객체 지향 기술을 이용한 PDM 프레임워크 개발

김 정 아†

요 약

PDM 기술은 제품 기획, 제조, 생산 분야에서의 생산성 향상의 기법으로, 시장이 점점 더 커지면서 여러 회사들에서 서로 다른 이유이긴 하지만 PDM 시스템 개발에 투자하고 있는 중이다. 동기들은 서로 다르지만 보다 좋은 품질의 제품을 넓기에 제공할 수 있는 회사 환경 구축이 공통 목표이다. 지금까지 개발된 PDM 시스템을 분석한 결과 그들의 개발 동기 및 개발 기법에 많은 공통점을 갖고 있지만, 실제 구현 전략상 약간씩의 차이를 보이고 있다. 반복되는 PDM 시스템 개발의 생산성 향상을 위하여 본 논문에서는 프레임워크 기술을 적용하였다. PDM 프레임워크 개발에 있어 PDM 도메인에 공통인 추상화 클래스들과 이들간의 기본 행위를 정의함으로써 프레임워크 자체가 어플리케이션에 대한 실질적인 템플리트가 되도록 하였다. 프레임워크의 개발은 단순한 어플리케이션 개발 보다 훨씬 더 많은 노력을 필요로 한다 본 논문에서는 PDM(Product Data Management) 도메인에 대한 프레임워크 개발 과정에 프레임워크 개발 공정(Process)을 적용하였다 또한 이를 지원하는 제정의(Customization)도구를 함께 제공하여 프레임워크 기반 어플리케이션 개발을 효과적으로 지원하도록 하므로써 프레임워크를 통한 재사용성과 생산성 향상의 효과를 증대하였다.

Applying Object-Oriented Technology for Development PDM Framework

Jeong-Ah Kim†

ABSTRACT

Many companies are investing in the development of the PDM(Product Data Management) system to improve the productivities of manufacturing since people believe that PDM technology can give a new solution from planning to development As the requirements of PDM grows, so many industries spend their own budgets on developing common requirements and functionalities In this paper, we describe the framework for a PDM application to promote reuse in a PDM system development area However, developing the framework is not easy. In this paper, the current state and results of our development are described 1) the phases of developing our framework. 2) abstraction strategies, 3) programming model based on repository Although frameworks improve software reuse, frameworks are so large and complex that application developers require to understand frameworks in order to customize the framework To support the customization of framework, development environments is developed, also

1. 서 론

PDM은 제품의 개념 정의에서부터 설계, 개발, 제조, 출하 그리고 고객 서비스에 이르기까지, 각종 정보들을 연관시켜 저장, 관리 및 제어하고, 제품의 전 라이프싸이클(LifeCycle)에 걸쳐 발생하는 각종 데이터와 프로세스에 의한 워크플로우(Workflow)까지 지원하는 시스템이다[11-13] PDM은 제품과 관련된 모든 주변 정보를 제공하는 제품 중심의 정보체계로서, 제조업체의 내부 또는 외부에서 제품과 관련된 일을 하는 모든

† 정 회 원 : 관동대학교 컴퓨터 교육과 교수
논문접수 : 2000년 3월 22일, 심사완료 : 2000년 4월 27일

조직의 구성원이 병렬적으로 작업을 진행할 수 있는 기반을 제공하는 정보 하부구조이다. 이로써 업체의 원가 절감 및 품질 향상에 기여하는 바가 크기 때문에 각 업체마다 PDM 시스템 개발에 투자를 늘리고 있는 실정이다.

본 연구에서는 반복되는 PDM 시스템 개발의 생산성과 효율성을 기하기 위하여 프레임워크 개발 기법을 적용하였다. 프레임워크 기반 PDM 어플리케이션 개발의 시도는 처음 시작하는 것은 아니다. 지금까지의 PDM 시스템 개발 과정을 보면 MetaPhase[11], WorkManager [13], Matrix[13]와 같은 프레임워크를 기반으로 재정의 과정을 거쳐서 이루어져 왔다. 이들 프레임워크는 기본적으로는 객체 지향 프레임워크가 아니며 재정의 역시 C로 정의된 API(Application Programming Interface)를 통해 이루어져 왔다. 또한 우리 나라 중소 기업 실정에는 이들 프레임워크는 너무 규모와 범위가 큰 단점을 가지고 있다. 그러므로 본 연구는 객체 지향 방법론에 근거한 중소기업용 PDM 프레임워크를 개발하는 것을 목표로 한다.

프레임워크는 시스템 전부나 일부에 대해 추상화 클래스로 정의된 재사용 가능한 요소들이고, 이들의 인스턴스가 상호 동작하는 방식을 함께 정의하고 있으며 클래스 라이브러리보다 큰 규모의 재사용 단위이다[1, 2, 4]. 프레임워크는 추상화 클래스들과 이들 간의 상호 동작 방법을 통해 어플리케이션에 필요한 기본 틀을 제공하며, 새로운 어플리케이션 개발자들은 이 프레임워크를 수정 및 조정하므로써 새로운 어플리케이션을 생성할 수 있다. 프레임워크의 재사용은 도메인 영역의 재사용을 가능하게 하며, 프레임워크는 미리 정의된 제어 흐름을 갖고 있어서 어플리케이션 설계자는 어느 특정 부품에만 집중하면 된다. 설계자가 수정한 부품은 프레임워크가 수행되는 동안 프레임워크에 의해 호출되어진다.

그러나 프레임워크를 개발하는 것은 하나의 어플리케이션을 개발하는 것에 비해 더 많은 노력과 공정을 필요로 하는 작업이다[7]. 본 논문에서는 중소기업형 PDM 프레임워크(UniPDM)와 이의 효과적인 재사용을 지원하는 재정의 도구의 개발을 목표로 하였다. 제안된 프레임워크 개발 공정[5]을 바탕으로 도메인 분석 과정을 거쳐 프레임워크 분석, 설계, 구현 공정을 통해 UniPDM Ver1.0을 개발하였다. 또한 이를 지원하는 도구를 모델링 도구, 클라이언트 위저드(Wizard), 데이터 베이스 관리 도구를 통합한 환경으로 구축하였다. UniPDM은 CORBA 기반의 클라이언트/서버 환경에서 운영 가능한 PDM 시스템을 지원하는 도메인 프레임워크와 White-box 프레임워크의 특성을 갖도록 개발하였다. 이 과정에서 프레임워크의 재사용성을 극대화하기 위해, 프레임워크를 구성하는 각 요소들간의 독립성을 보장하면서 각 요소들의 확장 및 수정을 지원하는 추상화 전략, PDM 도메인의 영역 특성을 반영한 추상화 전략, PDM 설계에 반영한 패턴 및 새롭게 정의한 패턴, 프레임워크 구현에 적용한 추상화 전략을 제안하였다. 이 추상화 전략은 UniPDM 설계 및 구현에 적용되었으며, UniPDM을 통한 새로운 PDM 어플리케이션 개발의 효율성 증진 및 UniPDM의 재사용성을 증진하는 효과를 얻었다.

