한국시뮬레이션학회 논문지
Vol. 15, No. 4, pp. 41-50 (2006. 12)

ASiM : 에이전트 시뮬레이터(1)

송병권[1†]

RRSiM : Agent SiMulator(1)

Byungkwen Song

ABSTRACT

The agent system for managing various equipments constituting communication networks is usually started to be developed after completing real resources or can be developed using network management platform independent from real resources. In this paper, we design real resource simulator which can perform in advance the function of real resources without real resources. The real resource simulator performs the role of automatically generating specific attribute value and informing it according to the specified random function. Moreover, in this paper, we design E-GDMO grammar in which the grammar for simulation is added to the existing GDMO grammar and its E-GDMO compiler to perform real resource simulator.

Key words : Network Management, Real Resource Simulator, Agent, GDMO, E-GDMO

요 약

통신망을 구성하는 각종 장비들을 관리하기 위한 에이전트 시스템은 보통 실제자원이 완성된 후에 개발을 시작하거나, 실제자원에 독립적인 망 관리 플랫폼을 이용하여 개발할 수 있다. 본 논문에서는 이와 같은 방법을 따르지 않고 실제자원이 없는 단계에서도 미리 실제자원의 역할을 수행할 수 있는 실제자원 시뮬레이터를 설계한다. 실제자원 시뮬레이터는 특정 애프리뷰트의 값과 통고를 지정된 random 함수에 따라 자동으로 발생시키는 역할을 수행한다. 또한 본 논문에서는 실제자원 시뮬레이터의 동작을 수행하기 위해 기존 GDMO 문법에 시뮬레이션 문법을 추가한 E-GDMO 문법과, E-GDMO 컴파일러를 설계한다.

주요어 : 망 관리, 실제자원 시뮬레이터, 에이전트, GDMO, E-GDMO

1. 서 론

정보화 사회의 기반인 통신망은 사회적인 요구에 따라 이종의 시스템들이 함께 나타나는 상황으로 점점 복잡하게 진화되고 있으며, 통신 서비스는 기존의 음성 서비스에서 데이터, 화상, 이동 컴퓨팅 서비스 등 미래의 환경을 갖춘 서비스로 점점 고도화, 다양화되고 있다. 이러한 통신망 환경에 대응하여 개별 운용시스템의 한계성을 극복하고 표준화된 개방형 방식의 총체적이고 일원화된 통신망 운용관리체제를 구축하기 위해 국제 전신 표준화 기구인 ITU-T가 M.3010 등의 권고안에서 TMN(Telecommunication Management Network) 관리 개념을 도입했다.

TMN은 관리 정보 모델링을 위해 OSI(Open System Interconnection)에서 제안한 객체지향 모델링 기법을 채택하고 있고, 관리 정보의 접근 및 교환을 위하여 OSI의 CMIP(Common Management Information Protocol)/CMIS(Common Management Information service)[1] 프로토콜을 채택하고 있다.

TMN은 망 관리 응용에 따라 관리자(manager)와 에이전트(agent)로 구분된다. 관리자는 에이전트에게 CMIP/CMIS 프로토콜을 이용하여 관리 요청을 보내고 에이전트로부터 관리 요청에 대한 수행 결과를 반환 받거나, 에

2005년 10월 11일 접수, 2006년 10월 24일 채택
[1] 서경대학교 정보통신공학학과
주 저 자 : 송병권
교신저자 : 송병권
E-mail; bksong@skuniv.ac.kr

이전트로부터 특별한 사건이 발생했음을 알리는 통고(notification) 메시지를 수신한다. 그리고 에이전트는 관리자로부터 관리요청을 수신한 다음 실제자원(real resource)에 접근하여 해당 값을 가져온 후, 그것을 관리자에게 반환한다. 따라서 관리자는 에이전트를 통하여 실제자원에 대한 관리를 종합적으로 수행한다.

실제자원은 전기통신망을 구성하는 교환기 및 각종 유무선 통신 장비 등 실제 관리하고자 하는 하드웨어 또는 소프트웨어적 요소다. 이러한 실제자원을 관리하기 위한 망 관리 시스템은 실제자원이 완성된 후에 개발을 시작하거나, 실제자원과는 독립적으로 개발할 수 있다. 전자의 경우는 실제자원이 완성되어 기존 통신망에 접속했을 경우, 해당 장비에 대한 관리 시스템이 완료되기까지 관리 공백이 초래되고, 후자 또한 개발이 완료된 실제자원과의 인터페이스 및 테스트를 위하여 부가적인 작업이 추가로 요구된다. 따라서 망 관리 시스템의 개발 기간을 단축시키고, 실제자원이 완성되기 전에 독자적으로 개발된 관리 시스템의 동작 상태를 검증하기 위해서는 실제자원의 기능을 대신할 수 있는 실제자원 시뮬레이터(simulator)가 필요하다.

현재 TMN에 관련된 연구는 이미 개발된 플랫폼(platform)을 이용하거나 독자 개발된 것을 사용하여 특정 통신망 환경에서 구동되는 관리 시스템의 기능이나 구현 메커니즘에 관한 것이 대부분이며,[11-14] 개발이 완료된 망 관리 시스템의 안정성이나 동작을 검증하고 또한 가상의 실제 자원을 이용하여 망 관리 시스템을 조기 개발하는 분야에 대해서는 연구결과를 찾기 어렵다.

본 논문의 구성은 다음과 같다. 우선 2장에서 기존에 개발된 플랫폼들에 대하여 기술하고, 3장에서는 실제자원 시뮬레이터를 설계한다. 4장에서는 확장된 GDMO(E-GDMO) 컴파일러를 설계하고, 5장에서 결론을 맺는다.

