'98 International Conference Digital Library & Knowledge
September 22, 1998, Seoul Korea

네트워크 관리 리파지토리 구축: 분산 객체 기술 중심으로

신 재호[1] 이 희석[2]

Implementing Network Management Repository From Distributed Object Technology Perspective

Jaeho Shin Heeseok Lee

<Abstract>

With rapid progresses in information technologies, managing enterprise network resources as a whole becomes more important. Telecommunication Management Network (TMN) has been used to integrate network elements. TMN is complex to develop and manage because it has a centralized management service. Common Object Request Broker Architecture (CORBA) can overcome this complexity by the use of transparent distributed object processing mechanism. Therefore, the main objective of this paper is to integrate TMN service with CORBA. A network management repository is built for this integration. In addition, a prototype that can partially support network management service is implemented to illustrate the usefulness of this repository.

---

[1] LG 정보통신㈜

[2] 한국 과학 기술원 테크노경영대학원

1 서론

통신 시장의 개방화 및 글로벌화는 통신 사업자들에게 새로운 경쟁 환경을 만들고 있다. 서비스의 통합성, 복잡성, 경쟁성은 통신 산업의 계속적인 발전을 요구하며 사용자들의 서비스에 대한 요구사항은 빠르게 변화되고 이에 따라 망 관리 또한 복잡해지고 있다. 네트워크가 기업의 중요한 자산으로 인식되어 이것을 어떻게 적절하게 활용하느냐 하는 것이 오늘날 기업의 성패를 좌우한다고 해도 과언이 아니다. 현재 이용되는 망 관리 프로토콜을 살펴보면 크게 SNMP(Simple Network Management Protocol)와 TMN(Telecommunication Management Network)으로 대별된다. 이들에 대해 간단히 살펴보면 다음과 같다.

SNMP 는 TCP/IP 환경에 기반을 두고 있는 인터넷 망의 관리를 위한 표준 프로토콜로 사용하고 있으며, 개발의 용이성, 저렴한 비용 등의 여러 장점이 있기 때문에 근거리 통신망 분야에서 많은 성장을 하고 있다. 그러나 많은 장점에도 불구하고 SNMP 의 여러 문제점(보안 문제, 다수의 요구에 대한 제약[11] 등) 때문에 대규모 시스템 관리에는 적합하지 않다. 따라서 새로운 관리 모델의 필요성이 대두되었고 이에 따라 TMN[2]이 등장하여 시스템 관리의 표준으로 채택되었다.

TMN 은 이질적인 정보통신망 구성 요소의 일원화된 통합 관리를 목표로 하고 있으며 OSI 시스템 관리 서비스 및 CMIP(Common Management Interface Protocol)[4] 기반의 관리 기능들을 이용하여 네트워크 요소들과 관리 시스템간에 관리 정보를 교환 및 처리함으로써 통합적으로 통신망을 관리한다. TMN 은 SNMP 에 비해 관리 요소들 중에서 처리될 객체를 식별하는 스코핑 기능과 필터링 기능 등을 제공하는 강력한 정보 모델이지만 설치하기가 복잡하고 비용이 많이 드는 단점이 있다. 그런데 최근에는 개방형 환경에서 분산처리를 지원하기 위하여 CORBA(Common Object Request Broker Architecture)[8]를 사용하여 TMN 에서 개방형 분산 관리를 지원하려는 시도가 있다.

CORBA 는 OMG(Object Management Group)에서 분산 객체지향 시스템의 표준을 마련하고자 하는 목적에서 제안한 표준안으로 분산 객체 지향 애플리케이션 개발을 가능하게 하는 중요한 기술이다. OMG 의 OMA(Open Management Architecture)에 상위레벨의 CORBA Service 와

CORBA 편의(CORBA Facility)들이 계속적으로 지원됨으로써, CORBA 에 기초한 아키텍쳐를 더욱 더 향상시키고 있으며 CORBA 를 더 많은 플랫폼에 적용하여 상호 운용성을 지원하도록 하고 있다. 이에 의해 CORBA 는 통신망 관리 표준과 다른 정보기술과의 차이(Gap)를 줄여주는 것으로 인식되어 지고 있다[13]. 네트워크 관리 표준인 TMN 과 CORBA 기술을 통합하는 것은 관리 애플리케이션이 표준 네트워크 관리 프로토콜을 지원할 뿐만 아니라 분산 객체를 이용하여 개발되도록 한다. 분산 객체를 사용함으로써 다양한 프로토콜, 애플리케이션들을 통합하기 쉽도록 하는 유연한 관리 구조를 제공한다.

본 논문에서는 이러한 요구 사항에 적합한 새로운 모델을 제안하고 이에 적용 가능한 기술을 제시하며, 이를 통하여 기존의 망관리 시스템 통합에 관한 방안을 제시하고자 한다. 본 논문은 다음과 같은 순서로 구성되어 있다.

제 1 장 서론에 이어 제 2 장에서는 TMN 에 대한 개요 및 아키텍처, 프레임워크, 관리 기능들에 대해 살펴본 후 분산 컴퓨팅의 표준인 CORBA 에 대한 개요 및 구성 요소들과 그들 상호간의 연결에 대한 아키텍처를 살펴본다. 제 3 장에서는 TMN 과 CORBA 와의 연결의 필요성 및 연결 방법을 제시하고 제시된 연결 방법들에 대한 장단점을 비교해 본다. 제 4 장에서는 3 장에서 제시한 연결 방식 중 하나를 택하여 그에 따르는 효율적인 관리를 위한 객체 리파지토리(Object Repository)를 제시한다. 제 5 장에서는 4 장의 모델을 기초로 관리 시스템을 개발하여 현재 진행중인 ATM(Asynchronous Transfer Mode) 교환기 개발 프로젝트에 적용하고 그 효율성에 대해 알아본다. 마지막으로 제 6 장에서는 본 연구의 성과를 정리하여 결론을 맺고 향후 과제에 대하여 설명한다.

2 TMN 및 CORBA

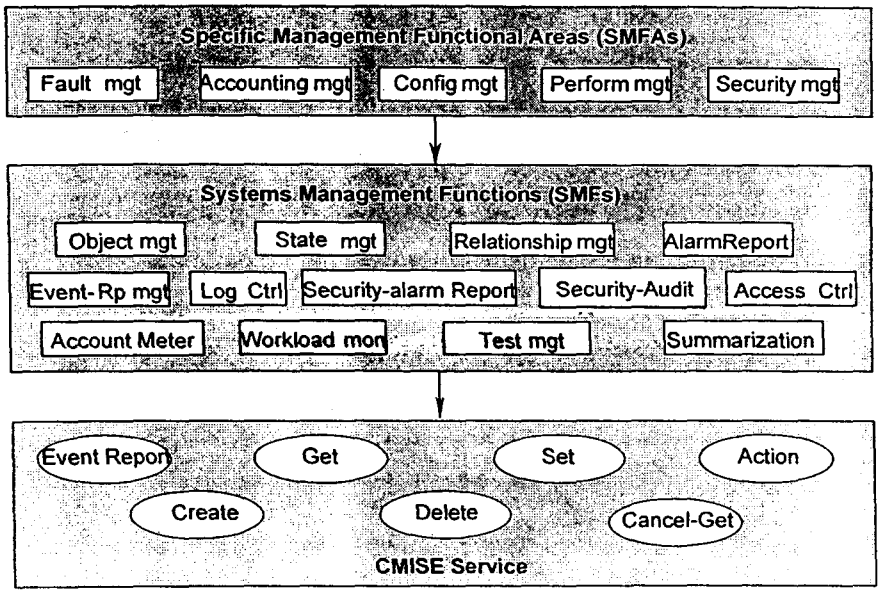
2.1 TMN (Telecommunication Management Network)

TMN 이란 전기 통신망과 서비스를 관리하기 위하여 운용 시스템과 통신 망 구성 장비들을 표준 인터페이스로 연결하고, 이 인터페이스를 통해 필요한 관리 정보를 상호 교환하는 논리적인 구조를 의미하며, 전기 통신망 관리를 체계적으로 지원하는 하부 구조이다[1]. TMN 은 다른 종류의 관리 시스템과 통신 장비들이 관리 정보를 서로 공유할 수 있도록 조직화된 아키텍

쳐를 지원한다. TMN의 가장 큰 특징은 객체 지향형 모델링 기술에 의한 망 관리 인터페이스를 규정한 것으로, 네트워크 관리는 이 네트워크 모델을 다룸으로써 행하여 진다. 이 모델은 관리되어지는 실제 네트워크 요소들에 대한 추상화(Abstract)된 관점을 나타내며 이러한 모델링 기술은 앞으로 전개될 다양한 정보통신 서비스의 통합관리 및 재사용을 위한 기반이 된다. TMN 기반의 망 관리 기술은 통신망의 고도화, 지능화 및 개방화 추세에 알맞은 표준모델을 제시함으로써 일원화된 망 관리 체계를 구성할 수 있도록 한다. TMN은 통신망과는 개념적으로 분리되어 있으며 여러 종류의 참조점을 설정하여 분산 환경하에서의 여러 개의 운용 시스템(Operation System)이 실현될 경우에 운용 시스템간의 상호 운용성(Interoperability)을 제공한다. 이 참조점을 통하여 망 관리 정보를 교환하며, 교환된 정보에 의하여 관리 대상인 통신망과 통신망 구성요소(교환기, 전송 시스템, 모뎀, 소프트웨어, 연결 실패와 같은 통계 정보 등)의 상태를 감시하고 적절한 제어를 가능하게 한다[1][21].

TMN 기능에는 망 관리자에게 시스템과 각 계층의 성능을 측정하고 관찰할 수 있게 해주는 성능 관리(Performance Management)기능, 망 관리자가 통신망과 관련된 문제점들을 인지하여 네트워크 상의 장애를 식별해 내고 결점의 원인을 고립시킨 후 결점을 바로 잡도록 하는 장애 관리(Fault Management)기능, 네트워크 관리 프로토콜을 사용하여 데이터를 수집하여 망 관리자가 망 요소와 OSI 관리 요소에 대한 제어를 할 수 있도록 하는 구성 관리(Configuration Management)기능, 망 관리자가 비용을 산출하고 망 자원 사용에 따른 비용을 부과하는 역할을 담당하는 과금 관리(Account Management)기능, 보호되어야 할 정보의 액세스 지점을 찾아 암호화, 인증 등의 방법으로 망 관리자가 통신 자원의 접근 보안에 관련된 서비스를 제공하는 보안 관리(Security Management)기능 등의 5가지 기능이 있다.

TMN 관리 프레임워크에서 정의한 관리 기능 분야는 망 관리자가 책임지어야 할 여러 분야를 설명하고 있는데 이들 분야는 서로 중복되는 경우가 많다. 따라서 다섯 개의 기능 분야로 표준화되지 않고 시스템 관리 기능이라고 불리는 여러 특정 기능을 정의한다. 하나의 시스템 관리 기능은 하나나 혹은 그 이상의 관리 분야를 지원할 수 있다. <그림 2-1>은 시스템 관리 기능[17]을 보여주고 있다.