2장에서는 프레임워크의 기본 특성 및 PDM 도메인에 대해 기술하였으며, 3장에서는 UniPDM의 개발 공정 및 제안하는 추상화 전략을, 4장에서는 개발한 UniPDM의 재정의를 지원하는 도구와 재정의 사례를 기술하고 5장에서 결론을 맺는다

2. PDM 프레임워크의 특성

PDM은 제품의 개념 정의에서부터 설계, 개발, 제조, 출하 그리고 고객 서비스에 이르기까지, 제품의 전 라이프싸이클에 걸쳐 발생하는 각종 데이터와 정보의 흐름을 효율적으로 제어하고 관리하여 주는 시스템 및 서비스가 통합된 것을 말한다. PDM 이란 부품 정보, 제품의 구성, 문서, CAD 파일, 결재 정보 등을 포함한 제품을 기술하는 제품과 관련된 정보와 결재와 배포 정보를 포함한 공정과 관련된 정의와 관리 정보를 포함한 제품과 관련된 모든 공정을 관리하는 기술이다.

PDM 시스템을 크게 두 부분으로 구분하면 자료 관리 및 프로세스 관리로 구분할 수 있다. 제조 업체에서 일반적으로 부품을 체계적으로 자료화시키고 문서, 도면을 모아두는 것뿐만 아니라 포괄적인 정보관리를 지원하는 부분이 자료 관리이다. 이는 부품의 분류, 문서의 분류 같은 분류 기능과 제품의 구조관리 기능 및 데이터 질의 기능의 기본 기능까지도 포함한다. 자료 관리에서는 데이터를 구성하고, 그 데이터의 접근을 용이하게 하며, 데이터를 조회하는 것을 다루는 반면 프로세스 관리는 데이터를 생성하고 수정하는 것과 관련된 실제의 행위를 관리하는 것이다 프로세스 관리는 Work 관리, Workflow 관리, Work History 관리의

세 가지 큰 기능으로 구성된다[13].

3. UniPDM 프레임워크 개발

3.1 UniPDM 프레임워크의 특성

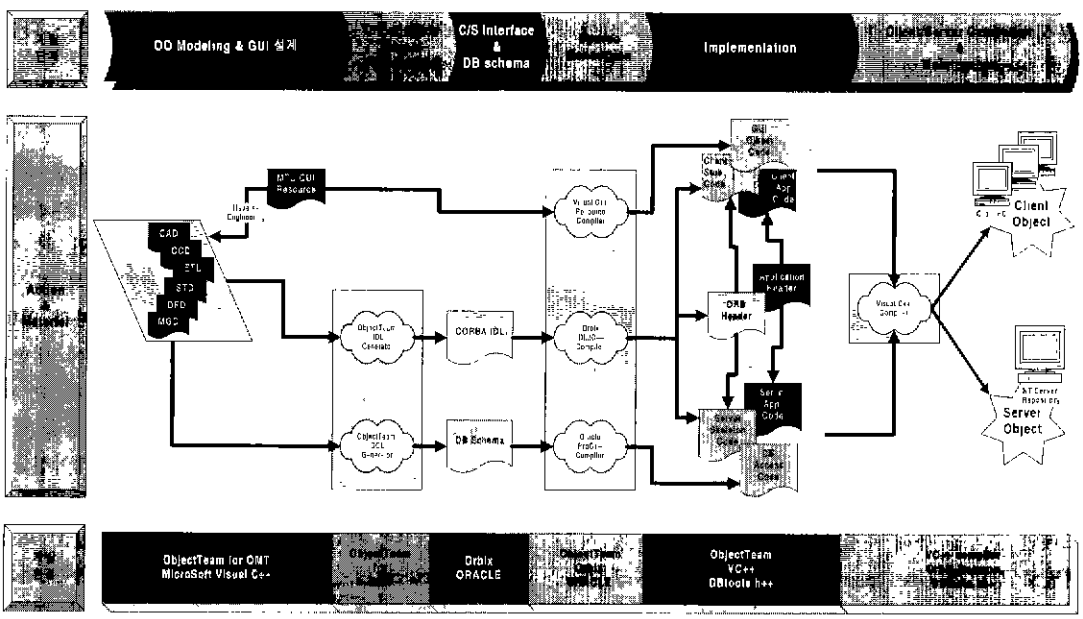
UniPDM의 소프트웨어 및 하드웨어 구성을 먼저 정의하면 다음과 같다. 운영 체제는 중소 기업의 실정을 고려하여 Window NT에 UniPDM 프레임워크 서버를 구축하였으며 기존의 시스템과의 연동을 고려하여 Oracle을 채택하였으나, NT에서의 Oracle의 사용 용이성과 데이터 베이스들간의 상호 운영성을 보장하기 위해 RogueWave사의 DBTools.h++ 라이브러리를 사용하였다. PDM의 발전 방향을 고려하여 분산 환경 및 서버 운영 체제의 이질성을 지원하기 위해 Orbix를 미들웨어로 채택하였다[10]. 클라이언트는 Window95 상에서 Visual C++의 MFC를 기반으로 하고 있다 (그림 1)은 PDM 프레임워크 개발 공정을 보여주고 있다 객체 모델링 단계에서 도출된 객체 모델을 바탕으로 데이터 베이스 스키마 및 클라이언트/서버간의 인터페이스를 정의한 후 IDL 컴파일러로부터 생성된 서버 스켈레톤(Skeleton) 및 클라이언트 스터브(Stub)를 이용하여 비즈니스 로직을 Visual C++로 구현하여 클라이언트 객체들과 서버 객체들을 개발하였다.

<표 1>에 UniPDM에서 개발한 기능의 범위를 정의하였다.

<표 1> UniPDM의 기능

기 능	특 성
사용자 관리	PDM시스템 사용자, 사용자그룹, 조직정보에 대한 등록, 삭제, 수정, 검색 기능 수행
전자결재 관리	도면/문서/모델/부품에 대한 결재 및 배포처리, 전자게시판 기능 우편함 관리 기능 수행
부품정보 관리	단품, 조립품에 대한 코드체번, 검색, 수정, 관련 도면과의 연결, 결재, 배포 기능 및 대, 중, 소, Category에 의한 품종 분류체계 관리
도면/문서 관리	이미지(tif), Pro-Engineering에서 발생된 hpgl형식의 도면파일에 대한 index정보관리, 검색, 출력, 결재, 배포, 수정, 삭제 기능 및 훈민정음, 엑셀, text, 이미지(tif) 등의 문서파일에 대한 index 정보관리. 검색, 출력, 수정, 삭제, 결재, 배포, 양식관리 기능 수행
계품구조 관리	제품구조체계, 구성품간의 관계, BOM관리 등 제품구성정보 및 대안품, 대체품, 옵션부, 유효성 등의 제품구성 부가정보 관리
과제정보 관리기	과제에 대한 기본정보, 프로세스정보, 인력정보에 대한 관리, 진행상황모니터링, 과제임무수행지원, 산출물관리 기능 수행
설계변경 관리기	설계변경에 대한 발의, 결재, 작업알림, 작업전송, 출력, 프로세스처리 기능
시스템 기본관리	시스템 공통사항, 하위시스템의 형상 등에 관한 시스템 전반적 정보에 대한 사항을 조절, 관리