2. 실제자원 시뮬레이터 설계

2.1 실제자원 시뮬레이터 기능

본 논문에서 제안한 실제자원 시뮬레이터는 실제자원의 동작을 예상하여 그 행동을 시뮬레이션(simulation)함으로써, 실제자원의 역할을 대신한다. 이를 위해 실제자원 시뮬레이터는 실제자원의 역할을 표현하는 데이터를 입력으로 받아들이게 되는데, 이것은 관리객체(managed object)를 표현하는 표준 GDMO(Guidelines for the Definition of Managed Objects)[2]에 시뮬레이션 신택스(syntax)를 포함하도록 설계된 E-GDMO에서 제공하게 된다. E-GDMO 문법과 E-GDMO 컴파일러는 4장에서 기술한다.

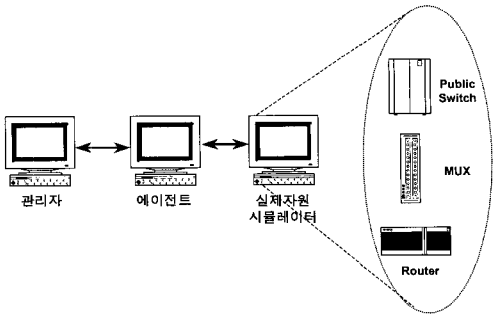

그림 1. 관리자, 에이전트, 그리고 실제자원 시뮬레이터

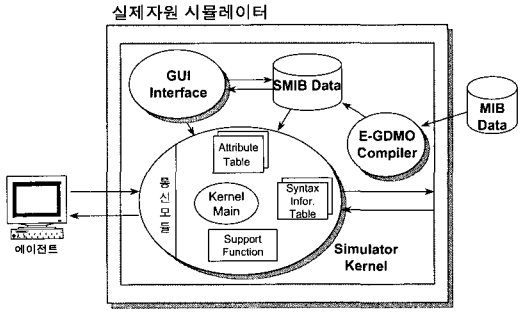

그림 2. 에이전트 시뮬레이터의 전체 구조

따라서 실제자원 시뮬레이터를 이용하는 사용자는 E-GDMO 문법에 따라 에이전트에서 요구하는 데이터 값을 마치 실제자원에서 생성한 것처럼 시뮬레이션할 수 있다. 이러한 실제자원 시뮬레이터에서 생성되는 데이터 값은 사용자로부터 실제자원의 특성을 파악하여 정의된 확률분포에 따라 생성됨으로써, 망 관리 시스템 개발자는 실제자원이 없이도 실제자원 시뮬레이터를 이용하여 해당 시스템을 개발하거나 개발된 시스템을 검증할 수 있다. 그림 1은 관리자, 에이전트 그리고 본 연구에서 설계한 실제자원 시뮬레이터와의 관계를 나타낸다.

2.2 실제자원 시뮬레이터의 전체 구조

그림 2는 실제자원 시뮬레이터의 전체 구조를 나타낸다. 실제자원 시뮬레이터는 시뮬레이터 커널, GUI 인터페이스, SMIB(MIB for Simulator) 데이타, 그리고 E-GDMO 컴파일러 등 4 개의 부분으로 구성된다.

실제자원 시뮬레이터는 E-GDMO 문법으로 정의된

MIB 데이타를 E-GDMO 컴파일러를 통하여 실제자원 시뮬레이터에서 사용될 수 있는 데이타 구조인 SMIB로 변환하고, 사용자는 GUI 인터페이스를 통해 SMIB에 저장되어 있는 애트리뷰트(attribute) 값들을 초기화하고, 각 애트리뷰트 단위로 시뮬레이션중에 발생할 support function을 지정한다. 또한 실제자원 시뮬레이터가 동작 중이라도 초기화된 애트리뷰트 정보(신택스, support function의 종류 및 파라미터 등)를 GUI 인터페이스를 통하여 변경할 수 있다. 실제자원 시뮬레이터의 각 부분별 기능은 다음과 같다.

2.2.1 시뮬레이터 커널

시뮬레이터 커널은 i) 에이전트로부터 요청받은 애트리뷰트의 random number 값을 생성하여 반환하고, ii) GUI 인터페이스를 통해 요청받은 애트리뷰트의 변경 값을 처리하며, iii) 일정 주기별로 통고 메시지를 생성하여 에이전트에게 전달하는 기능을 수행한다. 따라서 시뮬레이터 커널은 실제자원 시뮬레이터내의 모든 애트리뷰트를 관리하고, 애트리뷰트 신택스가 'REAL' 또는 'INTEGER'인 경우에 support function을 통하여 random number 값을 생성한다. 그리고 신택스가 'STRING'인 경우에는 사용자에 의해 설정된 고유한 값을 저장한다. 특히, 사용자는 GUI 인터페이스를 통하여 i) 시뮬레이터 커널의 실행 전 또는 실행 과정 중에도 애트리뷰트에 관한 정보(random number를 생성하는 시간, 주기, 사용하는 support function의 형태 및 파라미터)를 변경할 수 있다.

시뮬레이터 커널의 구성 모듈은 다음과 같다.

커널 main 모듈 : 시뮬레이터 커널의 main 프로그램으로, 시뮬레이터 커널이 수행하는 모든 동작이 커널 main에서 시작된다. 따라서 커널 main 모듈의 주 기능은 에이전트로부터 요청받은 애트리뷰트의 random number 값을 생성하거나, GUI 인터페이스를 통해 요청받은 애트리뷰트 정보의 변경 사항을 처리하고, 또한 특정 애트리뷰트를 지정하여 일정 주기별로 통지 메시지을 생성하여 에이전트에게 전달하는 기능을 수행한다.

