Journal of Korea Multimedia Society Vol. 9, No. 10, October 2006(pp. 1361-1370)

유비쿼터스 환경의 물류관리업무를 대상으로 한 계층구조 컴포넌트의 설계 및 구현

권 오 현†

요 약

최근 인터넷 기술의 빠른 발전은 웹 애플리케이션 관련 다양한 기술을 적극적으로 활용하는 방향으로 진화되고 있으며 다양한 장비들 간의 호환성 있는 정보 교환 소요도 급증하고 있다. 이러한 추세는 유비쿼터스 시대로의 진입에 따라 변화되는 부분들 중 일부라 할 수 있다. 본 논문은 다양한 클라이언트 유형들로 구성된 물류정보 시스템을 대상으로 상호간에 효과적으로 정보교환을 할 수 있는 계층구조를 지닌 컴포넌트의 설계 및 구현에 초점을 두었다. 설계 및 구현 시 재사용성과 확장성 증대를 위해 독립적 계층 구조 개념의 컴포넌트들을 제시 하였으며 클라이언트 간 상호 원활한 인터페이스를 위해 XML 웹 서비스 기능을 적용하였다. 설계 및 구현 시 다양한 시험을 통한 검증을 위해 RFID 입력 장치, PDA, 데스크 탑 등 다양한 유형의 클라이언트들을 적용 대상으로 하였으며 클라이언트 유형 및 플랫폼에 관계없이 정상적으로 작동됨을 확인하였다. 본 논문에서 구현된 컴포넌트는 향후 유사 업무를 개발 시 효과적으로 재사용될 수 있을 것이다.

Design and Implementation of Layer Structured Components for the Material Management System Under the Ubiquitous Environments

Oh-Hyun Kwon†

ABSTRACT

Recently fast innovation of Internet technology causes lot of application to change into web application and requirement for the inter-operable communication among various systems are increased rapidly. These trends are part of changes which is caused by ubiquitous world and it is just beginning of huge waves that is required to fit and change under the ubiquitous environments. This thesis is focused on the Design and Implementation of Layer Structured Components which can be interfaced effectively for the Material Management System under the Heterogeneous Client Server Environments. The key points to do that kinds of affairs are using XML web services that can communicate thru systems and also using independent layered components for the enforcement of reusability and inter-operability. For the various type of testing of implementation, we used RFID System, PDA, Desktop Systems and confirmed the normal operation without concerning the type of client system and platforms. The Components proposed in this thesis could be reused effectively in case of developing similar applications.

Key words: Component(컴포넌트), Interface(인터페이스), Object(객체), Layers(계층), RFID(알에프아이디), Tier(티어), Xml(엑스엠엘), Client(클라이언트), Server(서버), PDA(피디에이), Ubiquitous(유비쿼터스), Web Service(웹서비스)

※ 교신저자(Corresponding Author) : 권오현, 주소 : 부산시 남구 용당동 535번지(608-711), 전화 : 051)610-8200, FAX : 051)610-8039, E-mail : ohkwon@tit.ac.kr

접수일 : 2006년 4월 10일, 완료일 : 2006년 7월 20일

† 종신회원, 동명대학교 정보통신대학 컴퓨터공학과 교수

1. 서 론

앞으로의 유비쿼터스 시대에는 사물, 사람, 컴퓨터 등을 포함해 지구상에 존재하는 많은 것들에 컴퓨팅 환경이 제공되어 정보를 교환 할 수 있는 시대가 될 것이다. 예를 들면, 자신의 건강 정보에서부터 냉장고에 있는 음식의 상태정보에 이르기까지 새롭고 다양한 정보를 제공하는 기업들도 수도 없이 생겨날 것이다. 고객들은 각각의 기업들이 독자적으로 개발한 시스템들로부터의 정보뿐만 아니라 이와 연관된 수많은 타 시스템 정보도 요구하게 될 것이다. 이 경우, 상호간에 인터페이스를 위해 필요한 소프트웨어 및 데이터는 기하급수적으로 늘어나게 되고 서로 다른 방식으로 개발된 애플리케이션간의 커뮤니케이션 및 공동 활용 문제는 중요한 이슈로 대두될 것으로 예측된다[1]. 이러한 문제를 기존의 전통적인 애플리케이션 개발 방법으로 대처하는 것은 한계가 있을 수밖에 없으며 그 해결방안으로 유력하게 등장한 것이 재사용성과 확장성을 크게 강화시킬 수 있는 컴포넌트 기반의 소프트웨어 개발 방법이라 할 수 있다[2,3]. 또한 웹 애플리케이션의 발전에 따라 웹 서비스의 기능을 보다 지능화시켜 컴포넌트화 시킨 후 이를 다양한 클라이언트에게 서비스하는 방법론이 대두되고 있으며 이러한 기능들을 효과적으로 결합시켜 활용할 경우 고객에 대한 다양한 정보의 제공 및 공동 활용 측면에서 시너지 효과를 거둘 수 있을 것으로 예상된다[2].

