□ 기술해설 □

Java 기반 인터넷 어플리케이션 아키텍쳐 및 설계기법

숭실대학교 김수동*

1. 서 론

최근 들어 Java언어를 이용한 웹 어플리케이션 개발이 보편화되고 있으며, 이 언어를 기반으로 하는 웹 어플리케이션 개발 도구들도 다수 소개되고 있다. 이러한 배경에는 Java언어의 특징과 장치들이 HTML/CGI방식의 절차적 프로그래밍 방식의 한계점과 문제들을 잘 해결해 주기 때문이다. 그림 1에서 처럼, 인터넷 어플리케이션은 웹 브라우저를 표준 사용자 인터페이스로 사용하고 HTTP 프로토콜을 사용하여 웹 서버가 요청된 정보를 전송해 주는 클라이언트/서버 구조를 가지고 있다.

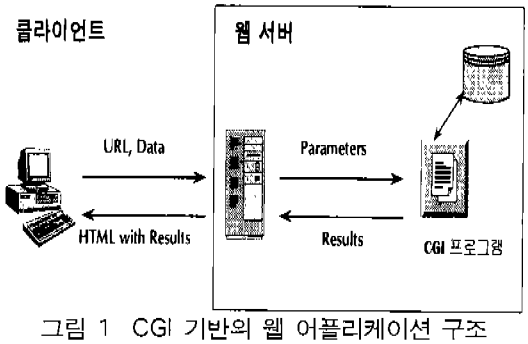

그림 1 CGI 기반의 웹 어플리케이션 구조

이러한 CGI방식의 웹 어플리케이션은 클라이언트가 모든 정보의 검색과 트랜스액션의 처리를 서버에 요청해야 하고, 서버는 요청된 CGI 프로그램을 바인딩하여 실행한다. 따라서, 서버는 네트워크 트래픽의 집중화와 프로세싱의 과중으로 인하여 시스템 성능이 크게 저하되는 문제를 안고 있다[1]. 그 결과 사용자가 요청한 작업에 대한 반응시간(Response Time)이 길어지게 된다.

Java는 이런 문제점들을 해결하기 위한 여러 장치들을 가지고 있어 지난 2~3년간에 그 활용이 두드러지게 높아지고 있다. 그러나, Java는 순수한 객체지향 프로그래밍 방식을 채택하고 있어 새로운 객체지향 클라이언트/서버 모형의 정립과 이를 기반으로 한 웹 어플리케이션의 아키텍쳐를 필요로 하고 있다. 또한, 이러한 아키텍쳐를 기반으로 웹 어플리케이션를 개발하기 위해서는 객체지향 분석 및 설계 기법의 활용이 필수적이다.

본 논문에서는 객체지향 클라이언트/서버 모형을 제안하고, Java기반의 대표적인 웹 어플리케이션 아키텍쳐들을 정의하며, 이를 지원하는 UML 객체지향 분석/설계 기법을 소개한다. 논문의 구성을 살펴보면, 2장에서는 Java언어의 객체지향적 특징과 장치들을 살펴보고, 3장에서는 객체지향 클라이언트/서버 모형과 계층별 구성 요소들을 제안하며, 4장에서는 Java기반의 웹 어플리케이션 개발에 적용될 수 있는 대표적 시스템 아키텍쳐들과 구현 방법들을 살펴보며, 5장에서는 웹 어플리케이션 분석 및 설계에 UML기법을 활용하는 방법들을 제시한다.

2. Java언어의 객체지향적 특징

Java언어의 일반적인 특징은 특정 하드웨어, 운영체제등의 플랫폼 독립성, C++에 비해 간단해 진 장치들, 보안성 등 10여 가지가 있는데, 본 장에서는 Java 언어가 객체지향 프로그

*종신회원

표 1 Java언어의 주요 객체지향 장치들

언어장치 분류(Criteria)	Java가 지원하는 장치
객체지향 기본 장치	객체(Object), 클래스(Class), 상속(Inheritance)
객체지향 고급 개념	추상클래스(Abstract Class), 인터페이스(Interface)
모듈화를 위한 그룹핑	복합객체(Composite Object), 패키지(Package)
인터넷 프로그래밍 지원	애플릿(Applet), URL
데이타 전송/통신	소켓(Socket), 데이타그램(Datagram)
기타 고급 장치	예외처리(Exception), 트래드(Thread), 바이트 코드

