 사항이 모의 되도록 하였다.

- ○전차 교전 과정은 기동, 탐지, 사격 준비 및 발사, 명중으로 이루어지도록 한다.
- ○기동은 교전간 회피 기동으로 간주하고 기동 간격은 전차별로 다를 수 있고, 탐지 시간, 사격 준비 및 발사 시간, 명중 확률도 전차 별로 다를 수 있다.



〈그림 4〉 SMG 방법

○표적은 무작위로 선정되며, 중복 사격을 허용한다.

○청군 전차와 홍군 전차는 위의 교전 과정을 거치면서 교전하는데, 최초 공격 개시선에서 교전을 개시하여 청군 전차 혹은 홍군 전차가 목표선에 도달하면 교전이 완료된다.

○교전 전장은 전장지도(raster 지도)를 배경으로 하고 교전간 전차의 기동, 탐지, 사격 준비 및 발사, 명중을 에니메이트(animate)하며, 필요에 따라서 메뉴바와 대화 상자를 이용하여 이들과 관련된 파라메타를 입력할 수 있도록 한다.

○에니메이션 수준은 전차가 실제 교전시 Turret을 회전하여 표적물에 정치하고 Gun으로 사격하는 것까지 상세히 묘사될 수 있도록 한다. 또한 사격시에는 포화가 생길 수 있도록 하며, 파괴시에는 파괴 잔해가 나타날 수 있도록 한다.

○시뮬레이션을 위한 입력 파라메터(회피기동거리,

사격준비 및 발사시간, 탄 비행시간, 명중확율, 기동속도, 교전 전차 대수)는 메뉴바와 대화박스를 통해 입력함으로써 사용자 인터페이스를 향상시킨다.

5.2 객체 지향 분석

전차 교전 시뮬레이션 소프트웨어 개발을 위하여 분석 단계에서는 객체와 메세지를 식별하여 객체/메세지 도표로 나타내고, 메세지 흐름도를 작성하여 시뮬레이션의 동적 과정을 분석한다.

여기서 객체을 식별하는 데 있어서 에니메이션 효과를 나타내는 것을 고려하여 객체를 실제 전차 교전에서 실 개체와 대응시켜 식별할 수 있다. 그래서 다음과 같은 객체로 식별한다.

먼저, 시뮬레이션을 통제하는 통제 객체로써 aController를 둔다. 여기서 교전간 청군 전차와 홍군 전차를 생성하도록 한다. 그리고 교전의 기본 단위인 전차를 객체로 식별하되 청군 전차를 aBlueTank 객체로 대응시키고 홍군 전차를 aRedTank 객체로 대응 시킨다. 한편, 각 전차의 Turret과 Gun을 각각 aTurret 객체와 aGun 객체로 둔다. 그리고 에니메이션 효과를 내기 위하여 전차 잔해와 폭음을 각각 aExplosion 객체와 aSmoke 객체로 나타낸다.

그리고 이들 객체에 대하여 메세지를 정의하는데, 이는 <표 3>과 같다.

한편, 전차 교전에 대한 동적 과정을 분석함이 필요한데, 이는 아래 <그림 5>의 메세지 흐름도를 사용하여 분석할 수 있다.

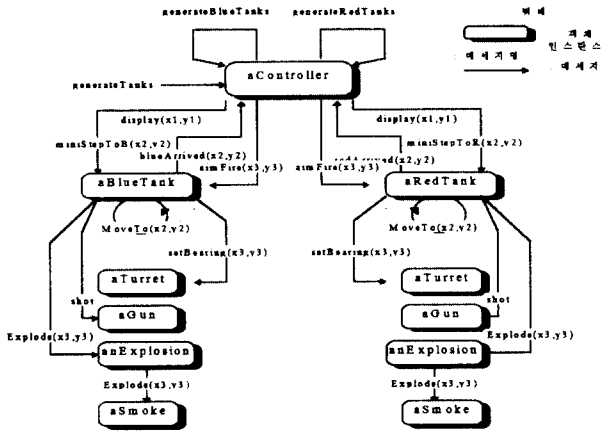
먼저, aController 객체에 generateTanks 메세지를 보내면, aController 객체에서 자체의 generate BlueTanks 메소드와 generateRedTanks 메소드를 동시에 호출하여 aBlueTank 객체와 aRedTank 객체를 생성하여 그들의 참조자를 버퍼에 저장한다. 그리고 aBlueTank 객체와 aRedTank 객체에게 각각 최초(x1,y1)지점에 전시 되도록 하고 , miniStepToB (x2,y2)와 miniStepToR(x2,y2)라는 메세지를 보낸다. 그러면 자체 MoveTo(x2,y2) 메소드를 호출하여 (x2,y2)지점으로 회피 기동하는 것이 에니메이트 되도록 한다. 물론 (x1,y1), (x2,y2)는 청군 전차와 홍군 전차에 따라 다를 수 있다.

