# BDL을 이용한 망관리 객체의 시간지원 능동특성에 대한 정형적 모델

최 은 복† · 노 봉 남††


## 요 약

본 논문에서는 시간속성과 능동특성을 지원하는 관리객체의 동적특성 표현언어인 BDL(Behaviour Description Language)을 이용하여 시스템 망관리 모델의 관리기능을 정형적으로 표현하였다. 그리고 BDL로 표현된 관리기능을 CORBA IDL로 변환하는 BDL_to_IDL 컴파일러를 설계·구현하였다. 특히, 망 관리 정보베이스에 저장되어있는 관리객체를 안전하게 보호하기 위해 ITU-T 권고안에 정의된 강제적 접근제어 모델과 역할기반 접근제어 모델을 상호연동한 접근제어모델을 정의하였다. 또한, 관리속성값을 제어하는 관리연산을 연관된 유형별로 묶어 역할로 정의하고 관리자와 관리객체에 인가등급과 보안등급을 부여하여 역할배정규칙과 제약조건에 따라 관리정보의 접근을 제어함으로써 보다 무결성을 보장받도록 하였다.


# A Formal Model of Managed Objects with Temporal and Active Properties Using BDL

Eun-Bok Choi† · Bong-Nam Noh††


## ABSTRACT

In this paper, we described network management functions using a language for specifying behavioral aspects of managed objects, BDL(Behaviour Description Language), that supports temporal and active properties. And, With a BDL to IDL translator, network management functions can be easily coded and transformed into CORBA IDL. Also we define a new access control model which combines Biba model, one of the mandatory access control policies stipulated by ITU-T Recommendations, and RBAC model for securing managed objects in Management Information Base. In that model, we categorize management operations for administrative attributes as a role and assign security levels to both managed objects and network managers. This model ensures the integrity MIB by controlling accesses to managed objects with constraint conditions and role assignment rules.


## 1. 서 론

통신과 컴퓨터 기술의 발달 그리고 이들의 통합으로 인하여 전송 시스템 및 교환 장비를 포함한 통신 장비를 비롯하여 컴퓨터, 인터페이스, 프로토콜 등과 같은 통신망을 구성하고 있는 구성요소들이 복잡해지고 규모가 커지고 있다. 이렇게 복잡한 전체 망을 관리하는 망 관리 시스템에서 문제가 발생할 경우 사용자 및 기업체 모두에게 서비스 중단, 인적 물적 손실 등과 같은 여러 가지 막대한 영향을 미치므로 대규모 망의 원활한 제어와 감시 기능을 제공하는 망관리 시스템이 필요하다. OSI에서는 이를 위해 구성관리, 고장관리, 성능관리, 보안관리 및 회계관리 기능을 제공하고 있으며 이러한 망 관리 기능을 수행하기 위해 망에서 수집되는 다양한 자료가 저장되고 처리되어야 한다.

† 정 회 원 : 전남대학교 대학원 전산학과
†† 종신회원 : 전남대학교 컴퓨터정보학부 교수
논문접수 : 2000년 6월 24일, 심사완료 : 2000년 7월 24일

망 관리기능을 정의하는 관리객체 클래스는 통신망 구성 자원들의 정적 속성 및 동적 특성, 연관된 관리 객체 클래스간의 상속관계 등에 대한 정보를 객체지향 개념을 기반으로 표현한다. 따라서 통신망 관리시스템은 관리객체 클래스에 정의된 속성과 연산을 통해 통신망 구성요소 내부의 구현방식과는 무관하게 관리대상 객체의 동작상태를 감시하거나 필요에 따른 적절한 제어기능을 수행할 수 있다[5].

관리객체를 정의하기 위한 가이드라인인 GDMO(Guidelines for the Definition of Managed Objects)를 표현하고 있는 ITU-T의 X.722 권고안은 관리객체 클래스의 정의를 위한 템플릿들에 대해 표현하고 있다. GDMO 표현법은 9가지 템플릿을 이용하여 관리객체의 정적 속성과 동적 특성 외에, 통신망 관리시스템에게 관리객체 내부에서 관리상에 주요한 사항이 발생했음을 알리는 통지에 대해서 자세히 표현하고 있다. 그러나 현재의 GDMO 표현은 관리객체의 구조와 속성 등을 포함한 대부분의 정적 특성을 적절히 표현하고 있는데 반해, 동적 특성을 표현하는 방법으로는 텍스트 형태의 자연어를 이용하고 있어 관리객체의 모든 특성을 완전하게 표현하지 못하는 문제점을 가지고 있다[8].

망관리 시스템에서는 중요한 상황이 발생할 때, 시기 적절하게 대처할 수 있는 기능이 필수적으로 제공되어야 하며, 망관리 정보 베이스의 시간 데이터(temporal data)를 이용한 자원관리 기능이 첨가되어야한다. 따라서 망관리 정보 베이스 상태를 감독하고, 정의된 조건을 만족시키는 연산이 수행되었을 때 이에 대한 적절한 행위를 시간적 제약사항에 따라서 수행하는 시간지원 능동 특성을 갖는 망관리 시스템이 필요하다. 또한 엄격한 시간 제한 내에 수행되거나 일정한 주기를 갖고 반복적으로 수행되는 관리 서비스의 경우 망관리 정보 베이스의 무결성과도 밀접하게 연관이 되어 시간지원 개념이 절실히 요구된다.

망관리 시스템에서 관리자나 대리자에 의해 유지되는 망관리 정보 베이스는 망 관리 시스템의 가장 핵심적인 부분으로서 망 관리에 필수적이며 중요한 모든 정보를 유지하고 있기 때문에 안전하게 유지되어야 한다. 이러한 망관리 정보 베이스를 안전하게 운용되기 위해서는 망 사용자에 대한 정확한 인증 뿐만 아니라 관리 객체에 대한 접근을 효율적으로 통제할 수 있어야 하며 또한 관리 객체에서 발생하는 사건의 통지에 대해서도 효율적으로 제어하여야 한다.

본 논문에서는 정적 및 동적 시간속성과 능동특성을 지원하는 동적특성 표현언어인 BDL(Behaviour Description Language)[15]을 정의하고 이 언어로 표현된 시스템 망관리 모델의 4가지 관리기능을 정형적으로 표현한다. 그리고 BDL 파서와 IDL 코드생성기로 구성된 BDL_to_IDL 컴파일러를 개발하여 CORBA IDL 파일로 변환한다. 이는 기존 IDL_to_C++이나 IDL_to_Java 컴파일러를 이용하면 BDL로 정의된 관리객체의 동적 행위부분을 구현할 수 장점을 제공한다. 특히, 망관리 정보베이스에 저장되어있는 관리객체를 안전하게 보호하기 위해 표준 권고안에 정의된 강제적 접근제어 모델과 역할기반 접근제어 모델을 상호연동한 접근제어모델을 정의하여 관리자와 관리객체의 보안등급과 역할, 그리고 이들간의 제약조건에 따라 관리정보의 접근을 제어함으로써 보다 무결성을 보장하고자 한다.