본 논문은 이에 발 맞추어 웹 서비스 및 개발한 컴포넌트들의 기능을 활용하여 서로 다른 클라이언트 또는 다른 시스템 환경에서도 객체들 간의 커뮤니케이션이 가능하도록 컴포넌트 기반의 시스템을 설계 및 구현하는 데 목표를 두었다. 이를 위해 유비쿼터스형 각종입출력 시스템의 필요성이 강조되는 물류 정보시스템을 적용대상으로 하였으며 개발환경은 닷넷 플랫폼을 적용하였다.

2. 관련 연구

2.1 계층구조

1990년대중 정립된 CBD(Component Based Development)는 기업용 소프트웨어의 구축에 있어 계층적 아키텍쳐(Tiered Architecture or Layered Architecture)를 전제로 한다. 1990년대 전반 클라이언트 Client/Server System의 기본 아키텍처가 정립되면서, 모든 기업용 소프트웨어는 최소한 사용자 인터페이스 계층(UI:User Interface Layer), 비즈니스 로직 계층(Business Logic Layer) 및 데이터베이스 계층(Database Layer)의 3계층으로 구성된다는데 전 소프트웨어 산업의 공감대가 형성되었다[4]. 사용자 인터페이스 계층은 클라이언트로부터 요청이 들어오면 그 요청을 비즈니스 로직 계층에 전달하며 비즈니스 로직 계층으로부터 받은 결과를 다시 클라이언트에게 보내주는 역할을 담당한다[5]. 비즈니스 로직 계층은 데이터베이스 계층과 직접적으로 통신을 하는 계층으로 어느 특정 도메인의 비즈니스 프로세스에 기여하는 단일 업무 기능을 처리해주는 서버측의 독립된 소프트웨어 모듈이라고 간주할 수 있다. 즉 비즈니스 컴포넌트는 기능적으로 응집력이 있는(Functionally Cohesive), 느슨하게 계층 결합된(Loosely Coupled), 캡슐화된(Encapsulated)소프트웨어 모듈인 것이다. 비즈니스 컴포넌트는 여러 프로젝트에서 재사용할 수 있어야 좋고, 나아가 기업 간, 업종 간, 응용 도메인 간에 널리 공용될 수 있으면 더욱 값진 것이다. 왜냐하면 한 번 잘 만든 부품이 여러 시스템에 쓰이면 그만큼 개발생산성과 품질이 향상되고, 비용은 저하되기 때문이다. 데이터베이스 계층은 데이터베이스에 접근하는 방법을 제공하는 계층으로 정의되었다. 이러한 계층구조는 컴포넌트의 응집력을 제고시켜 재사용성을 강화하기 위한 방향으로 지속적으로 진화되고 있다[6].

하지만 대부분의 상용 계층구조 시스템들은 대부분 복잡하고 무거운 구조를 가지고 있어 사용자 입장에서 자기 환경에 맞게 쉽게 조정하기가 어려울 뿐만 아니라 수행모듈 중심으로 컴포넌트가 공개되기 때문에 컴포넌트를 수정하거나 재활용하기가 어려운 실정이다[12].

2.2 웹 서비스

최근 들어 본격화되고 있는 웹 서비스(Web Service) 기술 및 산업의 발전은 IT 역사의 제4.5세대라 불러도 될 만큼 기업 정보시스템에 혁명적인 영향을 미칠 전망을 보인다. 웹 서비스를 정의하자면, 이는 느슨하게 계층 결합된(Loosely Coupled) 소프트웨어 컴포넌트로서, 그 서비스는 XML, SOAP 등 국

제표준 기술만을 이용하여 인터넷을 통해 제공된다. 웹 서비스 기술의 출현은 CBD에 의한 정보시스템 구축을 보다 쉽게 하면서 CBD의 이점을 보다 강화시키는 결과를 가져오고 있다[7]. 우선 웹 서비스 컴포넌트는 종래의 COM+, EJB 등과는 달리 각 컴포넌트를 실행시키고 있는 플랫폼에 무관하게 상호간 완벽한 호환성을 보장한다[8]. 이는 웹 서비스 컴포넌트 간의 호출이 XML, HTTP, SOAP 등 표준 인터넷 기술 만에 의존하기 때문이다. 또한 웹 서비스 컴포넌트들은 Universal Description, Discovery and Integration(UDDI)라는 컴포넌트 등록 메커니즘을 통해 동태적으로 결합되면서 복합적 서비스를 창출할 수 있다. 이와 같이 웹 서비스 기술은 종래의 COM+나 EJB 기술에 비해 비즈니스 컴포넌트들이 더 크고 복합된 기능을 가지면서도, 보다 쉽고, 비즈니스 컴포넌트의 조립에 의한 정보시스템의 구축을 보다 동태적으로 유연하게, 빠르게 하는 효과를 가지고 있다[9].

이에 따라 마이크로소프트사 등은 최근 웹 서비스 기술을 활용한 컴포넌트 재활용 분야에 역점을 두고 사업계획을 추진하고 있으나 아직은 초창기로 본격적으로 보편화되고 활용되는 데에는 좀 더 시간이 걸릴 것으로 예상된다.