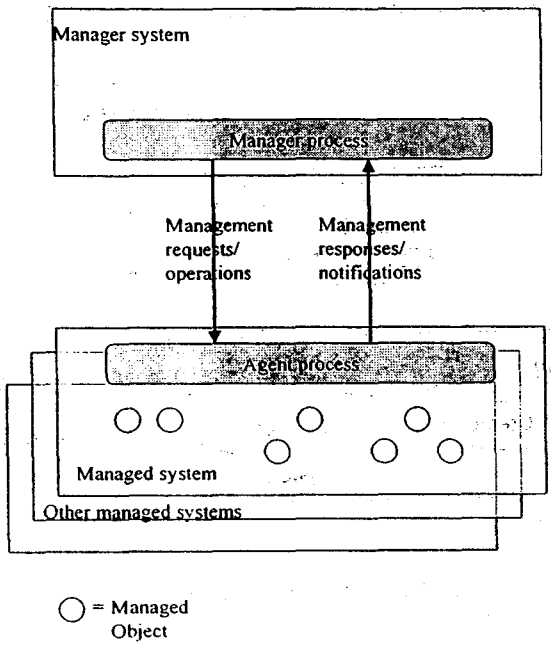


<그림 2-1> 시스템 관리 기능(SMF)

TMN 관리 정보 모델은 OSI 시스템 관리 정보에 대한 일반적 관리 정보를 제시하며, 관리 객체의 정보 모델과 그들의 속성, 관리 객체의 이름부여(Naming)규칙, 시스템 관리 정보의 논리적 구조, 관리 객체 클래스들의 개념과 관리 객체간의 관계 등의 정의를 제공한다[2]. 관리 객체는 그것들이 포함하는 속성들과 속성들에 대해 수행되는 동작 명령들, 각종 통고(notification) 그리고 다른 객체들과의 관계를 통하여 표현된다. MIB(Management Information Base)정보의 구조에 의하여 각 관리 객체들은 관리 객체 클래스의 한 구성원이 된다.

TMN 관리 동작 서비스에는 관리 정보 베이스로부터 자료를 가져오는데 사용되는 M-GET, 관리 정보 베이스의 자료를 수정할 수 있는 M-SET, 관리 객체의 일부로 명시된 절차와 동작을 실행시킬 수 있는 M-ACTION, 객체 클래스로부터 인스턴스를 만드는 M-CREATE, 관리 정보 베이스로부터 하나 또는 다수의 객체를 제거하는데 사용되는 M-DELETE, M-GET 요구의 결과가 긴 메시지일 경우 동작을 중지시키는데 사용되는 M-CANCEL-GET 서비스 등의 6 가지가 있다. TMN 에서 Manager-Agent 구조는 <그림 2-2>와 같다[17]. 그림에서와 같이 관리자, 관리 대행자 역할로 관리 기능이 구분되어지고 이들간의 상호 연결은 서비스와 프로토콜이 표

준화되어 있다.



<그림 2-2> Manager-Agent 구조.

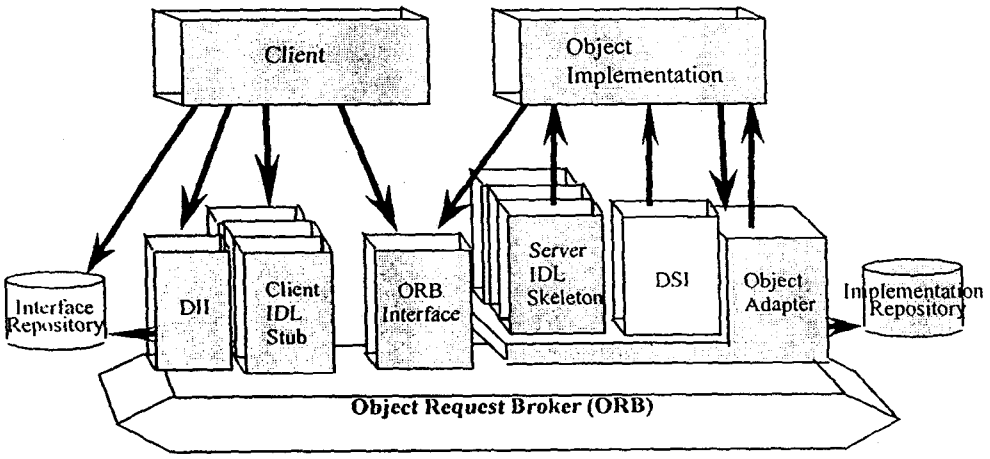
2.2 CORBA (Common Object Request Broker Architecture)

1989 년에 탄생한 비영리 단체인 OMG 의 주된 목표는 분산 객체를 지원하는 분산 애플리케이션에 대한 표준화된 프레임워크를 만드는 데 있다. OMG 는 객체지향 기술을 기반으로 이종의 분산된 환경 하에서 응용 프로그램들을 통합하고, 상호 연동할 수 있는 표준기술을 제정했는데, 바로 이 표준이 OMA(Object Management Architecture)이다[8].

OMA 는 응용 프로그램간의 통합뿐만 아니라 객체의 생성, 소멸에서부터 저장, 트랜잭션(transaction) 기능에 이르기까지 분산 객체 환경에서 필요한 모든 서비스를 통합할 수 있는 표준 프레임워크이다. 이들 기능 중 CORBA 는 컴퓨터 내부의 버스처럼 서로 다른 컴퓨터의 프로그램들 사이의 버스 역할을 하는 모듈로서 프로그래머가 원하는 메쏘드 등의 위치에 관계없이 마치 로컬에 있는 것처럼 사용할 수 있도록 하는 기능을 제공한다. CORBA 에서 가장 중요한 요소는 ORB(Object Request Broker)라 할 수 있는데, ORB 는 클라이언트의 요구를 목표객체에 전달하고 그 처리 결과를 클라이언트에 보내주는 역할을 수행하는 메커니즘과 인터페이스

들을 제공하는 객체 버스(Object Bus)이다. ORB는 객체들이 실행 시에 서로를 찾아내고 서로의 서비스를 호출할 수 있도록 한다. ORB는 정적 메쏘드 호출(Static Method Invocation)과 동적 메쏘드 호출(Dynamic Method Invocation) 모두를 지원한다.

<그림 2-3>은 CORBA ORB구조를 나타낸다. 클라이언트 응용 프로그램은 OMG IDL(Interface Definition Language)에 의해 정의된 객체들의 오퍼레이션들에 대한 요청을 호출한다. IDL은 선언적인 언어로서, 구현에 관한 상세 사항은 제외되어 있다. IDL을 사용하여 API를 정의하고, 운영체제나 언어에 독립적으로 호출이 가능한 개방형 인터페이스를 제공한다. 클라이언트와 서버는 ORB를 사용함으로써 서로간의 정보에 대해 저장할 필요가 없다. ORB는 클라이언트 애플리케이션으로부터 요청을 받아들여서 서버에 있는 구현 객체들 중에서 그 요청을 수행할 적당한 구현 객체를 선택하여 요청을 해당 구현 객체에게 보낸다.



<그림 2-3> CORBA ORB 구조

인터페이스 저장소(Interface Repository)는 네트워크상에 존재하는 응용 프로그램들과 데이터 객체에 대한 인터페이스들을 가지고 있다. CORBA IDL로 작성된 모든 인터페이스 정보는 인터페이스 저장소에 저장된다. 구현 저장소(Implementation Repository)는 구현 객체의 물리적인 위치나 서버 구동 방식, 사용 프로토콜 등 객체 어댑터가 구현 객체를 구동시키는데 필요한

정보들을 저장한다. 객체 어댑터(Object Adapter)의 역할은 구현객체 대신에 클라이언트에게 ORB 서비스를 제공하는 것이다. 요청을 받아들인 객체 어댑터는 해당 구현 객체를 호출하기 위해 구현 저장소를 참조한다. 객체 어댑터는 해당 구현 객체에 객체 참조를 부여하고 활성화 시키며 해당 메쏘드를 호출한다. 가장 일반적으로 쓰이는 객체 어댑터를 BOA(Basic Object Adapter)라고 한다. 모든 CORBA 벤더들은 그들의 시스템의 일부분으로 BOA 를 지원해야 한다. 서버 응용 프로그램은 하나 이상의 구현(Implementation)을 가지고 있는데, 이것들은 특정 객체의 오퍼레이션에 대한 클라이언트 요청을 수행하는 서버의 일부분이다. 클라이언트가 ORB 에게 한 요청을 보냈을 때 ORB 는 그 요청을 만족하는 구현을 선택하고 그 구현으로 직접 요청하기 위해 BOA 를 사용한다. BOA 는 그 구현 안의 메쏘드들을 부르기 위해 서버 스켈레톤(Server Skeleton)을 사용한다. 또한 서버 스켈레톤은 BOA 와 한 객체의 오퍼레이션을 수행하는 메쏘드들의 연결을 제공한다. 한편 CORBA2.0 에서는 이기종 ORB 간의 상호 연동을 위한 아키텍처를 정의했다. 일반적인 ORB 간 아키텍쳐는 GIOP(General Inter-ORB Protocol)에 기초하고 있으며 TCP/IP 전송을 위해 CORBA 표준을 준수하는 모든 ORB 제품에 IIOP(Internet Inter-ORB protocol)지원을 의무화하였다. IIOP 는 어떻게 GIOP 가 TCP/IP 전송상에서 구현될지를 결정한다. 이러한 상황에서 클라이언트 메쏘드 호출은 물리적으로는 클라이언트, DSI, DII, 그리고 서버 순서의 서비스 요청 과정을 거치지만 실제 사용자에게는 서버가 클라이언트와 지역적으로 동일한 곳에 존재하는 것으로 보인다.

3 CORBA 기반 TMN 통합 방법

3.1 통합 필요성

통합의 필요성을 살펴보기 위해 먼저 CORBA 와 TMN 를 서로 비교해 보면 다음과 같다. 분산 하부 구조를 살펴보면 분산 객체를 서비스하기 위해 CORBA 는 객체 버스를 제공하여 이를 여러 응용 객체들이 공유하여 사용하는 반면, TMN 은 포인트-포인트 통신만 이루어질 수 있다. 표준 API 로는 CORBA 는 여러 가지 언어들에 대한 인터페이스를 제공하는 반면, TMN 은 C 언어나 C++만을 제공한다. 한편 CORBA 는 자바 인터페이스를 제공함으로써 Web 액세

스가 가능한 반면, TMN은 어렵다. TMN이나 CORBA 둘 모두 실시간 서비스 지원을 하고 있으며, CORBA가 산업 전반적인 분야를 지원하는 반면 TMN은 통신 분야 관리를 위주로 하고 있다. CORBA는 TMN 정보 모델을 제공하기 위해 IDL로 변환이 필요한 반면, TMN은 GDMO[5]와 ASN.1을 기반으로 관리대상을 모델링하고 CMIP 프로토콜을 사용하여 통신함으로써 TMN 정보 모델을 제공한다. 한편 CORBA는 트레이더 서비스에 의해 객체의 이동이 자유로운 반면 TMN은 어렵다. <표 3-1>에서 CORBA와 TMN 간의 비교를 표로써 정리하였다.