래밍을 어떻게 지원하는지 살펴본다. 표 1은 Java언어가 제공하는 장치들을 주요 항목별로 정리한 것인데, 우선 Java는 객체지향 프로그래밍의 세 가지 핵심 개념인 객체(object), 클래스(class) 및 상속(inheritance)을 지원한다. 구문상(Syntax)으로는 C++와 유사하지만 friend 장치, 연산자 Overloading 등 복잡하거나 비 객체지향적 장치들을 제외시켜 순수 객체지향 방식을 채택하였다. 객체지향의 고급 개념인 추상 클래스를 abstract class와 interface의 두 가지 장치로 지원한다. 특히, interface는 순수 가상 함수들과 상수들 만을 가지고 있어 추상 데이타 타입을 정의하고 서브 클래스들이 반드시 구현/제공해야 하는 기능을 명세하는데 효과적으로 사용될 수 있다.

Java는 class보다 큰 그룹화 장치로 패키지(Package)를 사용하는데, 연관된 여러 클래스들과 객체 등을 한 모듈로 그룹화 할 경우에 유용하게 사용되는 장치로 Programming-in-the-large를 효과적으로 지원한다. Java의 package는 CORBA IDL 언어의 모듈(module)에 해당되어 JavaORB를 사용하여 웹 어플리케이션을 개발할 경우 두 언어간의 효과적인 매핑을 할 수 있다[2]. 또한, package는 재사용을 위한 보다 큰 단위로 사용될 수 있어, 최근에 소개된 Framework 개발에 적절한 단위가 된다[3]. Java는 기존의 private, protected, public의 가시성(Visibility) 종류 이외에도 한 package내부에서 정보 공유 및 교류를 효과적으로 처리할 수 있는 'package' 가시성이 제공된다.

Java는 인터넷 기반의 어플리케이션 개발을 지원하는 장치들을 가지고 있는데, 대표적으로 애플릿(Applet)을 들 수 있다. 애플릿은 여러 객체들을 포함할 수 있으며 HTML을 이용하여 서버로부터 클라이언트에게 직접 전송되어 클라이언트 컴퓨터에서 실행될 수 있는 복합 객체(Complex Object)이다. 따라서, 서버에서 CGI 프로그램으로 실행되던 프로세싱의 상당 부분을 클라이언트에게 이동시켜 실행함으로서 프로세싱의 균형화(Processing Load Balancing)를 가능하게 한다. 그 결과, 웹 어플리케이션의 전체적 성능이 크게 향상될 수 있다.

객체지향 프로그래밍에서는 객체와 객체간의 정보 교환이 메시지에 대한 매개 변수와 결과값 전달의 형태로 이루어 진다. 그러나, 서버 객체와 클라이언트 객체가 보다 효율적인 데이타 전송을 위해서 socket, datagram의 장치들을 제공하고 있다. 객체들간 보다 지속적이고 빠른 데이타 전송이 요구될 경우에 유용하게 사용될 수 있어서 웹 어플리케이션 개발에 유용하게 사용될 수 있다.

이외에도, Java언어는 예외처리를 위해 Exception 객체들을 제공하며, 스레드(Thread)도 역시 객체 단위로 모델링하게 하는 등 순수한 객체지향 방식을 채택하고 있어, 객체지향 프로그래밍의 모듈성, 재사용성, 유지보수 우수성 등의 장점들을 최대한 활용하고 있다. 따라서, 향후 인터넷 기반의 어플리케이션 개발에 이 언어의 활용이 더욱 본격화될 것으로 예상된다.

3. 객체지향 클라이언트/서버 모형

웹 브라우저에 사용되는 HTTP 프로토콜의 형태는 요청된 HTML을 서버가 클라이언트에게 전송하는 클라이언트/서버 형태를 가진다. 기존의 절차적 클라이언트/서버 시스템들은 주로 원격 프로시져 호출(Remote Procedure

Call) 방식과 미들웨어 방식을 이용하며, 3개의 계층으로 구성된다[4]. 사용자와의 인터페이스를 제공하는 사용자 인터페이스 계층, 업무논리(Business Logic)과 트랜잭션을 실행하는 프로세싱 계층, 데이타의 관리 및 처리를 제공하는 데이타 계층으로 구성되어 있다[4].