## 2. 관리정보 모델

### 2.1 개 요

관리정보 모델의 관리객체를 정의하기 위한 가이드라인인 GDMO는 OSI환경에서 자원관리를 위한 관리객체 클래스를 표현하는데 사용되는 표기법을 정의하기 위한 국제 표준으로서 관리객체 클래스 템플릿, 패키지 템플릿, 매개변수 템플릿 등 9개의 템플릿을 이용하여 관리객체 클래스의 정적 및 동적 특성을 표현하고 있다[8]. 또한 관리정보 모델에 포함된 추상적인 모델링 개념과 OSI 환경에서 특별한 자원 관리를 위한 관리객체 클래스의 표현 요구조건과의 관계를 매핑해 주는 역할을 수행한다. 여기에는 관리객체 정의자가 관리객체 클래스와 그들의 컴포넌트들을 표현할 때 사용하는 표기법의 문법과 의미에 대한 정의가 포함되어있다.

GDMO는 유일한 관리객체 식별자들을 컴포넌트들에 할당하는 등록 트리(registration tree) 구조를 갖으며 이들 식별자들은 정수값의 노드로 구성된다. 첫 번째 노드는 트리의 초기 루트로 두 번째 노드의 등록기관에 해당하며 ISO, CCITT 그리고 ISO-CCITT 중 하나를 갖는다. 두 번째 노드는 표준에 관련된 집단이나 관련된 영역이 배정되며 그 하위 노드로부터 특별한 관리객체에 이르기까지 트리 구조로 형성된다. 이러한 관리객체는 사용자가 알아보기 편하게 읽기 편한 문자형태의 이름으로 변환될 필요가 있는데 이런 변환

과정을 명칭변환이라 한다.

관리객체 클래스를 정의하는데 사용되는 표기법은 관리객체 클래스의 여러 관점의 명세를 표현하기 위한 형식인 템플릿 개념에 기반을 둔다. 템플릿에서는 구성요소들이 표현되고 삭제되고 반복되는가에 대한 문법적인 표현사항이 정의되며 표준화된 형식의 집합으로 간주되어진다. 각각의 템플릿들은 관리객체 클래스의 완전한 정의를 생성하기 위해 다른 템플릿들과 연합되어 사용되어진다.

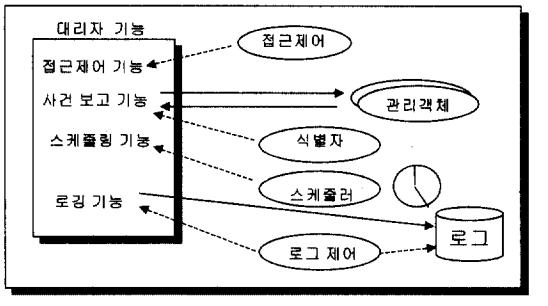
관리객체 클래스 템플릿은 관리객체 클래스의 중심부분에 해당하는 템플릿으로 NAME BINDING 템플릿을 제외하고 모든 다른 템플릿들이 관리객체 클래스 템플릿을 직·간접적으로 참조한다[5].

모든 관리객체 클래스는 하나이상의 상위클래스로부터 특성을 상속받는데 궁극적으로 top이라는 특별한 관리객체 클래스로부터 상속되어진다. 관리객체 클래스 템플릿에 있는 DERIVED FROM 절은 이러한 상속관계를 나타내는 클래스를 정의하고 있다. 또한 한 개 이상의 상속되는 패키지 템플릿이 존재하는데 여기에는 속성, 연산, 통지, 매개변수 그리고 정의되어진 행위 등이 포함된다.

CHARACTERIZED BY 절에서는 클래스의 모든 인스턴스들이 포함되는 필수패키지를 정의하고 있으며 CONDITIONAL PACKAGE 절은 조건을 만족했을 때 수행하는 패키지에 대한 정의로서 선택적으로 표현되어진다. 그리고 REGISTERED AS 절은 관리객체 클래스 정의에 해당하는 관리객체 클래스 이름을 정의하는 절로서 전역적으로 사용되는 유일한 관리객체 식별자를 할당하는데 사용된다.

### 2.2 대리자 관리 기능 모델

(그림 1)의 대리자 관리 기능 모델에서 대리자 관리 객체는 망 관리 서비스를 제공할 뿐만 아니라 관리 객체 인터페이스와 개방형 통신 인터페이스간의 매핑을 제공한다. 또한 대리자에서는 수행되는 연산을 선택적으로 필터링하는 메커니즘과 통지를 통해 발생하는 데이터의 흐름을 제어하는 기능을 제공한다. 국제표준안에서 제공하는 대리자의 기능에는 접근제어, 사건 보고, 스케줄링, 그리고 로깅 기능 등이 있다[10]. 관리자와 관리객체에 대한 접근 권한의 허가 여부를 결정하는 접근제어 기능과 관리객체에 대한 특정 사건이 발생하였을 때 관리자에게 통지하는 사건 보고 기능, 주기적이고 반복적인 스케줄링 계획에 따라 관리객체의 활동을 제어하는 스케줄링 기능 그리고 사건이나 통지, 관리객체 접근 등 전반적인 내용에 대한 감사 추적을 위한 로깅 기능 등이 있다.




(그림 1) 대리자 관리 기능 모델

## 3. 정형적 모델의 설계 및 구현

### 3.1 모델링 개념

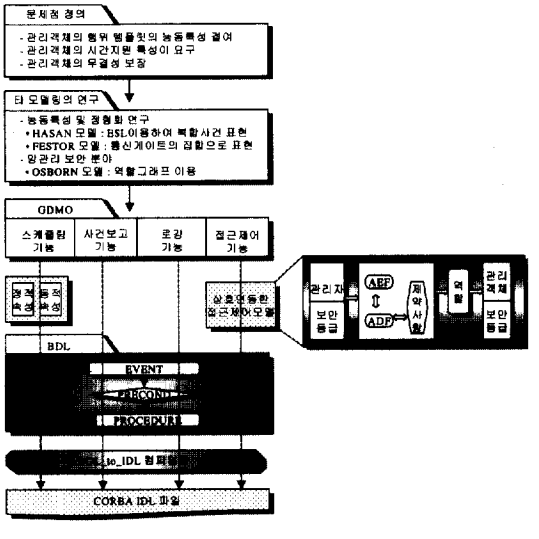
OSI에서는 객체지향 개념을 이용하여 관리객체들을 정의하고 전체적인 망관리 정보를 모형화하고 있으며, 이러한 관리객체 클래스를 정의하기 위한 가이드라인인 GDMO에서는 아홉 개의 템플릿을 제공하고 있다. 그러나 현재의 GDMO 표현은 관리객체의 구조와 속성 등을 포함한 대부분의 정적 특성을 적절히 표현하고 있는데 반해, 동적 특성을 표현하는 방법으로는 텍스트 형태의 자연어를 이용하고 있어 관리객체의 모든 특성을 완전하게 표현하지 못하는 문제점을 가지고 있다. 또한 망관리 시스템에서는 능동 특성과 함께 중요한 상황이 발생할 때 시간적 제약사항에 따라 시기적절하게 수행하는 시간지원 특성이 절실히 요구된다. 그리고 관리자나 대리자에 의해 유지되는 망관리 정보베이스가 안전하게 운용되기 위해서는 망 사용자에 대한 정확한 인증 뿐만 아니라 관리객체에 대한 접근을 효율적으로 통제할 수 있어야 한다.