<표 3-1> CORBA와 TMN 비교표>

항목 \ 표준 기술	CORBA	TMN
분산 하부 구조	분산 객체를 위한 객체 버스 제공	포인트-포인트 통신을 위한 파이프(Object pipe) 제공
표준화 기구	OMG	ISO, ITU-T
표준 API	OMG가 정의한 java, C, C++, Smalltalk, COBOL API	NMF(network Management Forum)가 정의한 C++
Web 액세스 가능성	Java지원에 의해 가능	불가능
실시간 서비스 지원	가능	가능
도메인	산업의 전반적 분야 지원	통신 분야 관리 위주
관리 기능 지원	개발중	지원
TMN 정보 모델	IDL로 변환 필요	GDMO/ASN.1에 의해 가능
이동성	가능	불가능

CORBA는 분산, 인터페이스, 통합 문제를 해결해주며, 통신 환경에서 분산 관리 애플리케이션을 개발하는 기술이지만 네트워크 관리 아키텍쳐를 지원하지 않는다. 즉 분산 객체 컴퓨팅 아키텍처만을 지원한다. 따라서 CMIP과 같이 표준 관리 프로토콜을 지원하는 기존 시스템과

상호 통합이 가능해야 한다. 이와 같이 두 기술을 통합하도록 하는 프레임워크를 개발하는 것은 다음과 같은 장점을 제공할 수 있다.

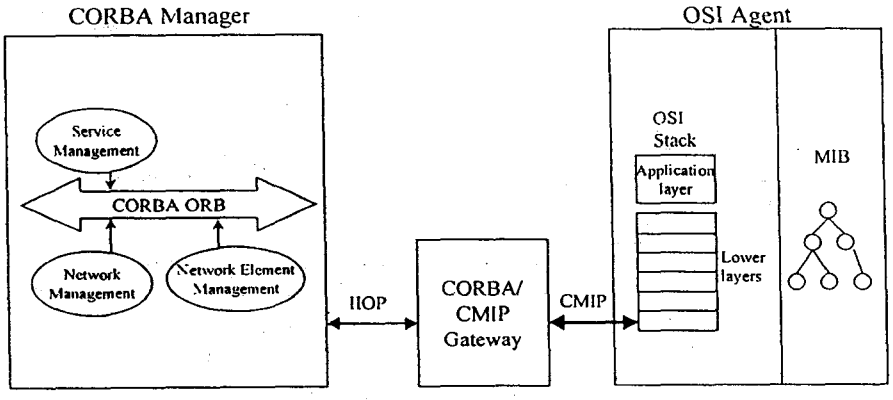
첫째, 기존의 시스템과 미래의 CORBA 방식의 시스템과의 연동을 쉽게 할 수 있다. TMN의 개념을 그대로 이용하며 CORBA로의 전환을 도모하는 것이 바람직하다. 둘째, 기업에게 GDMO(Guideline for the Definition of Managed Object)에 기반한 기존 TMN 정보 모델에 투자를 지속할 수 있도록 한다. 셋째, 현재 기업 내에 존재하는 TMN 시스템을 CORBA 프레임워크로 점진적으로 대체할 수 있도록 한다. 넷째, TMN 시스템은 제한된 명령어 집합인 CMIP 명령어만을 지원하고 있어 그에 따른 관리자와 관리 대행자간의 대규모 검색 명령어 실행 시에 여러 번의 명령어가 필요하여 네트워크에 부하가 많은 반면 CORBA는 하나의 명령어 인터페이스를 이용하여 처리할 수 있다. 다섯째, TMN에서 사용되는 CMIP에 의한 프로토콜수준의 관리보다 CORBA 객체 하부구조(Object Infrastructure)를 따르는 API 수준의 상호 운용기술이 관리 시스템을 만들기에 더 쉬운 방법을 제공한다.

3.2 통합 모델 비교

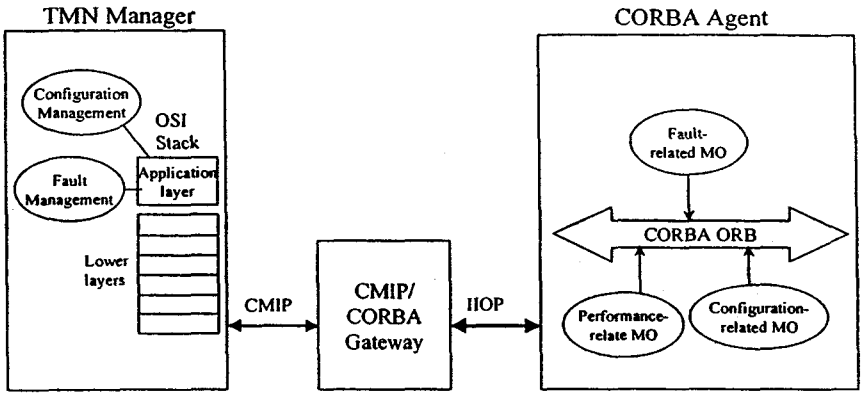
통합 모델로는 <그림 3-1>, <그림 3-2>, <그림 3-3>과 같은 모델들이 있다. 편의상 이들 각각을 다음과 같이 표현한다. 즉 첫번째 방법을 CMTA(CORBA Manager-TMN Agent)라 하며 이는 관리자 시스템으로써 CORBA를 사용하며 관리 대행자를 TMN시스템을 통해 구현하게 된다. 두번째 방법을 TMCA(TMN Manager-CORBA Agent)라 하며 이는 첫째 방법과 반대로 관리자 시스템 구현 시에 TMN을 이용하게 되며 관리 대행자로는 CORBA를 이용하여 구현하게 된다. 세번째 방법은 CMCA(CORBA Manager-CORBA Agent)라 하며 이 방법은 게이트웨이가 없고 관리자와 관리 대행자 모두 CORBA로 구축한 환경을 나타낸다.

<표 3-2>는 위의 3가지 방법을 다음과 같이 비교하고 있다. 첫째, 웹과의 연동을 살펴본다. CORBA환경은 Java등을 이용하여 쉽게 연동이 되지만 TMN환경은 어렵다. 둘째, 융통성(Flexibility)이란 시스템의 확장이 얼마나 쉬운가를 나타낸다. CORBA환경이 TMN보다 더 좋은 확장성을 지원한다. 셋째, 가용성(Availability)이란 현재 구축된 망 관리 시스템이 쉽고 비용이 적게 들면서 새로운 환경에 적용될 수 있는가를 살펴본다. CORBA Manager-CORBA

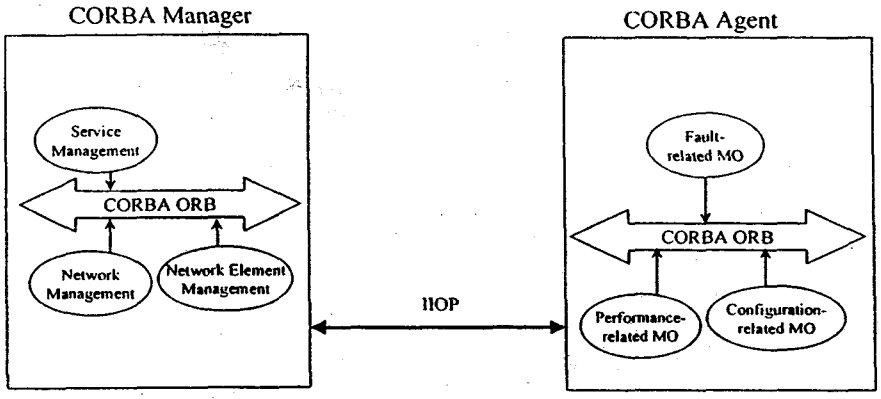
Agent시스템은 현재 사용하는 망 관리 시스템을 전반적으로 재 구성하여야 하기 때문에 상대적으로 비용이 많이 들고 이전하기도 어렵다. 넷째, Workflow등과 통합이 쉬운가를 살펴본다. CORBA환경의 관리자를 제공하면 분산객체 기술을 이용하여 다른 기업 정보망과의 통합을 쉽게 만들 수 있다. 다섯째, 망을 모델링하는데 노력이 얼마나 드는가를 살펴본다. 이 세가지 모델 중에서 웹을 통한 관리의 효율성과 융통성이 뛰어나며 통합 관리가 용이한 방법인 CMTA에 의해서 관리 시스템 모델을 구축하고자 한다 이는 현재 IT통합이라는 요구 사항에도 가장 적합하다. TMN과 CORBA를 통합할 경우 망 관리 서비스에서 사용할 수 있는 CORBA서비스들로는 표준화된 객체의 생성, 삭제, 이동, 복제를 담당하는 생명주기 서비스, 객체의 이름으로 객체를 찾도록 해주는 명명 서비스, 객체의 성격으로 해당 객체를 찾아주는 트레이더 서비스, 객체를 지속적으로 유지하도록 해주는 영속성 서비스(Persistent Service), 사건 처리를 담당하는 이벤트 서비스, 그리고 보안 서비스 등이다.



<그림 3-1> CMTA(CORBA Manager-TMN Agent)



<그림 3-2> TMCA(TMN Manager-CORBA Agent)



<그림 3-3> CMCA(CORBA Manager-CORBA Agent)

<표 3-2> 3가지 방법 비교표

방법 / 항목	CMTA	TMCA	CMCA
웹과의 연동	쉽다	어렵다	쉽다
융통성(Flexibility)	좋다	보통	좋다
가용성(Availability)	좋다	보통	적다
WM 등의 기업 정보통합 관리 기능	좋다	보통	좋다
모델링 난이도	많은 노력	적은 노력	많은 노력

3.3 CMTA 통합 모델

JIDM(Joint X/Open and Network Management Forum Working Group)에서는 현재 TMN 과 CORBA 를 통합하기 위한 방안을 제안하고 있다[13]. <그림 3-4>는 JIDM 에서 제안한 참조 모델(Reference Model)[16]을 바탕으로 하는 관리시스템을 나타낸다. 이 그림에서는 관리 시스템으로써 CORBA 를 사용하고 있다. 그리고 CMIP 과의 통합을 위해 관리 객체 인터페이스를 두고 있어 이 인터페이스가 게이트웨이 역할을 수행하게 된다. 즉 이 모델은 첫번째 방법인 CMTA(CORBA Manager-TMN Agent)에 해당된다. 이와 같이 OMG 내의 그룹인 JIDM 은 CORBA 와 다른 망 관리 시스템과의 연동에 대해 연구하고 있다. 세부 내용에 대해 알아보면 다음과 같다.

