논문 00-25-6A-10 한국통신학회논문지 '00-6 Vol.25 No.6A

네트웍 관리를 위한 MIB의 자동생성

정회원 유 재 우*, 김 영 철*, 김 성 근*

Automatic Generation of MIB for Network Management

Chae-Woo Yoo*, Young-Chul Kim*, Sung-Keun Kim* *Regular Members*

요 약

TMN에서의 네트웍 관리는 네트웍에 존재하는 운영체제나 통신 장비를 객체로 정의하고 관리한다. 이들 객체를 기술하기 위해 GDMO(Guidelines for the Definition of Managed Objects)를 사용하고 있다. GDMO는 네트웍 관리를 위해 직접 이용되지 않으며, 객체지향 파라다임을 가지는 언어로 변환되어 사용된다. 또한 GDMO는 객체들을 관리하기 위하여 ASN.1(Abstract Syntax Notation One)을 참조한다. 본 논문에서는 ASN.1 & GDMO 명세를 Managed object Instance Base(MIB) 생성에 사용되는 객체지향 언어로 자동 번역하는 번역기를 설계하고 구현하였다. 본 시스템은 기존의 네트워크 관리를 위한 원시코드 생성기와는 달리 MIB 생성에 이용될 수있는 다양한 객체지향 언어를 자동 생성할 수 있도록 고안되었으며, ASN.1과 GDMO를 하나의 부라우저에서 처리할 수 있는 통합 환경을 제공함으로써 응용프로그램을 개발하는데 편리하도록 하였다.

ABSTRACT

Network management in TMN concerns to the operating system and communication equipments in network, and defines them as objects. GDMO(Guidelines for the Definition of Managed Objects) is used to describe those objects. GDMO is not directly used for managing the network, but translated into a language with object-oriented paradigm. And GDMO refers to ASN.1(Abstract Syntax Notation One) for manage objects. This paper presents design and implementation techniques for the translator which automatically translates the specification of ASN.1 and GDMO to the object-oriented language for generating MIB(Managed object Instance Base). This system, unlike the existing source code generator, is designed to generate various object-oriented languages automatically, which are used to generate Managed object Instance Base(MIB). And the system includes various graphic user interface to enhance the developement environment of ASN.1 and GDMO

Ⅰ. 서 론

네트웍을 매개로 컴퓨터가 다른 컴퓨터의 파일이나 주변 기기에 고속으로 액세스하기 위해서는 다양한 종류의 운영체제와 통신 장비간의 정보 교환을 하기 위한 네트웍 관리가 필수적으로 요구된다. 이러한 상황을 극복하고자 하는 노력의 하나인 네트웍 관리의 기본 하부 구조인 통신 관리 네트웍 (Tele- communications Management Network : TMN) 개념이 ITU-T에 의해 1988년에 처음 정의되었다[1]. TMN의 구조는 통신 네트웍의 계획과 설계에 따른 물리구조, 기능구조, 정보구조 세 가지로 분류될 수 있으며, Q_3 인터페이스의 표준화와 공유정보의 모델링을 표준화하기 위한 작업을 진행하고 있다.

TMN은 다양한 종류의 운영체제와 통신 장비 사이에 표준화된 인터페이스를 이용하여 정보의 교환이 이루어지도록 하는 것이다. 이를 수행하기 위해서는 인터페이스와 행위를 표준화된 형태로 정의하

* 숭실대학교 컴퓨터학부
논문번호 : 99287-0723, 접수일자 : 1999년 7월 23일

고 관리할 수 있는 도구가 요구된다. 이를 위해서 ASN.1(Abstract Syntax Notation One) [2,3]과 GDMO(Guide line Definition Managed Object)가 표준으로 제정되었다.