그러나, Java는 순수 객체지향 방식을 채택하고 있어, Java기반의 웹 어플리케이션 개발에는 절차적 클라이언트/서버 모형이 잘 적용되지 않는다. 따라서, 본 장에서는 Java기반의 웹 어플리케이션 개발에 활용될 수 있는 객체지향 클라이언트/서버 모형과 그 구성 요소들을 제안한다. 객체지향 클라이언트/서버 시스템은 절차적 클라이언트/서버의 경우처럼 세 개의 계층들로 구성되고 있으나, 각 계층의 구성요소는 객체의 단위로만 구성되어 있으며 계층별 객체들의 기능과 역할이 다르다.

3.1 사용자 인터페이스 계층(UI Layer)

순수 객체지향 프로그램은 객체들로만 구성되어 있다. 객체지향 클라이언트/서버 구조에서 UI계층의 객체(U객체)들은 관련된 사용자 인터페이스 데이타와 관련된 함수들을 한 덩어리로 가지고 있다[5]. 사용자로부터의 입력을 받아 프로세싱 계층의 해당 객체들에게 전송하며, 프로세싱 계층의 객체들로부터의 트랜잭션 처리 결과를 받아 사용자에게 전달하는 사용자 인터페이스를 제공한다. Java AWT에서 제공되는 클래스들을 이용하여 윈도우, 메뉴, 버턴, 아이콘 등에 기본적인 Widget 객체들을 만들 수 있으며, 이들을 포함하는 보다 큰 복합 U객체들을 생성할 수 있다.

UI 객체는 프로세싱 계층에 있는 해당 객체를 참조(Reference) 형태로 가지게 되며, 프로세싱 계층도 해당 사용자 인터페이스 객체들을 참조하게 된다. 웹 어플리케이션에서는 대부분의 U객체들은 웹 브라우저에서 실행되는 애플릿으로 정의되고, 프로세싱 계층의 객체들은 클라이언트나 서버에서 실행될 수 있는데, 이들간의 참조 관계는 객체지향 분석 과정에서 OMT의 이벤트 추적도(Event Trace Diagram)나 UML의 순차도(Sequence Diagram)를 이용하여 발견할 수 있다.

3.2 프로세싱 계층(Processing Layer)

프로세싱 계층의 프로세싱 객체(P객체)는 UI 객체로부터 요청받은 트랜잭션을 처리하며, 필요한 입력값은 U객체들로부터 받고 데이타 계층의 데이타 객체들로부터 관련된 데이타를 전송 받아 트랜잭션을 실행시킨다. 그 결과값은 U객체들에게 전송되며, 필요시 데이타 계층에 변경된 객체의 상태(State) 정보를 저장한다.

트랜잭션 처리에 필요한 데이타는 데이타 계층에서 전송되어 오므로, P객체 안에 있는 데이타 부분은 현재 트랜잭션 처리에 필요한 버퍼(Buffer), 임시 변수의 값 및 중간결과값 등을 가지고 있다.

3.3 데이타 계층(Data Layer)

일반적으로 데이타 계층은 영구적인 데이타를 저장 및 관리하는데, 객체지향 방식의 데이타 계층은 데이타 객체(D객체)들로 구성되어 있다. 이들 객체들의 공통된 특징은 객체 내부의 데이타가 영구적인 정보라는 점과 여러 어플리케이션에서 공통으로 필요한 데이타 처리 기능을 함께 가지고 있는 점이다. 즉, U객체의 데이타를 처리하기 위해 잘 정의된 공통 기능들로서, 프로세싱 계층의 객체들이 각 응용 어플리케이션에 종속적인 특정 트랜잭션 처리를 위한 함수들을 가지고 있는 것과는 대조적이

표 2 객체지향 클라이언트/서버 모형의 3계층

	데이타 부분	기능(함수) 부분
사용자 인터페이스 객체(U 객체)	UI Widget데이타 및 입출력 데이타	입출력과 UI를 처리하는 함수들
프로세싱 객체 (P 객체)	버퍼, 임시변수, 중간결과값	특정 어플리케이션에 종속적인 트랜잭션 처리 기능
데이타 객체 (D 객체)	도메인의영구적 데이타	여러 어플리케이션들에 공통적인 트랜잭션/데이타 처리 기능

다.

