# 이기종 분산환경에서 PDM 통합환경 구현에 관한 연구
# - A Study on the Implementation of PDM Integration Environment in Heterogeneous Distributed Environment -

김 형 선*
Kim, Hyoung Seon

## Abstract

The typical characteristic of PDM(Product Data Management) System seperates the databases to store the meta data and applications. Therefore, meta data contains the information for the location of file, user profiles, relationships between the files, and proecess. PDM utilizes these information efficiently and does file management, configuration management, and process management. In this view, the integration strategy of PDM is to merge data and process. In the view of architecture, the interface between data and application and the actions of each application execute seamlessly.

This architecture is viewed as integrated data and process among enterprises and implemented with client/server technology in distributed process environment that interfaced with open object-oriented technology which is developed with business object in the object-oriented infrastructure.

In this paper, we studied the definition, function, and scope of PDM and researched the core technologies to implement the PDM integration environment. We also researched the PDM utilization in distributed enterprise environment and implementation of PDM integration environment with this technical background.

## 1. 서론

PDM의 개념 자체는 80년대 후반의 EDB(Engineering Database)로부터 근간을 이루는 개념으로 전개 되었다. 그러나 이제는 새로운 IT(Information Technology)의 진보에 의해 이를 실제 구현하는 기술을 갖추게 되었는데, 이러한 기술이 바로 객체기술(Object Technology)이다. 인간의 사고유형과 동일하게 기업의 업무활동 그 자체를 객체화 하여 즉, 개발 생산 판매하고자 하는 제품(Product)과 이를 실현하는 기업의 프로세스 그리고 집행할 사람들을 바로 객체화 하여 이를 객체기술로 그 객체들의 관계성(Relation)을 정의하고, 그 규칙(Rule)을 지원하여 비즈니스 환경 변화에 능동적으로 시스템 지원 및 변경을 함으로써 쉽게 하도록 한 것이다. 또한 현존하는 여러가지 시스템이나 어플리케이션에 대한 지원과 개방된 아키텍처와 분산된 자원 등의 통합을 용이하게 전개할 수 있어야 하는 확장성이 있어야 PDM시스템 구현 후에 소기의 목적을 원할히 수행할 수 있다. 즉, PDM시스템은 객체지향 기술을 이용한 개발업무 및 커스터마이징된 적용업무를 위한 소프트웨어의 재활용이 용이 하여야 한다.

* 시스템공학연구소 시스템통합연구부

PDM시스템은 이기종 분산환경하에서 클라이언트/서버 아키텍쳐를 구축할 수 있어야 하며, 각부분의 실무자들은 각자의 업무에 따라 사용하는 시스템의 환경이 다를수 있다. 예를 들면 설계자들은 CAE, CAD 작업을 위해 엔지니어링 워크스테이션 또는 PC 클라이언트 환경하에서 업무를 수행하고 있으며, 엔지니어링 워크스테이션도 업무지원 어플리케이션 특성상 서로 다를수 있다. 따라서 하나 이상의 서버에 다양한 하드웨어 및 오퍼레이팅 시스템상의 클라이언트에서 구축하여 PDM 어플리케이션을 동일한 사용자 인터페이스로 이용할 수 있어야 한다.

본 본문에서는 이러한 분산된 이기종 환경하에서 PDM 통합환경을 구현하기 위한 방안으로 먼저 PDM의 정의와 기능 및 범위에 대하여 간략하게 살펴보고, 업무용 응용프로그램의 기술적인 진화 과정과 지역적으로 분산된 환경에서의 PDM 활용 및 CORBA를 이용한 PDM 통합환경 구현에 대하여 기술하고자 한다.

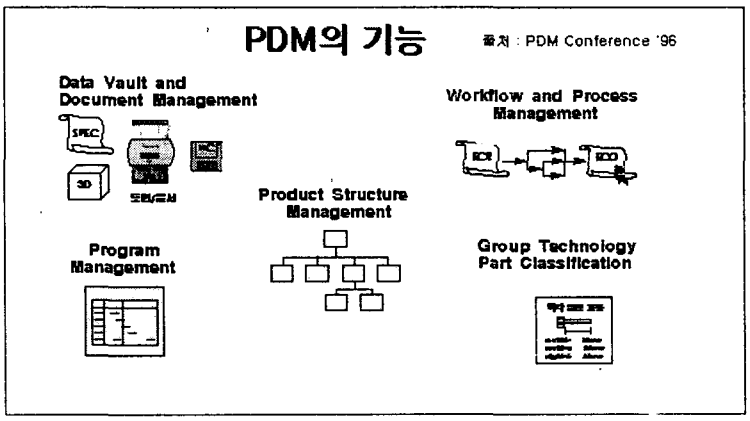
## 2. PDM(Product Data Management)

### 2.1 PDM의 정의

PDM은 개념 설계에서부터 개발, 제조, 서비스에 이르는 제품의 개발 프로세스에 걸쳐 발생하는 복잡하고도 다양한 정보와 업무 프로세스를 시스템화 하여 각종 정보들의 원할한 공유와 자료의 재창출이 가능한 시스템을 말한다. 또 한편으로는 모든제품과 관련된 정보 즉 부품정보, 제품의 구성, 문서, CAD파일, 결재정보 등을 포함한 제품을 기술하는 정보이며, 제품과 관련된 노든 공정을 의미한다.