통신 모듈 : 에이전트와 실제자원 시뮬레이터간에 서비스 프리미티브를 교환하는 기능을 수행한다.

애트리뷰트 테이블 : 시뮬레이션이 필요한 애트리뷰트에 대한 random number 값의 생성 정보가 저장된다.

신택스 정보 테이블 : 실제자원 시뮬레이터에서 사용될 신택스들의 인코딩, 디코딩 그리고 프린트 방법에 대한 정보를 저장된다. 따라서 실제자원 시뮬레이터는 에이전트에게 시뮬레이션된 결과 값을 반환할 때, 신택스 정보

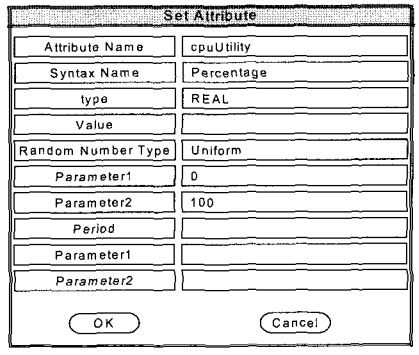

그림 3. 애트리뷰트 설정 화면의 예

테이블에 저장된 정보를 이용하여 반환될 애트리뷰트를 인코딩 한다. 이 정보들은 E-GDMO 컴파일러에 의해서 생성된다. 실제자원 시뮬레이터가 시뮬레이션할 수 있는 신택스 종류로는 'INTEGER', 'REAL', 그리고 'STRING' 등이다.

support function :실제자원 시뮬레이터는 애트리뷰트 신택스가 'INTEGER' 또는 'REAL'인 경우에 애트리뷰트 테이블에 저장된 random number 생성 정보에 따라 실제자원에서 발생되는 결과 값과 유사한 값을 생성한다. 사용자는 GUI 인터페이스를 통하여 ramdom number를 생성하는 support function의 타입과 파라메터를 지정하거나, 지정된 것을 변경할 수 있다. support function은 일반적으로 사용되는 확률 분포인 exponentially, continuous uniform, discrete uniform, binomial, poisson, 그리고 gaussian 등의 함수를 제공한다.

2.2.2 GUI 인터페이스

GUI 인터페이스는 i) 시뮬레이터 커널의 동작 이전에 SMIB 화일에 정의된 애트리뷰트들을 초기화하거나, ii) 관리자 커널이 수행 중일때, 시뮬레이터 커널에 저장된 SMIB 정보를 변경하기 위하여 사용된다. 그림 3은 SMIB에 저장된 애트리뷰트의 support function 타입, 파라미터, 그리고 주기에 관한 정보를 설정하는 예다.

2.2.3 SMIB(MIB for Simulator) 데이터

SMIB 데이터는 E-GDMO 컴파일러가 사용자가 정의한 MIB 파일을 컴파일하여 생성한다. 이러한 SMIB 데이터는 MIB에서 정의된 신택스에 관련된 정보를 포함하고 있다. 신택스 관련 정보는 실제자원 시뮬레이터에서 사용될 신택스 자료 구조, 인코딩, 디코딩, 프린트 방법 그리고 신택스가 'INTEGER' 또는 'REAL' 타입인 경우에

```
#include "BSSASN1Module-types.h"
#include "define.h"

struct _attrInfo attrInfo[] =
{
        {"bssNeAccessChannel.bssNeChannelId",
        _ZBssNeChannelIdBSSASN1Module,
        &_ZBSSASN1Module_mod},
        {"bssNeAccessChannel.bssNeBlockStatus",
        _ZBssNeBlockStatusBSSASN1Module,
        &_ZBSSASN1Module_mod},
        ...
```

그림 4. 애트리뷰트에 대한 SMIB 데이터

```
#include "define.h"

struct ntfInfo Notification_info[] =
{
        {"bssNeAccessChannel.testNotif1",},
        {"bssNeAccessChannel.testNotif2",},
        {"bssNeAccessChannel.objectCreation",},
        ...
```

그림 5. 통고에 대한 SMIB 데이터

random number 값을 생성할 수 있는 정보로 구성된다.

애트리뷰트 정보 파일 : 애트리뷰트의 정보를 유지하는 SMIB 데이터는 관리 객체의 이름, 애트리뷰트의 이름, 애트리뷰트의 타입을 정의한 신택스의 이름, 그리고 신택스의 모듈 이름으로 구성된다. 그림 4는 애트리뷰트에 대한 SMIB 데이터의 예를 보여준다. 그림 5에서 bssNeAccessChannel은 관리 객체의 이름을 나타내며, bssNeChannelId는 애트리뷰트의 이름, BssNeChannelId는 애트리뷰트의 타입을 정의한 신택스의 이름, 그리고 BSSASN1Module은 신택스가 정의된 모듈의 이름을 나타낸다.

통고 정보 파일 : 통고에 대한 SMIB 데이터는 그림 5과 같이 저장되며, bssNeAccessChannel은 관리 객체의 이름을 나타내고 testNotif1은 관리 객체의 통고의 이름을 나타낸다.

2.3 실제자원 시뮬레이터 동작 예

그림 6은 실제자원 시뮬레이터의 동작 과정을 나타낸 것으로서, 에이전트로부터 요청받은 애트리뷰트의 random number 값을 생성하여 반환하는 예를 나타낸다.

(1) 에이전트가 실제자원 시뮬레이터에게 특정 애트리뷰트 값을 요청한다.