GDMO는 관리 정보 각각의 구조적인 모델링을 가능하게 하는 기술 도구이다. GDMO는 우선 관리 객체 클래스 템플릿을 사용하여 네트워크 자원이나 관리 정보를 추상화된 관리 객체로 규격화하여 정의한다. 또한 그 템플릿 안에서 패키지(package), 행위(behavior), 속성(attribute), 속성 그룹(attribute group), 액션(action), 인지(notification), 네임 바인딩(name binding), 파라메터(parameter)와 같은 여러 템플릿을 다시 정의하거나 선언함으로써 그 관리 객체의 특성을 정의한다. 관리 객체 클래스 템플릿 내의 템플릿들은 자신의 특성을 관리 객체 클래스 템플릿이나 패키지 템플릿 내에서 정의하지 않고, 자신의 템플릿에서 정의한다. 이들 템플릿을 사용하는 관리 객체 클래스 템플릿이나 패키지 템플릿은 그 내부에서 선언하여 사용하도록 함으로써 GDMO의 객체 지향적 설계의 목적인 재사용성을 향상시킨다. 네임 바인딩 템플릿은 관리 객체 클래스 템플릿의 특성을 정의하기보다는 두 관리 객체 클래스 템플릿 간의 포함 관계를 설정 및 속성을 정의하는데 사용된다. 이들 템플릿의 데이터 타입들은 다시 ASN.1 모듈의 ASN.1 타입과 값으로 정의된다.

ASN.1은 다양한 인터페이스나 통신 매체를 통하여 전송되어야할 정보를 기술하기 위한 언어이며, ASN.1 컴파일러는 ASN.1 언어 명세를 C++ 언어 등으로 짜여진 암호(encoding)와 해독(decoding) 루틴으로 바꾸어주는 작업이다. 이러한 루틴들은 OSI 응용프로그램을 개발하는데 이용될 수 있으며, 비 OSI 통신 프로토콜을 구현하기 위해서도 이용될 수 있다. ASN.1으로 정의된 타입을 객체지향 프로그래밍 언어인 Java, C++[4] 등으로 변환하는 도구를 개발함으로써, 표준 데이터 타입을 처리하기 위한 인터페이스를 표준화할 수가 있다. 또한 C++ 언어를 사용하여 객체 지향적인 개념을 충분히 이용할 수 있다. 즉, 단일화 된 유틸리티 함수를 외부에 보여지도록 인터페이스를 정의함으로써 상세 내용을 캡슐화시켜서 모듈의 복잡도를 줄일 수 있다.

ASN.1 명세는 언어 자체에 "import"나 "export" 기능을 포함하고 있으므로, 표준 기관이나 비공식적인 기관에서 정의된 많은 모듈들을 데이터베이스에서 포함하고 있어야한다. Import 기능은 외부 모듈에서 정의된 타입을 이용할 수 있는 기능이고, Export는 본 모듈에서 정의된 타입을 외부 모듈에서 사용할 수 있게 해주는 기능이다. 그러나 지금까지 개발된 ASN.1 컴파일러들은 대부분 데이터베이스와의 연동이 되지 않고 있으며, 단지 구문분석과 의미분석에 의한 다른 언어로의 변환기 역할만을 수행하고 있다. 기존의 ASN.1 컴파일러에는 MAVROS [5], BBN [6], Snacc [7], ISODE의 PEPY/POSY [8], UBC의 CASN1 [9] 등이 있다.

Ⅱ. ASN.1&GDMO 개발 환경 모델

네트워크 관리를 위한 원시 코드 브라우저는 ASN.1 & GDMO 명세 언어를 입력으로, 즉 관리 객체와 ASN.1 정의를 포함한 텍스트 파일을 입력으로 ASN.1 & GDMO 컴파일러나 중간 자료 구조인 ASN.1 & GDMO 스키마 테이블로 번역하여 삽입한다. 이렇게 작성된 ASN.1 & GDMO 스키마 테이블은 템플릿 스키마 데이터베이스에 저장된다. 저장된 중간 자료 구조는 ASN.1 & GDMO 원시 코드 생성기에 의해 객체지향 파라다임을 가지는 언어로 번역되게 된다.

원시 코드 생성의 결과는 객체지향 파라다임을 지원하는 언어로 표현된다. 이것은 새로운 데이터베이스에 저장되어 네트워크관리에 사용된다. 그림 1은 확장된 ASN.1 & GDMO 개발 환경 구조이다.