3. 설계된 컴포넌트 구조

3.1 기본 설계구조

본 연구에서 설계된 컴포넌트 구조는 기본적으로 3계층 구조를 취하는데 프리젠테이션 계층, 비즈니스 계층, 데이터 계층 구조이다. 프리젠테이션 계층에는 UI Components와 UI Process Components로 구분하였는대 UI Components는 실제 사용하는 클라이언트들로부터 입출력되는 폼들을 컴포넌트화한 부분이며 UI Process Components는 UI Components로부터 요청되는 이벤트들을 하위 비즈니스 계층으로 전달해주고 처리결과를 다시 UI Components로 되돌려주는 역할을 담당한다. 본 계층구조에서 주의 깊게 보아야 할 부분은 비즈니스 계층이다. 개념적으로 볼 때, 비즈니스 계층을 제외한 계층들은 어떤 식으로 구성되어 있든 어떤 플랫폼을 사용하든 상관이 없다. 비즈니스 계층에서 나오는 모든 객체정보 및 데이터는 웹 서비스에 노출시켜 XML 데이터로 제공하는 구조로 설계를 하였기 때문에 클라이언트의 종류 및 유형에 상관없이 그 결과물을 전달 받을 수 있도록 하였다. 하지만 비즈니스 계층 자체는 해당 시스템의 특성에 상대적으로 영향을 많이 받는 부분인 관계로 비즈니스 계층의 변경시 타 부분에 미치는 영향을 최소화하기 위해서 별도로 독립적인 계층으로 설정하는 것이 바람직하다[10]. 따라서 비즈니스 계층은 프리젠테이션 계층의 요청을 처리하는 Service Interface와 데이터 계층의 Data Access Logic Components를 접근하여 결과값을 사용자 정의 객체로 변환시키는 Business Entities부분으로 구분하여 설계하였다. 그림 1은 핵심구조를 도식화한 것으로 애플리케이션 전체의 구조를 보여주고 있다.

3.2 세부 설계구조

3.2.1. 컴포넌트의 클래스 구성

본 설계에서 제시된 세부구조를 클래스 다이어그램 형태로 표현하면 그림 2와 같다. 먼저 우측상단에 있는 UIPressWD 와 UIPressInfo 클래스는 UI Process Components에 포함되는 클래스들이며 UI Components로부터 정보를 전달받아 비즈니스 계층의 서비스 인터페이스를 호출하는 역할을 담당한다. 그림 좌측의 UI Components는 해당 클라이언트 이벤트가 UI Process Components에게 요청을 전달하거나 특별히 데이터베이스로부터의 정보가 불필요한 경우는 직접 비즈니스 계층의 인터페이스를 호출하기도 하는데 ServiceCommand, ServicePlan, Service-ResourceM 등의 클래스 호출이 이에 해당된다.