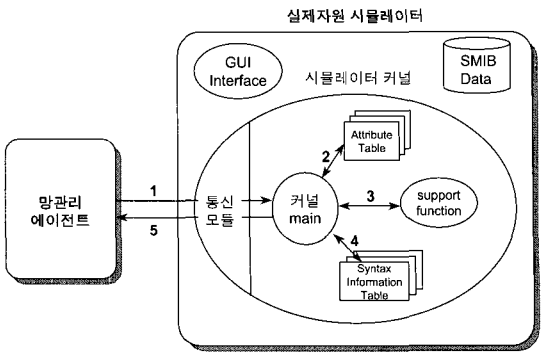

그림 6. 에이전트로부터 요청된 사건의 처리 과정

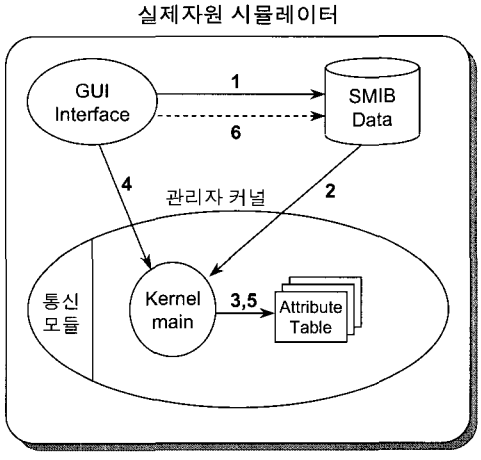

그림 7. GUI 인터페이스를 통한 제어 절차

(2) 실제자원 시뮬레이터는 에이전트로부터 수신한 요청에 따라 애트리뷰트 테이블을 검색하여 해당 애트리뷰트의 정보(support function의 타입 및 파라메터 값 등)를 가져온다.

(3) 2)의 과정에서 얻은 애트리뷰트 정보를 이용하여 support function을 실행시키고, 그 결과 값을 얻는다.

(4) 신택스 정보 테이블을 이용하여, (3)의 수행 결과 값을 encoding하는 데 필요한 정보를 얻은 후, 그 정보에 따라 해당 애트리뷰트 값을 encoding한다.

(5) (4)에서 encoding된 애트리뷰트 값을 에이전트에게 반환한다.

그림 7은 실제자원 시뮬레이터가 GUI 인터페이스를 통하여 애트리뷰트에 대한 제어 정보를 처리하는 과정을 나타낸다.

(1) 실제자원 시뮬레이터를 초기화하기 위해 사용자는 GUI 인터페이스를 통하여 SMIB에 저장된 애트리뷰트에 대한 support function의 타입과 파라미터를 설정한다.
(2) 시뮬레이터 커널은 사용자가 설정한 초기 값을 읽어온다.
(3) 시뮬레이터 커널은 SMIB 데이터에서 읽어온 값을 이용하여 자신의 애트리뷰트 테이블을 초기화한다.
(4) 사용자로부터 초기화된 애트리뷰트 정보의 변경 사항을 시뮬레이터 커널에게 전달한다.
(5) 시뮬레이터 커널은 GUI 인터페이스로부터 전달된 애트리뷰트의 변경 사항에 따라 애트리뷰트 테이블에 해당 내용을 수정한다.
(6) 변경된 애트리뷰트의 정보를 SMIB 데이터에 저장한다.

3. 확장된 GDMO 컴파일러 설계

본 논문에서 설계한 실제자원 시뮬레이터는 실제자원의 역할을 표현하는 데이터를 입력으로 받아들여 실제자원의 행동을 대신한다. 입력 데이터는 표준 GDMO 문법에 시뮬레이션 신택스을 포함한 확장된 형태의 GDMO 즉 E-GDMO 컴파일러에서 제공된다. 따라서 E-GDMO 컴파일러의 역할은 시뮬레이션에 필요한 문법을 추출하여 실제자원 시뮬레이터가 인식할 수 있는 데이터 파일 형태로 저장하는 기능을 수행하며, 기존 망 관리 플랫폼과의 연동을 할 경우 관리 객체 데이터 파일을 생성하는 역할도 수행한다.

3.1 E-GDMO 문법

E-GDMO의 문법은 관리되는 객체를 9개의 템플릿(template)으로 나누어 정의하도록 하였다. 이중에서 관리

```
<package-label>          PACKAGE
        [BEHAVIOUR       <behaviour-definition-label> [, <behaviour-definition-label>]* ; ]
        [ATTRIBUTES      <attribute-label> propertylist [<parameter-label>]*
                         [, <attribute-label> propertylist [<parameter-label>]* ]* ; ]
        [ATTRIBUTE GROUPS <group-label> [<attribute-label>]* [<group-label>
                                 [<attribute-label>]*]* ; ]
        [ACTIONS  <action-label> [<parameter-label>]* [, <action-label>
                         [<parameter-label>]*]* ; ]
        [NOTIFICATIONS   <notification-label> [period-definition <parameter-label>]*
                         [, <notification-label> [period-definition <parameter-label>]*]* ; ]
[REGISTERED AS object-identifier] ;

supporting productions
propertylist        →    [REPLACE-WITY-DEFAULT]
                         [DEFAULT VALUE              value-specifier]
                         [INITIAL VALUE              value-specifier]
                         [PERMITTED VALUES  type-reference]
                         [REQUIRED VALUES   type-reference]
                         [RANDOM VALUES              [random-value-specifier] ]
                         [get-replace]
                         [add-remove]
period-definition   →    RANDOM VALUES [random-function-definition]
value-specifier     →    value-reference | DERIVATION RULE <behaviour-definition-label>
random-value-specifier → period-function-specifier
                         generate-function-specifier
get-replace         →    GET | REPLACE | GET-REPLACE
add-remove          →    ADD | REMOVE | ADD-REMOVE
period-function-specifier → PERIOD random-function-definition
generate-function-specifier → GENERATE random-function-definition
random-function-definition → EXPONENTIAL <real-value>
                         | CONTINUOUS UNIFORM <real-value> <real-value>
                         | DISCRETE UNIFORM <integer-value> <integer-value>
                         | BINOMIAL <integer-value> <real-value>
                         | POISSON <real-value>
                         | GAUSSIAN <real-value> <real-value>
```