(x2,y2)지점에 도착하면 청군 전차의 경우 blueArrived(x2,y2) 라는 메세지, 그리고 홍군 전차의 경우 redArrived(x2,y2)라는 메세지를 aController에게 보낸다. 그러면 aController에서 버퍼를 참조하여 상대방의 전차를 표적으로 선정하도록한 후에, 각 전차에게 aimFire(x3,y3)라는 메세지를 전달하여 그 표적을 제압할 수 있도록 한다. 여기서 (x3, y3)는 표적의 위치가 되는데 청군 전차의 경우는 홍군 표적 전차의 위치가 되고, 홍군 전차의 경우는 청군 표적 전차의 위치가 된다.

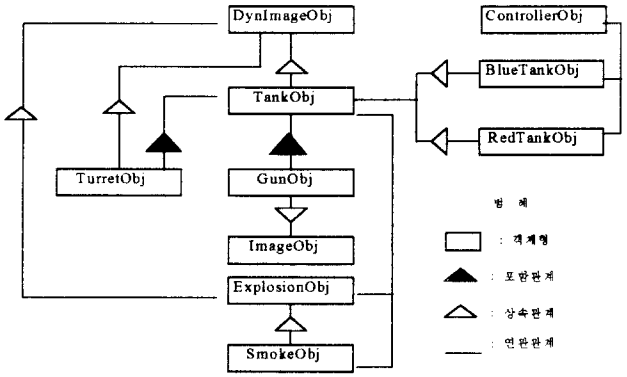
각 전차가 aimFire(x3,y3)라는 메세지를 전달 받으면 자신의 aTurret 객체에게 setBearing(x3,y3) 라는 메세지를 전달하여 Turret 화상 객체를 표적 위치 (x3,y3)로 선회하도록 에니메이션 한다. 그후 자신의 aGun 객체에게 shot이라는 메세지를 전달하여 사격을 실시하도록 한다. 그리고 임의 탄 비행시간후에 제압시에는 anExplosion 객체에게 explode(x3,y3) 메세지를 전달하여 잔해가 남도록 하고 동시에 aSmoke 객체에게 explode(x3,y3) 메세지를 전달하여 포화 효과가 나타나도록 한다.

객체	메세지	메세지 내용
aController	generateTanks	generateBlueTanks 메시지와 generateRedTanks 메시지를 동시에 호출하여 aBlueTank, aRedTank객체 생성
	generateBlueTanks	aBlueTank 객체 생성, 버퍼에 저장
	generateRedTanks	aRedTank 객체 생성, 버퍼에 저장
	blueArrived(aBlueTank)	aBlueTank가 회피 기동을 완료한후에 표적을 획득
	redArrived(aRedTank)	aRedTank가 회피 기동을 완료한 후 표적을 획득
	displayBlueDebris(x,y)	aBlueTank가 제압되어 잔해를 전시
	displayRedDebris(x,y)	aRedTank가 제압되어 잔해를 전시
aBlueTank	displayAt(x1,y1)	(x1,y1)에 aBlueTank 화상개체 표시
	miniStepToB(x2,y2)	현재 위치에서 aBlueTank 화상개체를 (x2,y2)로 이동
	aimFire(x3,y3)	표적 위치(x3,y3)로 사격
	returnTurret	Turret을 최초 위치로 이동
aRedTank	displayAt(x1,y1)	(x1,y1)에 aRedTank 화상개체 표시
	miniStepToR(x2,y2)	현재 위치에서 aRedTank 화상개체를 (x2,y2)로 이동
	aimFire(x3,y3)	표적 위치(x3,y3)로 사격
	returnTurret	Turret을 최초 위치로 이동
aTurret	setBearing(x,y)	현재의 Turret위치에서 (x,y) 방향으로 회전
aGun	shoot	표적에 대한 사격
aExplosion	explode(x,y)	(x,y)에 명중 폭팔/폭음
aSmoke	explode(x,y)	(x,y)에 포화 전시