이들 상세 기능은 기본적으로 워크플로우 기능을 필



(그림 1) UniPDM 개발 공정

요로 한다. UniPDM 프레임워크는 PDM 영역 중에서도 자료 관리 부분에 중점을 두고 개발하였다 그 이유는 프로세스 관리 부분은 기존의 워크플로우 엔진을 도입하는 것을 전제로 하였기 때문이다

3.2 UniPDM 개발 공정

본 논문에서는 UniPDM 개발을 위하여 [5, 10]에 제시된 프레임워크 개발공정을 적용하였다

3.2.1 도메인 분석(Domain Analysis)

도메인 분석은 주어진 PDM 영역에서 개발되어온 또는 앞으로 개발될 모든 어플리케이션 공통으로 적용될 비즈니스 프로세스를 식별하고 이에 필요한 객체를 식별, 추출하여 도메인 모델(Domain Model)을 구축하는 것이 목표였다. 도메인 분석을 위해서 먼저 PDM에 관한 일반적 영역 정보를 추출하기 위하여 PDM Buyer's Guide[13]를 참조하였는데, 국제 표준 문서로서 PDM 시스템이 갖추어야할 기능적 측면 뿐 아니라 사용의 편이성에 관한 내용까지 정의하고 있기 때문이다. 또한 기존의 MetaPhase, Matrix, WorkManager 등의 기존 시스템을 분석하여 일반적으로 외국에서 개발되어진 PDM 시스템의 기능성을 분석하였다 무엇보다 UniPDM의 목표는 국내 중소기업을 위한 것이므로, 우리 나라에서 개발된 적이 있거나 개발 계획을 갖고 있는 회사를 중심으로 분석하였다. 이들을 중심으로 도메인에 관한 지식의 습득과 정형화된 모델의 구축을 위해서 Use Case Diagram 및 각 Use case 별로 사용 시나리오(use scenario)를 지연어 형태의 기술서로 작성하였다 참조 및 분석한 각각의 시스템마다 객체 모델을 작성하여 각 시스템의 개념적 모델을 구축하였다. 서로 다른 시스템의 객체 모델을 비교 하므로써 PDM 도메인에서의 일반적인 Use case, 사용 시나리오, 공통의 객체 모델의 추출이 가능하였다. 이는 추후 프레임워크를 설계할 때 Template로 정의할 수 있는 후보로 정의하였다 또한 시스템마다 다른 부분을 식별하므로써 프레임워크의 Hot Spot의 후보로 정의할 수 있었다.

3.2.2 시스템 분석

도메인 분석을 통해 추출한 공통의 Use Case 및 시스템별로 상이한 Use case를 구분하여 프레임워크 개발을 위해 일반적인 객체 지향 요구 분석을 수행하였다. 먼저 프레임워크의 논리적 관점을 표현하는 객체 모델을 작성하였으며 이는 도메인 분석에서 추출한 모든 도메인에 공통인 객체들로만 구성된 객체 모델과 비슷하였다. 단, 시스템 개발의 효율성을 위해 PDM 프레임워크를 5개의 서브 시스템으로 구성한 후 각각에 대한 객체 모델을 구축하였다. 5개의 서브 시스템은 사용자 관리, 문서 관리, BOM, 프로젝트 관리, 변경 관리등이다 이 객체 모델을 기반으로 프레임워크의 동적 특성을 정의하는 객체 협력 다이아그램(Object Collaboration Diagram)와 객체 상호 작용 다이아그램(Object Interaction Diagram)과 필요한 경우에 상태 전이 다이아그램(State Transition Diagram)을 작성하였다. 분석 과정은 UML[9]을 기반으로 수행하였다.

3.2.3 일반화 및 추상화

프레임워크에서 확장성과 재사용성을 보장하기 위해 가장 중요한 것은 일반화와 추상화이다. 일반적인 모델을 바탕으로 확장의 여지를 담고 있는 추상화 클래스(Abstract Class)들이 정의되어 있어야, 새로운 어플리케이션을 개발 할 때, 이들 클래스를 상속과 폴리모피즘(Polymorphism)의 특성을 통해 확장할 수 있기 때문이다 일반화의 목적은 서로 다른 시스템의 분석의 결과를 프레임워크에 Template 클래스로 정의할 부분과 Hot spot으로 정의할 부분을 정의하는 것으로, 객체 지향의 일반화/상세화(Generalization/Specialization) 기법을 기반으로 하였다. 추상화 전략은 다음절에 상세하게 기술하였다.

3.2.4 패턴 적용

설계 패턴은 프레임워크를 문서화하는 가장 보편적이며 효과적인 방법이다[2, 8] 설계 패턴은 문제에 대한 일반적인 해결안을 제안하는 것으로[3, 8]. 본 연구에서도 프레임워크 개발을 위한 도메인 분석 후 프레임워크 설계 단계에서 Hot spot의 특성에 따라 일반적이고 표준화된 설계 패턴을 적용하였다. 이에 대한 지침의 일부를 <표 2>에 기술하였다 가장 유용한 패턴들로는 Strategy, State, Builder, Command 등을 들 수 있다 객체 모델에 정의된 객체의 각 오퍼레이션들 중 각 회사마다 서로 다른 비즈니스 정책을 가진 경우들이 많았는데, 이는 Strategy를 통한 확장성을 보장할 필요가 있는 예이다. 어떤 경우에는 서로 다른 어플리케이션 마다 객체에 대한 개념은 동일하지만, 이 객체를 운영하는 과정에서 객체가 가질 수 있는 상태의 개

수가 서로 다르게 정의되고 이에 따른 행위가 서로 다른 경우를 볼 수 있었다. 이에 유연하게 대처하기 위하여 State와 Strategy의 패턴을 함께 적용하였다. 패턴을 적용하여 설계한 UniPDM 프레임워크 모델의 일부는 (그림 2)와 같다.