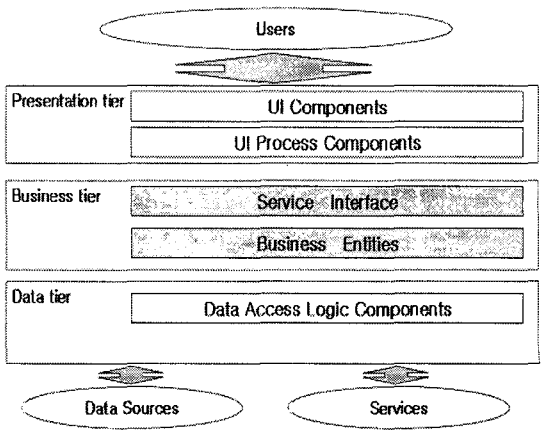

그림 1. 애플리케이션 전체 구조

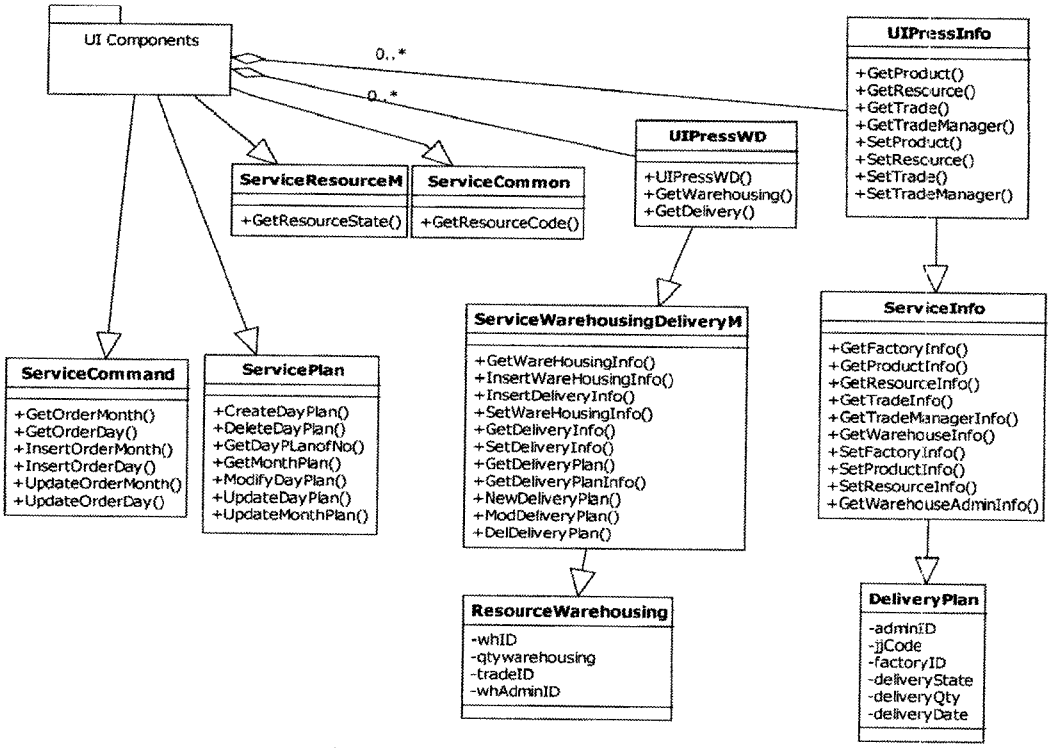

그림 2. 컴포넌트 설계구조의 클래스 다이어그램

Service Interface 컴포넌트에 속하는 클래스들 중 ServiceWarehousingDeliveryM 과 ServiceInfo 클래스는 DeliveryPlan 등의 Business Entity의 클래스들을 준비한 후 Data Tier를 액세스하여 필요한 정보를 처리하는 기능을 수행한다.

3.2.2 계층별 구조

(1) Presentation Tier Components

UI Process 컴포넌트는 사용자와의 인터페이스를 통해 발생하는 입출력 정보를 처리하기 위한 컴포넌트로서 여러 개의 UI Process 컴포넌트들의 조합으로 활용되는 경우에는 UI 컴포넌트가 대체 활용되기도 한다. 본 설계에서는 크게 2가지 유형의 컴포넌트를 제안하였으며 그 요약된 기능은 표 1과 같다. 그 중 UIPressInfo 컴포넌트는 기초정보를 처리하는 기능을 수행하며 그림 3처럼 GetProduct() 등 다수의 메소드로 구성되어 있고 이들은 여러 UI에서 재사용할 수 있다.

(2) Service Interfaces 컴포넌트

Business tier의 Service Interface 관련 컴포넌트는 모두 5개로 이루어졌으며 클래스별 수행 기능은 표 2와 같다.

Service Interface 컴포넌트는 비즈니스 로직을 수행하도록 조치를 취한 후 리턴되는 결과를 웹 서비스의 지원을 받아 XML 타입의 결과물로 반환 시킨다.

표 2의 ServiceWarehousingDeliveryM 이라는 서비스 인터페이스가 웹 서비스와의 연동을 통해 XML 타입으로 변환한 후 해당 클라이언트에게 노출시킨 결과가 그림 7-1에서 보여 진다.

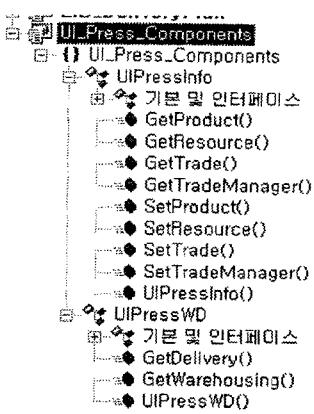

그림 3. UI Process 컴포넌트의 구성

표 1. UI Process 컴포넌트

컴포넌트명	클래스 명	기 능
UI Process Components	UIPressInfo	기초 정보를 가져오거나 업데이트하기 위해 Service Interface를 호출하고 결과를 받는다.
	UIPressWD	입고, 출고 정보를 가져오기 위해Service Interface를 호출하고 결과를 받는다.

표 2. Service Interface 관련 컴포넌트

컴포넌트 명	클래스 명	기 능
Service Interface	Service WarehousingDeliveryM	자재 입출고 정보관련 정보 검색 및 업데이트
	ServiceInfo	거래처 정보, 제품정보, 입고정보 등의 기본정보 관리
	ServicePlan	일간, 월간 계획정보 관리
	ServiceResourceM	자재 재고현황 관리
	ServiceCommon	각종 코드정보 관리

(3) Business Entites컴포넌트

대부분의 애플리케이션 컴포넌트는 그들 상호간에 데이터를 전달하게 된다. 데이터 전달시에 사용해야 할 엔티티를 표현하기 위해 사용자 정의 객체지향 클래스를 만든다. 예를 들면, 발주, 출하, 자재 등과 같은 객체가 해당되며 이렇게 객체로 만들어 놓음으로써 유지보수, 확장성이 용이하게 된다. 그림 4는 Business Entities 구현의 일부분인 출하계획 객체 구현부를 보여주고 있다.

그림 4의 DeliveryPlan은 운송을 위한 각종 계획정보를 포함한 컴포넌트로 각종 프로퍼티들을 패러미터로 갖으며 이 컴포넌트는 3계층 Presentation 부분의 출하계획을 하고 확인 버튼을 누르는 과정에서 각 계층단계를 거쳐 호출되며 이 경우 그림 4의 DeliveryPlan 객체에 새 자재출하계획에 해당하는 정보를 넣게 되고 새로운 자재 출하계획이 포함된 객체변수를 서비스 인터페이스 컴포넌트 Service-Warehousing-DeliveryM 의 인자로 넘겨주게 된다.