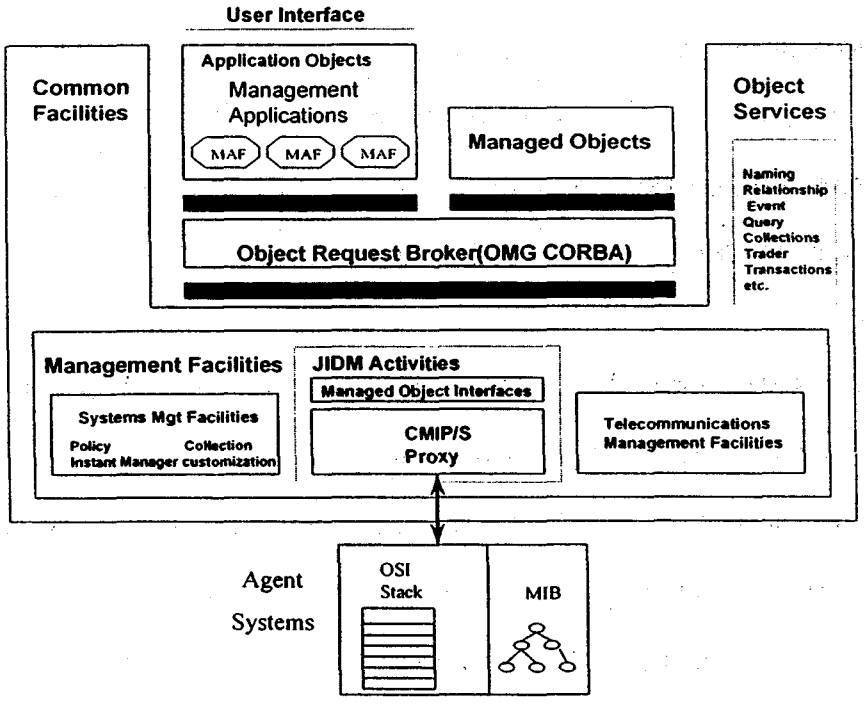
첫째, TMN 에서의 ASN.1 데이터 타입은 IDL 타입으로의 변환을 위하여 헤더 파일 내에 typedef 문을 사용하여 IDL 에 맞는 구조(Structure)를 미리 정의하여 둔다. 이것을 #include 문을 사용하여 import 하게 된다. 예를 들어 ASN.1 파일내에 정의되어진 CHOICE 타입에 대해서 ChoiceStruct 타입으로 정의해 놓으며 이를 ASN.1 IDL 에서 import 하여 번역하게 된다.

둘째, GDMO 를 IDL 로 번역하기 위해서는 좀더 복잡하다. GDMO 에서의 클래스 간의 상속 관계는 IDL 파일에서 인터페이스들 간의 상속 관계로 번역되어 진다. 또한 각 속성

(Attribute)과 액션(Action)들은 IDL 파일에서 인터페이스 내의 오퍼레이션으로 번역되어 진다. 또한 각 통보(Notification)들은 OMG 이벤트 서비스에서 사용되는 오퍼레이션으로 번역되어 진다. 한편 파라메타 템플릿은 IDL 타입 정의로 바뀌어지며 행위(Behavior)는 IDL 에서 설명문(Comment)으로 바뀐다.

셋째, IDL 이 GDMO 로 번역되기 위해서는 각각의 IDL 모듈은 하나의 GDMO 문서로 대응되어지고, 각 IDL 인터페이스는 하나의 GDMO 관리 객체 클래스로 바뀌어진다. 또한 IDL 속성(Attribute)은 GDMO 속성(Attribute)으로 바뀌어지며 IDL 오퍼레이션은 GDMO 액션(Action)으로 번역되어지고 IDL 타입과 상수(Constant)들은 ASN.1 타입으로 번역되어 진다.

위의 방안은 미리 헤더파일로 정의함으로써 액세스 속도가 빠르다는 장점이 있다. 반면에 융통성이 떨어진다. 즉 GDMO 정의가 달라질 때마다 이에 따르는 정보를 갱신하기 위하여 매번 컴파일을 다시 해야 한다는 단점이 있다. 또한 관리자 애플리케이션을 개발하는 사람은 바뀐 내용에 대한 정보를 빨리 획득할 수 없다는 단점이 있다. 이러한 단점을 해결하기 위한 방안으로 위에서 제안한 객체 리파지토리(Object Repository)는 보다 유연성 있는 시스템을 개발하도록 한다.



<그림 3-4> CMTA 통합 모델

4. 객체 리파지토리 (Object Repository)

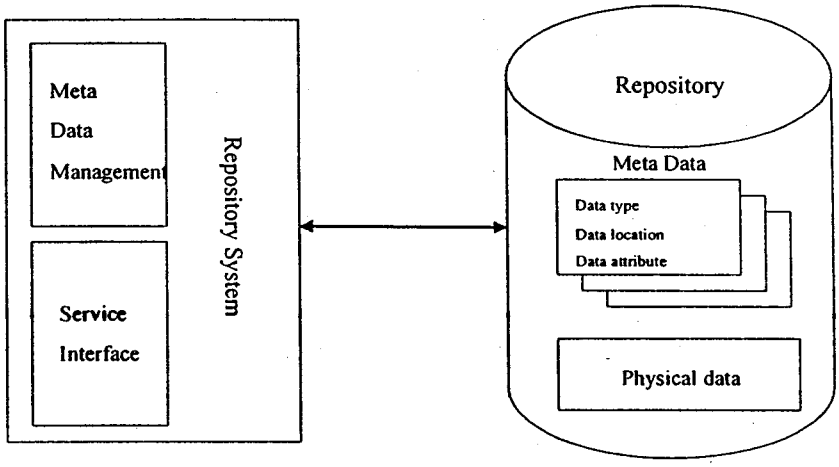
4.1 리파지토리 개요

리파지토리는 저장된 모델의 타입에 관계없이 통합된 데이터를 저장하는 장소이다. 그것은 여러 모델 타입들간의 상이한 점을 서로 관련지울 수 있다[14]. 또한 리파지토리는 기업의 업무처리와 자동화, 관련된 분석 정보 등을 유지, 관리하기 위한 지식 위주의 관리 시스템 환경을 말한다. 리파지토리는 데이타에 관한 정보, 즉 메타데이터(MetaData)라고 하는 정보를 저장하고, 논리 정보와 물리 정보사이를 연결하며, 그들 사이의 관계를 정의하는데 융통성(Flexibility)을 제공하며, 메타데이터를 관리하고 질의하는데 유용한 툴을 제공한다. 한편 리파지토리 애플리케이션은 신속하게 시스템을 개발하고 급속한 변화에 대처할 수 있도록 지원하며, 재사용성을 높이며, 소프트웨어 개발팀을 지원하는 기능을 제공한다. <그림 4-1>은 리파지토리 구조를 나타낸다. 그림에서 리파지토리는 메타데이터와 그에 따르는 물리적 데이터로 구성되며, 메타데이터는 데이터의 형태, 위치, 속성 등의 정보를 가진다. 한편 리파지토리는 리파지토리 시스템에 의해서 관리되어진다. 리파지토리를 구축하는 이유는 다음과 같다.

첫째, 협의의 수준에서는 시스템 구축 시에 공유되어지는 데이터를 공통 메모리에 저장하여 각 기능 블록들이 이것을 이용하도록 하는 것이다. 이 방법은 공통 메모리 사용에 따르는 효율적 관리가 이루어지도록 하는 장점이 있다.

둘째, 광의의 개념으로는 흔히 OM(Organizational Memory)이라고 하는 것으로써 조직에서의 경험적 지식을 서로 공유하여 조직의 노하우를 생산성으로 이어지게 하는 것을 말한다. 이를 이루기 위해서는 대규모의 데이터를 효과적으로 저장하는 기술이 필요하며, 의사결정을 지원할 수 있는 알고리즘이 필요하게 된다.

본 논문에서의 리파지토리 구축은 두번째 방법을 이용하고자 한다. 리파지토리는 시스템 개발에 필요한 자료의 메타데이타를 담고 있어야 한다. 예를 들어 Entity-Relationship Diagram(ERD)에 의한 메타스키마가 작성되면 그에 따르는 시스템의 정보들이 저장되어, 시스템 관리가 용이하며 유지, 보수가 쉽도록 설계되어야 한다.



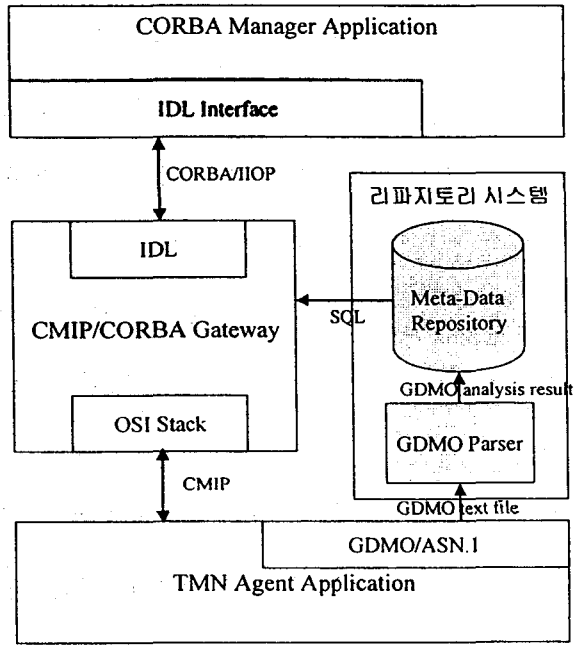
<그림 4-1> 리파지토리 구조

<그림 4-2>는 TMN환경하에서의 리파지토리를 구현하기 위한 구조를 제안하고 있다. 제안하는 리파지토리 시스템은 많은 다른 종류(Heterogeneous)의 정보를 데이터의 제구성없이 액세스할 수 있도록 한다. CMIS(Common Management Information Service) IDL은 CMIP 프로토콜의 자세한 내부처리를 감추면서 기능들을 수행하게 된다. 또한 GDMO Parser를 이용한 메타데이터를 사용하여 실시간으로 번역이 이루어지도록 한다. 리파지토리에 저장된 데이터는 정의된 IDL에 의해서 엑세스되어진다. CORBA 관리자는 IDL을 이용하여 관리 대행자 애플리케이션과 서로 통신하게 된다. 다음 절에서는 효율적인 통합을 위해서 제안하는 리파지토리 구조를 구축하는 방법에 대해 살펴본다.

4.2 객체 리파지토리 (Object Repository) 구조

앞에서 설명하였듯이 TMN은 관리 객체를 모델링하기 위하여 GDMO 즉, 객체 모델링 기법을 사용한다. 또한 CORBA도 인터페이스 정의어로 기술할 때 객체 모델링을 통해 상속 관계를 나타내고 있다. 이들을 통합하기 위해서 두 가지 모델을 저장할 수 있는 메타데이터 저장소를 구현하면 관리가 쉬워지고 효율적인 액세스가 가능하다. 메타데이터 저장소 구현에 따

르는 요구 사항은 다음과 같다. 첫째, 두 모델간의 전이(Transformation)가 쉬워야 한다. 둘째, 검색 등의 다른 작업에 사용하기 쉽도록 확장성이 있어야 한다. 셋째, 모델링 작업이 쉬워야 한다. 제안한 방법은 새로운 데이터 타입을 제공하고 색인(Indexing)기법을 제공한다.