그림 8. E-GDMO의 패키지 템플릿 문법

객체 템플릿은 관리객체의 구성 요소를 정의한다. 정의되는 구성 요소는 부모 클래스와 관리 객체에 포함되는 패키지들이다. 패키지 템플릿은 관리 객체 템플릿에서 참조하는 패키지의 구성 요소들을 정의한다. 정의되는 구성 요소는 애플리뷰트와 애트리뷰트 그룹, 동작, 통고 등이다. 이외에 애트리뷰트 템플릿, 애트리뷰트 그룹 템플릿, 동작 템플릿, 통고 템플릿, 파라미터 템플릿, 네임바인딩(namebinding) 템플릿, 행동(behaviour) 템플릿이 있다.

```
samplePackage          PACKAGE
  ATTRIBUTES
  ObjectId             GET,
  CurrentUser          GET-REPLACE
       RANDOM VALUES PERIOD EXPONENTIAL 20
       GENERATE DISCRETE UNIFORM 10 100,
       CurrentProcess  REPLACE-WITH-DEFAULT
                  GET-REPLACE;
    NOTIFICATIONS
         objectCreation,
         objectDeletion,
         attributeValueChange RAMDOM VALUES;
REGISTERED AS    { samplePackage 1 };
```

그림 9. E-GDMO로 기술된 패키지 템플릿

본 논문에서 설계한 에이전트 시뮬레이터가 수행하는 시뮬레이션의 대상은 관리 객체의 애트리뷰트와 통고이며, 이에 따라 시뮬레이션 문법은 관리 객체에 포함되는 애트리뷰트와 통고에 대한 값의 생성을 시뮬레이션할 수 있도록 구성된다. 관리 객체의 애트리뷰트와 통고를 기술하는 GDMO 문법의 템플릿은 패키지 템플릿이므로 기존 패키지 템플릿의 문법에 포함되는 애트리뷰트와 통고에 대해 시뮬레이션을 위한 추가의 문법을 구성하였다. 그림 9는 E-GDMO의 패키지 템플릿에 대한 문법을 나타낸다.

그림 8과 같이 패키지 템플릿에서 애트리뷰트와 통고를 실제 값이 아닌 시뮬레이션하는 값으로 사용한다는 것을 명시할 수 있다. 망 관리자는 패키지 템플릿을 기술하면서 애트리뷰트와 통고를 시뮬레이션한다는 것과, 시뮬레이션에 사용하는 주기 및 생성 함수의 종류와 각각에 대한 파라미터를 정의할 수 있다. 또한 주기 및 생성 함수를 기술하지 않은 경우는 실제자원 시뮬레이터에서 망관리자가 사용 함수와 파라미터를 지정할 수 있다. 그림 10은 E-GDMO로 기술된 패키지 템플릿을 보여준다.