### 2.2 PDM의 기능

PDM기능은 <그림 1>과 같이 개발을 지원하는 PDM의 기능적인 측면에서 볼때 전자금고 및 문서관리, 분류 및 코딩시스템, 제품구성관리(BOM), 작업흐름과 프로세스 관리(Workflow), 제품개발 일정관리 등 5가지 주요기능으로 나눌수 있다. 이러한 기능들은 대량의 정보를 조회 검색하는 기능에 따른 정적(Static)인 정보관리와 정보가 조직내에서 유통되고 변화하는 과정에 대한 추적하는 기능에 따른 동적(Dynamic)인 정보관리로 구분할 수 있다. 제품구성 관리기능, 저장소 및 문서관리기능, 분류 및 코딩 시스템은 정적인 정보 관리 측면의 기능이며, 작업

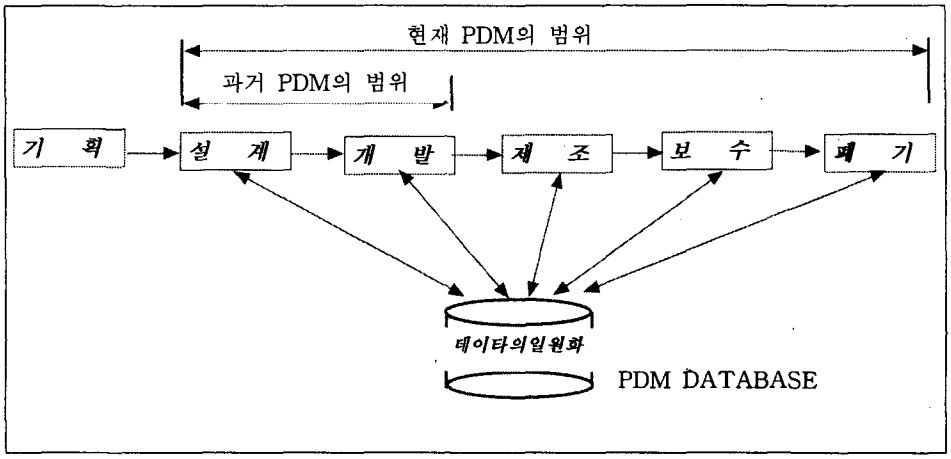



<그림 1> PDM의 기능

흐름과 프로세스 관리기능 및 제품 개발 일정 관리는 동적인 정보관리 측면의 기능이라고 볼 수 있다

### 2.3 PDM의 범위

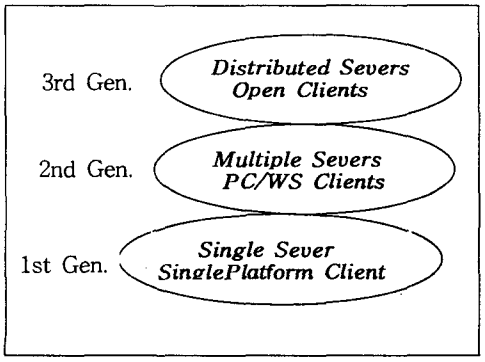
PDM의 범위는 <그림 2>에서 보는바와 같이 제품에 관련된 정보를 개발 단계에서부터 제품의 Life Cycle 전반에 걸친 범위까지 일원적 관리를 위한 것이며, 개발에서 폐기까지의 Data 중심의 Management 체계 확립을 위한 System적으로 접근하는 것을 의미한다.



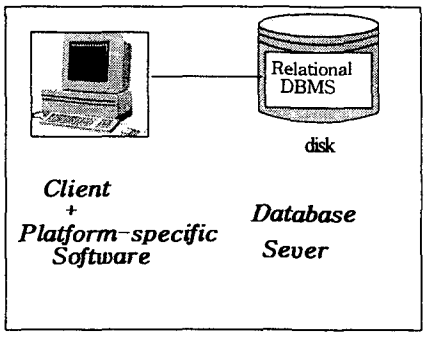

<그림 2> PDM의 범위

## 3. PDM 통합환경을 위한 핵심기술의 진화과정

호스트 중심으로 출발한 소프트웨어 어플리케이션은 클라이언트/서버 환경으로 오면서 <그림 3>에서 보는 바와 같이 제1세대는 단일서버, 제2세대는 복수서버, 제3세대는 분산서버 환경으로 발전해 왔다.