이러한 문제점을 해결하기 위해 여러 가지 다양한 연구가 진행되었다. 동적 특성에 영향을 주는 요인을 몇가지 게이트로 정의하고 동작을 위한 조건과 행위를 규칙형태로 정의한 Festor 모델이 있으며 ER 모델을 기본으로 관련성있는 관리객체들간의 사건을 제약사항으로 표현한 ERC 모델이 제시되었다[2]. Hasan은 통신망을 데이터베이스로 간주하고 통신망을 구성하는 관리객체에서 발생하는 사건 발생조건을 표현하는 사

건명세언어를 제시하고, 발생한 사건에 대한 처리과정을 ECA 규칙으로 모델링하였다[3]. Osborn은 정보의 비밀성을 보장하는 BLP 모델과 역할기반 접근제어모델을 역할그래프모델에 적용하였다. 하지만 역할그래프는 일반적인 역할기반 접근제어 모델에 비해 매우 제한적으로 이용된다는 단점이 있다[4].

이렇듯 여러 관련연구를 살펴보아도 관리객체의 동적특성을 표현하기 위해 게이트나 인터페이스 그리고 사건간의 관계, 관계계층 등 E-R 모델을 통해 표현하고 있을 뿐 관리정보를 구현할 경우 필요한 프로그램 형태의 정형적 표현 방법은 미흡하다.

그러므로 본 논문에서는 시스템 망관리 모델의 4가지 관리기능을 동적특성 표현언어인 BDL을 사용하여 표현한 후 CORBA IDL로 변환하는 컴파일러를 개발하여 실제 망관리에 적용될 수 있는 기반을 제공한다. 특히, 망 관리 정보베이스에 저장되어있는 관리객체를 안전하게 보호하기 위해 표준 권고안에 정의된 강제적 접근제어 모델과 역할기반 접근제어 모델을 상호연동한 접근제어모델을 정의하여 관리자와 관리객체의 보안등급과 역할, 그리고 이들간의 제약조건에 따라 관리정보의 접근을 제어함으로써 보다 무결성을 보장하고자 한다(그림 2).




(그림 2) 시간지원 능동특성의 정형적 모델 구조

### 3.2 동적특성의 구성요소

관리 객체의 시간속성을 지원하는 능동특성의 시스템 망관리 모델의 관리기능을 정형적으로 표현하기 위한 구성요소는 다음과 같다.

#### 3.2.1 외부요인(EVENT)

관리객체의 동작 절차의 결정요소로서 통신망 관리자로부터 전달되는 관리객체에 대한 관리연산의 실행요청과 다른 관리객체로부터 전달되는 통지 또는 사건보고 등이 있다.

#### 3.2.2 선행조건(PRECOND) 및 불변조건(INVARIANTS)

선행조건은 관리연산의 정상적인 수행이나 통지가 발생되기 위한 논리조건을 표현하고 불변조건은 관리객체의 동작 전체 과정에서 지켜져야 하는 조건을 표현한다. 선행조건과 불변조건은 참, 거짓을 결정할 수 있는 논리식 형태로 표현한다.

#### 3.2.3 동작절차(PROCEDURE)

관리객체로 전달되는 외부요인이나 관리객체 내부의 속성값, 조건부 패키지의 선행조건 등을 이용한 ECA 규칙을 기반으로 작성된다. 관리객체에 대한 사건(EVENT)이 발생되면, 발생된 사건의 처리여부를 결정(PRECOND)하고 사건에 대한 적절한 조치(ACTION)를 수행하는 구조를 가진다.

### 3.3 동적 시간지원 관리객체 클래스

관리객체의 동적특성은 능동적 특성 외에 시간지원 특성에 따라 동작절차가 달라질 수 있다. 관리객체의 동작은 통신망 관리 목적이나 관리객체의 특성에 따라 반복적이고 주기적인 정적인 시간 개념 뿐만 아니라 다른 관리자의 속성에 따라 사용권한이 결정되거나 특정기간 동안에만 권한이 부여되는 동적인 시간개념이 첨가된다면 매우 복잡한 운용조건을 갖는 관리객체의 동적 특성을 정확히 표현할 수 있다.

현재 표준권고안의 시간 관리객체의 활동을 제어하는 스케줄링 관리 객체는 관리자에 의해 시간이 지정되는 정적인 시간개념으로 동적특성을 갖는 시간지원 특성을 표현하기에는 다소 미흡하다. 그래서 다른 관리자의 속성에 따라 사용권한이 결정되거나 특정기간 동안에만 권한이 부여되는 동적인 시간개념이 추가된다면 동적특성이 더욱 강화될 것이다. 그러므로 본 논문에서는 X.746 권고안에 표현된 스케줄러 관리객체

클래스에 동적인 시간지원 속성을 지원할 수 있는 패키지로, 특정 관리자에 대한 관리객체의 접근모드에 따라 해당 관리자의 접근여부가 결정되는 dependencySchedulingPackage와 관리자가 관리객체에 접근요청을 시작하는 시점을 기준으로 하여 종료시점을 부여하는 timestampSchedulingPackage를 정의하였다.

3.3.1 dependencySchedulingPackage

특정 관리자에 대한 관리객체의 접근모드에 따라 해당 관리자의 접근여부가 결정되는 'dependencySchedulingPackage'를 정의하였다. 여기서 적용되는 'dependencyMode' 모드에는 특정관리자가 관리객체에 대해 접근모드를 수행하기 전에만 수행할 수 있는 'UNLESS' 모드와 특정 관리자가 관리객체에 대해 접근모드를 수행하지 않는 그 이외의 시기에만 수행할 수 있는 'WHENEVERNOT' 모드가 있으며, 이와 상반된 개념으로, 특정 관리자가 관리객체에 대해 접근모드를 수행할 때만 접근할 수 있는 'WHENEVER' 모드와 특정관리자가 관리객체에 대해 접근모드를 최초로 수행할 때만 접근할 수 있는 'ASLONGAS' 모드가 있다[1]. 이에 해당하는 규칙을 정의하면 다음과 같다.

[규칙 1] 관리자 $m_i$와 관리자 $m_j$의 실제 관리객체 $mo_i$를 관리하는 대리자 $a_i$의 접근모드 p의 연산은 dependencyMode에 따라 결정된다.