〈표 3〉객체/메세지 도표



〈그림 5〉 메시지 흐름도



〈그림 6〉객체 관계 모형

한편, aController 객체에 generateTanks 메세지를 교전 시나리오에 맞게 주기적으로 보냄으로써 이 과정을 반복하게 된다.

5.3 객체지향 설계

설계 단계에서는 분석 단계에서 작성한 객체/메세지 도표와 메세지 흐름도를 참고하여 객체 관계 모형과 동적과정 분석모형을 작성한다. 그리고 이들 모형을 바탕으로 전차 교전 시뮬레이션을 위한 객체형을 설계한다.

5.3.1 객체관계 모형과 상태전이 모형

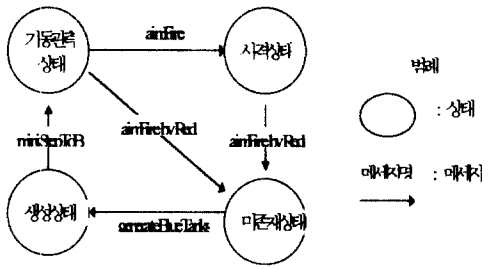
객체관계 모형은 객체/메세지도표를 사용하여 유사 객체들을 하나의 객체형으로 만들고 객체간에 IS_A 관계는 상속관계로, IS_A_PART 관계는 포함관계로, 그리고 IS_USING을 포함한 기타 관계를 연관관계로 설정하여 객체 계층 구조를 만드는데, 이는 <그림 6>과 같다.

여기서, 각 객체에 대한 객체형은 뒤첨자에 Obj를 붙여서 구분하였다. DynImageObj와 ImageObj는 애니메이션과 이메지 처리를 위하여 MODSIM II에서 제공되는 객체형이다. 여기에는 에니메이션을 위한 각종 메소드가 정의되어 있다. 그래서 먼저, TankObj라는 객체형을 만들어 DynImageObj객체형으로 부터 상속받아 에니메이션 기능을 상속받고, 동시에 TurretObj 객체형과 GunObj 객체형을 포함하도록 한다. 또한 TankObj 객체형은 ExplosionObj 객체형과 SmokeObj 객체형을 연관관계로 유지하여 필요에 따라 TankObj 객체형에서 일시적으로 사용할 수 있도록 한다. ExplosionObj 객체형은

DynImageObj 객체형으로 부터 상속받아 에니메이션 기능을 상속받도록 한다. 그리고 SmokeObj 객체형은 ExplosionObj 객체형으로 부터 상속받도록 한다. 한편, BlueTankObj 객체형과 RedTankObj 객체형은 TankObj 객체형으로부터 상속받도록 한다. 그리고 ControllerObj 객체형은 BlueTankObj 객체형과 RedTankObj 객체형을 연관관계로 두어 ControllerObj 객체형에서 이들 객체를 사용할 수 있도록 한다.

상태전이 모형은 각 객체형마다 다르게 나타내는데, 예를 들어 aBlueTank에 대한 상태전이 모형은 <그림 7>과 같다.

최초 aBlueTank는 미존재상태에서 generate-BlueTanks 메세지를 받아 메모리에 생성되고 miniStepToB 메세지를 받아 기동관측 상태로 전이한다. 그리고 aimFire메세지를 받으면 사격상태로 전이되고, aRedTank에의해 aimFire메세지를 받아 제압당했을 때 미존재 상태가 되어 메모리에서 제거된다.