<그림 3> 클라이언트/서버
애플리케이션의 진화과정




<그림 4> 제1세대 클라이언트/서버
애플리케이션 구조

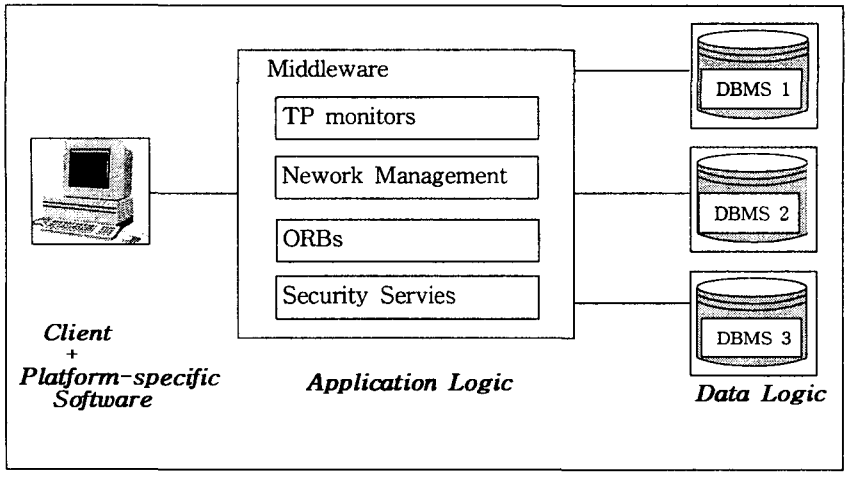
### 3.1 단일서버, 단일 플랫폼 클라이언트

<그림 4>의 제1세대 클라이언트/서버 아키텍처는 워크그룹이나 부서단위 업무처리를 위한 단일 플랫폼 클라이언트와 단일 서버의 2-Tier 구조로 이루어진 것이었다. 이러한 2-Tier 시스템은 사용자의 증가와 서버의 추가 및 새로운 클라이언트 플랫폼의 증가가 어려웠으며, 실행속도의 저하, 높은 유지보수 비용, 기업전반에 걸친 어플리케이션과 업무로직 공유의 어려움이 있었다. 이에 대한 대안으로 데이터베이스 엔진의 기능을 이용하는 방법으로, 업무용 로직을 저장프로시져(Stored Procedure)라는 형태로 데이터베이스 서버에 구현하는 기술이었는데, 단일업체 제품을 사용하는 환경에서는 어플리케이션간의 재사용을 보장해 주었지만 다른 어플리케이션과의 업무 공유는 어려웠다.

대부분의 기업환경은 여러가지 상이한 데이터베이스에 분산되어 있는 정보를 처리해야 함에도 불구하고 상이한 데이터베이스 환경하에서는 어플리케이션간의 상호운용성(Interoperability)이 보장되지 않아 심각한 제약 사항이었다.

### 3.2 복수서버 및 복수 클라이언트

<그림 5>의 제2세대 클라이언트/서버는 대부분 클라이언트 소프트웨어, 어플리케이션 코드 및 데이터로 대별되는 3-Tier 아키텍처를 채용하였다. PC환경에서는 GUI를 운영 하였으며, 업무로직은 UNIX나 NT를 운용하는 Mid-Tier 서버에 의해 처리 되었고, 데이터는 UNIX 서버 또는 메임프레임이나 미니컴퓨터에 저장 되는 형태가 일반적인 제2세대 클라이언트/서버 모델이었다.




<그림 5> 제2세대 클라이언트/서버 애플리케이션 구조

3-Tier 아키텍처는 업무로직을 데이터베이스와 프리젠테이션 로직으로부터 분리하여 데이터베이스간의 전환을 용이하게 하였으며, 다중 데이타베이스 환경에서 어플리케이션을 통합하게 해주었으며, 객체지향기술(Object Technology)을 도입하는데 밑거름이 되었다.

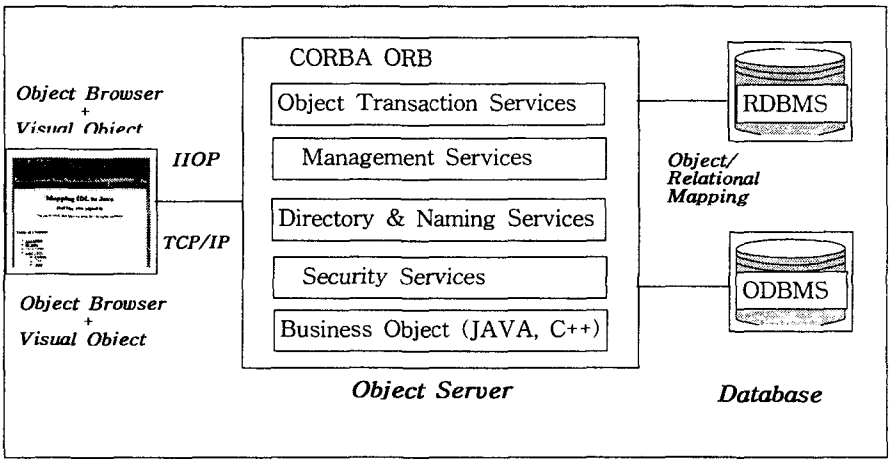
### 3.3 차세대 클라이언트/서버 아키텍처

제3세대 클라이언트/서버 어플리케이션을 위해서는 먼저 전사적으로 인터넷이라는 인프라

스트럭처를 가지고 있어야 하며, 위에서 언급한 인프라스트럭처는 사내에서 서버와 클라이언트를 연결 하는데 TCP/IP 네트웍이 사용된다는 것을 의미하며, 사내의 네트웍 자체가 '네트워크 방화벽'을 경유하여 전체 인터넷에 연결될 수도 있다는 뜻이다. 제3세대 애플리케이션 도래와 더불어, 오브젝트 브라우저(Object Brower), 크로스플랫폼(Cross-Platform), 아키텍처 중립 컴포넌트(Architecture Neutral Components)들이 기존의 클라이언트 측에 존재하는 플랫폼 의존형의 소프트웨어를 대체하게 된다. 또한 CORBA(Common Object Request Broker Architecture)표준을 따르는 JAVA 오브젝트들이 범세계적 규모의 인터넷 서비스를 가능하게 만들었다. JAVA를 이용하면 플랫폼에 상관없이 이식성이 뛰어나고 인테넷상에 분산시킬 수 있는 클라이언트 어플리케이션을 개발하는 것이 가능 하였기 때문이다.

OMG(Object Management Group)의 CORBA와 결합된 JAVA는 어플리케이션 컴포넌트들이 인터넷상의 다중공유 Back_End 서비스(Multiple Shared Back_End Service)를 활용할 수 있는 길을 열어 주었다. 그러나 현재의 브라우저는 HTTP 중심의 정적인 하이퍼텍스트 문서를 전송하여 준다는 점에서 우수하다고 볼 수 있지만, 네트웍을 통하여 오브젝트나 어플리케이션을 전송할 수 있는 환경은 제공하지 못했다.