⇨ **$(m_i, a_i, mo_i, p)$ <dependencyMode> $(m_j, a_i, mo_i, p)$**

단, $m_i$, $m_j \in M$ : 관리자($i \neq j$), $a_i \in A$ : 대리자, $mo_i \subseteq MO$ : 관리객체, $p \subseteq P$ : ACCESS_MODE

3.3.2 timestampSchedulingPackage

권고안에서 표현된 시간속성은 시작시간과 종료시간을 지정하여 수행하는 'durationPackage'나 'dailySchedulingPackage' 그리고 'weeklySchedulingPackage'등이 있다. 하지만 이러한 패키지는 관리자에 의해 정적인 시간개념으로 부여가 되며 관리객체 또한 중요도에 따른 차별화를 두지 않아 정보의 보안성이 침해될 우려가 있다. 이러한 관리객체에 대해서는 접근시간을 실시간 환경에 적용될 수 있는 동적 개념으로 변환하므로서 보다 안전하게 유지할 필요가 있다. 여기에서는 관리자가 관리객체에 접근요청을 시작하는 시점을 기준으로 하여 종료시점을 부여하는 타임스템프 개념을 도입하여 정보의 안전성을 도모하였다.

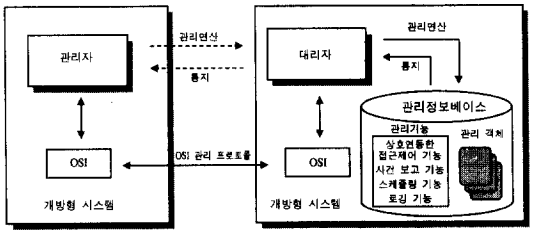
[규칙 2] 관리자 $m_i$와 실제 관리객체 $mo_i$를 관리하는 대리자 $a_i$의 접근모드 p의 연산은 타임스템프 $T_i$ 동안에만 수행된다.

⇨ **$(m_i, a_i, mo_i, p, T_i)$**

단, $m_i \in M$ : 관리자, $a_i \in A$ : 대리자, $mo_i \subseteq MO$ : 관리객체, $p \subseteq P$ : ACCESS_MODE, $T_i \in \mathbf{N} \cup \infty$ : EXPIREDTIME

3.4 동적특성 표현방법 설계

(그림 3)은 OSI의 기본적인 시스템 망관리 모델로서 관리표준의 관리기능과 GDMO를 포함하는 관리정보의 구조 및 세부내용을 갖는다. 특히, 관리기능에는 상호연동한 접근제어 기능과 사건보고, 스케줄링 그리고 로깅기능으로 구성된다. 이러한 관리객체 클래스들의 동적특성을 표현하기 위하여 본 절에서는 각 기능들을 ITU-T X.720에 정의된 관리객체 클래스의 동적 특성 요소인 관리연산이나 통지 그리고 관리객체의 속성과 조건부 패키지를 반영하였다. 특히 접근제어 기능모델은 역할영역과 제약조건을 기반으로 하여 상호연동한 접근제어 모델의 접근제어 결정함수와 접근제어 집행함수의 동작과정을 표현하므로써 보다 체계적이고 명확한 제어과정을 서술하였다.




(그림 3) OSI 시스템 망관리 모델

3.4.1 상호연동한 접근제어 기능

접근제어 기능은 관리객체의 보안을 위하여 사용자의 접근을 제어 및 관리하는 기능이다. 접근제어를 위한 메카니즘은 접근행렬, 접근제어 리스트, 능력리스트에 의한 방법인 자율적 접근제어와 관리자와 관리객체의 보안 등급에 의한 방법인 강제적 접근제어 그리고 관리자의 역할에 의한 역할기반 접근제어가 있다. 본 절에서는 강제적 접근제어 모델중에서 무결성을 보장하는 Biba 모델과 실생활에 적용될 수 있는 역할기반 접근제어 모델을 상호연동한 접근제어 모델의 규칙을

기반으로 표현하였다. rule 관리객체는 관리자가 특정 관리객체에 접근하고자 할 때 규칙에 근거하여 접근여부를 결정하는 기능을 수행하는데, 현재 권고안에는 'rule' 관리객체의 행위 템플릿을 자연어로 표현하고 있어 관리객체의 동작 조건이나 절차, 결과 등이 명확히 표현되지 못한 문제점을 가지고 있다. 또한 접근제어 프레임워크[13]에서는 입출력 정보와 접근제어 메카니즘, 그리고 관련된 관리객체 클래스와 속성들만 표현되어있고 실질적인 접근제어 과정은 세부적으로 표현되지 않았다.

이에 본 절에서는 상호연동한 접근제어 모델의 역할 영역과 제약조건을 정의한 후 접근제어 규칙인 'rule' 관리객체 클래스를 BDL로 표현하였다. 특히, 표준안에 표현되어있지 않은 접근제어 관리 기능인 접근제어 결정함수와 접근제어 집행함수의 행위 템플릿을 세부적으로 명확하게 표현하였다.

**1) 역할 영역**

관리정보 모델에는 관리연산을 크게 전반적인 관리객체에 적용되는 연산과 속성값에 적용되는 연산으로 구분하고 있다. 전반적인 관리객체에 적용되는 연산에는 관리객체의 인스턴스를 생성하고 삭제하는 create, delete 연산과 개별적인 관리객체의 요구조건을 정의하는 action 연산이 있다. 그리고 속성값에 적용되는 연산에는 속성값을 읽는 get 연산, 속성값을 쓰는 replace 연산, 그리고 관리객체 정의시 명기되어있는 값으로 재정의하여 쓰는 replace with default 연산 등이 있다. 또한 특별한 속성 타입을 정의하기 위한 것으로 동일한 데이터 타입의 멤버들의 비순서 집합을 추가, 삭제하는 addMember와 removeMember 등이 있다[5].


(그림 4) 역할 영역

본 논문에서는 실질적으로 관리정보베이스의 속성값을 수정하고 읽는 관리연산들에 대해서 (그림 4)와 같이 읽기전용역할, 쓰기전용역할 그리고 이들 두 연산을 포함하는 읽기쓰기 역할로 세분하였다.

**2) 제약조건**

인가등급을 갖는 관리자는 역할영역에 속해있는 배정된 역할에 따라 관리객체의 관리연산을 수행할 수 있으며, 관리객체에도 보안등급이 부여되어 정보의 무결성을 보장토록 한다. 이러한 관리연산을 수행하기 위한 역할 배정규칙과 이에 따르는 제약조건을 다음과 같이 표현하였다.

**● 읽기 전용 역할[제약조건 1]**

주체의 집합 $\mathbb{S}$에 해당 주체 S가 속하고, 역할의 집합 $\mathbb{R}$에 읽기 전용 역할 $R_r$이 속할 때, 주체 S가 읽기 전용 역할($R_r$)에 배정되기 위해서는 주체의 인가등급이 읽기 전용 역할($R_r$)에 속해있는 보안등급중 제일 낮은 등급인 최소 보안등급[r-level($R_r$)]에 지배되어야 한다.