<그림 7>상태전이 모형

5.3.2 객체형 설계

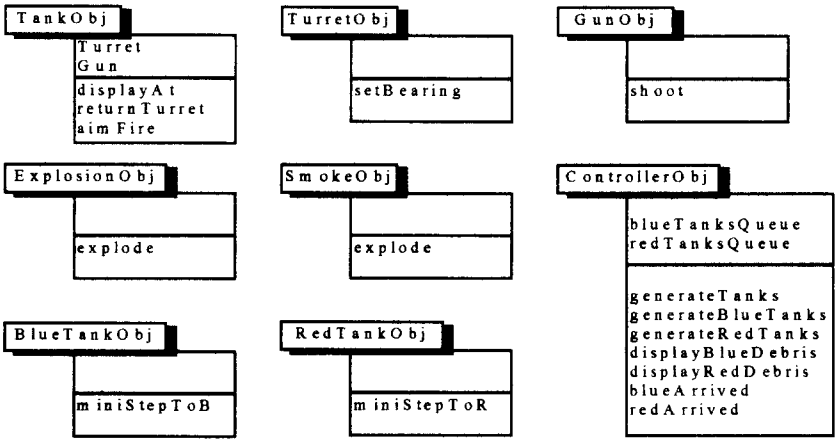
객체관계 모형과 상태전이 모형으로부터 객체형을 설계하는데, 객체형에 대한 구분은 객체 관계모형으로 부터 가능하고, 이 모형과 상태전이 모형을 참고하여 객체형의 필드와 메소드를 정의할 수 있다.

객체 관계 모형으로부터 객체형을 ControllerObj, TankObj, BlueTankObj, RedTankObj, TurretObj, GunObj, ExplosionObj, SmokeObj로 구분한다. 그리고 이들에 대한 필드와 메소드를 정의하여 객체형을 설계할 수 있는데, 지면의 한계상 주요 필드와 메소드를 중심으로 설계된 객체형을 형틀(template)로 나타내면 <그림 8>과 같다.

여기서 각 Template의 Tag은 객체형의 이름이고 위의 박스에 있는 것은 필드이고 아래 박스에 있는 것은 메소드를 나타낸다. 메소드를 나타낼때는 ASK 메소드, TELL 메소드, 그리고 WAITFOR 메소드로 구분하여 나타낼 수 있다.

5.4 객체지향 구현

전차 교전 시뮬레이션 프로그램은 7가지의 모듈로 구성하였다. 먼저 DTank.Mod와 ITank.Mod에서는 TankObj, ExplosionObj, SmokeObj, BlueTankObj, 그리고 RedTankObj 객체형에 대한 정의와 구현을 했고, DController.Mod와 Dcontroller.Mod 에서는 ControllerObj에 대한 정의와 구현을 했다. 그리고 Dmenu.Mod와 Imenu.Mod에서는 시뮬레이션 파라메터를 대화식으로 입력 할 수 있는 MenuObj에 대한 정의와 구현을 했다. 마지막으로 MTank-Engagement. Mod는 메인 프로그램으로 하였는데,



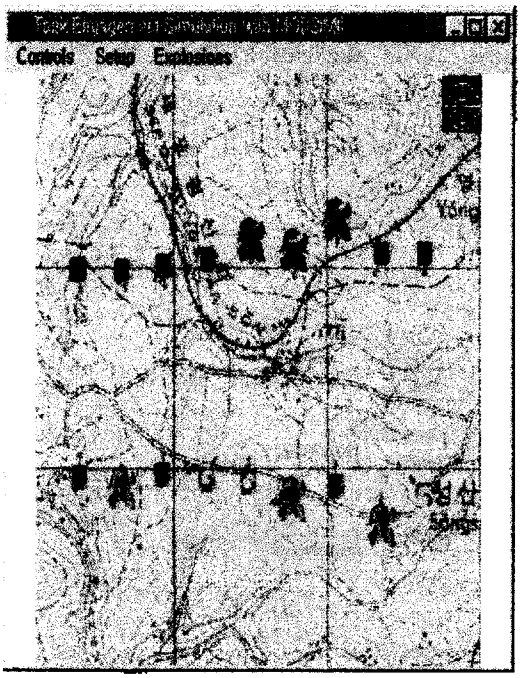
<그림 8>객체 형틀

여기서 화면 배경 지도를 포함하여 에니메이션을 위한 화상 객체를 정의하고 구현했다. 이들에 대한 구체적인 프로그램은 생략한다.

5.5 실행 예

전차 교전 시뮬레이션 프로그램 구성은 1개의 메인 프로그램과 6개의 정의 화일과 구현 화일로 이루어 있는데 이를 컴파일한 후에 실행하면 시뮬레이션이 실행되며, <그림 9>는시뮬레이션이 실행되고 있는 화면을 캐치한 것이다.