그림 1에서 ASN.1과 GDMO 명세가 입력되어 각각 해당 컴파일러에 의해 파싱된다. 컴파일러는 어휘분석, 구문분석, 의미분석등을 수행한다. 특히 의미분석 단계에서는 데이터베이스를 활성화시키며,

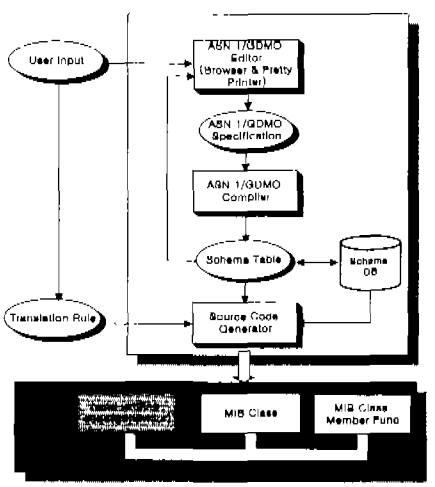

그림 1. ASN.1 & GDMO 개발 환경 모델

컴파일된 결과 물은 메이터베이스 스키마 테이블에 입력된다. 생성된 메이터베이스 스키마 테이블은 변환 규칙을 통하여 원시 코드 생성에 이용된다. 또한 메이터베이스 스키마 테이블은 나중에 편집기나 브라우저를 통해 사용자에게 보여줄 때 이용되기도 한다. 이렇게 생성된 원시 코드는 네트워관리에 적합한 응용프로그램, 유틸리티, 라이브러리 등에 이용된다.

Ⅲ. 개발 환경 모델의 구성 요소

1. ASN.1 & GDMO 편집기

ASN.1 & GDMO 편집기는 ASN.1 & GDMO 각각의 명세를 생성, 수정, 삭제 등의 편집을 지원하는 메뉴를 제공한다. ASN.1 & GDMO 편집기는 여러 선택 사항을 제공하고 있다. 메이터베이스에 저장된 ASN.1 & GDMO 템플릿을 그래픽 형태로 나타내어 사용자 인터페이스를 강화하고 ASN.1 & GDMO 템플릿에 익숙지 않은 사용자도 쉽게 이용할 수 있도록 한다. 편집기는 사용자에 의해 입력된 ASN.1 & GDMO 명세 내용을 컴파일하여 ASN.1 & GDMO 스키마 메이블의 형태로 바꾸어 데이터베이스에 저장한다. ASN.1 & GDMO 편집기의 내부구조는 그림 2와 같다.

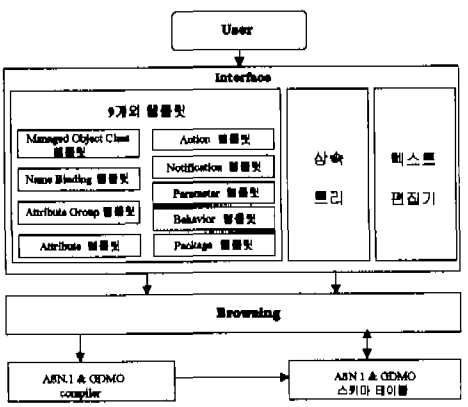

그림 2. ASN.1 & GDMO 편집기의 구조

2. ASN.1 & GDMO 컴파일러

ASN.1 & GDMO 컴파일러를 구현하기 위해서는 많은 컴파일러 보조 도구들이 필요하다. 대부분 이용되는 컴파일러 보조 도구들은 어휘분석 도구로서 lex를 이용하며, 구문분석 도구로 yacc을 이용한다[11]. 그림 3은 ASN.1 & GDMO 컴파일러에 대한 환경을 보여준다.

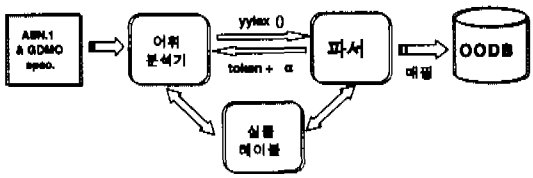

그림 3. ASN.1 & GDMO 컴파일러 개발환경

구문 분석 과정에서 이용한 문법은 1990년에 제정된 x.208 규약에 의한 문법을 기준으로 하였다. 표준으로 정의된 문법은 구문 분석기의 구성에 사용되는 yacc과 같은 도구를 사용하기에는 부적합하므로 정의된 문법을 동일한 부류의 구문의 LALR 문법으로 변환하였다. 문법의 변환 과정은 의미를 변환시키지 않고 기본적인 관계 표현을 적용함으로써 변환이 가능하며, 다음과 같은 관계 연산자를 적용하였다.