반면에 오브젝트 브라우저 또는 웹브라우저는 IIOP(Internet Inter-ORB Protocol)를 채택함으로써, 기존의 하드웨어나 소프트웨어를 최대한 활용하는 동시에 정보의 검색 또는 접근방법과 분산된 객체지향의 업무용 어플리케이션의 새로운 표준으로 부상하고 있다. 일반적인 JAVA 애플릿이 IIOP 프로토콜을 이용하여 인터넷상의 CORBA 오브젝트가 가지고 있는 메소드를 직접 기동할 수 있게 해준다.




<그림 6> 제3세대 클라이언트/서버 애플리케이션 구조

즉, CGI(Common Gateway Interface)나 HTTP를 완벽하게 우회하게 되어 ORB(Object Request Borker)를 이용한 클라이언트와 서버간의 직접적인 연결이 이루어 진다. 차세대 클라이언트/서버 아키텍처는 HTTP와 오브젝트 브라우저를 동시에 지원할 수 있는 서버가 되며, Middle-Tier 서버상의 CORBA 오브젝트는 어플리케이션 업무로직을 Encapsulate 시킬수 있게 된다. 업무용 오브젝트들은 CORBA의 ORB(Object Request Broker)를 이용하여 오브젝트 브라우저상에서 상호연동 되거나 클라이언트 컴포넌트와 연동될 뿐만 아니라 SQL이나 다른 형태의 미들웨어를 통하여 데이터베이스 Tier에 존재하는 기존의 서버 애플리케이션과도 연결시킬수 있다.

데이터베이스 Tier상에서 CORBA 오브젝트와 통신이 가능한 DBMS, ODBMS, Lotus Notes, TP-Monitor 등을 포함하게 된다. 이러한 업무용 오브젝트는 오브젝트 브라우저의 요청에 대하여 효율적인 처리를 복수의 서버에서 수행되며, ORB는 짧은 시간내 요청에 대한 처리를 마치게 된다.

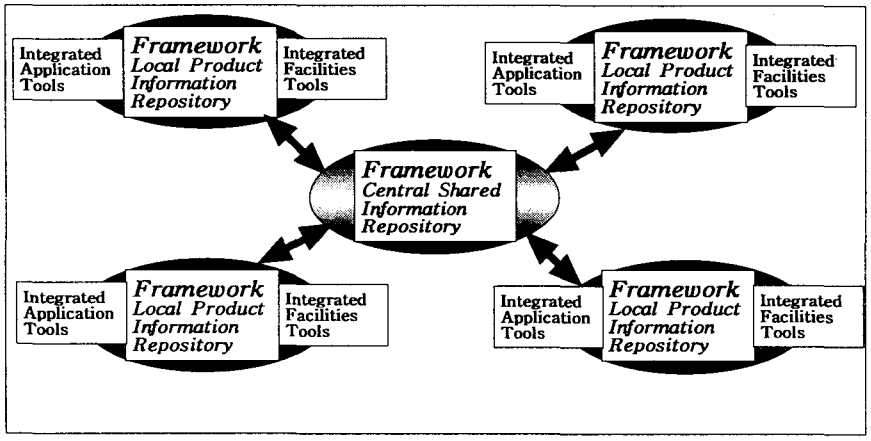
## 4. 분산된 환경에서의 PDM 활용

본 장에서는 지역적으로 분산된 이기종 기업 환경에서의 PDM 활용 방안에 대하여 서술하기로 한다. 엔지니어링 환경하에서 분산된 기업일수록 정보의 효율적 관리 및 활용이 용이하여야 하며, 제품 라이프사이클 변혁에 민감할 수 있게 된다. 이러한 요구들에 의하여 클라이언트/서버 구조하에서 분산형 데이터베이스를 활용함으로서 그 제한성을 극복하는 해결책을 제공한다.

PDM 시스템을 근간으로 한 통합제품 개발 환경은 조직적으로 분리된 환경에서 뿐만 아니라 지역적으로 분산된 환경에서도 효과적으로 전사적 목표를 달성할 수 있도록 지원할 수 있는 환경인 것이다. 분산된 개발환경을 지원하는 PDM 시스템은 다음과 같은 역할을 제공함으로서 지역적인 분산환경의 문제점을 극복할 수 있도록 하고 있다.

(1) 시스템 정보를 저장하기 위한 분산형 정보 저장장소
(2) 운영 방법론을 실현할 수 있는 Core 역할
(3) 정보의 저장장소로서의 접근을 관리하는 구조
(4) 다양한 어플리케이션 소프트웨어를 공통적으로 이용할 수 있는 환경
(5) 분산된 개발환경을 지원하는 목적의 일반적 기능

다음 <그림 7>은 분산환경하에서의 PDM 시스템 주요 요소들을 보여주고 있는데 각 요소들이 가지고 있는 역할에 대하여 알아보면 다음과 같이 정의할 수 있다.




<그림 7> 분산환경하에서의 PDM 통합구조

### 4.1 정보 저장장소(Repository)

분산된 환경하에서의 첫번째 요소는 제품정보의 저장장소이며, 저장장소는 각각의 개발 프로젝트의 시작단계에서 부터 형성되어지는 정보에 대한 개념적이고 물리적인 의미이다. 이 저장장소는 <그림 7>에서 보는바와 같이 논리적으로는 상호 연결된 지역(Local)과 중앙공유

(Central Shard) 제품정보 저장장소로 구분되고 있다. 지역 저장장소는 각각의 분산조직에서 운영되어 지는 것이며, 중앙공유 저장장소는 형식적으로 전체를 관장하는 모습으로 존재한다. 한 담당자가 자신의 정보를 완성하고 지역에서 중앙으로 그 정보를 등록하게 되면 다른 타지역에서는 중앙공유 저장장소로부터 가상적으로 보여지는 정보를 활용하게 된다. 이 정보들은 제품 구조하에서 논리적인 링크관계를 가지고 있기 때문에 지역에서 일어나는 변경관리와 똑같은 방법으로 새로운 버전의 변경 정보를 생성하거나 활용할 수 있다.