〈표 2〉 패턴 적용 지침

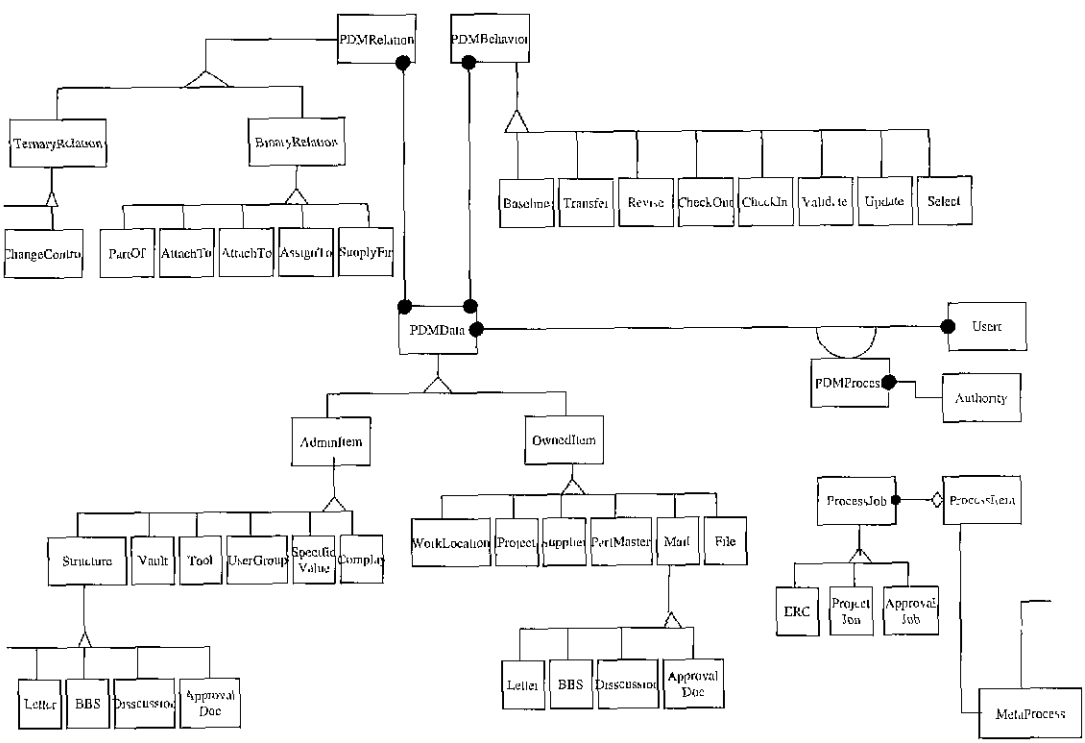
Guideline	Pattern Name
동일한 PDM 자료 항목들의 집합	Collection Member & Iterator
동일한 상위 클래스를 갖는 상이한 PDM 자료들의 집합	Composite & Iterator
새로운 속성이 추가될 수 있는 가능성이 있는 클래스	Decorator & Interpreter
기존의 모델에 정의된 객체와 같은 비즈니스 개념이기는 하지만 새로운 정책이 나타날 수 있는 경우	State & Strategy
새로운 행위가 추가될 수 있는 경우	Strategy & Adapter
새로운 오퍼레이션이 추가될 수 있는 경우	Command
사용자 인터페이스와 기존의 도메인 클래스간의 연결성이 추가될 경우	Memento & Prototype
기 타	Inheritance

3.3 UniPDM 개발을 위한 추상화 전략

본 절에서는 UniPDM 프레임워크 개발에 적용한 추상화 전략을 중심으로 기술한다. PDM 프레임워크를 개발한 목적은 복잡한 PDM 도메인을 쉽게 이해하고 프레임워크를 구성하는 컴포넌트를 독립적으로 수정, 확장할 수 있는 재사용성을 증대하므로써 PDM 어플리케이션 개발을 쉽게 하려는 것이다. UniPDM의 재사용성과 확장성을 높이기 위해서 객체 지향의 기본 추상화 개념인 자료 추상화와 타입 추상화를 바탕으로 보다 강력한 추상화 규칙을 정의하여 개발에 적용하였다[10, 16]

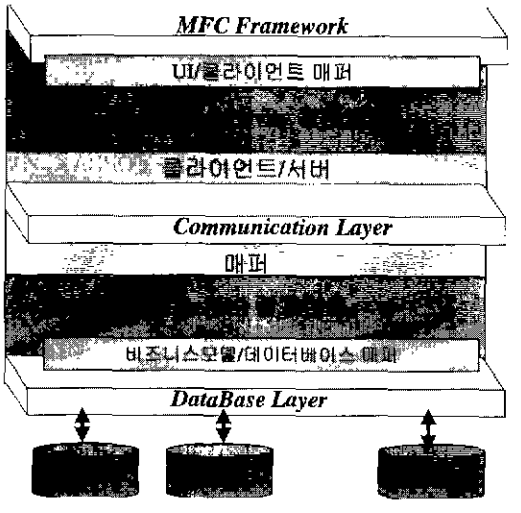
3.3.1 클라이언트/서버를 지원하는 UniPDM 아키텍치

먼저 UniPDM 소프트웨어 아키텍처 측면에서의 확장성을 보장하기 위한 것으로 UniPDM 개발에 적용한 첫 번째 추상화 전략이다. UniPDM은 도메인 프레임워크이지만 도메인 지식만을 프레임워크로 구축한 것이 아니라 어플리케이션이 운영될 수 있는 하드웨어 및 소프트웨어를 포함하고 있다. UniPDM은 CORBA를 기반으로 서버와 클라이언트 부분으로 구분되어 있으며



〈그림 2〉 패턴까지 적용한 UniPDM 객체 모델 - 일부

PDM의 저장소로 Oracle을 사용하고 있다. 그러나 데이터 베이스의 상호 운영성 보장을 위해 RogueWave사에게 개발한 DBTools.h-+ 라이브러리를 사용하고 있다. 다시 말해서 UniPDM 프레임워크 개발에는 PDM 도메인을 지원하는 도메인 프레임워크와 네트워크를 지원하는 CORBA, 사용자 인터페이스를 지원하는 MFC, 데이터 베이스를 지원하는 RogueWave등의 어플리케이션 프레임워크들이 통합되어 있다. 이러한 어플리케이션 프레임워크들은 다른 PDM 어플리케이션에서는 얼마든지 변경할 수 있는 부분들이다 PDM 도메인 프레임워크의 독립성과 확장성을 보장하기 위하여 (그림 3)과 같은 아키텍처를 적용하였다