지금까지 설명된 객체지향 클라이언트/서버의 계층별 구성 요소들을 살펴보면 표 2와 같다.

4. Java 기반의 웹 어플리케이션 아키텍쳐

객체지향 클라이언트/서버 모형은 위에서 정의된 3개의 계층에 해당되는 객체들을 클라이언트와 서버 중 어느 곳에 위치시키는가에 따라 여러 가지 모형들로 분류된다. 본 장에서는 Java로 웹 어플리케이션을 개발할 경우에 적용될 수 있는 시스템 아키텍쳐를 제안하며, 각 모형별 장/단점과 구축전략을 살펴본다. 다음에 제안되는 클라이언트/서버 아키텍쳐들은 Java의 특성, 웹 어플리케이션의 특성 및 효율성, 현재의 개발 및 운영 도구 등을 고려하여 시스템 구축에 실제로 활용될 수 있는데 주안점을 두었다.

4.1 Data Server 클라이언트/서버 아키텍쳐

이 아키텍쳐는 Java 애플릿을 이용하여 서버의 부하를 줄이고 클라이언트의 CPU등 컴퓨팅 자원을 최대한 활용할 수 있도록 한다. 그림 2에서 U는 사용자 인터페이스 객체을 의미하며, P객체는 프로세싱 계층의 객체, D는 데이타 객체를 의미하는데, Java 에플릿을 서버로부터 클라이언트 컴퓨터로 이동하여 트랜잭션 등 정보처리를 클라이언트 호스트에서 직접 실행할 수 있기 때문에 서버의 부하를 줄이며, 보다 빠르고 능동적인 클라이언트를 구현할 수 있다. P객체는 U객체들과 함께 클라이언트에서 실행되므로 사용자로부터의 입력을 U객체들로 부터 전달 받아, 직접 트랜잭션을 처리할 수 있다.

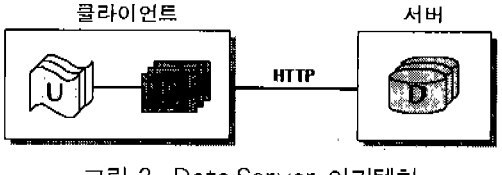

그림 2 Data Server 아키텍쳐

Data Server 모형은 모든 P객체가 클라이언트에서 실행되므로 여러 어플리케이션간에 그룹으로 처리해야 하는 트랜잭션을 처리하는 서버용 P객체는 구현하기 어렵다. 따라서, 이 모형은 클라이언트 전용 트랜잭션들이 대부분인 경우에 사용되어야 한다.

데이타 계층의 D 객체들은 서버에 위치하여 여러 어플리케이션에서 공통적으로 요구되는 데이타와 간단한 데이타 처리 기능들을 가지게 된다.

이 아키텍쳐를 구현할 때 사용되는 Java의 주요 장치로는 애플릿과 Socket, Datagram 등의 장치들이다. 한 HTML 페이지 안에서는 객체간, 애플릿간에 메시지 교환이 가능하므로 U객체와 P객체간의 정보교류가 가능해 진다. 그러나, 클라이언트에 위치한 P객체와 서버에 위치한 D객체간의 메시지 교환을 위해서는 Java Remote Method Invocation(RMI)을 사용하여야 한다. RMI는 CORBA의 Static Interface Invocation(SII)와 유사한 방식으로 서버에 있는 객체들에게 메시지을 보내는 장치이다.

4.2 Process Balanced 클라이언트/서버 아키텍쳐

Process Balanced 아키텍쳐는 Data Server 아키텍쳐와 유사하나, 그림 3과 같이 P객체들이 클라이언트와 서버에 분산되어 있는 구조이다. 즉, 클라이언트에서 독립적으로 실행될 수 있는 P′ 객체들을 클라이언트에 할당하게 되고, 여러 클라이언트들 간에 공통적으로 실행되는 트랜잭션이나 서버 전용 트랜잭션을 제공하는 P″객체들은 서버에 할당하는 방식이다.