### 4.2 Framework

Framework는 제품정보 저장장소에 대한 접근관리까지 포함하는 소프트웨어 도구이다. 이것은 여러가지 타 어플리케이션을 통합하여 이들이 가지고 있는 갖가지 정보관리 및 구성관리 등의 기능을 사용자에게 일관된 환경으로 제공한다.

### 4.3 Tools

이것은 각 지역에서 특성적으로 선택되어 활용할 수 있는 소프트웨어 어프리케이션들로서 이를 통하여 정보가 생성 및 분석되어 지고 저장장소에서 보관되게 된다. 이러한 Tool들이 Framework와의 통합을 통해 전지역으로부터 다중작업이 가능하다.

### 4.4 Facilities

Facilities는 Tool과 마찬가지로 소프트웨어 어플리케이션이지만 전체 환경에서 공통적으로 활용 되어지는 소프트웨어 어플리케이션이다. 이 Facilities에 속하는 소프트웨어 도구들은 전자메일 등을 포함하는 통신도구(Communication Tools)나 분산된 자료관리나 이미지 저장, 관리 등을 포함하는 문서관리도구, 서로 다른 종류의 정보를 구조적으로 연결하는 정보 연계 도구, 그리고 리포트 발행 도구들이 Facilities에 속하는 것들이다. 이러한 분산형 정보의 관리 및 활용 측면이 고려된 PDM 시스템은 지역적, 특성적으로 분리된 환경에서 사용자가 아무런 불편도 느끼지 않고 자신의 업무를 실행할 수 있는 환경을 효과적으로 제공한다.

## 5. CORBA를 이용한 PDM 통합환경 구현방안

### 5.1 PDM 통합환경 구현시의 고려사항

실제로 CORBA를 사용하여 정형화된 개발과정을 찾기에는 상당히 어려웠다. 왜냐하면 학계, 연구계, 산업계에서 CORBA를 사용해 많은 프로토타입을 구축했지만, 개발과정에 대한 자세한 정보는 제공하지 않기 때문이며, 본 논문에서 CORBA를 이용하여 시스템 구축과정에 대해 정리한 사항은 다음과 같다.

* 1단계, 문제정의와 요구분석

가장 먼저 해야할 작업은 구축하고자 하는 도메인의 현상황을 직시하는 것이다. 이를 위해 현재 도메인에서 필요로 하는 요구사항을 분석하고 문제를 정의한다. 이때 일반적인 것으로 요구사항 명세와 사용 시나리오를 정리한다. 그리고 이들 요구사항을 해결하는 시스템을 구축하기 위한 방법을 결정한다. 이를 위해 먼저 CORBA를 적용한다는 생각에서 벗어나 현재의 공통적인 개발환경인 네트웍 클라이언트/서버 환경에서 다른 종류의 하드웨어

와 소프트웨어에 무관한 시스템을 구성하는데 필요한 기본 프레임웍을 선택할 때 필요한 몇가지 기준은 다음과 같다.

첫째, 프레임웍은 사용하기 쉬워야 한다.

- 짧은 기간에 쉽게 습득할 수 있고 간단한 사용법을 갖고 있어야 한다.

둘째, 프레임웍은 유연해야 한다.

- 다양한 표준을 지원하고 서로 다른 응용 프로그램이나 모델에 상관없이 사용할 수 있어야 한다.

셋째, 프레임웍은 확장성이 있어야 한다.

- 적용하려는 분야에 필요한 기능을 구현하여 쉽게 확장할 수 있어야 하며, 새로운 표준으로 쉽게 전이될 수 있어야 한다.

넷째, 프레임웍은 결합력이 있어야 한다.

- 웹이나 OLE, 오픈독(OpenDoc)같이 기존의 새로운 기술이나 향후 출현하는 기술과 쉽게 결합해 사용할 수 있어야 한다.

* 2단계, 분석과 디자인

작성된 요구사항 명세와 사용 시나리오에 기초해 분석과 디자인을 시작하는데, 일반적으로 OMT같은 객체지향 분석과 설계 기법을 사용하고, 해당 시스템에서 필요한 객체를 정의하고, 이들 객체의 속성과 메쏘드를 정의하며 객체 사이의 연관성을 정의한다.

* 3단계, IDL정의

분석과 디자인이 정의되면 디자인 스펙에 따라 CORBA IDL 인터페이스를 작성한다.

* 4단계, IDL을 프로그램 언어로 컴파일

작성된 CORBA IDL 파일을 원하는 형태의 개발언어로 컴파일 할 수 있다.

* 5단계, 원하는 응용프로그램 로직 추가

IDL 컴파일을 통해 생성된 파일을 바탕으로 원하는 객체의 메쏘드와 로직을 구현한다.

* 6단계, 해당 시스템에 개발된 구성요소 배치

개발된 구현객체들을 원하는 시스템 구성방침에 따라 배치한다.

* 7단계, 시스템 구동

구축된 시스템을 가동한다.

위에서 제시한 7단계의 개발과정을 염두에 두고 PDM 시스템 통합환경을 구현하면 안정된 시스템을 개발할 수 있을 것이다.