- **분해** : 분해 과정은 문법 생성 규칙에서 "prefix", "suffix", "bifix"의 형태를 가지며, 다음과 같이 변환된다. 이러한 분해 과정은 부분적으로 적용될 수 있다.
  prefix : X → $\beta\gamma$ 1 | $\beta\gamma$ 2 | ... | $\beta\gamma$ n | B는 X → β Y | B와 Y → γ 1 | γ 2 | ... | γ n이 된다.
  suffix : X → α 1β | α 2β | ... | α nβ | B는 X → Yβ | B와 Y → α 1 | α 2 | ... | α n이 된다.
  bifix : X → $\alpha\beta$ 1γ | $\alpha\beta$ 2γ | ... | $\alpha\beta$ nγ | B는 X → α Yγ | B와 Y → β 1 | β 2 | ... | β n 이 된다.

- **축소** : 축소 과정은 문법 생성 규칙에서 같은 규칙을 합치고, 새로운 비단말 기호를 만들어내는 과정이다.
  X → α 1 | α 2 | ... | α n은 X → Y | α i+1 | α i+2 | ... | α n과 Y → α 1 | α 2 | ... | α i가 된다.

- **제거** : 제거 과정은 문법을 전체적으로 확장한 후에 필요없는 규칙을 제거하는 과정이다. 제거하는 과정에서 주의할 점은 문법 규칙의 "의미"를 변환시켜서는 안된다. 예를 들면 문법 규칙 X → X | A는 X → A로 제거될 수 없으며, X → lower*val* | lower*id*는 X → lower가 될 수 있다.

- **재명명** : 모든 문법 규칙에서 판독성을 위하여

문법 규칙의 식별자를 재명명한다.

- **확장** : 문법 규칙의 확장 과정은 위의 4가지 규칙을 적용하여 전체적으로 확장될 수 있다. 예를 들면 다음과 같은 문법 규칙 A → a B C, B → b A, B → b, C → c는 A → a b A C, A → a b C, A → B, B → b A, C → c로 확장될 수 있다.

일반적으로 lex의 많은 버전들은 제한된 토큰 크기를 가지고 있다. ASN.1에서는 따옴표("")로 이루어진 문자열에서와 같이 매우 긴 토큰을 가질 수 있으므로 사용할 수 없다. 그러나 GNU의 flex[12]는 토큰 저장을 위한 메모리 할당을 동적으로 수행하므로 이러한 제약을 가지고 있지 않기 때문에 본 시스템 구현에 flex를 이용하였다.

3. ASN.1 & GDMO 스키마 테이블

ASN.1 & GDMO 템플릿 스키마 테이블은 ASN.1 & GDMO 컴파일러에 의해 컴파일된 문법 정보를 데이터베이스에 저장하기 위한 구조이다. ASN.1 & GDMO 컴파일러가 생성하는 중간 자료 구조는 객체지향 형태로 되어있다. 그래서 이러한 구조를 저장하는 스키마 테이블의 형태는 객체지향 데이터베이스가 적당하다. 객체지향 데이터베이스를 사용하면 자료 접근시 인터페이스가 쉽고, 객체지향 파라다임의 특성을 가진다는 장점이 있다. 데이터베이스 객체 모델을 설계할 때 중간 자료 구조를 정의한다. 중간 자료 구조의 형태는 C++ 언어의 클래스와 유사하다. 이를 위해 Objectivity의 DDL (Data Definition Language)을 사용하여 스키마 테이블을 구성하였다[13]. ASN.1 & GDMO에서는 Objectivity DataBase (ODB)를 사용하는데 그림 4는 ASN.1 & GDMO 모듈과 ODB를 나타낸다. 그림 5에서과 같이 ODB 구조는 다수의 Federated DataBase (FDDB), DataBase (DB), ContainedDataBase(CDB) 체계로 이루어져 있다. 가장 상위 계층인 FDDB는 ODB에서 여러개 만들 수 있으며, 하나의 FDDB는 다수의 DB를 포함할 수 있다. 따라서 하위 계층의 DB(혹은 CDB)를 생성하거나 열기를 할 경우, 상위 계층의 FD(혹은 DB)는 이미 열기가 수행되었을 때만 가능하다. 또한 FD를 삭제하였을 경우에 그 하위 계층에 있는 모든 DB 및 CDB가 함께 삭제된다.