$\Rightarrow \forall S \in \mathbb{S},\ \forall R_r \in \mathbb{R}$

RoleAssign(S, $R_r$) $\Rightarrow \lambda(S) \leq$ r-level($R_r$)

**● 쓰기 전용 역할[제약조건 2]**

주체의 집합 $\mathbb{S}$에 해당 주체 S가 속하고, 역할의 집합 $\mathbb{R}$에 쓰기 전용 역할 $R_w$이 속할 때, 주체 S가 쓰기 전용 역할($R_w$)에 배정되기 위해서는 주체의 인가등급이 쓰기 전용 역할($R_w$)에 속해있는 보안등급중 제일 높은 등급인 최대 보안등급[w-level($R_w$)]을 지배하여야 한다.

$\Rightarrow \forall S \in \mathbb{S},\ \forall R_w \in \mathbb{R}$

RoleAssign(S, $R_w$) $\Rightarrow \lambda(S) \geq$ w-level($R_w$)

**● 읽기쓰기 역할[제약조건 3]**

주체의 집합 $\mathbb{S}$에 해당 주체 S가 속하고, 역할의 집합 $\mathbb{R}$에 읽기쓰기 역할 $R_{rw}$이 속할 때, 주체 S가 읽기쓰기 역할($R_{rw}$)에 배정되기 위해서는 읽기 전용 역할($R_r$)의 최소 보안등급이 쓰기 전용 역할($R_w$)의 최대보안등급을 지배하여야 하고, 주체의 인가등급이 읽기 전용 역할($R_r$)의 최소 보안등급은 지배하고 쓰기 전용 역할($R_w$)의 최대 보안등급에는 지배되어야 한다.

$\Rightarrow \forall S \in \mathbb{S},\ \forall R_{rw} \in \mathbb{R}$

RoleAssign(S, $R_{rw}$) $\Rightarrow$ r-level($R_r$) $\geq$ w-level($R_w$) AND $\lambda(S) \leq$ r-level($R_r$) AND $\lambda(S) \geq$ w-level($R_w$)

다음은 BDL를 이용한 접근제어 관리객체의 동적 특성중 상호연동한 접근제어의 역할영역과 제약조건에 근거한 관리객체 접근을 제어하는 과정과 수행절차를 정형적으로 표현하고 있다. 여기에서 이태릭체로 표현된 부분은 정적 시간속성을 확장한 동적 시간속성 패키지와 상호연동한 접근제어 기능을 나타낸다. 특히, 권고안에서 간략히 텍스트형태로 표현된 접근제어집행함수와 접근제어 결정함수를 세부적으로 명확하게 정형화하였다.

```
rule MANAGED OBJECT CLASS
DERIVED FROM accessControl;
CHARACTERIZED BY rulePackage PACKAGE
BEHAVIOUR ruleBehaviour BEHAVIOUR
DEFINED AS
EVENT :
  AccessControlEnforcementEvent
  accessControlObject : accessControlObjectName;
INVARIANTS :
  startTime <= stopTime;
PRECOND:
  administrativeState == unlocked AND
  operationalState == enabled AND
  availabilityStatus != Offduty;
PROCEDURE:
if ( NOT (EXISTS durationPackage AND
   currentTime VALID IN [startTime, stopTime] )
   OR
  (EXISTS dailySchedulingPackage AND
   currentDay VALID IN [startDay, stopDay] )
   OR
  (EXISTS weeklySchedulingPackage AND
   currentWeek VALID IN [startWeek, stopWeek])) then
    emit timeDomainViolation notification;
    abort;
    if (NOT( EXISTS  dependencySchedulingPackage
     OR timestampSchedulingPackage ) ) then
     emit securityServicOrMechanismViolation notification;
     abort;
    endif;
  endif;
  emit AccessControlDecisionEvent notification;
  if (enforcementAction == allow) then
   "access is permitted";
   validAccessAttempts = validAccessAttempts + 1;
   "send to a security audit trail log";
  emit usageReport notification;
  if (enforcementAction == deny with response) then
   "access is denied";
   invalidAccessAttempts = invalidAccessAttempts + 1;
   "send to a security audit trail log";
  emit usageReport notification;
  endif;
 endif;
END;
EVENT : AccessControlDecisionEvent
  accessControlObject : accessControlObjectName;
PRECOND:
  administrativeState == unlocked AND
  operationalState == enabled AND
  availabilityStatus != Offduty;
PROCEDURE:
switch(Role)
{
case readonlyRole :
 if r-level(Rr) DOMINATES initiator-level
  then      enforcementAction = allow
  else   enforcementAction = deny with response ;
 endif;
case writeonlyRole :
 if initiator-level DOMINATES w-level(Rw)
  then   enforcementAction = allow
  else   enforcementAction = deny with response ;
 endif;
case  readwriteRole :
 if initiator-level(Rr) DOMINATES target-level(Rw)
 then
  if (r-level(Rr) >= w-level(Rw) ) AND
    (initiator-level <= r-level(Rr) ) AND
    (initiator-level >= w-level(Rw) ) then
  enforcementAction = allow
   else   enforcementAction = deny with response ;
  endif;
 endif;
END;
```

(그림 5) BDL을 이용한 확장된 rule 관리 객체 클래스의 동작 특성 표현

3.4.2 사건보고 기능

사건 보고 기능은 시스템에 이미 발생한 사건이나 발생할 가능성이 있는 사건 등을 관리자에게 통지해 주는 역할을 수행한다. 사건에 대한 통지는 사건 전송 식별자(Event Forwarding Discriminator)에 의하여 선택되어 CMIS의 M-EVENT-REPORT 서비스를 이용하여 관리자에게 보내어진다[6].

다음은 BDL를 이용한 사건 보고 기능에 대한 시간 지원 동적 특성을 정형적으로 표현한 예이다[15].

```
eventForwardDiscriminator MANAGED OBJECT CLASS
DERIVED FROM discriminator;
CHARACTERIZED BY efdPackage PACKAGE
BEHAVIOUR eventForwardingDiscriminatorBehaviour
BEHAVIOUR
DEFINED AS
  EVENT : PortentialReportEvent
          managedObjectClass: ObjectClass,
          managedObjectInstance: ObjectInstance,
          eventType: EventTypeID,
          .....
```

(그림 6) BDL을 이용한 eventForwardDiscriminator MO Class 동작 특성 표현

3.4.3 스케줄링 기능

특별한 관리객체 인스턴스 내에 명세되어 있는 시간 관리객체의 활동을 제어하여 관리객체 내에서 자체적으로 발생하는 'trigger scheduling'인 비주기적인 스케줄링과 관리객체의 활동에 사용되는 연산의 시간구간을 지정하여 일정하게 반복적으로 제어하는 'interval scheduling' 주기적인 스케줄링으로 나뉜다. 특히 스케줄링을 수행하는 관리객체를 스케줄링 관리 객체(Scheduling Managed Object)라 하며 스케줄링 관리 객체에 적용되는 스케줄링 값과 타입을 관리하는 객체를 스케줄러 객체(Scheduler Object)라 한다[9].