여기서 시뮬레이션이 개시되면 위쪽에는 홍군 전차가 아래쪽에는 청군 전차가 나타난다. 시뮬레이션이 진행됨에 따라 각 전차는 중앙의 목표선으로 회피 기동을 하면서 전진하고 상대방의 전차와 교전



<그림 9>실행 예

한다. 교전간에는 표적물을 발견시 표적 방향으로 Turret을 회전하여 표적물에 Gun을 정치하여 사격하고, 사격시에는 포화가 발생하며, 파괴시에는 파괴잔해가 나타난다. 그리고 교전간 어느 한쪽이 전멸되거나 어느 한 쪽의 전차가 중앙 목표선을 먼저 탈취하면 시뮬레이션은 종료된다.

6. 결 론

본 연구에서는 MODSIM II 환경에서 객체지향 시뮬레이션 소프트웨어를 개발하는데 SMG 방법을 사용한 예를 보이고 SMG 방법의 적용성에 대하여 연구하였다. SMG 방법은 기존의 연구를 토대로 일반적인 객체지향 소프트웨어 개발 방법과 같이 시뮬레이션 개발 단계를 문제기술, 객체지향 분석, 객체지향 설계, 객체지향 프로그램 단계로 나누고 시뮬레이션 대상 시스템에 대한 정적 구조 및 동적 구조를 나타내어 시뮬레이션 클라스 설계와 구현을 용이하게 하고 있다.

그래서 본 연구를 통하여 볼 때 SMG 방법은 MODSIM II 환경에서 객체지향 시뮬레이션 소프트웨어를 개발할 때 충분히 적용될 수 있음을 알 수 있었다. 그러나 분석 모형에서 일부 메소드, 예를 들면 ASK 메소드, TELL 메소드 등에 대해 표시하는 데에는 한계가 있었다. 이러한 문제는 Bailey 교수가 제안한 OOSPICs 방법[5]에서 객체 인스탄스와 객체형, 그리고 각종 메소드를 표시하는 방법을 접목 시킨다면, SMG 방법은 일반적인 객체지향 시뮬레이션 소프트웨어 개발 방법으로서 뿐만 아니라 MODSIM II 종속 개발 방법으로도 사용될 수 있을 것으로 사료된다. 이에 대해서는 추가적인 연구가 필요하겠다.

참고 문헌

1. 정창성, 최상영, 김성식, 동적 시스템을 위한 고속 객체지향 시뮬레이션에 관한 연구(보고서), 과학재단, 1996.6.
2. 최상영, 객체지향 기법의 시스템 모델 개발 적용(보고서), 국대원, 1993.9.
3. 최상영, 군사 시뮬레이션 모델링 기술(강의노트), 국대원, 1996.8
4. 최상영, Object-Oriented Techniqueue and its Application to the Development of the Combat System Simulation Model LADEM, 한국 군사운영분석 학회지, 20권, 1호(1994), pp.138-156.
5. Bailey, M., Object-Oriented Simulation Pictures for Design and Testing, ---.
6. Booch , G ., Object - Oriented Design, The Benjamin/Cunning Publishing Company Inc., 1991.
7. Duncan C.M., SIMNET:The Advent of Simulator Networking, Proceeding of the IEEE, Vol.83, No.8, August 1995, pp.1114-1123.
8. Embley, D.W., et al., Object-Oriented System Analysis, Prentice Hall, Englewood Cliffs, N.J., 1992.
9. Jacobson, I., et al., Object-Oriented Software Engineering, Addison-Wesley Publishing Company, 1992.
10. Jones, J.G., Modeling and Simulation in MODSIM II(Course Notes), CACI Products Company.
11. Lalit K.P. et al., System Acquisition Managers

Guide for the Use of Models and Simulations, DSMC, USA, 1994.

12. Rumbaugh,J., et al., Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, N.J.,1991.
13. MODSIM II : The Language for Object-Oriented Programming(Users Manual), CACI Products Company.
14. MODSIM II : The Language for Object-Oriented Programming(Reference Manual), CACI Products Company.
15. SIMGRAPHICS II : Users Manual for MODSIM II, CACI Products Company.