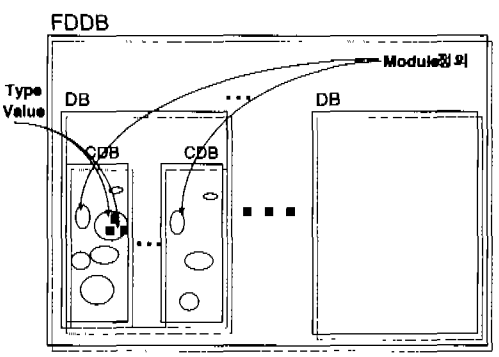

그림 4. ASN.1 & GDMO 모듈과 Objectivity DataBase 구조

그림 4에서 보면 ASN.1 & GDMO 모듈과 DB와의 연동 측면에서 살펴보면 다음과 같다. 하나의 모듈은 하나의 CDB로 정의한다. 모듈 내부의 값과 타입은 CDB 내부에 생성되어 들어간다. 따라서 CDB 내부에는 여러개의 객체들이 정의되어 있다. GDMO 템플릿들은 각각 하나의 CDB로 정의하였다. 9개의 템플릿에 해당하는 중간구조는 클래스 형태인 DDL로 구성된다. 이렇게 구성된 클래스는 GDMO 번역기와 브라우저에 의해 필요한 정보를 추출하여 DDL의 인스턴스에 삽입한다.

4. 원시 코드 생성기

GDMO 코드 생성기에 의해 생성되는 코드는 객체지향 파라다임을 제공하는 어떠한 언어로도 번역될 수 있다. 이를 위해서는 각 언어로 번역하는데 필요한 번역 규칙이 필요하다. 본 논문에서는 객체지향 언어인 C++로 변환하는데 중점을 두어 C++ 번역 규칙을 작성하였다.

```
class [PREFIX] behavior-label
  [POSTFIX]: {
 private:
 public :
  [PREFIX] behavior -label
  [POSTFIX]()
  ~[PREFIX] behavior -label
  [POSTFIX]()
  ERRORS Behavior();
  Attributed *Get- OID();
   ? Class Behavior behavior-label;
 /* other Behavior template... */
   ? Class Parameter parameter-label;
 /* other Parameter Template...*/
 /* 이하 생략 */
}
```

그림 5. 관리 객체의 번역 클래스

GDMO를 C++ 클래스로 변환하기 위한 기본적인 원칙은 GDMO 템플릿을 하나의 C++ 클래스로 변환하는 것이다. 템플릿 스키마 데이터베이스에 들어있는 각 템플릿의 값을 그림 5와 같이 GDMO 스키마 테이블의 형태로 가져오고 이를 원시코드 생성기가 입력받아 C++ 클래스를 생성한다. 그림 6은 관리 객체의 번역 규칙을 표현한 것이며, 번역 규칙은 각 템플릿에 각각 다르게 작성되어져 있다.

Ⅳ. ASN.1&GDMO 환경 구현 및 평가

본 시스템의 환경은 SunOS 5.5에서 구현되었으며 SPARC compiler version 4.0.1 C++를 사용하였다. 본 구현의 중간 자료 구조를 저장하는 데이터베이스는 객체지향 데이터베이스인 Objecti- vity를 사용하였으며, 중간 구조를 정의하기 위해 Objectivity의 DDL를 사용하였다. 구문 분석기에는 UNIX의 yacc을 이용하였으며, ASN.1 & GDMO 브라우저에는 Tcl/Tk[10]를 사용하였다.