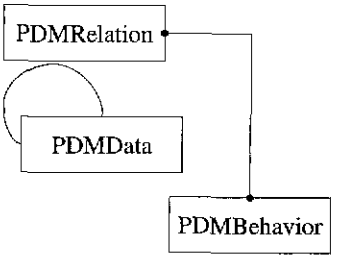
(그림 3) UniPDM의 아키텍처

CORBA에 해당하는 통신 계층(Communication layer), DBTools.h++에 해당하는 데이터 베이스 계층, Visual C++로 구성되는 MFC 프레임워크 계층은 기존에 상용화된 어플리케이션 프레임워크이다. CORBA를 중심으로 PDM 도메인 프래임워크를 클라이언트 Activity 컴포넌트와 비즈니스 서버 컴포넌트로 구분하였다 또한 각 도메인 컴포넌트와 기존의 어플리케이션 컴포넌트간의 직접적인 종속성을 없애주기 위한 UI/클라이언트 매퍼, 클라이언트 서버 매퍼와 비즈니스모델/데이터 베이스 매퍼를 두었다 도메인 프레임워크 구성요소들이 다른 어플리케이션 프레임워크와의 유기적 통합을 위해 3가지의 메퍼를 정의한 것이다. 첫 번째는 사용자 인터페이스 계층과 클라이언트 컴포넌트간의 연결성을 정의하고 있는 것이 UI/클라이언트 매퍼이다. 윈도우, 각종 콘트롤들로 구성된 사용자 인터페이스와 이들에게 실 사용자(end-user)가 보내는 이벤트들과 클라이언트 컴포넌트간의 연결 정보를 관리하는 매퍼이다. 이로써 사용자 인터페이스를 구성하는 콘트롤 또는 윈도우를 변경, 대체 할 수 있고 사용자와의 상호작용 방법을 사용자 인터페이스와 무관하게 변경할 수 있는 서로간의 독립성을 보장한다 두 번째는 클라이언트 컴포넌트와 서버 컴포넌트간의 인터페이스를 정의하는 메퍼이다. 이는 클라이언트의 요구를 어느 서버의 인터페이스에 전달해야 하는지 서버의 결과를 어느 클라이언트 컴포넌트에게 전달할 것인 지의 연결 정보를 관리한다 이로써 클라이언트 컴포넌트와 서버 컴포넌트간의 독립성을 보장한다. 세 번째는 서버 컴포넌트와 영구 저장소간의 대응 관계를 정의하는 메퍼로 서버에 정의된 클래스와 데이터 베이스의 테이블과의 대응 정보 및 클래스의 속성과 테이블의 필드간의 대응 정보들을 관리한다. 이로써 클래스 스키마의 변경이나 데이터 베이스 스키마의 변경은 서로 독립적으로 이루어질 수 있다

3.3 2 PDM 도메인 추상화 클래스

UniPDM은 도메인 프레임워크로 PDM 도메인을 지원하는 프레임워크이다 그러므로 PDM 어플리케이션 개발을 적극적으로 지원할 수 있어야 하고, 이를 위해서는 PDM 도메인에서의 Hot spot을 정의하는 것이 가장 중요한 일이다. 두 번째는 PDM 도메인에 대한 추상화 전략으로 도메인 개념의 확장 및 수정을 용이하게 하려는 목적을 갖는다. 이를 달성하기 위하여 PDM 도메인 분석을 통해 PDM 도메인의 일반적 특성을 추상화 클래스로 정의하였는데, 이들은 PDM 도메인에 공통인 클래스 구조와 행위들을 정의하고 있는 템플리트 클래스의 특성을 지니고 있으면서, 이들 클래스 안에 virtual 함수의 형태로 Hot spot을 정의하였다. UniPDM은 자료 관리 부분을 중심으로 개발되었으므로 이번에 제안한 추상화 클래스 역시 자료 관리를 지원하는 클래스들로만 구성하였다. 도메인 추상화 모델은 (그림 4)와 같다.

간단한 예를 들어서 (그림 4)의 모델이 갖는 배경을 설명하면 다음과 같다. 부품 정보 관리 영역을 분석해 보면 Part라는 클래스를 추출할 수 있다. 부품에 필요한 기본적인 속성과 오퍼레이션을 정의하고 있다. 도메



(그림 4) PDM 도메인 프레임워크 추상화 모델

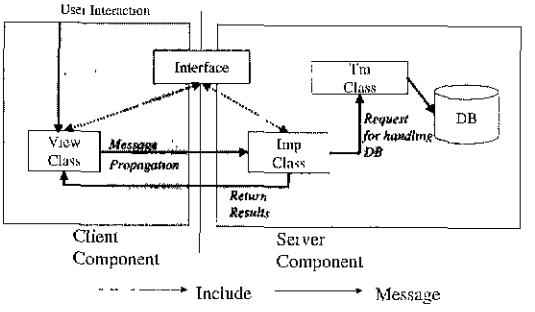
인 분석 과정에서 다양한 어플리케이션 분석을 통해 Part라는 개념은 동일하지만 업체마다 서로 다른 종류의 속성과 행위를 요구한다는 것을 발견하였다. 더욱 흥미로웠던 것은 각 어플리케이션 마다 Part라는 일반적 개념을 재정의하기 위한 요구 사항이 동일하지 않다는 것이다. 어떤 경우는 동일한 행위를 유지하면서 속성의 구조만 재정의 해야 하는 경우가 있는가 하면, 동일한 오퍼레이션에 대한 행위 처리 전략이 다른 경우도 존재했다 가끔은 도메인 개념들간의 관련성을 재정의 해야 하는 경우도 있었다. 예를 들어 Part 클래스에 미리 정의한 attachToDoc 연결 관계를 attachToMaterial로 변경해야 하는 경우도 있었다. 이를 분석한 후 PDM 도메인 역시 일반적인 객체의 재정의 개념과 유사성을 지니고 있으며, 이들의 변경 가능한 요인을 다음의 3가지로 정리하였다.

① 속성의 첨삭
② 오퍼레이션의 첨삭 또는 행위의 재정의
③ 관련성의 첨삭

이들 재정의 가능성에 유연하게 대처하기 위해 도메인 프레임워크의 최상위에 (그림 4)의 추상화 클래스를 정의한 것이다. PDMData 클래스는 속성들과 이들 속성을 다루는데 필요한 아주 간단한 오퍼레이션들을 정의하고 있는 자료 중심적(Data-Centric) 클래스이다. PDMBehavior 클래스는 PDM 자료들을 처리하는 비즈니스 처리 논리 또는 전략에 해당하는 클래스로 추후 재정의 가능하다. 클래스에 정의되는 오퍼레이션과는 반드시 구분되어야 한다. 즉 클래스에 정의되는 모든 오퍼레이션을 PDMBehavior의 하위 클래스화 할 수 없다. 복잡한 객체들간의 관련성으로 인한 업무 처리 논리만을 PDMBehavior의 하위 클래스로 정의하였다. PDMRelation 클래스는 PDMData 간의 연결성에 대한 추상화 클래스이다. 연결 관계 자체를 객체화하는 것은 새로운 개념은 아니다. 본 논문에서도 연결 관계로 인한 속성이나 행위들은 갖고 있는 관련성이나 PDMData 들간의 연결 관계가 변화 가능한 Hot spot으로 식별된 경우 PDMRelation의 하위 클래스로 정의하였다 예를 들면 AttchTo, Authorization등은 PDMRelation의 하위 클래스들이다.

3.3 3 UniPDM 프로그래밍 추상화

설계된 프레임워크 모델을 실제 프로그램으로 개발함에 있어 실질적 프로그램의 확장성과 각 모듈의 독립적 재사용성을 보장하기 위하여 (그림 5)의 프로그래밍 모델을 제안하였다. 물론 이 모델은 UniPDM의 개발 환경인 CORBA 및 Visual C++ 환경과 MVP모델을 적용한 것이다[6].