P″객체의 예를 들면, '그룹 작업' 등 여러 클라이언트들이 공통으로 실행하는 트랜잭션을 제공하는 객체나, 인터넷 상거래에서 전자결재(Electronic Payment) 등 트랜잭션의 성격상

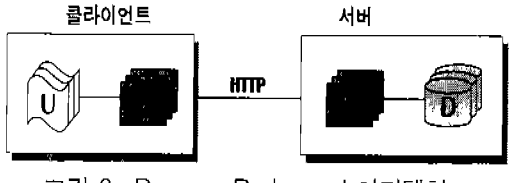

그림 3 Process Ba lanced 아키텍쳐

서버에서만 실행되어야 되는 기능을 가진 객체이다. 따라서, 이 구조에서 서버는 여러 클라이언트들이 공통으로 실행하는 작업들, 예를 들면 그룹 작업등을 지원하는 P″객체들과 D객체들을 가지게 된다. 이 아키텍쳐에서 U객체들은 주로 P′객체들과 메시지 교환을 하지만, P′객체나 P″객체는 모두 D객체들과 메시지 교환을 할 수 있다.

Process Balanced 아키텍쳐의 장점은 서버에서 실행되는 P″객체들을 모델링할 수 있어서, 도메인에서 요구되는 다양한 트랜잭션 모델들을 지원할 수 있는 점이다. 또한, P 객체들이 클라이언트와 서버에 분산되어 있어 보다 효율적인 클라이언트/서버 시스템을 구축할 수 있다. 또한, 클라이언트와 서버에 분산되어 있는 P객체들간의 IPC(Inter-Process Communication)이 가능해 진다.

이 아키텍쳐를 구현하는 방법은 클라이언트에 Java애플릿으로 P′객체를 만들고, 서버에는 Java 어플리케이션으로 P″객체들을 만들어 객체간의 메시지 교환을 제공하면 된다. 또한, Java 서브릿(Servlet)을 이용하여 서버에 위치한 P″객체들을 만들 수 있다.

4.3 Data/Process Balanced 클라이언트/서버 아키텍쳐

Data/Process Balanced 아키텍쳐는 그림 4와 같이 P객체들 뿐아니라 D객체들도 클라이언트와 서버에 분산되어 있는 구조로서, 클라이언트에도 D객체들을 할당하여 지역 데이타베이스를 구축할 수 있다.

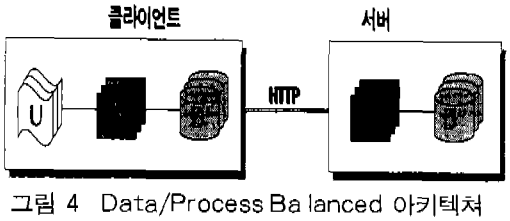

그림 4 Data/Process Ba lanced 아키텍쳐

이 아키텍쳐에서 D′객체는 클라이언트 전용 데이타를 가진 객체로서 P′객체가 트랜잭션을 실행할 때, 지역 데이타베이스에 있는 D′객체들로부터 직접 필요한 데이타를 전송 받을 수 있다. 서버에 위치한 D″객체는 여러 클라이언트들이 필요로 하는 공통된 데이타를 유지하며 이 데이타를 처리하는 함수들을 가지고 있다.

예를 들면 인터넷 상거래에서 옷을 구매하려 할 때 D′객체는 고객의 사진정보를 그래픽으로 가지고 있고 P′객체가 서버에 있는 D″객체로부터 구매하려는 옷에 대한 그래픽 정보를 전송받아 클라이언트 컴퓨터에서 고객의 사진정보와 옷 모양 그래픽을 합성하여 고객이 직접 그 옷을 입은 모습을 보내주는 기능이다. 이 경우, 고객 정보가 지역 데이타베이스의 D′객체에 있으므로 매우 효율적인 시스템 구성을 할 수 있다.

이 아키텍쳐는 지역 데이타베이스를 구현하여 데이타를 클라이언트와 서버에 분할하므로 네트워크 트래픽을 크게 줄여 시스템 효율성을 올릴 수 있는 장점을 가지고 있다. 또한, 프로세스와 데이타가 모두 분할될 수 있어서 진정한 객체지향 클라이언트/서버 구조로 볼 수 있다.