<그림 4-2 > Manager-Agent 와 리파지토리 간의 관계

리파지토리는 구현 방법에 따라 데이터베이스를 이용하거나 파일 시스템을 이용하여 구축할 수 있다. 데이터베이스를 이용한 방법은 체계적으로 데이터를 관리할 수 있다는 장점이 있는 반면, 비용이 많이 들고 액세스 속도가 느릴 수 있다. 반면에 파일 시스템을 이용하는 방법은 구현이 어렵다는 단점이 있는 반면, 비용이 적게 들고 액세스 속도가 빠르다는 장점이 있다. 현재 구현을 위해서 데이터베이스를 이용한 방법을 채택하여 구축한다. 앞에서도 언급 한 것처럼 CORBA Manager-TMN Agent 시스템을 구축 시 CORBA-TMN 게이트웨이에 해당되는 기능 블록에 리파지토리를 구축함으로써 관리 대행자(Agent)에서의 요구가 바뀌어지면 바뀐 요구 사항을 관리자(Manager)가 쉽게 반영할 수 있는 구조를 제안한다.

<그림 4-2>에서와 같이 Manager 와 게이트웨이 간의 통신은 IIOP 를 이용하게 되며 게이트웨이와 관리 대행자간의 통신은 CMIP 을 이용한다. GDMO 와 ASN.1 내용이 바뀌게 되면

리파지토리에 그 내용이 저장되어지며 게이트웨이 블록에서는 리파지토리 내용을 통하여 구현 객체 내용이 바뀌어지게 된다. 관리자는 GDMO/ASN.1 내용 변화에 따르는 구현 객체의 변화를 모르더라도 인터페이스 정보를 이용하여 수행하고자 하는 요구를 관리 대행자에게 보낼 수 있다. 이렇게 함으로써 관리자와 관리 대행자 개발에 걸리는 시간을 단축시키고 분리하여 개발하는데 용이함을 제공한다. 리파지토리를 구성하기 위해서는 구조적이고 체계화된 데이터가 필요하게 된다. 이러한 데이터를 메타데이터라고 하는데 검색 시에 색인과 같은 역할을 수행하게 된다. 메타데이타는 일반적으로 데이터의 데이터라고 일컬어지며, 인스탄스의 구조를 정의하고 클래스간의 관계를 표현한다. 메타데이타를 사용하게 되면 정보 시스템에 저장되어지는 데이터에 대한 사용을 용이하게 하는 장점을 얻을 수 있다. 먼저 메타데이터를 추출하기 위한 모델링 작업이 필요하게 되는데 데이터 모델링이란 기업의 정보구조를 실체와 관계를 중심으로 명확하게 체계적으로 표현하고 문서화하는 기법을 말한다. 모델링 작업에 의해 얻어지는 잇점은 첫째, 연관되어져 있는 정보에 대한 요구 시에 정확하게 이해할 수 있도록 하고 둘째, 분석자, 개발자, 사용자간의 의사 소통 수단을 제공한다. 또한 신규 시스템 개발 시 기초 자료를 제공해 준다.

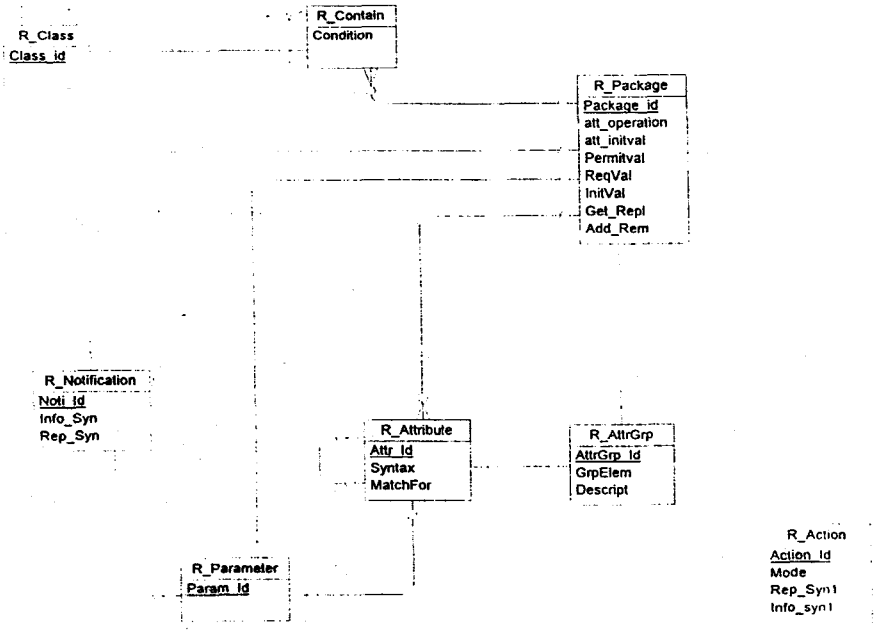
현재 모델링 방법에는 ER(Entity-Relationship)모델, 객체 지향 모델 방법(Object Oriented Modeling)이 있다. 이 논문에서는 구현의 편의를 위해서 E-R 모델을 사용하여 메타데이터를 모델링한다. 본 논문에서의 TMN 과 CORBA 를 통합한 시스템 개발에 따르는 리파지토리 개발에서의 절차는 다음과 같다.

첫째, GDMO 를 컴파일한다. 이는 GDMO 에서 기술된 각 정보들에 대한 문법(Syntax)을 얻는 방법을 제공한다. 둘째, 생성된 정보에 의한 클래스, 속성(Attribute), 액션(Action), 통보(Notification), 파라메타 등의 정보를 이용하여 제안한 모델을 통해 데이터 정보를 저장하는 과정이 필요하다. GDMO 에 기술된 내용에 의한 모델은 다음과 같은 것들로 구성되어 진다[5].

- 객체(Object) : 리소스 등의 개체를 나타내는 속성 값들의 고유한 집합을 말한다.
- 클래스(Class) : 공통된 특성을 갖는 객체들의 집합을 말한다.
- 패키지(Package) : 관련된 관리 객체 클래스의 특성들의 집합을 논리적으로 하나로 묶어주는 역할을 수행한다.

- 속성(Attribute) : 객체의 특별한 속성을 말한다. 각 속성은 스트링, 정수, 실수 등의 기본 타입 또는 사용자 정의 클래스의 타입을 갖는다.
- 속성 그룹(Attribute Group) : 오퍼레이션들이 그룹으로써 수행되어지도록 속성들을 하나로 묶어주는 역할을 수행한다.
- 파라메타(Parameter) : CMIP 에서의 필드들의 문법(Syntax) 정의를 포함한다.
- 행위(Behavior) : 관리 객체의 행위를 설명하는 역할을 하며 텍스트 형태로써 제공된다.
- 행동(Action) : 클래스내의 객체에 적용되어지는 function 을 말하며 같은 오퍼레이션이 다른 클래스에서 다르게 적용될 수 있다.
- 통보(Notification) : 객체의 상태에 영향을 미칠 수 있는 내부 또는 외부의 사태가 탐지되었을 때 관리 객체로부터 보내지는 사건을 말한다.
- 관계(Name Binding) : 두개 이상의 클래스간의 관련된 사항을 나타낸다.

GDMO 내용을 구성하는 요소들에 대하여 E-R(Entity-Relationship)로 모델링을 하면 <그림 4-3>과 같은 메타스키마를 얻을 수 있다.



<그림 4-3> GDMO 메타스키마

5. 시스템 구현 사례

5.1 사례 및 개발 범위

L정보 통신은 통신 기기를 제조하는 회사로서 크게 국설/사설 교환기, 이동 통신 교환기 및 각종 단말, 기지국 장비, ATM 교환기들을 생산하고 있다. 본 논문에서 설명한 CMTA 모델 관리 시스템을 적용 하고자 하는 곳은 ATM 교환기 망 관리 시스템이다. 우리 나라에서 개발되어지는 ATM 교환기는 국책 사업으로 한국 통신을 중심으로 각 개발 업체들이 공동 개발하고 있는 실정이다. ATM 교환기에 대한 관리를 위해서 현재 TMN을 구현하고 있으며, 가입자 관리를 위하여 ILMI(Interim Local Management Interface)등의 일부 기능에 대해서는 SNMP(Simple Network Management Protocol)를 사용하여 개발중이다. 관리자와 관리 대행자간의 통신은 CMIP이 사용되고 있다. 따라서 CMIP 위주의 서비스 개발 과정중인 시스템에 CORBA를 이용하여 관리가 이루어지도록 하는 방향을 적용하려고 한다. 구현 시스템에서의 요구 조건은 다음과 같다.

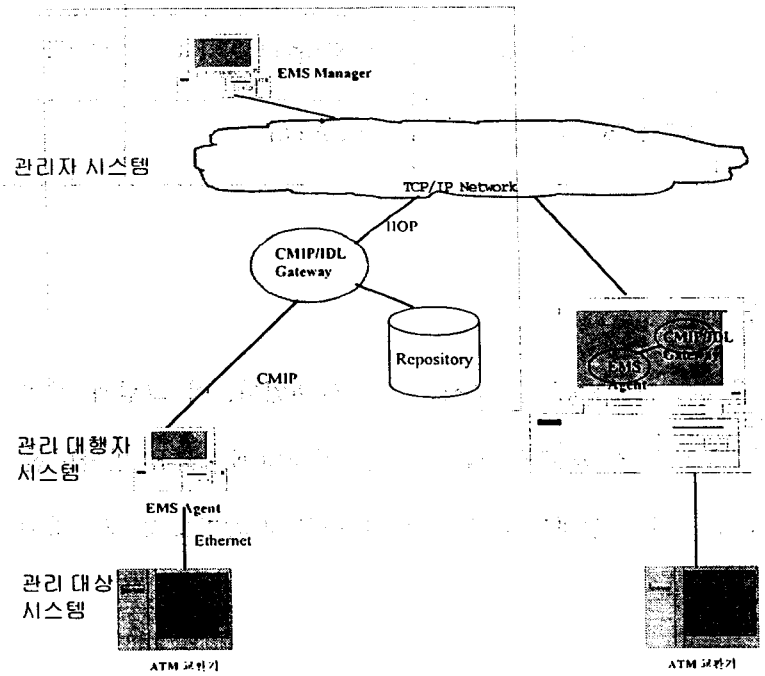
- 사건 보고 기능 : 로그 제어 기능 및 EFD(Event Forward Discriminator)기능의 지원
- 구성 관리 기능 : 통신망 자원에 대한 데이터를 제공
- 시스템의 통합 관리 기능 : 다른 정보 시스템과의 연동 기능의 제공
- 기타 기능 : 스코핑 및 필터링 기능의 제공

개발 범위는 다음과 같다. 먼저 Network Element에 해당되는 것은 ATM 교환기이며 이 교환기를 관리하기 위한 관리 대행자가 워크스테이션에서 구현되어 진다. 관리자는 같은 워크스테이션에 구현되어질 수도 있고 또는 분산 환경하에서 다른 워크스테이션에서 구현되어 질 수 있다. 관리자는 CORBA를 이용하여 구현되어지며, 관리 대행자와의 통신을 위한 게이트웨이가 지원되어 진다. 게이트웨이는 두 가지 블럭으로 나뉘어진다. 첫째로, GDMO/ASN.1 정보를 각 릴레이션 정보에 맞게 추출할 수 있도록 번역 작업을 하는 모듈과 둘째로, 이 정보에 의해 CORBA 서비스와 TMN 서비스 간의 정보 변환을 효과적으로 처리하기 위한 리파지토리가 구현되어 진다.