그림 9에서 망 관리자는 CurrentUser라는 애트리뷰트를 실제자원 시뮬레이터에서 생성하는 값으로 사용한다

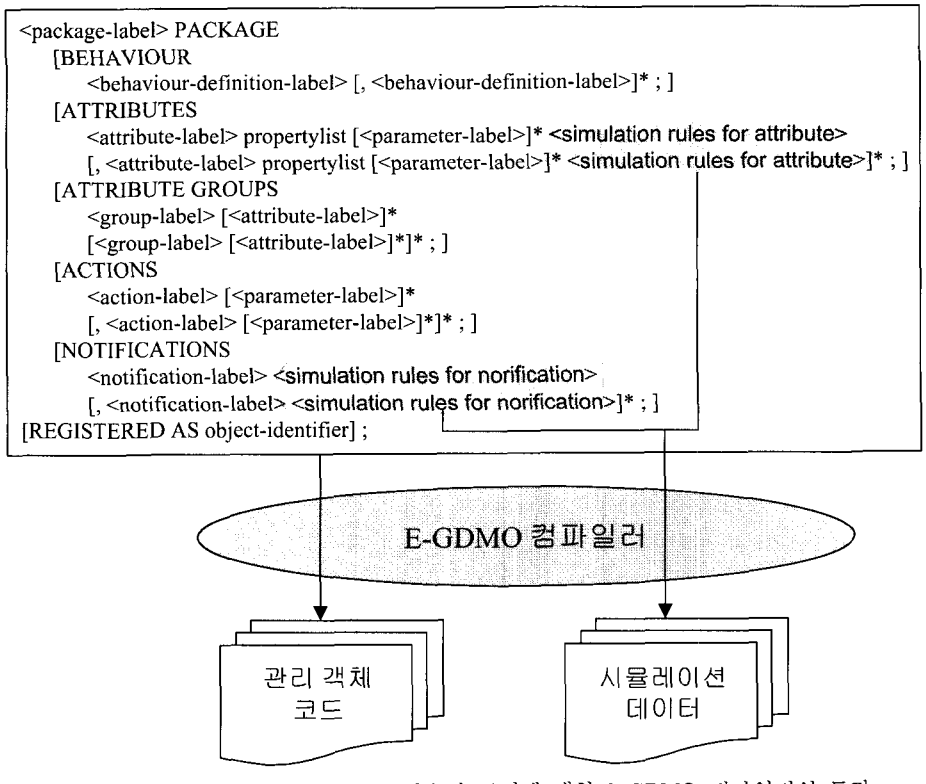

그림 10. E-GDMO로 기술된 문서에 대한 E-GDMO 컴파일러의 동작

는 것을 선언하였으며, 주기 함수로 Exponential 함수를 사용하고, 값 생성 함수로 Discrete Uniform 함수를 사용한다고 선언하였다. 또한 attributeValueChange라는 통고를 시뮬레이터에서 생성하는 값을 이용한다고 선언하였으며, 주기 함수는 시뮬레이터에서 초기화하도록 하기 위해 선언하지 않았다.

3.2 E-GDMO 컴파일러의 동작

E-GDMO 컴파일러는 그림 10과 같이 새로운 패키지 문법에 대해 기존의 문법과 추가된 문법을 분리하여 기존 문법을 이용하여 관리 객체 코드를 생성하고, 추가된 문법을 이용하여 SMIB 데이터를 생성한다. 관리 객체 코드의 생성은 실제 망관리 시스템이 구현된 상태에서 새로운 장치에 대한 시뮬레이션을 수행할 경우에는 필요하지 않은 부분이나, 본 논문에서 설계한 E-GDMO 컴파일러를 이용하여 망 관리 시스템을 구축할 경우에는 필요한 동작이다. 이에 따라 본 논문에서는 망관리 시스템의 개발을 고려하여 E-GDMO 컴파일러가 기존 GDMO 컴파일러의 역할을 포함하도록 하였다.

E-GDMO 컴파일러는 사용자가 기술한 E-GDMO 문서를 GDMO 문법과 시뮬레이션 문법으로 분리하여 내부 클래스(class)에 저장한다. 이와 같이 분리된 데이터는 관리객체 코드와 SMIB 데이터에 대한 파일을 생성한다. 시뮬레이션 데이터 파일은 사용자가 기술한 E-GDMO 문서에서 실제자원 시뮬레이터가 사용할 데이터를 저장한다.

SMIB 데이터는 시뮬레이션에 사용할 애트리뷰트에 대한 이름과 타입, 통고, 그리고 시뮬레이션을 위한 random number 생성 함수, 각 함수의 초기 파라미터 값 등의 정보를 포함한다. 관리객체를 위한 데이터 파일은 실제자원 시뮬레이터가 연동하는 망 관리 플랫폼마다 다르기 때문에 설계된 E-GDMO 컴파일러는 실제자원 시뮬레이터의 검증을 위해 국내·외 대학 및 연구소에서 학술연구 용도로 가장 많이 사용되는 UCL의 OSIMIS 플랫폼에 종속적인 출력 코드를 제공하도록 한다. 그림 12는 E-GDMO 컴파일러의 수행 과정을 나타낸다.

그림 11에서 초기화 파일은 E-GDMO 컴파일러가 수행하기 위해 필요한 기본 데이터화일들이다.