스케줄링 기능에 대한 스케줄러 관리객체의 시간 지원 동적 특성의 정형적 표현은 다음과 같다[15].

```
scheduler MANAGED OBJECT CLASS
DERIVED FROM : top;
CHARACTERIZED BY schedulerObjectPackage,
                 duration;
schedulerObjectPackage  PACKAGE
BEHAVIOUR  schedulerObjectBehaviour BEHAVIOUR
DEFINED AS
    EVENT : LockSchedulingManagedObject
            schedulerOjectName : schedulerObject,
            .....
```

(그림 7) BDL을 이용한 scheduler MO Class의 동작 특성 표현

3.4.4 로깅 기능

다양한 객체에 의해 수행되어진 연산이나 이미 발생한 사건에 관한 정보를 유지할 필요가 있는데 OSI 관리 모델에서는 'log'나 'log record'라는 관리객체클래스를 사용하여 정보를 보관한다. 'log' 관리객체는 외부로부터 전달되는 사건 보고나 로컬 시스템에서 발생된 통지의 저장 기능을 수행한다. 'log record'는 로그에 저장되는 정보의 단위를 나타낸다.

다음은 BDL을 이용한 로그 관리객체의 동적 특성중 다른 객체로부터 전달된 통지에 대한 처리과정과 통신망 관리자로부터 전달된 속성값 변경에 대한 관리명령 수행 절차를 나타내고 있다[15].

```
logPackage PACKAGE
   BEHAVIOUR
   logBehaviour BEHAVIOUR
   DEFINED AS
     EVENT: internalNotification
        notification: smi2Notificaiton
        notificationID: NotificationIdentifier;
        .....
```

(그림 8) BDL을 이용한 log 관리객체 클래스의 동작 특성 표현

3.5 표현 모델의 구현

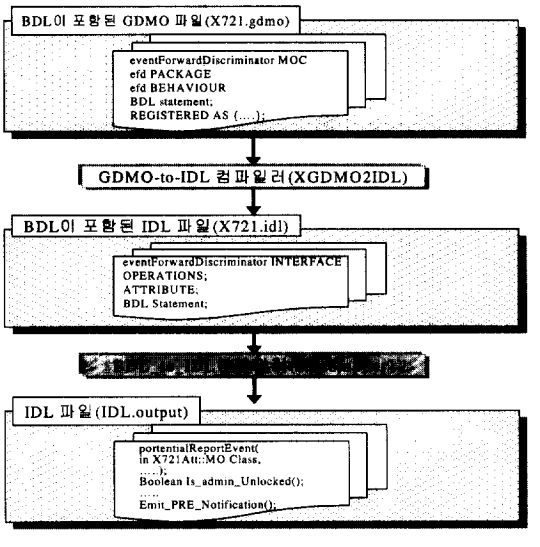
본 절에서는 관리객체의 행위부분을 정형화한 BDL 문법의 정확성을 검증하기 위하여 BDL_to_IDL 컴파일러를 구현하여 제시하였다.

3.5.1 BDL_to_IDL 컴파일러 설계

관리객체의 동적특성을 CORBA IDL로 변환하는 소프트웨어의 구현은 다음의 두 단계로 구성된다. 먼저, 관리객체의 행위부분을 표현하고 있는 BDL 파일에 대한 에러 검색과 문법을 점검하는 BDL 파싱 단계와, 파싱 작업이 성공적으로 끝났을 때 BDL 파일을 CORBA의 IDL로 변환하는 IDL 코드 생성 단계이다.

BDL_to_IDL 컴파일러는 Solaris 2.7을 사용한 SUN 워크스테이션에서 구현되었으며, 컴파일러 생성 도구로 lex와 yacc을 이용하여 개발하였다. 또한 IDL 코드 생성기를 개발하기 위해 C 컴파일러로 gcc 2.95.1을 사용하였다.

BDL 파일을 IDL 파일로 변환하기 전에 BDL 파일을 제외한 관리객체 클래스 정의부분과 관련된 패키지 그리고 속성들을 IDL로 변환하기 위해 경북대에서 개발한 XGDMO2IDL이라는 컴파일러를 이용하였다[14].



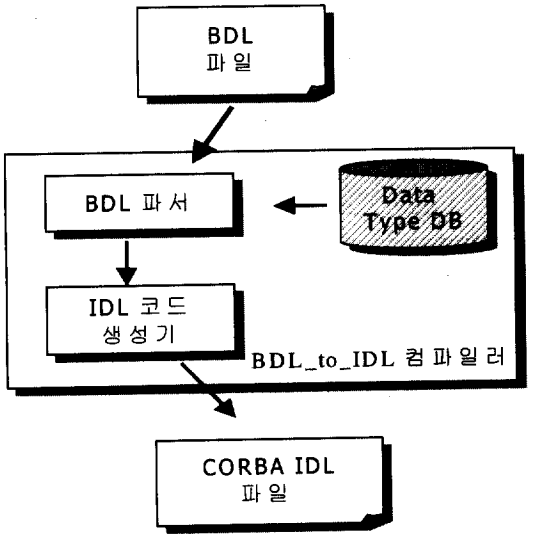

(그림 9) BDL 파일의 IDL 변환 과정

BDL_to_IDL 컴파일러는 BDL을 CORBA의 IDL 파일로 변환시켜 주는 프로그램이다. 먼저, 이 컴파일러를 실행시키기 전에 XGDMO2IDL 컴파일러를 통해 관리

객체 클래스와 관련된 패키지 그리고 속성들을 CORBA IDL로 변환해주어야 한다. 이 컴파일러에는 X721.gdmo라는 관리정보베이스가 입력값으로 요구된다. X721.gdmo는 X.721 권고안에 표현되어있는 관리객체 클래스를 데이터베이스화한 GDMO 파일로서 관리객체 클래스와 관련된 패키지 그리고 클래스에서 정의된 속성들로 구성되어 있다. 이러한 X721.gdmo에 BDL 파일을 삽입한 후 XGDMO2IDL을 실행시킨다. 물론 BDL 파일의 처음과 끝부분에는 특정 문자를 삽입하여 실행하였으므로 출력되는 IDL 파일에는 주석으로 처리되어 나온다. 주석으로 처리되어진 BDL 파일은 BDL_to_IDL 컴파일러를 통해 IDL 파일로 변환되어진다(그림 9).

**1) BDL 파서**

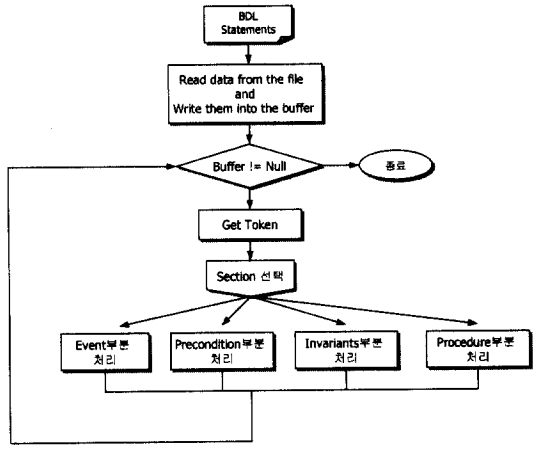
BDL 파서는 컴파일러 생성 지원 도구인 Lex와 Yacc을 이용해서 구축되었다. (그림 10)에서 DataType DB에는 BDL 파일의 속성 정의부분에서 사용되는 Data Type으로 ObjerctClass, ObjectInstance, EventTypeID, SecurityAlarmSeverity, BackUpStatus, ProbableCause, DiscriminatorConstruct 등을 가지고 있다. 이 DataType은 ASN.1에 정의된 타입이어야 한다.