<그림 5-1>은 구현하는 시스템에 대한 전반적인 구성도이다. 그림에서 구현하고자 하는

부분은 EMS Agent 와 게이트웨이, EMS Manager 에 해당된다. EMS Agent 는 TMN 시스템으로 구현하고 있다. NE(Network Element)에서의 이벤트 정보가 전달될 경우 해당 처리를 관리자에게 통보하게 되고, 또한 관리자의 요구가 NE 로 전달되어 지도록 하는 역할을 수행한다. 게이트웨이는 CORBA 관리자와 TMN 관리 대행자간의 정보의 변환을 담당하게 되며 편의를 위해 리파지토리 시스템을 이용하여 구현되어진다. EMS Manager 는 CORBA 를 이용하여 구현하며 Agnet 시스템을 관리하게 된다.

TMN 서비스를 제공하는 망 관리 시스템을 개발하기 위한 시스템 구성과 사양은 <표 5-1>과 같다. EMS Agent 는 TMN 툴킷으로써 DSET 사의 TMN toolkit 을 사용하여 현재 구현되어진 것을 이용하여 수정하였다. EMS Manager 는 IONA 사의 Orbix 3.0 Web 을 이용하여 CORBA 와 자바를 통해 구현하였고, 게이트웨이는 모델링 툴로써 PowerSoft 사의 SDesigner 를 사용하였다. DBMS 로는 마이크로 소프트의 SQL Server 를 이용 구현하였다. 객체들간의 통신 및 필요한 객체 서비스를 제공하기 위하여 IONA 사의 Orbix 3.0 을 사용하였으며, 이벤트 처리를 하기 위하여 이벤트 서비스를 사용한다.



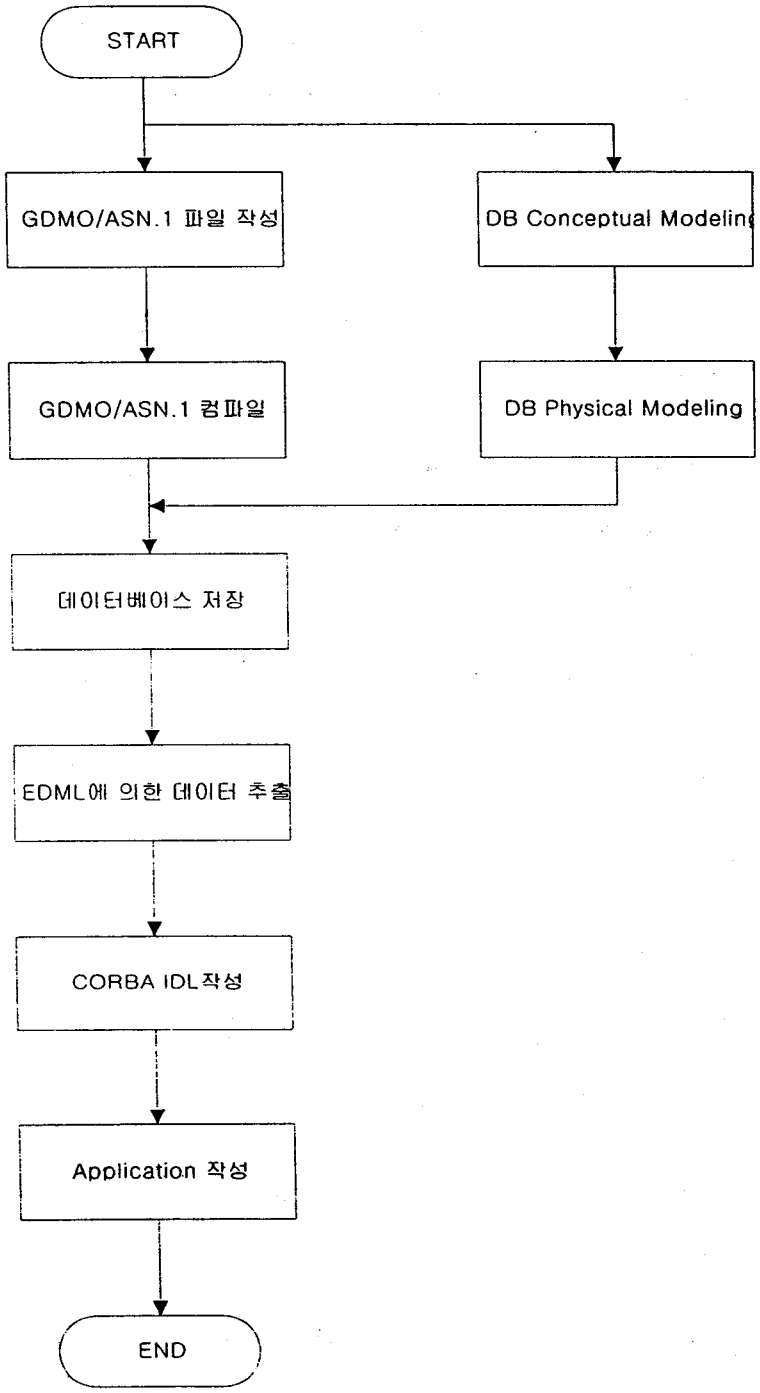
<그림 5-1> 시스템 구성도

<표 5-1> CMTA 모델 관리 시스템 개발 도구 목록

구분		구성 요소	사용 제품	비고
관리자 시스템	관리자 콘솔	GUI	Java AWT	
		클라이언트 객체	JDK 1.1	이동성 코드
		하드웨어 플랫폼	SUN SPARC10	SUN 사 제품
	관리 시스템	CORBA 관리자	Orbix ORB	IONA 사 제품
	리파지토리	DBMS	MS SQL Server	MS 사 제품
		MO 분석 도구	OSIMIS GDMO/ASN.1 컴파일러	UCL(University College London)제품
		DB 모델링 도구	SDesigner	PowerSoft 사 제품
관리 대행자 시스템		TMN 관리 대행자	TMN Agent Tolkit	DSET 사 제품
관리 대상 시스템		ATM Switch	TDX-ATM	

5.2 프로토타입 구축

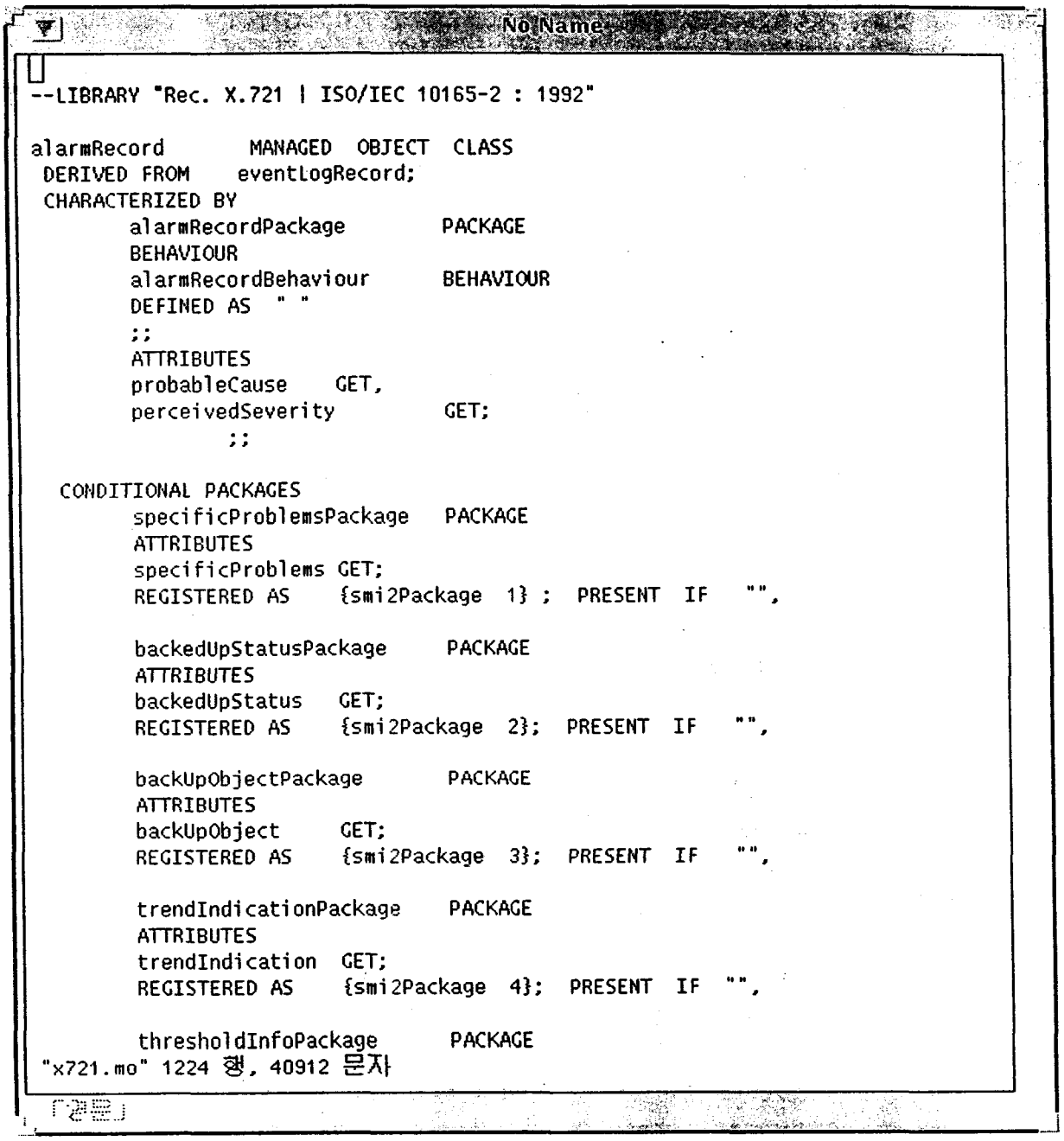
<그림 5-2>는 개발에 따르는 과정들을 순서도로 나타내었다. 이 그림과 같이 먼저 통신망 설계에 따르는 모델링을 통하여 GDMO/ASN.1 파일을 작성하게 된다. 만들어진 파일에 따라 GDMO/ASN.1 파서(Parser)를 이용 컴파일을 하게 되고 이를 리파지토리 모델링에 따라서 데이터베이스에 입력, 저장하게 된다.