(4) Data Access Logic Components

비즈니스 프로세스의 진행과정 동안 데이터 저장소에 접근해야 할 경우가 있다. 이 경우 데이터 접근에 필요한 로직을 데이터 액세스 로직 컴포넌트로 분리하여 데이터 액세스 기능을 집중화 시키면 유지 관리와 구성이 쉬워지기 때문에 본 연구에서도 이 같은 방식을 채택하였다[11]. 본 설계에서 제시한 Data Access Logic Components 는 데이터베이스 저장소에 액세스하여 정보를 검색하거나 수정하는 기능만을 수행한다. 예를 들어 새로운 출하계획과 관련하여 데이터베이스를 접근하는 로직은 아래 그림 5와 같다.

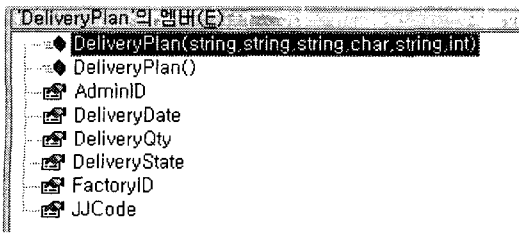

그림 4. 출하계획 객체 구현부

```
NewDeliveryPlan(LIB_DeliveryPlan.DeliveryPlan
dp)
{

  // 데이터베이스 Connection 객체 생성
    myConnection=new SqlConnection(Configura-
    tionSettings.AppSettings["ConnectionString-
    WebConfig"]);

  //전달받은 출하계획 객체(dp)를 데이터베이스에
    저장하기 위한 쿼리문
  SqlCommand myCommand = new SqlCommand
  ("Insert into WM_자재출하계획 (관리자ID, 자재코
  드, 생산공장코드, 출하예정일, 출하예정량) values
  ('"+dp.AdminID+"','"+dp.JJCode
  +"','"+dp.FactoryID +"','"+dp.DeliveryDate +"',
  "+dp.DeliveryQty+")", this.myConnection ;

  //데이터베이스와 연결,수행 및 종료
  this.myConnection.Open();
  myCommand.ExecuteNonQuery();
  this.myConnection.Close();
}
```

그림 5. Data Access Logic 사례

위의 NewDeliveryPlan은 다기종 클라이언트로부터 그림 4의 DeliveryPlan 객체를 전달받게 되면 DeliveryPlan 객체의 인스턴스를 dp라 명하고 dp의 관리자ID, 자재코드 등을 사용하여 쿼리문을 수행하게 된다. 이렇게 미리 사용자가 정의한 Business Entity의 객체로 데이터를 주고받아 사용 및 관리함으로써 유지보수, 확장성을 보장할 수 있다.

3.2.3 계층 컴포넌트간의 업무 수행 흐름

지금까지 살펴본 각 계층별 구조를 종합하여 상호간의 인터페이스와 제어 흐름을 순서 다이어그램으로 나타내보면 그림 6과 같다. 예를 들어 PDA, RFID 장치,PC 등의 클라이언트로부터 창고와 관련된 정보를 액세스하려 할 때 그림 6 좌측의 UI Components는 각 클라이언트별로 구현된 입출력 폼을 의미하며 고객이 요구사항에 대해 이벤트를 발생시키면 UI Process 컴포넌트에 있는 UIPressWD 클래스로 패싱되며 UIPressWD 클래스는 서비스 인터페이스 컴포넌트 클래스인 ServiceWarehousing-DeliveryM을 호출하여 처리토록 하며 ServiceWarehousing-DeliveryM은 비즈니스 엔티티의 하나인 Resource-Warehousing 엔티티를 통해 데이터베이스를 액세스하고 Data Access Logic을 통해 얻은 결과를 역순으로 되돌려 주는 역할을 수행한다.

최종적으로, 작업을 요청했던 UI Components에게 결과를 돌려줄 때에는 XML형태의 타입으로 변환하게 되며 각각의 클라이언트들은 수신한 결과를 자신의 시스템의 특성에 맞추어 파싱한 후 원래의 응용폼의 양식에 맞도록 전환해서 고객에게 전달해 준다. 이 과정은 웹 서비스와의 연동을 통해 수행하게 되는데 그 이유는 관련연구에서 살펴본 바와 같이 다양한 클라이언트들 간의 호환성 측면에서 매우 효과적임이 이미 검증되었기 때문이다.

4. 구현 결과

계층 구조적 설계에 따라 구현된 결과물이 그림 7-1부터 그림 7-6까지 보여 진다. 그림 7-1은 개발된 계층별 컴포넌트를 사용자가 직접 선택을 하여 수행할 수 있도록 하기위한 화면으로 ServiceWarehousing-DeliveryM 컴포넌트의 각종 기능을 웹 서비스와의 연동을 통해 직접 활용할 수 있도록 하였다. 이렇게 사용자가 활용하는 업무 기능이 개발자가 설계한 컴포넌트 단위와 일치하는 구조는 시스템의 유지보