(그림 10) BDL_to_IDL 컴파일러 구성

**2) IDL 코드생성기**

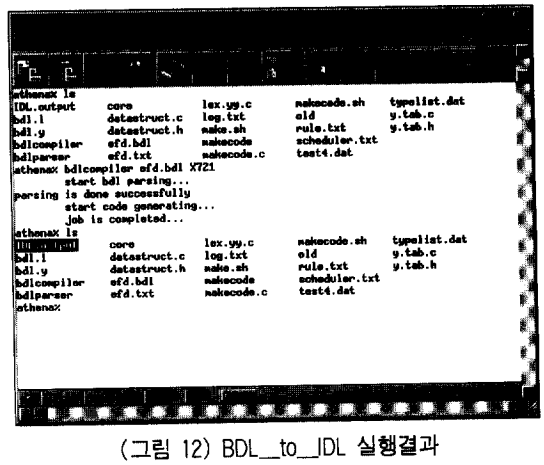
IDL 코드생성기는 BDL 파서에 의해 어휘분석과 구문분석이 성공적으로 끝난 BDL 파일을 받아서 IDL 파일로 변환한다. IDL 코드생성기는 먼저 BDL 파일을 버퍼로 읽어 들이고 이 버퍼에서 토큰을 가져와 Event 부분, Precondition부분, Invariants부분, Procedure부분 중 하나를 선택해서 적당한 IDL 파일을 생성한다[부록]. 이때, BDL 파일은 IDL 파일에 주석으로 처리된 후 그것에 해당하는 IDL 파일이 삽입된다. BDL 파일을 주석으로 처리하는 것은 프로그램 개발자가 고급언어로 변환하고자 할 경우 도움이 되고자 하는데 그 이유가 있다.




(그림 11) IDL 코드생성기 흐름도

3.5.2 실행결과 고찰

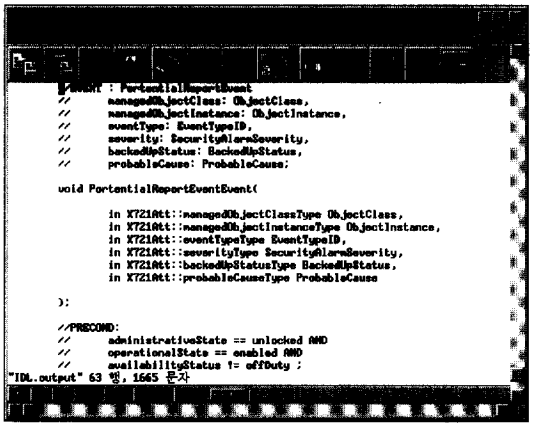
BDL_to_IDL 컴파일러인 bdlcompiler는 XGDMO2-IDL에서 출력값으로 생성된 IDL 파일중에서 행위부분을 표현하고 있는 BDL 파일만을 추출하여 IDL로 변환해주는 컴파일러이다. bdlcompiler는 BDL 파서 기능




(그림 12) BDL_to_IDL 실행결과

과 IDL 코드생성기 기능을 수행하는 컴파일러로 BDL 파일과 속성값을 출력하기 위한 권고안 타입이 입력값으로 들어간다. BDL 파일에 대한 어휘분석과 구문분석을 마치면 CORBA의 IDL 파일로 변환된 IDL.output 파일이 생성된다.

(그림 12)는 BDL_to_IDL 컴파일러인 bdlcompiler의 실행화면이며 (그림 13)은 bdlcompiler에 의해 생성된 IDL.output 결과 내용을 보여주고 있다.




(그림 13) IDL.output 결과 내용

## 4. 결 론

통신 장비를 구성하고 있는 요소들이 복잡해지고 규모가 커지고 있는 요즘, 복잡한 전체 망의 원활한 제어와 관리기능을 제공하는 망관리 시스템은 필수적이다. ISO에서는 통합 망 관리 시스템의 기능을 정의하기 위해 객체지향 개념을 기반으로 하는 ITU-T의 X.722 권고안의 GDMO 표현법을 9개의 템플릿을 이용하여 관리객체의 속성을 표현하고 있다. 그러나 관리객체의 구조와 속성 등을 포함한 대부분의 정적 특성을 적절히 표현하고 있는데 반해, 관리객체의 능동적 요인들과 동작절차 및 그에 따른 통지 발생 등의 동적 특성을 표현하는 방법으로는 텍스트 형태의 자연어를 이용하고 있어 관리객체의 모든 특성을 완전하게 표현하지 못하는 문제점을 가지고 있다.

본 논문에서는 정적 및 동적 시간속성과 능동특성을 지원하는 동적특성 표현언어인 BDL로 표현된 시스템 망관리 모델의 4가지 관리기능을 정형적으로 표현하였다. 그리고 BDL 파서와 IDL 코드생성기로 구성된 BDL_to_IDL 컴파일러를 개발하여 CORBA IDL 파일로 변환하였다. 이는 기존 IDL_to_C++이나 IDL_to_Java 컴파일러를 이용하면 BDL로 정의된 관리객체의 동적 행위부분을 구현할 수 장점을 제공한다. 특히, 망 관리 정보베이스에 저장되어있는 관리객체를 안전하게 보호하기 위해 표준 권고안에 정의된 강제적 접근제어 모델과 역할기반 접근제어 모델을 상호연동한 접근제어 모델을 정의하여 관리자와 관리객체의 보안등급과 역할, 그리고 이들간의 제약조건에 따라 관리정보의 접근을 제어함으로써 보다 무결성을 보장한다.