그러나, 이 아키텍쳐를 구현하는 데는 Java의 보안에 관한 제약사항으로 인하여 다소 어려움이 있다. 우선, Java의 보안관리자(Security Manager)는 P′객체가 클라이언트 컴퓨터의 파일이나 지역 데이타베이스의 접근을 허용하지 않는다. 따라서, 프락시 객체(Proxy Object)를 이용하여 D′객체를 우회하여 접근하거나 ORB(Object Request Broker)와 같은 미들웨어를 사용해야 한다. 또한 이 아키텍쳐로 개발할 때, 프로세스 객체와 데이타 객체가 분산되어야 하므로 객체 그룹화(Object Clustering) 작업이 논리적이며 최적의 상태로 이루어져야 전체 시스템의 성능을 크게 향상시킬 수 있다.

5. Java 어플리케이션 개발을 위한 OOA/D

미국Rational사에서 개발하여 Microsoft, HP, Oracle 등 여러 전산 회사들과 컨소시엄으로 OMG에 제출된 UML 1.1버전은 97년11월 OMG 표준 OOA/D기법으로 채택되었다[6]. 그러나, UML은 다양하고 풍부한 도식(Diagram)과 표기법(Notation)들을 제공하

나, 분석과 설계를 수행할 때의 지침은 매우 부족하다. 본 장에서는 UML 기법을 Java 기반의 웹 어플리케이션을 개발할 때 어떻게 활용할 수 있는가 살펴본다.

5.1 Use-Case 모델링

UML 개발 프로세스의 첫 작업인 Use-Case 모델링은 도메인에서 주요 액터(Actor)들과 Use-Case들을 발견하여, 어떤 액터가 어떤 Use-Case를 사용하는가 관찰하는 작업이다. Java기반 웹 어플리케이션 개발시, Use-Case 도식에서 액터들과 Use-Case들에 대한 역할(Role)을 클라이언트 및 서버로 구분하여 표기한 후, 객체 모델링 시 U, P′, P″, D′ 및 D″로 상세히 분류할 때 활용하도록 한다.

액터들 중에는 상당부분 U객체로 모델링되거나 U객체가 정보교류를 해야 되는 대상이 된다. 각 Use-Case를 사용하는 액터와 Use-Case 명세(Use-Case Description)에 나타난 역할을 살펴보면 상당수의 P′ 혹은 P″ 객체들을 찾을 수 있게 된다. 따라서, Java 어플리케이션 개발시에는 Use-Case 모델링 단계에서부터 클라이언트와 서버 역할을 분류하면 이후의 개발 작업에 유용한 정보를 제공할 수 있다.

5.2 객체 모델링

UML에서의 객체 모델링은 클래스 다이어그램을 작성하는 일인데, UML에서 제공하는 스테레오 타입(Stereo Type)을 이용하여 각 클래스에 대한 클라이언트/서버 모형에서의 해당 계층을 표시해 둔다. 예를 들면 'Payment'가 서버에서 전자 결재를 담당하는 클래스이면 클래스 이름 아래에 《P″》이라고 표기하면 이후 객체 클러스트링 단계에서 어느 컴포넌트에 위치를 결정할 때 유용하게 사용된다.

또한, Java는 Abstract Class와 Interface를 효과적으로 지원하므로, 클래스 다이어그램에서 해당되는 클래스에 {abstract}란 제약사항과 《interface》란 스테레오 타입을 표기해 두면 상세설계나 구현시 Java의 Abstract Class나 Interface로 직접 매핑이 된다.

5.3 메시지 흐름 모델링

UML에서 메시지 흐름 모델링은 순차도(Sequence Diagram)나 협력도(Collaboration Diagram)을 이용하여 작성한다. 각 Use-Case별로 작성된 순차도를 보면 해당되는 객체들이 상단에 나열되는데, 각 객체는 클래스 다이어그램에서 스테레오 타입으로 그 종류가 이미 표기되어 있으므로 클라이언트와 서버 노드간에 네트워크로 전송되는 메시지들을 쉽게 확인할 수 있다. 네트워크상의 메시지는 한 컴퓨터내의 메시지 교환보다 더 많은 시간과 자원을 필요로 하므로 메시지 교환의 부하를 최소화하는 최적화 작업을 할 수 있다.