(그림 5) UniPDM 프로그래밍 모델

① View 클래스는 사용자 인터페이스를 구성하는 클래스로 MFC 프레임워크를 기반으로 작성한 클래스이다. 단, 기존의 Visual C++가 사용자 인터페이스를 관리하는 RC 방식이 아니라 사용자 인터페이스에 대한 스크립트를 따로 정의하고 이를 해석하여 MFC용 사용자 인터페이스를 만드는 방식을 채택하였으며 이를 처리하는 PDMView 클래스를 정의하였다 이는 프로그램 수행시에 객체를 대체할 수 있도록 하는 RunTime Object 개발 기법을 도입한 것이다[14]. 이를 위해 UniPDM의 모든 사용자 인터페이스 클래스는 PDMView의 하위 클래스로 정의하였다. 이를 따르는 기본적인 예를 들면 다음 (리스트 1)과 같다.

```
#include "TKUIClassFig.h"          // TKUIClassFig.h : 프레임워크내에 정의된 PDMView 클래스
class CDefaultView : public CPDMView          {
// 프레임워크에 추가될 새로운 사용자 인터페이스 클래스
public:
        CString m_szViewName;
        TKUIClassFig*  m_pUIClassFig,          // 사용자 인터페이스에 대한 스크립트에 대한 포인터

public:
        BOOL  QueryForm();                     // 스크립트를 번역하는 오퍼레이션
        BOOL  CreateControls();                // 스크립트에 따라 MFC 콘트롤을 생성하는 오퍼레이션
        ...
};
```

(리스트 1) PDMView를 통한 새로운 View 클래스 정의 규칙

② 인터페이스 : CORBA의 IDL 파일로 클라이언트와 서버간의 인터페이스 역할을 담당한다. UniPDM에서는 IDL 파일과 이를 컴파일해서 얻어지는 클라이언트 스터브 및 서버 스켈레톤 뿐만아니라 클라이언트 컴포넌트와 서버 컴포넌트간의 대응성을 따로 정의하고 관리하는 클래스를 따로 정의하였으며, 새로운 대응성은 이 클래스를 상속받아 정의한다.

③ ImpClass는 비즈니스 논리를 만족하는 모든 서버 컴포넌트를 의미한다. ImpClass와 데이터 베이스 저장소간의 연결 역시 저장소에 대한 직접적인 접근이 아닌 TmClass에게 그 책임을 일임한다. ImpClass에는 서버 클래스들간의 업무처리 흐름만을 정의하고 있을 뿐이다.

④ TmClass는 데이터 베이스를 다루는 논리를 담당하는 코드를 구현한 클래스이다. 개념적으로는 도메인에 속한 비즈니스 개념 하나에 대응하는 TmClass를 정의해야 하지만 TmClass의 구현적 관점에서 보면 데이터 베이스를 관리하는 기본 논리는 동일하다. 이런 특성으로 UniPDM에서는 하나의 TmClass로 구현하였다. 단, 다양한 비즈니스 개념을 처리하고 앞으로 정의될 새로운 비즈니스 객체에 대한 저장소 관리를 위해 CORBA에 정의된 NVList와 namedValue개념을 도입하였다. 이를 프로그램에 적용하는 규칙은 (리스트 2)와 같다.

```
UfObjectFactory*Create (const char* businessObject,
UfNVList nvlist) {
        a. 비즈니스모델/데이타베이스 매퍼로부터
                businessObject에 대응하는 테이블명을 받아옴
        b. RWDBTable aTable = 받아온 테이블명을 이용하여 RWDBDatabase 로부터
                실제 테이블을 얻어옴
        c. RWDBInserter  anInserter = aTable.inserter(),
        d  for (nIndex = 0; nIndex < nvlist의 length;
                nIndex++) {
                NamedValue      namedValue = nvlist[nIndex],
                anInserter[namedValue.name] <<
                namedValue.value
          }             // 데이블 생성을 위한 정보 생성
        e 마크로 코드를 이용하여 businessObject에 해당하는 객체 생성
                UfObject            obj = new 마크로(interfaceName),
                obj->setInfo(nvlist)
                return  obj;
}
```

(리스트 2) 비즈니스 객체를 생성하는 ObjectFactory의 Create 메소드 재정의 기법

⑤ DB : PDM 시스템이 관리해야 하는 실제 정보를 저장하고 있는 저장소이다

4. 재정의 도구를 통한 UniPDM의 재사용성 증진

4.1 재정의 도구의 설계

본 논문에서 개발한 재정의 도구는 프레임워크에 정의된 CORBA 컴포넌트를 기반으로 비즈니스 논리, 연관된 사용자 인터페이스, 데이터 베이스 저장소를 포함한 이플리케이션을 개발할 수 있도록 지원하는 프레임워크 활용 도구이다.

이 도구의 특성을 정의하면 다음과 같다.

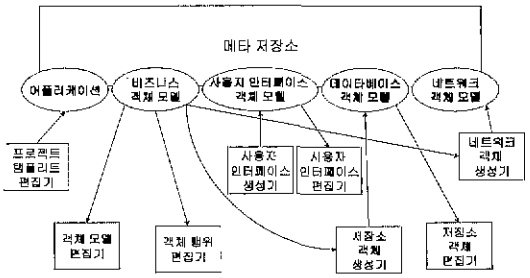
1) 비즈니스 모델링 요소를 확장한다. 이는 PDM 비즈니스 모델을 기반으로 모델의 확장 및 수정을 통해 전체적인 프레임워크의 재정의가 이루어진다. 새로운 PDM 비즈니스 객체가 등장하거나 프레임워크에 정의된 PDM 객체의 행위 및 구조를 변경해야 할 경우 비즈니스 객체 모델을 재구성할 수 있다.

2) 컴포넌트 모델링 도구이며 컴포넌트 생성 도구이

다 · 프레임워크를 구성하는 각각의 구성요소인 컴포넌트를 재정의하므로써 새로운 또는 제 정의된 컴포넌트를 생성하여 추가/변경된 요구사항을 만족하는 어플리케이션을 생성한다

3) PDM 프레임워크를 기반으로 동작한다. : 재정의 도구가 생성한 어플리케이션이 동작하기 위해서는 기본 행위를 제공하는 프레임워크를 필요로 한다.
4) 사용자 인터페이스의 재정의를 지원한다. : 프레임워크에서는 모든 것을 객체로 간주한다. 사용자 인터페이스 역시 하나의 객체로 인터페이스 외관에 해당하는 구조와 인터페이스의 행위를 재정의할 수 있다.
5) 컴포넌트 이해 및 재정의 대상 선정 과정을 지원한다. : 프레임워크를 구성하는 컴포넌트별로 컴포넌트의 구조와 행위를 이해하므로써 프레임워크가 제공하는 기본 행위를 새로운 요구 사항에 맞도록 재정의해야 하는 요소를 식별하게 된다.

이를 위해 기본적으로 프레임워크에 정의된 PDM 프레임워크 모델에 대한 메타 정보를 저장한 저장소(Repository)가 필수적이다[10, 15]. 앞에서 정의한 재정의 도구의 특성을 만족하도록 (그림 6)과 같이 설계하였다.