본 시스템의 초기 화면은 그림 6과 같이 상단의 메뉴, 9개의 템플릿을 작성할 수 있는 선택버튼들,

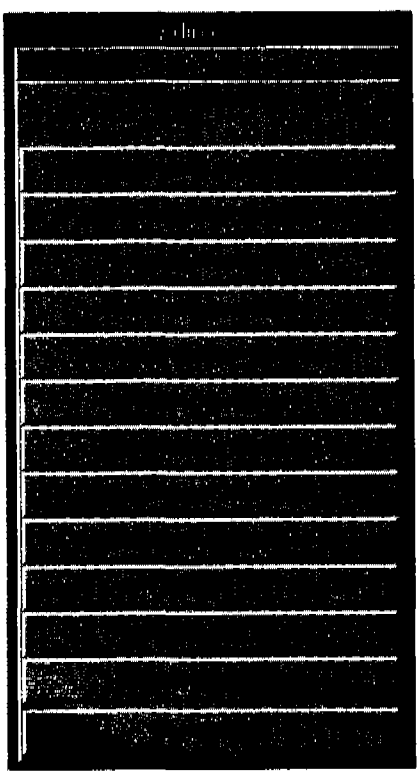

그림 6. 주 메뉴 화면

그리고 텍스트 편집기를 호출할 수 있는 버튼으로 이루어진다. 데이터베이스 메뉴에는 FDDB New, FDDB Open, FDDB Close, DB New, DB Open, DB Close, Contained DB New, Contained DB Open, Contained DB Close, Exit 가 있다. Tools메뉴 내에 있는 찾기 항목을 선택하면 두가지 방식으로 템플릿 찾기를 할 수 있다. 그 중 한가지는 템플릿의 이름을 직접 써서 찾는 방법이다. 또다른 한가지는 템플릿 찾기로서 여러 템플릿들 중에 하나를 선택하면 템플릿(예:관리 객체 클래스 템플릿 등등)에 해당하는 템플릿 목록들이 화면에 출력되고 그 중 하나를 선택하면 해당 템플릿화면이 제시된다. 각 템플릿 버튼을 클릭하면 템플릿 버튼명에 해당되는 템플릿 편집기가 나타나게 된다. 이 템플릿 편집기를 이용하여 템플릿의 내용을 삽입 및 변경을 할 수 있다.

그림 7은 ASN.1 버튼을 클릭하였을때 나타나는 브라우저로서 각 타입과 값의 정보를 표시하여 준다. 또한 포함 트리와 상속 트리 버튼은 각 템플릿 간의 포함, 상속 관계를 그림으로 표시하여 준다. GDMO의 Managed Object Class 템플릿은 Derived From이라는 문법을 가지고 있다. 이것은 다른 Managed Object Class 템플릿에 의해 상속을 받을 수 있다는 것을 의미한다. 이렇게 한 Managed Object Class가 다른 어떤 클래스에 의해 상속되었는지를 그래픽으로 보여 줌으로써 사용자의 이해를 도와주기 위한 도구이다.

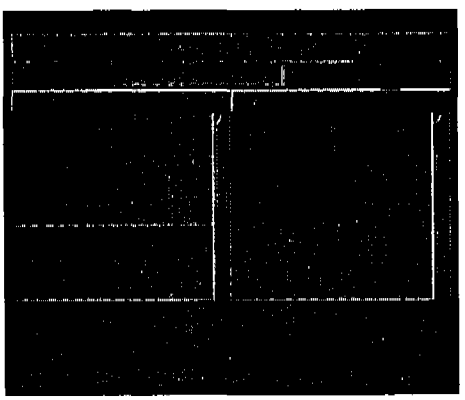

그림 7. 타입 및 값 브라우저

본 시스템의 특징은 네트웍 관리를 위한 통합 환경을 제공하는 것이다. 본 시스템은 표준안('93)에 개정된 ASN.1 및 GDMO 명세 언어를 완전히 파싱하여 처리할 수 있으며, 데이터베이스와 연동이

가능하도록 구현하였다. 구현을 위한 인터페이스 언어로 Tcl/Tk를 사용하였는데, 이 언어의 특성상 버튼의 2번을 클릭해야 메스트를 받아들일 수가 있으며, 멀티 브라우저를 생성하기 위해서는 반드시 다른 프로세서를 생성하여야 하는 문제점이 있었다. 다음의 표 1은 본 시스템과 기존의 ASN.1 연구 사례의 특징을 비교하였다.

표 1. ASN.1 연구 사례 비교

특징 / 컴파일러	DB와의 연동	원시코드 생성	브라우저	특징	비고
MAVROS	지원 안함	C, header 및 makefile	지원 안함	전처리기와 후처리기, 디버깅 도구 지원	
BBN	지원 안함	C++	지원 안함	거의 모든 OS에 이식가능, 자동 구문 검사	버전 1.4
Snacc	지원하지 않으나 링킹가능	C, C++	지원	IMPORTS/EXPORTS 링킹 가능	버전 1.2
CASN1	지원 안함	C	지원 안함	3 단계로 구성	
본 시스템	**지원**	**C++**	**지원**	**DB와 연동, ASN.1 & GDMO 통합환경 지원**	

V. 결 론

본 논문은 네트웍관리 시스템에서 ASN.1 & GDMO의 프로그램 개발환경을 구현하였다. ASN.1 & GDMO는 다양한 인터페이스나 통신 매체를 통하여 전송되어야 할 정보를 기술하기 위한 언어로 ASN.1 & GDMO 언어의 개발 환경은 네트웍 관리 정보 모델에서 신뢰성 있는 네트웍 기능을 제공하게 된다.