## 5.2 PDM 통합환경 구현시의 문제점

PDM 통합에 관련된 일반적인 문제점은 QOS(Quality Of Service)의 문제라고 할 수 있고, 일단 통합에 참여하는 개별적인 어플리케이션 시스템들은 독자적(Autonomy)으로 쓰일수 있음은 물론이고, 그들의 데이터 파일과 개별적인 데이터베이스들을 서로 다른 어플리케이션과 심각한 성능의 저하나 기능성의 Trade-Off없이 고유할 수 있어야 하며(Interoperability), 각각의 프로세스에 최적인 하드웨어 및 운영체제, 어플리케이션 버전간의 다양한 조합이 가능하도록 할 수 있어야 할 것이다(Diversity). 또한 기존의 어플리케이션들은 물론이고 향후 확장을 위한 고려가 충분히 이루어져야 하고(Openness), 프로젝트나 부서, 프로세스별로 확장 가능성(Scalability)이 높아야 한다. 통합 시스템의 신뢰성(Reliability)과 유용성(Availability)은 높아야 하며, 기업간 정보 교환의 경우 보안(Security) 역시 중요한 요소가 된다.

위에서 언급한 문제들은 분산환경을 구축 하고자할 때 따르는 일반적인 문제들이고, CORBA를 이용하고자 할 때 특별히 대두될 수 있는 문제는 역시 성능(Performanace)상의 문제이다. 대량의 데이터를 전송하는 것과 같은 성능에 민감한 어플리케이션의 도메인에서는

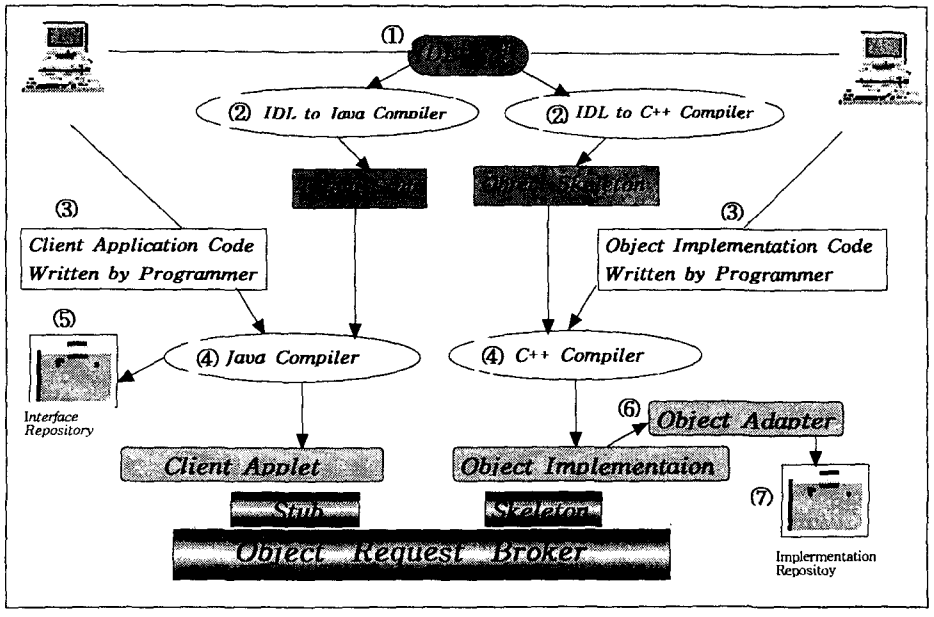
TCP/IP 소켓보다 CORBA를 사용하는 것이 성능면서는 최소 25%에서 75% 정도까지 속도가 떨어져 있는 것으로 알려져 있는데, ATM이나 FDDI와 같은 고속 네트워크상에서는 이러한 차이가 더욱 극명하게 나타날 수 있다.

CORBA를 통한 통합환경의 구축에는 이와 같은 문제점 이외에도 아직까지 많은 문제점이 있는데, 우선 검증되어 재사용되고 있는 추상적인 어프리케이션 프래임웍이 알려져 있지 않으며, 표준 API가 부족하고 분산환경하에서 어플리케이션 개발에 적합한 디버깅 도구도 나와 있지 않은 상태이다. 무엇보다도 가장 큰 문제점은 소프트웨어 인프라 자체가 아직 그리 유연하지도 않고, 신뢰도도 높지 않다는 것이다.

### 5.3 PDM 통합환경 프로토타입 구현예와 개발환경

객체지향 미들웨어인 OMG(Object Management Group)의 CORBA 메카니즘을 이용한 프로그래밍 예를 <그림 8>에서 보여주고, 이 그림은 일반적으로 CORBA 메카니즘을 이용하여 분산 클라이언트/서버 프로그램을 구현하는 과정을 도식화 한것이며, PDM 통합환경 프로토타입 구현도 이 순서를 기본으로 하여 기술하였다.

(1) OMG IDL을 사용하여 인터페이스를 정의한다.
(2) 인터페이스 정의를 IDL 컴파일러로 컴파일 하여 IDL stub와 IDL skeleton을 생성한다. IDL stub 및 IDL skeleton은 OMG IDL로 기술한 인터페이스 목표 프로그래밍 언어(C++, C, SmallTalk, Java)로의 매핑코드이다.
(3) 목표프로그래밍 언어로 인터페이스를구현한다. 즉 서버 프로그램을 작성한다.




<그림 8> CORBA 메카니즘을 이용한 프로그램 구현 예

(4) 객체구현을 인스턴트화(예: C++ 객체생성) 하기 위한 서버 응용프로그램을 작성하고, 그 다음 초기화가 완료되고 서버가 클라이언트의 요청을 받아들일 준비가 되는 시점을 ORB에게 알려준다.
(5) 서버를 구현 저장소(Implementation Repository)에 등록 한다.
(6) 서버에 연결하여 서버의 객체들을 사용할 클라이언트 응용프로그램을 수행한다.