5.4 논리적 클러스트링

논리적 클러스트링은 목표 어플리케이션의 논리적인 시스템 구조를 의미하는데, UML은 컴포넌트 다이어그램(Component Diagram)을 이용한다. 클래스 다이어그램에 나타난 클래스들 간의 결합도(Coupling)를 기준으로 관련된 클래스들을 컴포넌트 단위로 그룹화한다. 이때 결합도는 클래스의 여러 속성들을 이용하여 수학적으로 측정할 수 있다[7].

결합도 이외에도 메시지 흐름도의 메시지 교환 형태와 빈도를 감안하여 클러스트링하여야 한다. Java기반 웹 어플리케이션 개발에서의 논리적 클러스트링은 물리적 클러스트링를 위한 기본 자료가 되어 전체 시스템 성능에 영향을 주므로 최적의 클러스트링을 하여야 한다.

5.5 물리적 클러스트링

물리적 클러스트링은 논리적 클러스트링의 결과를 이용하여 물리적인 컴퓨터나 노드의 단위로 그룹화하는 작업이다. UML에서는 Deployment Diagram을 사용하는데, 논리적 클러스트링의 결합도 측정기법을 컴포넌트 단위로 적용하면 된다[7]. Deployment Diagram에 정의된 노드들은 물리적으로 클라이언트이거나 서버로 구현하면 된다.

5.6 기타 UML 모델링 작업

위에서 설명된 5가지의 작업 이외에도 객체 상태전이 분석도(State Diagram), 활동도(Activity Diagram) 등이 있는데, Java 기반

웹 어플리케이션의 특성을 감안하여 설계시 여러 가지를 고려해야 한다.

6. 결 론

본 논문에서는 인터넷 기반의 객체지향 클라이언트/서버 모형 및 구성 요소들을 설명하고, Java기반의 대표적인 웹 어플리케이션 아키텍쳐들과 구현 방법을 살펴보았다. 제시된 3가지의 클라이언트/서버 아키텍쳐들은 웹 어플리케이션의 특성에 따라 다양하게 적용할 수 있으며, 순수한 객체지향 패러다임(Paradigm)을 적용함으로써 재사용성을 향상시킬 수 있을 뿐만 아니라 시스템 유지보수도 용이하게 한다. 또한, OMG표준인 UML 분석/설계 기법을 Java 기반 웹 어플리케이션 개발에 어떻게 활용할 수 있는지의 지침들을 제안하였다. 이런 일련의 아키텍쳐와 개발 기법들이 Java기반의 웹 어플리케이션 개발에 효과적으로 사용될 때 목표 소프트웨어의 품질과 개발생산성 향상을 기대할 수 있을 것으로 여겨진다.

참고문헌

[1] Orifali, R. and Harkey, D., *Client/Server Programming with JAVA and CORBA,* Jon Wiley & Sons, 1997.

[2] Object Management Group, *The Common Object Request Broker : Architecture and Specification,* Revision 2.0, July 1995.

[3] Fayad, M. and Schmidt, D., "Introduction", Special Issue on Frameworks, *Communications of the ACM,* Oct. 1997.

[4] Jenkins, N., et el., *Client/Server Unleashed,* SAMS Publishing, 1996.

[5] Cho, E., Kim, S., Rhew, S., Lee, S. and Kim, C., "Object-Oriented Web Architecture and Development Strategies for Web Application", *Asia Pacific Software Engineering Conference '97,* Hong Kong, Dec. 1997.

[6] Object Management Group, *Unified Modeling Language 1.1,* September 1997.

[7] Kim, Soo Dong, 'Object Clustering Techniques for Client/Server and Distributed Software Development", ICSE98 Workshop on Software Engineering over the Internet, Japan, April. 1998.

김 수 동

1984 전산학 학사, Northeast Missouri State University
1988 전산학 석사, The University of Iowa
1991 전산학 박사, The University of Iowa
1991~1993 한국통신 연구개발단 연구실장/선임연구원
1993~1994 The University of Iowa, 교환교수
1995~현재 숭실대학교 컴퓨터학부 조교수

관심분야 : 객체지향 개발방법론, 분산 객체 컴퓨팅, 인터넷 상거래, 프레임웍 재사용