(그림 6) 메타저장소와 재정의 도구간의 관계

지원하는 PDM 프레임워크와 동일한 운영 환경에서 개발하였으며, 저장소는 Window NT, 오라클을 기반으로 구현하였으며 저장소를 통한 재정의 클라이언트는 Window 9x, Visual C++ 프로그래밍 환경을 지원하도록 개발하였다. 재정의 도구는 (그림 6)의 모듈 구성도에 따라 크게 5개의 서브 시스템으로 개발하였다. 각각의 서브 시스템의 구현된 기능은 다음과 같다

1) 도메인 클래스 생성기/관리기 : 프레임워크를 구성할 Model 클래스, View 클래스, Persistent 클래스등의 프레임워크 컴포넌트 클래스들을 생성하고 이들에 대한 메타 정보 및 매핑 정보를 관리한다. 프레임워크를 구성하는 도메인 클래스들의 계층 구조와 상호 연결 관계를 파악하고 있어 프레임워크 재정의한 결과를 바탕으로 새로운 어플리케이션 코드를 생성한다.
2) 사용자 인터페이스 클래스 생성기/관리기 : 본 연구 대상의 프레임워크는 실제 어플리케이션을 개발할 수 있는 사용자 인터페이스 컴포넌트까지 포함한 것으로 가정하였다. UI 관리 서브 시스템은 도메인 클래스를 서버 객체인 도메인 클래스와 클라이언트의 사용자 인터페이스 클래스와의 연결성을 관리한다. 도메인 클래스에서의 변경은 저장소에 저장한 매핑 관계를 통해서 자동으로 사용자 인터페이스 클래스에 반영시킨다. 또한 사용자 인터페이스는 기본적으로 사용자들의 편이성을 고려하여 빈번한 수정 보완될 수 있는 요소로 UI 관리 시스템은 Visual C++를 이용한 화면의 생성 및 편집을 지원한다.
3) IDL 클래스 생성기 : 본 연구에서 대상으로 하는 프레임워크는 CORBA상에서 운영되는 컴포넌트이고 객체들이므로 도메인 클래스의 경우는 CORBA객체로의 생성 및 기존의 CORBA 객체와의 매핑 관계를 관리하여야 한다. IDL 클래스 생성기는 도메인 클래스 생성기에 의해 정의되는 클래스들 중 CORBA객체로 지정되면 이를 IDL로 생성하고 CORBA 객체와의 매핑 정보를 생성 관리하게 된다.
4) 영속적 클래스 생성기 : 도메인 클래스 중 데이터베이스에 저장될 필요가 있는 경우 영속적 클래스를 정의하고 이 클래스에서 데이터 베이스 관련 처리를 담당하도록 하는 것이 일반적인 프레임워크 구현 방법이다. 영속적 클래스 생성기는 도메인 클래스와 영속적 클래스간의 대응 관계를 정의하므로써 도메인 클래스의 변경에 따른 영속적 클래스의 변경을 담당하게 된다
5) 프레임워크 메타 정보 저장소 설계 및 구현 : 각 도구들로부터 생성된 프레임워크 기본 클래스들에 대한 메타 정보와 클래스들간의 매핑 정보 및 참조 관계에 대한 메타 정보를 관리하는 저장소이다.

4.2 UniPDM의 재사용성에 대한 평가

재사용에 관한 평가를 두가지로 나누어 수행하였다. 하나는 프레임워크로써의 확장성 및 재사용성은 개발한 UniPDM을 기반으로 새로운 중소기업용 PDM을 만드는 과정을 통한 평가와 재정의 지원 도구의 기능성에 관한 평가이다 UniPDM을 기반으로 새로운 PDM 어플리케이션 개발시, UniPDM을 기반으로 도메인 분석을 수행하였다. 이는 UniPDM이 제공하는 클래스 프레임워크를 기반으로 새로운 도메인 클래스의 변경 없이 새로운 어플리케이션 개발이 가능한가와, 새로운 요구사항을 만족하지 않을 경우 필요한 비즈니스 클래스를 추가하거나 재정의하는 과정이 얼마나 쉽게 이루어지는가에 대한 평가이다. 완전한 평가가 이루어지기 위해서는 여러 번에 걸친 어플리케이션 개발이 필요하지만, 본 연구에서는 최근에 시작한 새로운 PDM 개발 사이트의 사례를 들어 정리하였다. 새로운 요구 사항의 경우는 크게 비즈니스 개념이 확장되어야 할 필요가 있지 않았으므로, 도메인 분석시 UniPDM이 제공하는 프레임워크 모델을 바탕으로 PDM에 가져야 하는 기능적 요구 사항의 분석을 수행했으며, 이 과정에서 완벽하게 새로운 비즈니스 논리보다는 미세한 속성의 첨삭이 필요하였다. 이는 PDMData와 NamedValue의 개념으로 새로운 속성을 정의하고 이에 필요한 Tm 클래스를 재정의하므로써 쉽게 처리할 수 있는 내용이었다. 또한 새로운 사용자 인터페이스 개발의 경우는 PDMView 클래스의 상속과 새로운 사용자 인터페이스 모델의 정의를 통해 이루어졌다. 이를 바탕으로 볼 때 UniPDM은 도메인 분석과 어플리케이션 분석의 기본으로 재사용할 수 있을 뿐 아니라, 실제 코딩 측면에서도 UniPDM이 제공하는 클래스의 상속을 통해 새로운 요구 사항을 만족하는 클래스를 쉽게 확장 할 수 있는 것으로 평가할 수 있다

두 번째 평가로는 재정의 도구에 대한 평가로, <표 3>에 제시한바와 같이 UniPDM의 재정의 도구가 제공하는 기능과 다른 PDM 프레임워크의 재정의 도구를 평가 해 보았다. 다른 재정의 환경이 보통 API를 기반으로 이루어지는 반면 UniPDM의 경우는 통합 및 대화식 환경으로 개발되어 있어, 프레임워크의 시각적 재정의가 가능하다는 장점을 지닌다. 또한 Mapper를 기반으로한 전체 아키텍처의 재사용을 지원하는데 다른 도구에 비해 장점을 지니는 것으로 평가할 수 있다.

〈표 3〉 프레임워크 재정의 도구 평가 표

분 류	UniPDM 기능 항목	WorkManager	Metaphase	Matrix
비즈니스 모델러	비즈니스 클래스 정의	○ Class 정의 기능	◎ MODEL로 관리	◎
	관련성 정의	○ Class 정의 후 외부적인 Relationship 정의 기능	◎ MODEL로 정의	◎ 상속관계 제공
	비즈니스 모델 관리	○ 계층구조 형식 관리 가능	◎ MODEL Viewer 제공	◎
	Component Repository 등록	◎ 재사용 가능	◎ Migration Tool 제공	◎
UI Builder	UI 및 모델 정의	◎ Template Editor 이용 Classification 유지	◎ ICE Browser 지원	○ Form이나 Report로 정의, 재정의
	스크립트 언어 정의 및 변환	◎ Macro API로 제안	◎ IMI 제공	◎ Form이나 Report가 자체 스크립트 내장
	리소스 변경	×	× GUI Server 제공	× 제공되지 않음
	파일 관리	◎ Template Editor를 이용 [illegible]	◎ ICE Browser 제공	◎ 디렉토리 관리