<그림 5-2> Network Management System 개발 흐름

다음에 만들어진 데이터베이스내용을 가지고 IDL을 작성하고 이에 따르는 응용 프로그램을 작성한다. <그림 5-3>은 GDMO 파일의 예를 보여 주고 있다. 이 그림에 나타나 있는 것과 같

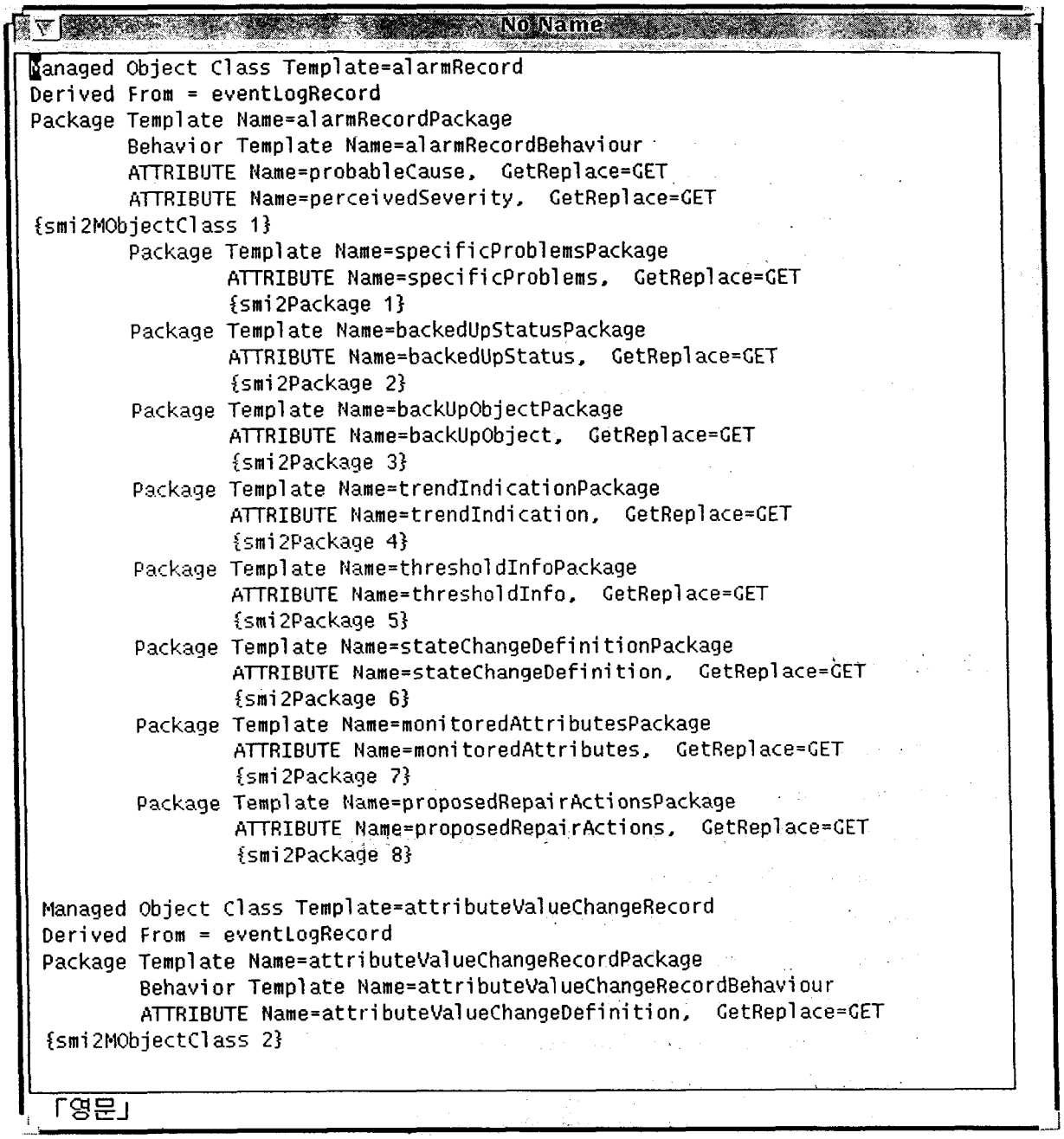
이 듯이 X.721 에서는 시스템의 로그를 저장하기 위한 logRecord 에 대한 관리 객체 모델뿐만 아니라 이벤트 서비스 처리를 위한 EFD(Event Forward Discriminator) 관리 객체 모델을 정의하고 있다.



<그림 5-3> GDMO 파일 예

<그림 5-3>을 입력으로 하여 GDMO 파서(Parser)를 이용, 컴파일을 하게 되면 <그림 5-

4>와 같은 결과를 얻을 수 있다. 이 그림에서는 텍스트 파일로 정의된 GDMO 파일을 데이터베이스에 저장할 수 있는 형식으로 바꾸어주는 역할을 수행하게 된다. <그림 5-4>의 컴파일 결과를 이용하여 리파지토리를 구성한 예는 <그림 5-5>와 같다.



<그림 5-4> 컴파일 결과 예

다음은 CORBA IDL의 내용을 보여준다. <그림 5-5>에서 구성되어진 리파지토리를 이용

하여 관리자 애플리케이션 개발자가 데이터를 추출하여 이에 따르는 IDL을 작성할 수 있도록 한다.

ASN1.idl

```
module ASN1 {
        typedef any ANY;
        typedef any DEFINED_BY;
        typedef boolean BOOLEAN;
        typedef long INTEGER;
        typedef dobule REAL;
        typedef long ENUMERATED;
        typedef sequence<octet> OCTET_STRING;
        typedef sequence<octet> IA5String;
        typedef string OBJECT_IDENTIFIER;
        typedef string ObjectDescriptor;
        ...
};
```

X721_ASN.idl

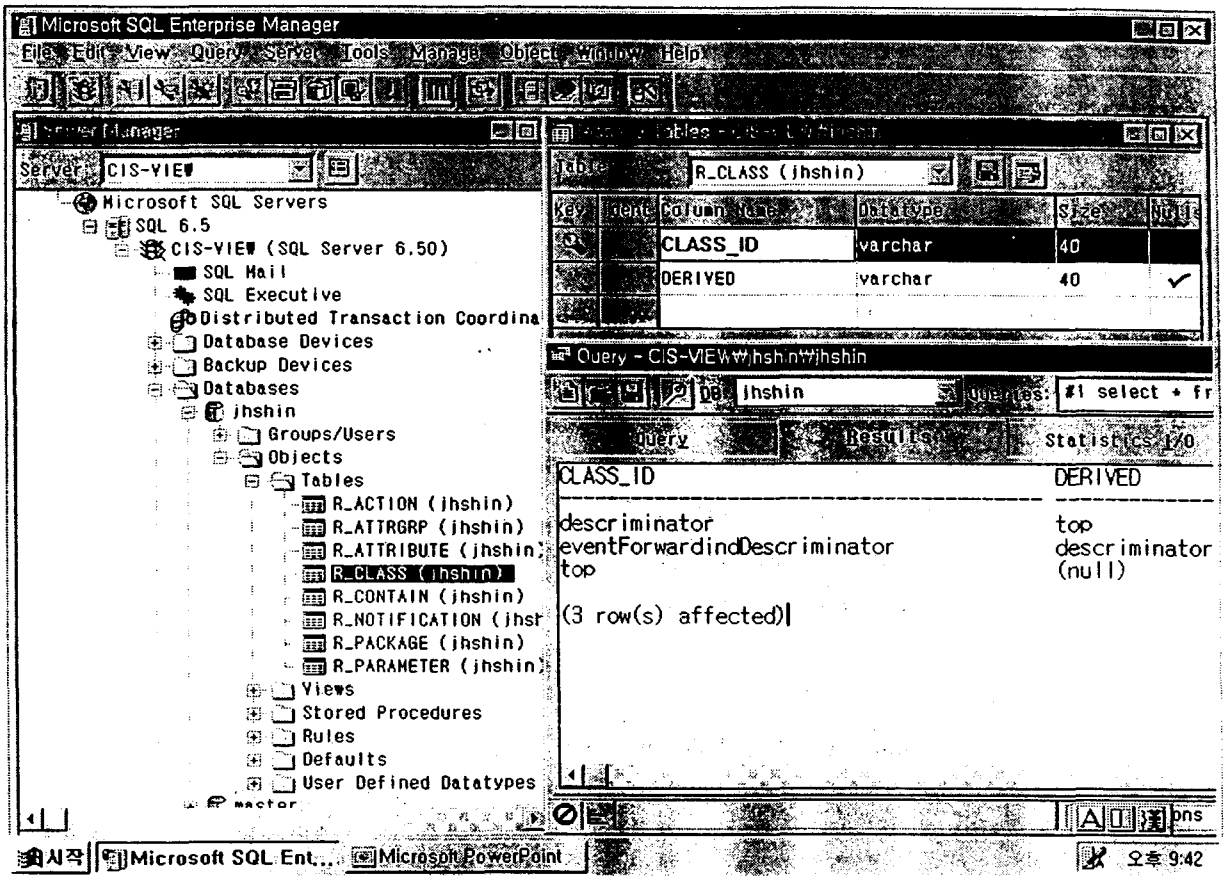
```
#include "ASN1.idl"
module Attribute_ASN1Module {
        typedef ASN1::OBJECT_IDENTIFIER ObjectClassType;
        typedef string NameBindingType;
        typedef string AdministrativeStateType;
        typedef string ArarmStateType;
        typedef string DiscriminatorIdType;
        typedef string DiscriminatorConstructType;
         interface AgentChannelCB {
                string AgentCreate (in string AgentName);
         };
};
```

EventService.idl

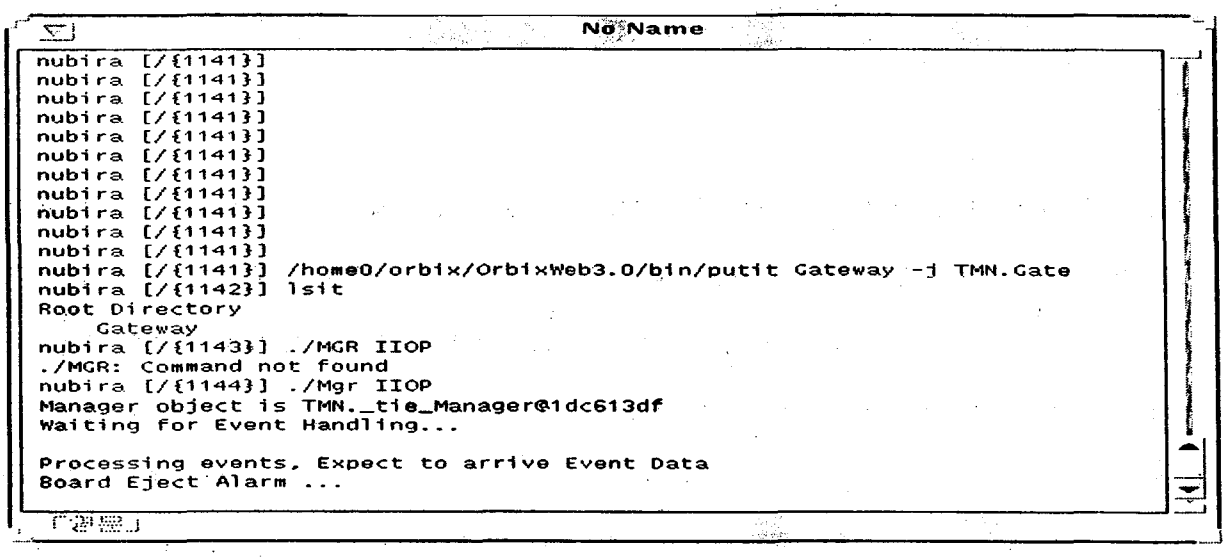
```
/**
 * This program is manager program for event handling
**/
#include "X721_ASN.idl"

module TMN {
        interface Manager {
                void EventService (in string Event);
        };
        interface Gateway {
                oneway void Eventfunc (in Manager cb);
                Attribute_ASN1Module::AdministrativeStateType
                    administrativeStateGet();
        };
        interface top {
                Attribute_ASN1Module::ObjectClassType objectClassGet();
                Attribute_ASN1Module::NameBindingType nameBindingGet();
        };
        interface discriminator : top{
                Attribute_ASN1Module::DiscriminatorIdType discriminatorIdGet();
                Attribute_ASN1Module::DiscriminatorConstructType
                        discriminatorConstructRepl();
        };
};
```