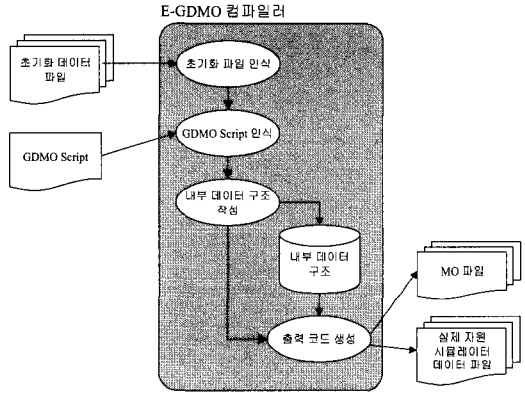

그림 11. E-GDMO 컴파일러의 수행과정

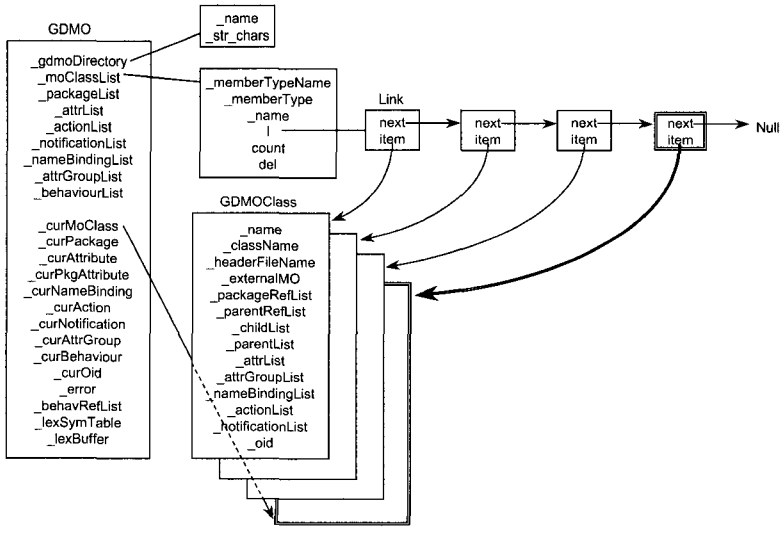

그림 12. GDMO 클래스의 구조

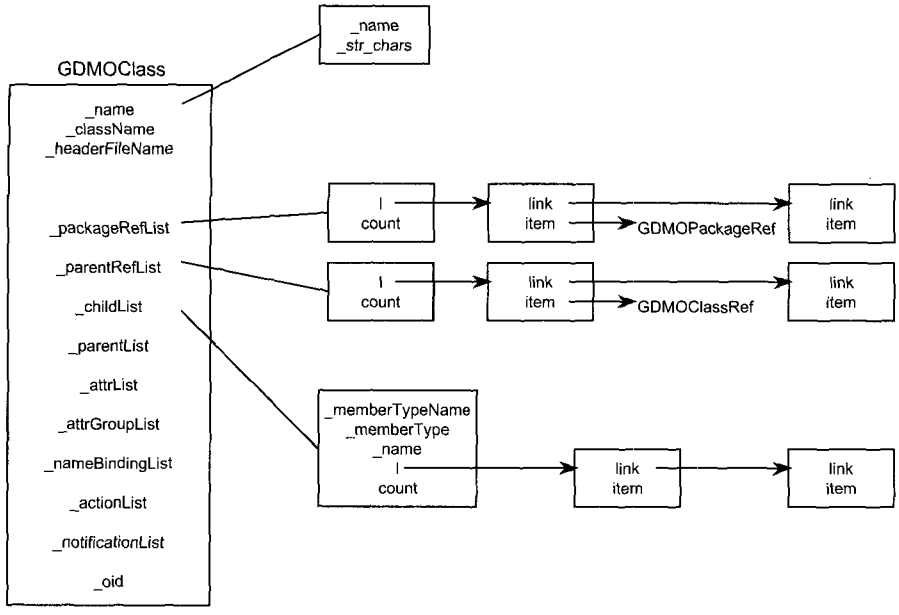

그림 13. GDMOClass 클래스의 구조

3.3 E-GDMO 컴파일러의 데이터 구조

E-GDMO 컴파일러는 E-GDMO 문서에서 기술한 관리 객체 요소를 저장하기 위해서 각 템플릿에 대응하는 클래스를 정의하고, 클래스를 인스턴스화하여 읽어들인 요소를 그 내부에 저장한다. 또한, 인스턴스화된 클래스들을 관리하고, 전체적인 접근이 가능하도록 GDMO 클래스에서 각 클래스에 대한 포인터를 유지한다. 그림 1은 E-GDMO 컴파일러에서 사용하는 GDMO 클래스를 보여준다.

GDMO 클래스에는 스크립트에 읽어들인 모든 정보들을 저장할 수 있다. 클래스 내부에서 관리하는 정보는 GDMO의 데이터베이스가 존재하는 디렉토리, 관리객체 클래스, 패키지, 속성, 액션, 사건보고, 네임바인딩, 속성 그룹, 파라미터와 행동을 저장하는 클래스에 대한 포인터들의 리스트, 그리고 현재 분석 중인 관리 객체 요소를 가리키는 포인터로 구성된다.

그림 12은 GDMO 클래스의 구성 요소중 관리객체 클래스에 해당하는 구조만을 보여 준다. _moClassList는 사용자의 정의에 따라 생성된 관리객체 클래스에 대한 리스트를 유지하며,여러개의 GDMOClass 클래스를 가리키는 것이 아니라 현재 인식을 수행하고 있는 관리객체 클래스를 가리킨다. 다른 관리 객체 요소에 대하여도 현재 분석중인 정보를 관리하기위해서 템플릿 이름 앞부분에 _cur를 붙인 변수를 사용하며, 관리객체 클래스와 같은 구조로 저장되어 사용된다. 그림에서 두 개의 실선으로 표시된 부분은 기존에 인식된 관리객체 클래스에 또다른 하나의 관리객체 클래스를 인식한 후의 데이터 구조를 나타낸다.

E-GDMO 컴파일러는 스크립트를 읽고 그 내용을 템플릿 단위로 분석해서 해당하는 클래스의 인스턴스를 만든다. 즉 관리객체 클래스 템플릿에 대하여는 GDMO-Class를 인스턴스화하고, 패키지 템플릿이면 GDMO-Package를 인스턴스화한다. 인스턴스로 된 클래스들은 인식되는 정보에 따라 필요한 데이타를 추가하게 되며, 각 클래스는 OId를 저장해두는 GDMOOid 클래스를 포함한다.

E-GDMO 스크립트 내에는 같은 템플릿의 관리 객체 요소를 여러 개 정의할 수 있으므로 GDMO 클래스에서는 이러한 정보를 저장하는 리스트를 제공한다. E-GDMO 컴파일러는 관리 객체 요소를 템플릿 단위로 분석하고 그에 대한 클래스를 인스턴스한 다음 그 인스턴스를 GDMO 클래스 내의 리스트에 붙인다. 물론 GDMO 클래스 내에는 템플릿의 종류에 따라서 같은 종류를 하나의 리스트에 저장하기 위해서 템플릿의 종류 만큼의 리스트를 준비하고 있다. 관리객체 클래스들은 GDMOList에서 포인터를 유지하며, 패키지들은 GDMOPackageList에서 포인터를 유지한다.

E-GDMO 스크립트에서 사용하는 템플릿에 대응하는

클래스들의 내부 구조는 템플릿에서 정의한 내용을 저장하기 위한 변수들로 이루어 진다. GDMOClass는 관리객체 클래스 템플릿의 정의에 따라 상속상 부모인 클래스와 필수 패키지, 조건부 패키지와 조건, 그리고 등록 OID가 정의되어 있다. 그림 13은 하나의 관리객체 템플릿에 대한 정보를 저장하는 GDMOClass 클래스의 구조를 보여준다.