분 류	UniPDM 기능 항목	WorkManager	Metaphase	Matrix
UI Builder	Component Repository 등록	◎ Template Editor 이용	◎ Migration Tool 제공	◎
코드편집기 기능	코드편집	○ Macro API 이용 VI Editor로 편집 가능	◎ 에디터 이용	◎ MQL module에서 제공
	코드버전관리	×	◎ 소스코드 Check-In/Out	×
	UI/DB 코드편집	○ Macro API 이용 VI Editor로 편집 가능	◎ 에디터 이용	○
	패턴관리	△	○ Hot Spot [illegible]	◎
Visualizer	Trace	○ Trace Component를 통해서 Object history 저장	◎ IMI Debugger 제공	○ verbose on 기능
	데이터 계층	×	◎ IMI Debugger 제공	×

분 류	UniPDM 기능 항목	WorkManager	Metaphase	Matrix
UI/DB Mapper	Mapping	○ Template Editor에서 정의된 Class간의 Mapping	○ MODEL로 정의	○ Form, Report의 Field의 자체 속성으로 Mapping
	Mapping관리	○ Template Editor에 Class간의 Mapping 관리	○ MODEL로 정의	○
	Wrapping	△ 정의된 Class의 래핑으로 가능	△ IMI 데이터 타입 관련 지원	△
	화면출력	△ Template Editor에서 UI Tool 제공	○ UI 컴포넌트 API 지원	△
메세지/UI Mapper	Mapping	○ Template Editor에서 정의된 Class 관리	◎ MODEL로 관리	△ UI Component의 자체 메세지 처리 [illegible]
	Event Handler	○ Template에 [illegible] Event Message Macro로 구현	◎ Prototype Event 제공	○ 폼에서 Event에 Event Handler를 정의할 수 있음
	Mapping관리	○ Macro API를 통해 가능	◎ MODEL로 정의	△

■범례
- × 기능을 제공하지 않음
- △ 구현가능하나 복잡하고 쉽지 않음
- ○ Customizing하여 쉽게 구현가능함
- ◎ 우수함(조금만 Customizing하여 그대로 사용가능함)

5. 결 론

본 연구에서는 반복되는 PDM 시스템 개발의 생산성과 효율성을 기하기 위하여 프레임워크 개발 기법을 적용하였다. 프레임워크는 시스템 전부나 일부에 대해 추상화된 클래스로 정의된 재사용 가능한 요소들이다. 또한 인스탄스가 상호 동작하는 방식을 함께 정의하고 있으며 클래스 라이브러리보다 큰 규모의 재사용 단위로 도메인에 대한 일반적인 모델을 바탕으로 구체적인 어플리케이션을 개발할 수 있다. 이로써 프레임워크의 재사용은 도메인 영역의 재사용을 가능하게 한다.

그러나 프레임워크를 개발하는 것은 하나의 어플리케이션을 개발하는 것에 비해 더 많은 노력과 공정을

필요로 하는 작업이다. 본 논문에서는 중소기업형 PDM 프레임워크(UniPDM)와 이의 효과적인 재사용을 지원하는 재정의 도구를 개발하였다. 제안된 프레임워크 개발 공정을 바탕으로 도메인 분석 과정을 거쳐 프레임워크 분석, 설계, 구현 공정을 통해 UniPDM Ver1.0을 개발하였다. 또한 이를 지원하는 도구를 모델링 도구, 클라이언트 위저드(Wizard), 데이터 베이스 관리 도구를 통합한 환경으로 구축하였다. UniPDM은 CORBA 기반의 클라이언트/서버 환경에서 운영 가능한 PDM 시스템을 지원하는 도메인 프레임워크와 White-box 프레임워크의 특성을 갖도록 개발하였다 이를 위해서 프레임워크에 대한 메타 저장소를 구축하여, PDM에 대한 메타 데이터를 모델화 하여 정의하고 이를 바탕으로 새로운 어플리케이션을 재정의할 수 있는 재정의 지원 도구를 설계, 개발하였다. 재정의 지원도구를 통해 프레임워크의 기본 행위를 이해하고, 프레임워크 구성 컴포넌트를 이해하며 재정의가 필요한 요소를 식별하여 이들의 객체 모델, 사용자 인터페이스 모델, 저장소 모델을 하나의 환경에서 재정의할 수 있도록 하였다. 이로써 해당 도메인에서 어플리케이션 개발 경험이 부족한 개발자들도 짧은 교육 과정을 거쳐 새로운 어플리케이션을 개발할 수 있다. 또한 하나의 환경 하에 컴포넌트 모델링 및 컴포넌트 생성을 통한 어플리케이션의 개발이 이루어질 수 있다는 장점을 갖는다.

참 고 문 헌

[1] Pree, W., Framework Patterns, SIGS Books, New york, NY, 1996

[2] Pree,. W., Design Patterns for Object-Oriented Software Development, Addison-Wesley/ACM Press, Reading MA, 1995.

[3] Gamma, E., Helm, R, Johnson, R., and Vlissides, J Design Patterns Elements of Reusable Object-Oriented Software, Addison-Wesley/ACM Press, MA, 1995.

[4] R. Johnson, Frameworks = (Components+Patterns), CACM Vol.40., No.10, Oct., 1997. pp.39-42.

[5] Michael Mattsson, Object-Oriented Frameworks : A Survey of methodological issues, LU-CS-TR : pp.96-167.

[6] Mike Potel, MVP : Model-View-Presenter, The Taligent Programming Model for C++ and Java, Technical Report of Taligent, 1996.

[7] Paul Dustin Keefer, An Object Oriented Framework for Accounting Systems, MS Thesis of University of Illinois at Urbana-Champaign, 1994.

[8] Hans Albrecht Schmid, Design patterns for constructing the hot spots of a manufacturing framework, JOOP June, 1996, pp.25-37.

[9] Rational, *Object-Oriented Analysis and Design Using UML : Professional Services*, Notes, Rational Software Corporation, 1997.

[10] Jeong Ah Kim, Jin Hong Kim, Nam Kyu Park, Development of PDM framework and its customization environment, TOOLS Pacific28, 1998. pp.45-54.

[11] HP Metapahase : Object Management Framework, PDM Manual.

[12] *PDM Enablers : Joint Proposal to the OMG in Response to OMG Manufacturing Domain Task Force RFP1*, mfg/98-02-02, OMG.

[13] CIMData, PDM Buyer's Guide, 1994, pp.5-54

[14] Ed Motak, Jean Caputo, Dynamic Runtime Objects Building Applications Your Users Can Modify at Runtime, Microsoft Systems Journal, July. 1997, pp. 49-74.

[15] Martine Devos, Michel Tilman, A Repository-based framework for evolutionary software development, Technical Report, AF1179-1.

[16] 김정아, "객체 지향 프레임워크 개발을 위한 추상화 전략", 제1회 한국 소프트웨어 공학 학술대회 발표 논문집, 1999. pp.19-23.

김 정 아

e-mail clara@mail.kwandong.ac.kr

1990년 중앙대학교 컴퓨터 공학과 대학원 졸업(공학석사)

1994년 중앙대학교 컴퓨터 공학과 대학원 졸업(공학박사)

1994년~1996년 중앙대학교 기술과학 연구소 객원 연구원

1996년~현재 관동대학교 컴퓨터 교육과 교수

관심분야 : 객체 지향 방법론, 컴포넌트 개발 방법론, 소프트웨어 재사용 소프트웨어 품질 개선