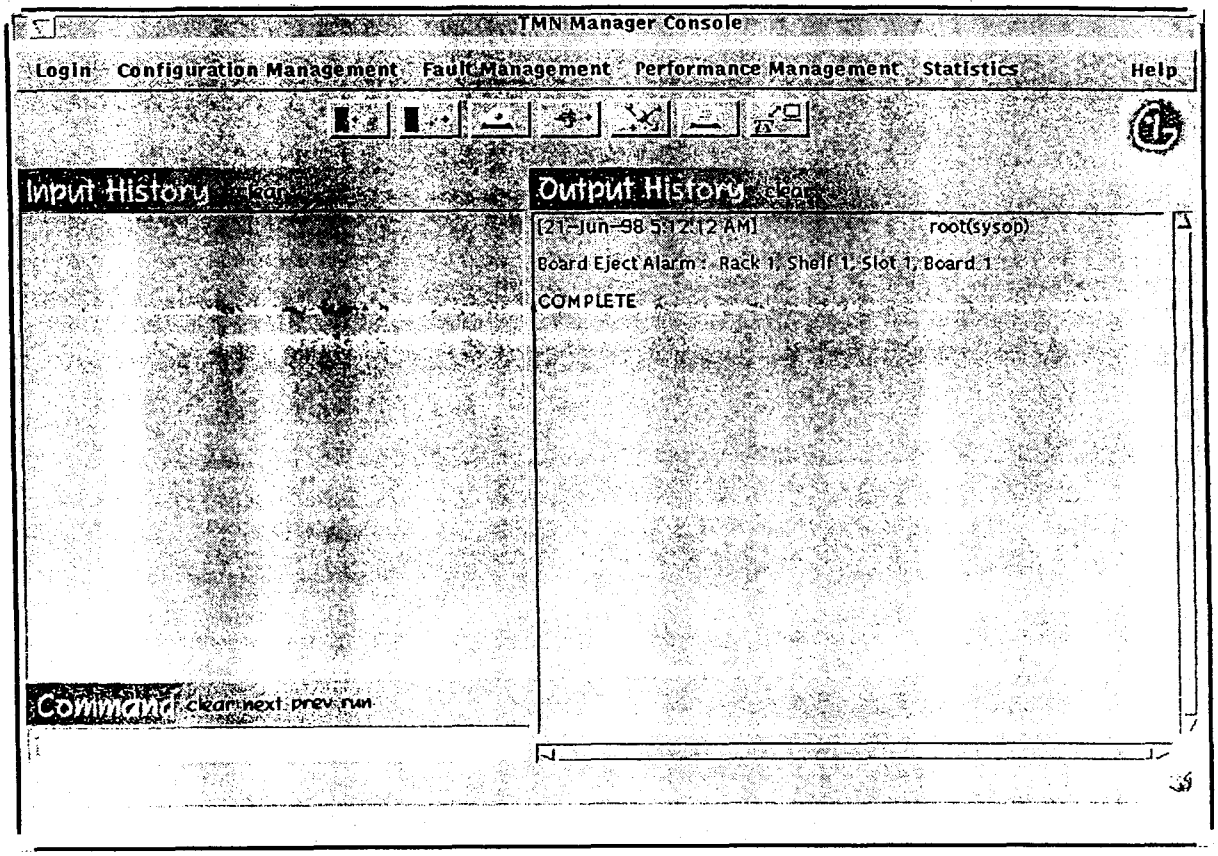
IDL에 의해 인터페이스를 정의한 후 ORB 서비스를 구현하게 된다. 구현된 프로그램의 실행을 위해서는 Orbix 데몬을 실행해야 한다. 데몬은 네트워크에서의 객체의 위치를 알려주는 네이밍 서비스를 제공하게 되어 객체간의 통신을 가능하게 한다. Orbix 실행 데몬을 수행시킨 후에는 최종적으로 서버와 클라이언트 프로그램을 수행시켜 원하는 작업을 수행하게 된다 [18][19]. 데몬을 실행한 후에 CORBA 관리시스템 에플리케이션을 수행하게 되는데 이 실행 화면은 <그림 5-6>과 <그림 5-7>과 같다. 즉, <그림 5-6>은 게이트웨이를 등록한 후 실행 화면을 보이고 <그림 5-7>은 개발되어진 관리자 프로그램의 실행 예를 보여주고 있다. 이 화면은 현재 장애가 발생했음을 나타내고 있다.



<그림 5-5> 리파지토리 구성 예



<그림 5-6> 게이트웨이 실행 화면



<그림 5-7> 관리자 프로그램 실행 화

6. 결론

본 논문에서는 첫째로, 네트워크 관리시스템에 적용되는 대표적인 기술인 TMN에 관한 고찰을 통하여 이질적인 관리 시스템간의 통합의 필요성에 대해 알아 보았다. 또한 통합을 위한 여러 아키텍처를 비교하였다. 이에 따라 개발 시스템으로써 CORBA Manager-TMN Agent 구조의 시스템을 선택하였고, CORBA 를 이용하여 TMN 환경에서의 관리 프레임워크에 기반한 네트워크 관리 프로토타입 시스템을 구현하였다. 프로토타입은 구현의 용이성과 범용성을 위해 IONA 사의 Web based CORBA-Java 개발툴을 이용하였다. 둘째로, 선택된 아키텍처를 근간으로 시스템 구현 과정에서의 상이한 두 모델간의 변환이 쉽도록 게이트웨이 구조를 적용했으며 개발의 용이성을 위하여 E-R 모델링에 의한 리파지토리를 제안하였다.

본 논문에서의 기존 연구와 가장 큰 차이점은 두 가지 객체 모델 즉 TMN 정보 모델과

CORBA 인터페이스 모델간의 변환을 위하여 중간 형태의 새로운 모델을 E-R 모델에 의해서 제시했다는 것이다. 현재 표준으로 상정 중인 안에서는 정적인 형태의 정보를 제공하여 유연한 변환이 어려운 반면, 본 논문에서 제한한 리파지토리 방식은 모델에 의해 만들어지는 데이터베이스를 이용하여 유연성을 제공하며 보다 많은 정보를 시스템 개발자와 사용자에게 제시할 수 있다. 본 논문에서 향후 보완해야 할 점을 기술하면 다음과 같다. 첫째, 변환하고자 하는 두 가지 모델이 모두 객체 모델인 반면에 변환된 모델은 개체-관계 모델로써 나타내고 있다. 그로 인하여 나타날 수 있는 모든 정보를 포함하는데 한계가 있다. 따라서 객체 모델로의 전환이 필요하다. 둘째로, 이 시스템에서 제안한 방법론과 기존의 연구들에서의 방법론을 평가요소별로 비교 분석하는 과정이 필요할 것이다. 셋째, 현재 진행중인 웹 위주의 관리 아키텍처(Web-Based Management Architecture)와의 연동 방안에 대한 연구가 필요하다. 리파지토리를 이용하여 좀더 다양한 정보를 관리자에 제공하는 방안이 모색된다면 웹 위주의 관리 아키텍처를 구축 시 좀더 가벼운 클라이언트(Thin Client) 구조를 제공할 수 있을 것이다. 마지막으로 TMN 정보 모델을 바탕으로 파싱(Parsing)에 의해 중간 데이터 형태를 얻게 되는데, 이렇게 중간 데이터를 만드는 응용프로그램을 이용하여 내장된 SQL(Embeded SQL)형태로 데이터의 추가, 삭제를 하게 되면 리파지토리의 자동 생성이 가능하다. 또한 작성되어진 데이터를 검색하는 응용프로그램에서 검색된 데이터를 이용하여 IDL 로 자동적으로 만들 수 있는 모듈을 추가하는 작업이 주요 향후 연구과제의 하나이다.

참 고 문 헌

[1] [M.3010] ITU-T Recommendation M.3010 : Principles for a Telecommunications Management Network, October 1992.

[2] [M.3100] ITU-T Rec. M.3100 : Generic Network Information Model.

[3] [M.3400] ITU-T Rec. M.3400 : TMN Management Functions.

[4] [X.711] ITU-T Rec. X.7112 : Common Management Information Protocol Specification.

[5] [X.722] ITU-T Rec. X.722 : Guideline for the Definitions of Managed Object for ITU-T

Applications.

[6] [John Westgate 92] Technical Guide for OSI Management, NCC Blackwell.

[7] [Hong 96] J. W. Hong., "Supporting Distributed Management in a Telecommunications Management Network(TMN) Environment" Technical Report, CS-DPE-96003, POSTECH, January 1996.

[8] [OMG Architecture 95] OMG, CORBA : Architecture and Specification, OMG Publications, Aug. 1995.

[9] [OMG CORBA Services 95] OMG, CORBA services, OMG Publications, April. 1995.

[10][OMG CORBA Facilities 95] OMG, CORBA facilities, OMG Publications, Aug. 1995.

[11][Alan 91] Alan Beale, Telecommunication Journal of Australia Vol. 41 No. 2 1991.

[12][Kong and Chen 96] Kong, Q. And Chen G., Integration of CORBA-based Management with OpenView DM, OpenView Forum International Members' Conference, St. Louis, USA, June 1996.

[13][OMG Interworking 97] OMG, Interworking Between CORBA and TMN Systems Revision 1.0, OMG Documentation: telecom/97-09-04.

[14][A. Tannenbaum 94] Adrienne Tannenbaum ., Implementing a Corporate Repository.

[15][Ashrafi et al. 95] Noushin Ashrafi, and Jean-Pierre Kuilboer., The Information Repository : A Tool for Metadata Management., 1995.

[16][OMG White Paper 96] CORBA-based Telecommunication Network Management System., 1996.

[17][Stallings 93] William Stallings., SNMP, SNMPv2 and CMIP : The Practical Guide to Network Management Standards., 1993.

[18][IONA 97] IONA Technologies PLC November 1997 : OrbixWeb Programmer's Guide., 1997.

[19][Orfali et al. 97] Robert Orfali, and Dan Harkey., Client/Server Programming with Java and CORBA, John Wiley & Suns,Inc., 1997.

[20][Wang et al. 97] Zhenxin Wang, Keith Allen, Zhenjun Zhu., A White Paper on The Design of a CORBA Notification Service for OSI Management, Southwestern Bell Technology Research, Inc., 1997.

[21][조영현외 94] 조영현, 김영명, 석승학., TMN을 향한 첫걸음, 1994.