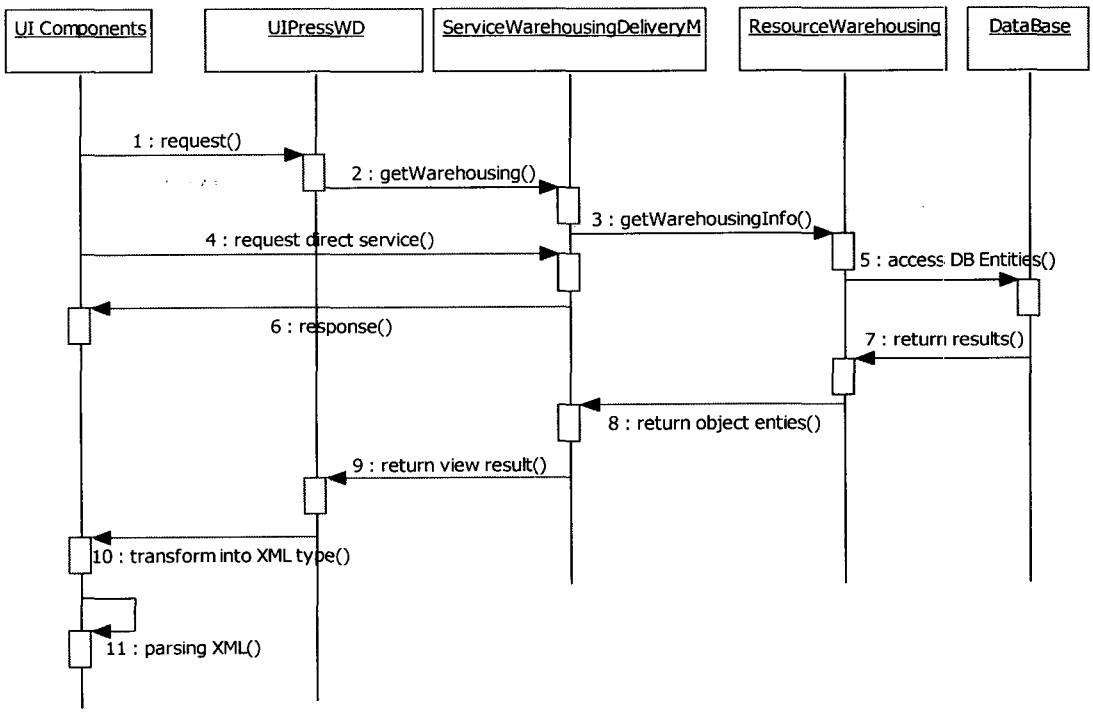

그림 6. 계층별 클래스 간의 제어흐름 다이어그램

수성 및 이동성(portability)이 뛰어나 클라이언트 시스템이 변경될 경우에도 다른 부분에 미치는 영향을 최소화하고 쉽게 변경을 할 수가 있다는 장점을 지니고 있다. 그림 7-2는 각종 클라이언트를 이용한 애플리케이션을 수행하기 전 이와 관련된 드라이버와 응용소프트웨어 등을 다운로드 받아 수행할 수 있도록 준비하는 과정이며 클라이언트 장비가 다양한 시스템 종류로 구성되었거나 수시로 변경된다 할지라도 해당 장비 특성에 부합되는 드라이버를 현장에서 간편하게 변경할 수 있도록 하기 위해 사용된다. 그림 7-3과 그림 7-4는 각각 RFID 클라이언트와 PDA 클라이언트를 대상으로 구현한 결과이며 그림 7-5와 그림 7-6은 데스크탑 환경에서 윈도우 폼과 웹 폼으로 구분하여 각각 구현한 결과이다.

이처럼 수시로 시스템 환경이 바뀔 수 있는 다양한 클라이언트 환경에서도 현장 사용자가 별 불편을 겪지 않고 자기에게 맞는 환경을 조정 및 선택할 수 있도록 한 것이 주요한 특징이라 할 수 있다.

5. 분석 및 평가

소프트웨어 시스템 평가시 주로 사용하는 기준[12,13]들을 도입하여, 지금까지 설계 및 구현한 시스템의 특징을 타 유사 시스템과 비교 평가해보면 표 3과 같다.