## [부록] IDL 코드 생성기 알고리즘

**[IDL 코드생성기]**

```
BDL_file = Open BDL_document;
size = sizeof( BDL_file );   /* get file size */
buffer = malloc( size+1 );
for( i=0 ; i < size ; i++ ){
        read a character from the BDL_file;
        write the character into the buffer[i];
}
buffer[i] = '\0';
current_pos = 0;
while( buffer[current_pos] is not null ){
  get token from the buffer;

  switch( token ){
   case EVENT:
    event_procedure();
    break;
   case PRECOND:
    precond_procedure();
    break;
   case INVARIANTS:
    invariants_procedure();
    break;
   case PROCEDURE:
    procedure_procedure();
    break;
   }
}
```

**[EVENT 부분 처리]**

```
void envent_procedure(file, buffer, pos, event_name)
{
 char token1[TOKEN_SIZE], token2[TOKEN_SIZE], token3
 [TOKEN_SIZE];
  /* 이벤트 함수명을 얻는 과정*/
 get_token(token1); /* 버퍼로부터 토큰을 얻음 */
 토큰(이벤트 함수명)을 배열에 저장;
 token1을 주석으로 처리;
  // EVENT : event_function_name
 /* 이벤트 속성들 추출 */
  get_token(token1);
  while( strcmp(token1, ";") )
   /* 마지막 속성일 때까지 반복 */
  {
```

```
    if( !strcmp(token1, ",") )
    /* strcmp()는 문자열을 비교하여 같을 때 0 반납 */
     {
     get_token(token1);  /* managedObjectClass */
     token1(이벤트 속성)을 배열에 저장;
     }
    get_token(token2);   /* : */
    get_token(token3);   /* ObjectClass */
    token1, token3을 배열에 저장;
    token1, token3을 주석으로 처리;
    get_token(token1);
    }  /* end of while */

for (i=0; i < line; i++)
 배열을 파일로 출력;
} /* end of event_algorithm */
```

**[PRECOND 부분 처리]**

```
void precond_procedure(file, buffer, pos)
{
 char token1[TOKEN_SIZE] token2[TOKEN_SIZE], token3
 [TOKEN_SIZE];
 get_token(token1); /* 버퍼로부터 토큰을 얻음 */
  /* 선행조건 속성 추출 */
 while( strcmp(token1, ";") )
  /* 마지막 속성일 때까지 반복 */
  {
   if( !strcmp(token1, "AND") )
    /* strcmp()는 문자열을 비교하여 같을 때 0 반납 */
    {
     get_token(token1);
    }
   get_token(token2);
   get_token(token3);
   if ( !strcmp(token2, "==") )
    "Boolean Is_token1_token3();"을 배열에 저장;
     else if ( !strcmp(token2, "!=") )
      "Boolean IsNot_token1_token3();"을 배열에 저장;
     else
      "Boolean Default_token1_token3();"을 배열에 저장;
    token1, token2, token3을 주석으로 처리;
    get_token(token1);
   } /* end of while */

   for (i=0; i < line; i++)
    배열을 파일로 출력;
 } /* end of precond_algorithm */
```

**[INVARIANTS 부분 처리]**

```
void invariants_procedure(file, buffer, pos)
{
  char token1[TOKEN_SIZE] token2[TOKEN_SIZE], token3
  [TOKEN_SIZE];
  get_token(token1); /* 버퍼로부터 토큰을 얻음 */
/* 불변조건 속성 추출 */
  while( strcmp(token1, ";") )
   /* 마지막 속성일 때까지 반복 */
   {
    if( !strcmp(token1, "AND") )
     /* strcmp()는 문자열을 비교하여 같을 때 0 반납 */
    get_token(token1);
    get_token(token2);
    get_token(token3);
    if ( !strcmp(token2, "==") )
     "Boolean Is_token1_token3();"을 배열에 저장;
      else if ( !strcmp(token2, "!=") )
       "Boolean IsNot_token1_token3();"을 배열에 저장;
      else
       "Boolean Default_token1_token3();"을 배열에 저장;
     token1, token2, token3을 주석으로 처리;
     get_token(token1);
    } /* end of while */

  for (i=0; i < line; i++)
   배열을 파일로 출력;
 } /* end of invariants_algorithm */
```

**[PROCEDURE 부분 처리]**

```
void procedure_procedure(file, buffer, pos, event_name)
{
  char token1[TOKEN_SIZE], token2[TOKEN_SIZE];
  "void Emit_event_name_Notification();" 배열에 저장후 파일
  출력;
  start = 프로시져 시작 위치;
  반복문 사용하여 토큰을 얻음;
  end = 프로시져 끝 위치;

  for (i=start; i < end; i++)
   배열을 파일로 출력;
} /* end of procedure_algorithm */
```

## 참 고 문 헌


[1] Elisa Bertino, Claudio Bettini, Pierangela Samarati, "A Discretionary Access Control Model with Temporal Authorizations," IEEE New Security Paradigms Workshop, 8, 1994.

[2] Oliver Festor, Georg Zorntlein, "Formal Description of Managed Object Behavior - A Rule Based Approach," Proceedings of the IFIP TC6/WG6.6 3rd International Symposium on Integrated Network Management, 1993.

[3] Masum X. Hasan, "An Active Temporal Model for Network Management Databases," Proceedings of the 4th International Symposium on Integrated Network Management, 1995.

[4] Sylvia Osborn, "Mandatory Access Control and Role-Based Access Control Revisited," Second ACM Workshop on RBAC, pp.31-40 11. 1997.

[5] Morris Sloman, Network and Distributed Systems Management, Addison-Wesley, 1994.

[6] ITU-T X.720 : "Information Processing Systems - Open Systems Interconnection - Management Information Model," Geneva.

[7] ITU-T X.721 : "Information Processing Systems - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 2 : Definition of Management Information," Geneva.

[8] ITU-T X.722 : "Information Processing Systems -

Open Systems Interconnection - Management Information Services - Structure of Management Information Part 4 : Guidelines for the Description of Managed Objects," Geneva.

[9] ITU-T X.723 : "Information Processing Systems - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 5 : Generic Management Information."

[10] ITU-T X.734 : "Information Technology - Open Systems Interconnection - System Management - Event Report Management Function."

[11] ITU-T X.741 : "Information Technology - Open Systems Interconnection - System Management - Objects and Attributes for Access Control."

[12] ITU-T X.746 : "Information Technology - Open Systems Interconnection - System Management - Scheduling Function."

[13] ITU-T X.812 : "Information Technology - Open Systems Interconnection - Security Frameworks For Open System - Access Control Framework."

[14] 경북대학교 AIN Lab, "XGDMO2IDL-GDMO/ASN.1 to CORBA IDL Translator," http://ain.kyungpook.ac.kr/nexus.

[15] 최은복, 이형효, 노봉남, "동적 시간지원 특성을 지원하는 망관리 객체의 정형적 모델링", 정보처리학회 논문지, 제7권 제1호, 2000.

**최 은 복**

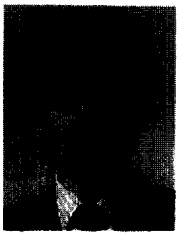
e-mail : eunbog@chonnam.ac.kr

1992년 전남대학교 전산학과 (이학사)

1996년 전남대학교 대학원 전산학과(이학석사)

2000년 전남대학교 대학원 전산학과(이학박사)

관심분야 : 통신망관리, 정보보안, 멀티미디어시스템 등



**노 봉 남**

e-mail : bongnam@chonnam.ac.kr

1978년 전남대학교 수학교육과 (이학사)

1982년 한국과학기술원 전산학과 (공학석사)

1994년 전북대학교 대학원 전산통계학과(이학박사)

1983년~현재 전남대학교 컴퓨터정보학부 교수

관심분야 : 객체지향시스템, 통신망관리, 정보보안, 컴퓨터와 정보사회 등