GDMOClass 내부에는 상속상 부모인 클래스와 상속상 자식인 클래스를 저장하는 리스트가 있고, 포함되는 패키지에서 정의한 속성, 속성 그룹, 액션과 사건보고를 저장하는 리스트가 있다. 그리고 외부적으로 선언한 네임바인딩을 저장하는 리스트와 OID를 저장하는 변수가 있다. 템플릿을 읽으면 우선 정의되어있는 상속상의 부모들을 리스트에 저장하고, 그 다음 포함되는 패키지를 리스트에 저장하게 되다. 이 리스트는 다음 요소를 가리키는 포인터와 요소의 총 개수를 가지고 관리된다.

그 다음 원소는 리스트의 다음 원소를 가리키는 포인터와 그 내용인 클래스를 가리키는 포인터를 갖는다. 클래스를 가리키는 포인터는 상속상 부모인 경우는 GDMOClassRef를 가리키고, 포함된 패키지의 경우는 GDMOPackageRef를 가리키게 된다. 그리고 패키지에서 정의한 각 요소들과 네임바인딩은 ListSymbol을 상속하는 클래스에 의해서 관리된다. 이 클래스는 리스트에 붙일 수 있는 원소의 타입을 정의한 부분과 리스트를 유지하기 위한 원소로 이루어져 있다. 원소의 타입은 예를 들면 속성은 GDMOAttribute 클래스를 인스턴스한 것이므로 GDMOAttribute와 클래스의 심볼 타입인 Object TypeDef이 해당하는 원소의 타입을 나타낸다. 리스트는 위에서 설명한 것과 마찬가지로 다음 요소를 가리키는 포인터와 리스트의 크기로 유지되며 리스트의 한 요소는 다음 요소를 가리키는 포인터와 그 내용인 클래스를 가리키는 포인터를 갖는다.

관리객체 클래스의 등록과 관련된 OID도 String SymbolList라는 클래스를 통해서 관리된다. 이 클래스도 위의 ListSymbol을 상속한다. 그 내부는 String과 StringTypeDef로 이루어지고 OID의 각 요소들이 하나의 스트링을 이루며 리스트에 저장된다.

GDMOClass는 템플릿에서 정의한 것보다 더 많은 요소를 이루어져 있지만 그 근본은 템플릿 정의에 있다. 이와 비슷하게 다른 템플릿의 경우도 그 내용을 그대로 저장할 수 있도록 대응되는 클래스를 GDMO 컴파일러에서 정의하여 놓았다.

4. 결 론

본 논문에서는 실제자원이 없는 환경에서도 망관리 시스템을 구축하기 위한 실제자원 시뮬레이터와 E-GDMO 컴파일러를 설계하였다.

실제자원 시뮬레이터는 도입해야할 장비를 대신하여, 해당 장비에서 수행되는 망관리 동작을 미리 시뮬레이션 할 수 있는 기능을 제공한다. 실제자원 시뮬레이터는 관리 객체의 애트리뷰트에 대한 값과 통고를 지정된 random 함수에 따라 자동으로 생성시키는 역할을 수행한다. 또한 기존 GDMO 문법에 시뮬레이션 문법을 추가한 E-GDMO 문법은 본 논문에서 개발한 E-GDMO 컴파일러가 실제자원 시뮬레이터에서 사용하는 데이터를 생성시켜 실제자원 시뮬레이터에서 사용하도록 한다.

본 논문에서 설계한 실제자원 시뮬레이터와 시뮬레이션 데이터를 생성하는 E-GDMO 컴파일러를 이용함으로써 i) 실제자원이 완성되기 이전에 에이전트와 관리자 시스템을 구현할 수 있고, ii) 자체 개발된 망 관리 시스템의 동작을 검증할 수 있으며, iii) 망 사업자 측면에서는 도입할 통신 장비에 대해 실제자원 시뮬레이터 및 확장된 GDMO 컴파일러을 이용하여 관리 시스템을 개발할 수 있기 때문에 장비 도입 후 즉시 그것에 대한 운용과 관리가 가능해진다.

참 고 문 헌

1. ITU-T M.3010, "Principles for a Telecommunication Management Network".
2. ISO7498-4/ITU-T X.700, "OSI Basic Reference Model Part 4: Management Framework".
3. ISO10040/ITU-T X.701, "Systems Management Overview".
4. ISO9595/ITU-T X.710, "Common Management Information Service Definition".
5. ISO9596-1/ITU-T X.711, "Common Management Information Protocol Specification".
6. ISO10165-1/ITU-T X.720, "Management Information Model".
7. ISO10165-2/ITU-T X.721, "Definition of Management Information".
8. ISO10165-4/ITU-T X.722, "Guidelines for the Definition of Managed Objects".
9. ISO10165-5/ITU-T X.723, "Generic Management Information".
10. ISO10164-5/ITU-T X.734, "Event Management Func-

tion".
11. George Pavlou, "The OSIMIS Platform:Making OSI Management Simple".
12. George Pavlou, "High-Level Access APIs in the OSISMIS TMN Platform: Harnessing and Hiding".
13. George Pavlou, "Experience of Implementing OSI Management Facilities".
14. G.Pavlou, J.Cowan, J.Crowcroft, "A Generic Management Information Base Browser".
15. John Larmouth, "ASN.1 Complete", Open Systems Solutions, 1999.

송 병 권 (iamsong@naver.com)

1984년 고려대학교 전자공학과 학사
1986년 고려대학교 전자공학과 석사
1995년 고려대학교 전자공학과 공학박사
1995년~현재 서경대학교 정보통신공학과 교수

관심분야 : 망관리, 이동 컴퓨팅, 고속통신망 프로토콜