위의 과정에 따라 String을 조작하는 분산 소프트웨어를 개발하는 과정이고, 이 과정에 따라 Visigenic사의 Visibroker 3.0을 사용하여 개발과정을 보여준다.

첫째, IDL syntax을 이용하여 다음과 같은 IDL 파일 작성한다.

```
Struct Approval
{
     long     Id;
     long     Status_id ;
     char     R_level[129]
};
Interface     Seri_Pdm
{
    Approval     Getapproval(in long al)
};
```

둘째, 위의 IDL 파일을 IDLtoC++ 컴파일러로 컴파일하여, 다음과 같은 C++ 코드 Skeleton을 생성한다.

```
class_sk_pdm : public Seri_Pdm
{
    protected:
        _sk_pdm(const char*object_name=(const char *)NULL);
        _sk_pdm(const char*service_name, const CORBA::ReferenceData& data);
        virtul~_sk_pdm(){}
    public:
        static const CORBA::TypeInfo_skel_info;
        virtul Approval GetApproval(CORBA::long al) = 0 ;
        static void_Getapproval(void *obj, CORBA::MarshalStream &strm,
                                CORBA::Principal_ptr principal,
                                const char *oper,
                                void *priv_data);
 };
```

또, IDLtoJAVA 컴파일러로 컴파일하여 JAVA stub을 생성한다.(생성된 코드 내용 생략)

셋째, 위의 C++ Skeleton 코드에 실제 Getapproval의 기능에 해당하는 부분을 C++로 작성한다.

넷째, 생성된 JAVA stub코드를 이용하여 Client를 구현한다.

다섯째, 클래스의 정의를 인터페이스 저장소에 바인딩 한다.

(넷째,다섯째는 Compiler 및 시스템 제공 유틸리티가 자동으로 제공함)

여섯째, 객체구현을 run-time시에 생성시켜 주기 위해 다음과 같은 C++코드를 작성한다.

```
main(int argc, char **argv)
{       CORBA::ORB_var orb=CORBA::ORB_init(argc,argv);
```

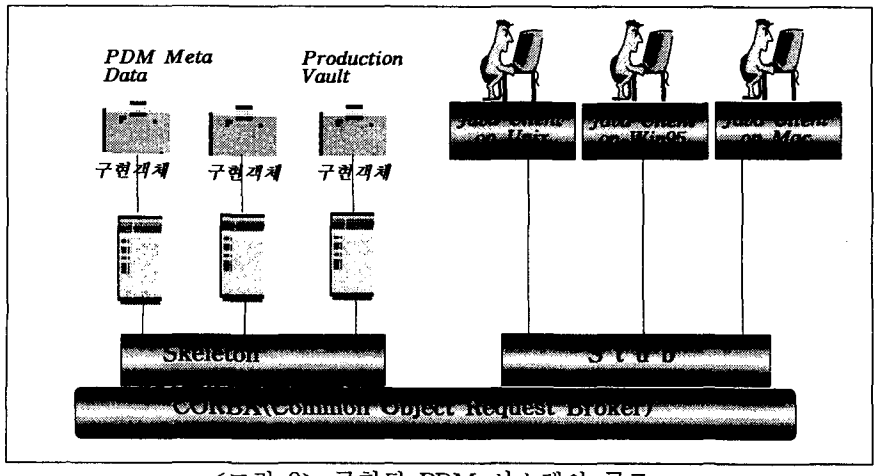
```
CORBA::BOA_var boa=orb->init(argc,argv);
Seri_Pdm PdmObj;
boa->obj_is_reaby(&Seri_Pdm);
boa->impl_is_ready();
```

일곱번째, run-time 객체의 등록이 끝나면 클라이언트 객체가 객체구현으로부터 서비스를 요청할 수 있는 준비가 완료 되었으며, 클라이언트 객체는 다은과 같은 JAVA 코드를 통해 서비스를 요청할 수 있다.

```
CORBA.ORB orb = CORBA.ORB.init(this);
Seri_Pdm pdmobj=Ser_Pdm_var.bind();
Approval    al = new Approval();
for(int j=0; j<NumberOfProduct; j++)
{
        ap = para.Getapplication_context(j);
}
```

위의 과정은 사용자가 이기종 분산 환경하에서 원하는 작업을 쉽게 수행할 수 있도록 CORBA 와 JAVA가 통합된 Java 애플릿을 제공하고, 이 클라이언트를 이용하여 사용자는 인터넷 웹페이지 내에서 원하는 제품의 데이터를 엑세스 하기 위하여 CORBA를 통하여 네트웍상에 존재하는 상응하는 구현객체(Implementation Object)에게 요청하고, 그 구현객체에 의해 처리된 요청결과의 객체를 되돌려 받아 보려는 작업을 수행한다.