이를 제공하기 위해서는 TMN 기능을 지원하는 정형화된 소프트웨어 기본구조의 연구와 네트웍 관리 정보모델을 기반으로 하는 자동화된 네트웍 관리 원시코드 발생기가 필요하다. 따라서 본 논문은 네트웍관리 시스템에서 ASN.1 & GDMO의 프로그램 개발 환경 모델을 설계하고 구현하였다. 또한 사용자 인터페이스를 강화하기 위해 다양한 그래픽 브라우저로 구현되었으며, 객체지향 파라다임을 가지는 ASN.1 & GDMO 스키마 테이블을 설계하고 구현하였다. 또한 ASN.1 & GDMO를 다양한 객체지향 언어로 번역할 수 있는 원시 코드 자동 생성기를 구현하였다.

향후에는 본 연구에서 생성된 결과 물을 토대로 ASN.1 및 GDMO 유틸리티를 위한 라이브러리를 구현하여야 하며, 데이터 베이스에 저장된 목적 코드를 이용하여 코드의 단일화 및 정보 교환에 필요한 응용 프로그램을 개발하여야 한다. 또한 ASN.1 & GDMO 개발 환경 상에서 객체지향 언어의 생성에 필요한 여러 단계를 최적화하여 보다 효율적인 ASN.1 & GDMO 환경을 구축하여야 할 것이다.

참 고 문 헌

[1] U. Black, *Network Management Standards*, McGraw-Hill, 1994.

[2] D. Steedman. *ASN.1 The Tutorial and Reference*, 1990.

[3] J. Embry, *ASN.1/C++ Application Programming Interface*, Issue 1.0, Draft 9 - Submission to X/Open. 1996.

[4] B.Stroustrup, *The C++ Programming Language*, Addison Wesley, 1991.

[5] C. Huitema, *General Presentation of the MAVROS Compiler* ,INRIA, 1990, available via at http://www-sop.inria.fr/rodeo/mavros/ mavros-home.html.

[6] IOS developers, *BBN Systems and Technology ASN.1 Compiler* version 1.4, 1995, available via at http://ests.bbn.com/ASNSRC.html.

[7] M. Sample, *Snacc ASN.1 Compiler version 1.2*, 1997, available via at http://www.fokus.gmd.de/ovma/mug/ archives/mug-software/snacc.html.

[8] ISODE, 1997, available via at http: //ftp.doc.ic.ac.uk/packages/isode.

[9] N. Drakos, *CASN1-ASN.1 to C Compiler*, 1995, available via at http://www.atri.curtin.e.edu.au/~duke/honours/compiler/casn1/casn1.html.

[10] J. K. Ousterhout *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

[11] R. Corbett, *BYACC parser generator version 1.9*, 1993, available via anonymous ftp.cs.berkeley.edu:/ucb/ 4bsd/byacc.tar.Z.

[12] V. Paxson, *On-line manual pages distributed with the Flex software package*, 1990, available via anonymous ftp from ftp.ee.lbl.gov or prep.ai.mit.edu.

[13] Objectivity Inc, *Objectivity DB C++ Application Development : Version 3*, 1995.

유 재 우(Chae-Woo Yoo) 정회원
1976년 : 숭실대학교 전자계산학과(공학사)
1978년~1985년 : 한국과학기술원 전산학과 (공학 석사, 박사)
1986~87, 96~97년 : Cornell Univ.와 Univ of Pittsburgh의 Visiting Scientist
1983년~현재 : 숭실대학교 컴퓨터학부 교수
<주관심 분야> 컴파일러, 프로그래밍 환경, 인간과 컴퓨터

김 영 철(Young-Chul Kim) 정회원
1990년 : 한남대학교 전자계산학과 졸업
1996~현재 : 숭실대학교 컴퓨터학과 박사과정
<주관심 분야> 컴파일러, 웹 프로그래밍 언어, 망 관리

김 성 근(Sung-Keun Kim) 정회원
1993년 : 숭실대학교 전자계산학과 졸업
1995년~현재 : 숭실대학교 컴퓨터학과 박사과정
1998년~현재 : 가톨릭상지대학 컴퓨터정보계열 교수
<주관심 분야> 프로그래밍 환경, 에이전트 프로그래 밍 언어