표 3의 내용 중 먼저 적용클라이언트의 다양성과

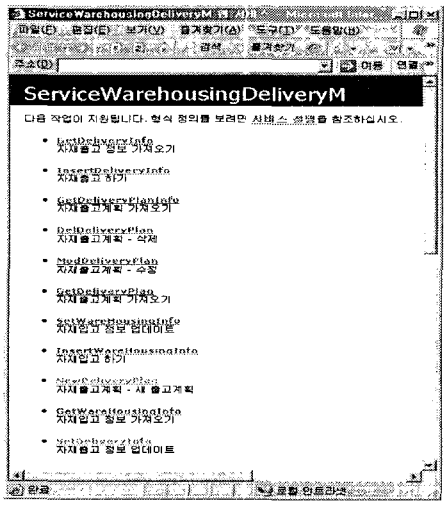

그림 7-1. 컴포넌트 선택

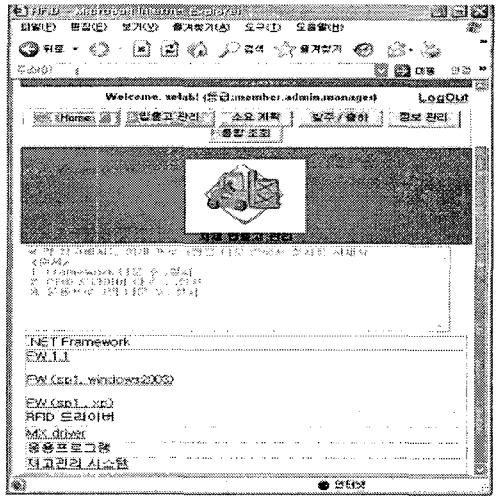

그림 7-2. 클라이언트 특성관련 소프트웨어 다운로드

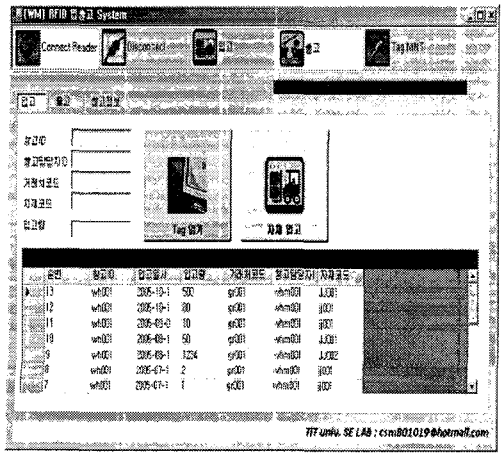

그림 7-3. RFID 클라이언트 구현

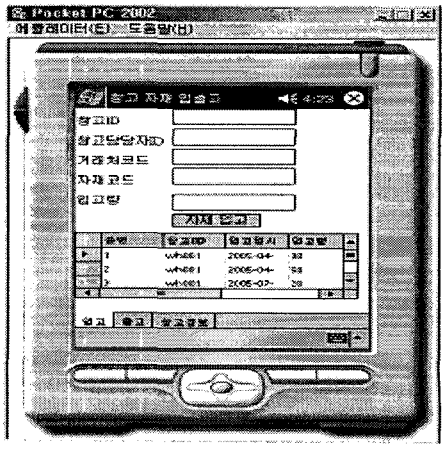

그림 7-4. PDA 클라이언트 구현

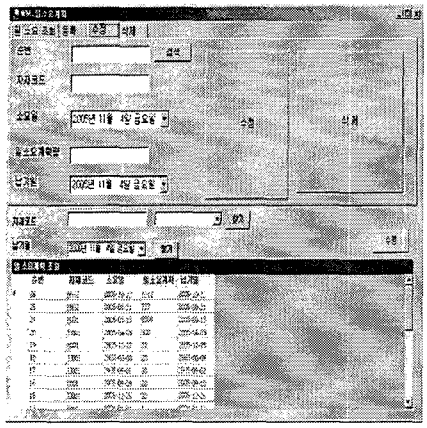

그림 7-5. 윈도우 폼 클라이언트 구현

그림 7-6. 웹을 통한 통합관리 구현

확장성 그리고 편리성 측면을 살펴보면 대부분의 유사 시스템은 너무 비대하고 다양한 클라이언트 환경에 신축적으로 적용하기에는 어려운 구조를 지닌데 반해 본 시스템은 작고 가볍기 때문에 소규모의 다양한 업무환경에 쉽게 적용할 수 있을 뿐 아니라 필요시 쉽게 확장이 가능하다.

또한 계층적 컴포넌트 구조로 설계된 시스템을 XML 웹 서비스와의 연동을 통해 어떠한 형태의 클라이언트이건 손쉽게 적용이 가능트록 하였다는 점이 주요한 차이점이라 할 수 있다. 특히 본 연구의 대상이 된 물류업무는 타 업무에 비해 상대적으로 다양한 종류의 장비와 소프트웨어 플랫폼 환경으로 구성될 수밖에 없으며 이런 환경하에서 해결하기 위한 좋은 방안의 하나는 그림 7-2에서 제시한 바처럼 해당 클라이언트에서 인터페이스 되는 부분을 직접 다운로드시켜 손쉽게 연동시키는 방법이라 할 수 있다.

기존의 타 상용 시스템들은 대부분 설계부터 적용되는 클라이언트 타겟을 미리 설정하였기 때문에 관련 클라이언트 시스템에서는 정상적으로 작동을 할 수 있도록 한 반면, 그렇지 않은 경우는 새롭게 개발을 해야만 하거나 사용자 편리성 면에서 문제가 되는 사례가 적지 않다.

시스템 변경 용이성은 일반적으로 J2EE 등을 기반으로 한 상용 프레임워크(Framework) 시스템이 EJB(Entity Java Bean)처럼 무거운 설계를 동반[12]하고 있는 데 비해 본 연구에서 설계된 시스템은 시스템 변경시 발생되는 오버헤드가 적은 구조를 갖추

표 3. 제안된 시스템과 타 시스템과의 비교 평가

구 분	제안된 시스템	유사 타 시스템
적용 클라이언트의 다양성 및 확장성	컴포넌트가 XML과의 연동을 통해 결과물을 디스플레이하기 때문에 XML이 수용되는 모든 시스템은 적용가능토록 설계 및 구현	대부분 설계 시부터 대상 클라이언트를 설정한 관계로 제한된 클라이언트에 적용
사용자 편리성	현장 작업요원이 현지 시스템에 맞는 기능을 다운로드 시켜 필요한 업무 수행함으로 편리	중앙 서버의 기능을 수정해야하므로 현장 작업요원은 불편
시스템 유지 보수성	-소스 및 수행모듈의 컴포넌트 화로 시스템 구조 변경 용이 -컴포넌트 단위와 응용분야 기능을 일치시켜 보수유지 용이	-수행 모듈 만 제공되어 시스템 구조 변경 곤란 -무거운 설계구조 및 복잡성 증가로 보수유지 어려움
시스템 안정성	트랜잭션의 안정화를 위한 모듈 추가 개발 필요	상대적으로 안정성 있음