이종 분산환경에서 프로그램의 작성은 다른 프로그래밍 언어로 다른 객체들을 구현하는 것이 가능 하며, 본 PDM 통합환경 구현에 있어도 이런 원리를 사용하고 있다. 즉 서버구현에는 C++ 언어를 사용하고, 클라이언트 구현에는 Java언어를 사용한다. 그리고 클라이언트 프로그램 작성은 IDL 컴파일러로 컴파일 하여 목표 프로그래밍 언어(여기서는 Java)로 작성된 IDL stub 코드를 생성하는 것으로 시작 한다. 생성된 IDL stub 코드에 선언되어 있는 함수들을 이용하여 PDM Meta Data 구현 객체(서버)에 연결하여 이 구현객체에 사용할 클라이언트 응용프로그램을 작성한다. 이렇게 CORBA와 Java가 통합된 클라이언트는 사용자가 인터넷상의 웹페이지 내에서 그래픽 사용자 인터페이스(Java Applet)를 통하여 응용 구현객체와 대화식으로 수행할 수 있게 한다. 전형적인 CORBA 클라이언트 프로그램은 객체 서비스 요청을 위하여 원거리 객체(Remote Object)를 찾고, 객체 바인딩(Object Binding)를 통하여 그 객체에 대한 객체 참조(Object Reference)를 획득하며, 그리고 나서 그 객체에 대한 오퍼레이션을 호출한다.




<그림 9> 구현된 PDM 시스템의 구조

<그림 9>에서 보는 바와 같이 이 클라이언트는 JAVA 애플릿으로 구성되며, PDM Meta Data와 Production Vault의 데이터베이스를 CORBA를 통하여 네트웍상의 서버에게 요청하고, 그 서버에 의하여 제공된 데이터를 웹페이지 내에서 디스플레이 한다. 사용자는 디스플레이 된 데이터를 필요에 따라 변경하고 그 변경된 데이터를 서버에게 전달 함으로써 데이터베이스를 갱신할 수 있다. PDM의 Meta Data는 다른 지역 에 있는 서버에 저장 하였으며, PDM 실행 모듈들은 또 다른 지역에 있는 Java 클라이언트에서 실행한다.

시스템의 개발환경은 다음과 같다.

CORBA의 객체구현

| | |
|---|---|
| Language | : C++ |
| OS | : Wnndows NT |
| CORBA IDL Compiler | : Visibroker for C++ (Version 3.0) |
| Database System | : 7.3.2.1.0 |

CORBA의 클라이언트

| | |
|---|---|
| Language | : JAVA |
| OS | : Wnndows 95 |
| CORBA IDL Compiler | : Visibroker for JAVA (Version 3.0) |

OMG의 CORBA를 이용하여 분산환경에서의 PDM 시스템을 구현하면 지역적인 분산환경을 느끼지 못하면서 서로 다른 지역에서 동시에 PDM 자료의 생성 및 변경, 삭제 등의 작업을 할 수 있는 투명성을 제공하며, 제품의 생명 주기 활동에 관여하는 각각의 사람들이 다루게 되는 하드웨어 및 소프트웨어, 플랫폼, 데이타베이스의 종류에 상이함에 관계없이 쉽게 생산, 관리할 수 있는 환경으로 구축할 수 있다.

## 6. 결론

본 논문에서는 이기종 분산환경에서 응용프로그램을 개발하여 사용자들에게 제공하는데 기반이 되는 객체 지향 미들웨어의 표준인 OMG(Object Management Group)의 CORBA를 이용하여 PDM 시스템의 통합환경을 구현하기 위한 방안을 제시하였다.

OMG의 CORBA는 어플리케이션간의 호환성을 증진 시키기 위한 노력에 의해 생겨나게 되었으며, PDM 시스템처럼 서로 다른 어플리케이션간에 통합을 위해 기존의 특수 목적의 API(Application Program Interface)가 해결할 수 없던 부분을 OMG의 CORBA를 활용함으로써 표준화된 오브젝트 전달 기술에 의해 PDM 통합환경이 가능하게 된 것이다.

컴퓨팅 환경이 이기종의 시스템들로 구성된 분산환경으로 발전하고, 통신 서비스 및 통신망 관리와 관련된 소프트웨어가 커지면서 객체지향 개념을 도입한 PDM 시스템 통합환경 구현은 필수적이라 하겠다. 그러나 현재로서는 CORBA의 여러가지 문제점으로 인하여 많은 시간과 비용을 투자하여 어플리케이션간의 통합작용을 수행하고 있는 실정이다. 향후 각 어플리케이션들이 CORBA를 적용하여 개발되면 그 어플리케이션간의 PDM 통합환경 구현작업은 매우 획기적으로 개선될 전망이다.

## 참고문헌


[1] John MacKrell and Patrice Romzick, *PDM Conference '96 Tutorial Proceedings PDM Overview*, CIMdata, 1996

[2] OMG, *"CORBA:Arcchitecture and specification"*, *OMG Publication*, 1995.

[3] Robert Orfali, *The Essential Client/Sever Survival Guide*, 2nd edtion, Wiley, 1997

[4] Robert Orfali, Dan Harkey, *Client/Sever Programming with Java and CORBA*, 2nd edtion Wiley, 1997.

[5] G. Almasi, V juggy Jagannathan, *Integrating the WWW and CORBA-based environment*, West Virginia Unversity, 1996.

[6] Randy Otte, Paul Patrick, Mark Roy, *Understanding CORBA : The Common Object Broker Architecture*, Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1996

[7] Jon Siegel, *CORBA : Fundamentals and Programming*. John Wiley & Sons, 1996

[8] Ashley McClenaghan, *Building a Web from Distributed Objects, ANSA Project*, 1996.

[9] Philippe Merle, *Distributed Object Technology abd the Web, WWW conference*, 1996

[10] jaehyun park, WWW와 분산객체시스템, WWW-KR Workshop, 1996

[11] jaehyun park, 비즈니스 환경에서의 CORBA, KIECO, 1996

[12] 김수동, "CORBA를 이용한 분산 객체컴퓨팅", 정보통신기술 10권 1호, 1996

[13] 심재찬,고병도, "OMG의 분산객체기술 CORBA와 상용화 동향", 전자통신동향분석 제12권 2호. 1997