었으며 개발된 컴포넌트의 소스도 확보하고 있기 때문에 새로운 아이디어에 따른 시스템 확장이 유리하다. 또한 대부분의 상용 프레임워크는 초기 설계 시 벤더가 설정한 비즈니스 요구사항이 사용자들의 요구사항과 맞지 않는 경우가 자주 있기 때문에 지속적인 수정과 릴리스가 필요하다.

시스템 안정성 측면에서는 트랜잭션 처리시의 안전성 강화를 위해 EJB 트랜잭션 처리 모듈 등을 도입 운영하는 타 상용 시스템들에 비해 상대적으로 미흡하다 할 수 있으며 추가 개발 및 보완이 필요한 것으로 판단된다.

6. 결 론

본 연구에서는 물류시스템의 일부를 대상으로 하여 계층구조를 갖는 컴포넌트 시스템을 설계 및 구현하였다. 즉, Presentation Tier, Business Tier, Data Tier로 구분되는 각각의 계층구조별 설계 내역을 제시하였으며 계층별 상관관계와 연동을 통해 작업이 수행되는 절차와 그 구현 결과물에 대해서도 언급을 하였다.

본 연구에서 제시한 시스템 설계 구조는 어떤 플랫폼이든지 구애 받지 않고 다기종 클라이언트 환경에서 요구되는 데이터 및 정보를 제공할 수 있다. 즉, 대부분의 비즈니스 로직을 통해 요구되는 작업이 수행된 후 제공되는 결과 서비스를 웹 서비스로 노출시켜 원하는 데이터 및 정보를 XML 형태로 제공해 줌으로써 XML이 지원되는 시스템은 대부분 적용이 가능하다. 또한 대규모 플랫폼에서만 제대로 적용될 수 있는 기존의 프레임워크들에 비해 경량화 되어 있고 소스가 확보되어 있어 확장성 측면에서 유리한 여건을 지니고 있다.

따라서 본 연구에서 제시한 방법은, 미래의 유비쿼터스 시대에 등장할 서로 다른 플랫폼으로 개발된 수많은 애플리케이션 및 다기종 클라이언트와의 커뮤니케이션 문제를 해결하기 위한 효과적 방안중 하나가 될 수 있을 것이며, 다양한 오픈 아키텍처(Open Architecture) 환경 발전에도 도움이 될 것으로 판단된다. 향후에는 개발된 시스템의 보수 유지성을 더욱 강화하기 위해서 각 컴포넌트의 설계에 디자인 패턴 개념을 추가 적용하는 방안과 정량적이고 객관적인 성능분석에 대한 추가연구도 필요할 것으로 보인다.

참 고 문 헌

[1] Alan W. Brown, *Large-Scale Component Based Development*, NJ, Prentice Hall, 2000.

[2] Marinescu, F. "2001: The Year of Web Services," *Java Developer's Journal*, Apr. 2001.

[3] H.K.Kim, Oh.H.Kwon, "SCTE: Software Component Testing Environments," *LNCS3481*, Vol. 2, Part 3, pp. 138-146, May 2005.

[4] Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, MA, Addison Wesley, 2003.

[5] Martin Fowler and David Rice, *Pattern of Enterprise Application Architecture*, MA, Addison- Wesley, 2002.

[6] Park, J.S. "Component-Based e-Business Engineering," *4th International Conference on Electronic Commerce Research*, Dallas, TX, November 2001.

[7] Smith, D.et al, "The Future of Web Services: Dynamic Business Webs," *Gartner Research Note*, Apr. 2001.

[8] Scott Short, *Building XML Web Services for the Microsoft. NET Platform*, Microsoft Press, 2002.

[9] James McGovern, *Java Web Services Architecture*, San Diego, Morgan Kaufmann, 2003.

[10] Gregor Hohpe, Bobby Woolf, *Enterprise Integration Patterns*, MA, Addison-Wesley, 2003.

[11] Larsen,G., "Component-Based Enterprise Frameworks," *Communications of ACM*, Vol. 43, No. 10, Oct. 2000.

[12] 김경회, "소프트웨어 아키텍처의 성숙평가 모델에 관한 연구," 한국컴퓨터 정보학회 논문지, 제6권 제1호, pp. 167-176, 2005.10.

[13] 정혜정, "소프트웨어 신뢰도 품질평가 메트릭에 대한 연구," 한국인터넷정보학회 논문지, 제7권 제2호, pp. 151-159, 2006.4.

권 오 현

1975년 3월 해군사관학교 졸업
1980년 6월 미국NPGS computer system 석사
1989년 2월 중앙대학교 전산학과 박사
1975년~2000 해군전산분야 근무
2001년 3월~현재 동명대학교 컴퓨터공학과 부교수
관심분야 : 시스템 분석.설계, 컴포넌트 소프트웨어