무선 인터넷을 위한 컨텐츠 변환 시스템의 설계 및 구현

양 해 술[†] · 최 민 용[††] · 황 석 형[†††]

요 약

최근 전 세계적으로 IT 분야의 새로운 키워드로 부각되고 있는 것 중에 하나가 바로 무선 인터넷이다. 무선 인터넷이란 무선단말기를 사용하여 무선 환경에서 인터넷에 접속하여 데이터 통신이나 인터넷 서비스를 제공받을 수 있는 아키텍처와 기술 및 서비스를 말한다. 이러한 무선 인터넷 기술은 기존의 유선 인터넷 기술과 더불어 정보의 활용 범위를 확장시키고 있다. 그러나 무선 인터넷 기술을 효과적으로 사용하기 위해서는 컨텐츠의 질적, 양적 공급이 확대되어야 한다. 이를 위해서는 기존의 유선환경에서 서비스되던 양질의 컨텐츠를 재사용하는 방안이 효과적일 수 있다. 이에 본 논문에서는 무선 컨텐츠의 새로운 구축의 개념에서 벗어나 컨텐츠 자동 재작성 기술을 바탕으로 부분조합방식의 컨텐츠 생성기법을 설계·구현하고자 한다.

Design and Implementation of Contents Conversion System for Wireless Internet

Hae-Sool Yang[†] · Min-Yong Choi[††] · Suk-Hyung Hwang[†††]

ABSTRACT

Wireless internet is embossed the new keyword of IT technology in the world lately. The wireless internet is the architecture, technology and service to be can taken data communication and internet service using wireless device through connecting internet in the wireless environment. These wireless internet technology is extends the scope of using information with the existing wire internet technology. But, It is necessary to be extend supply of quantitative and qualitative for using wireless internet technology efficiently. For these matters, it is efficient to reuse good quality contents that is serviced on wire environments. Then, in this paper, design and implement the technique of creating contents that is made by the method of compounding parts based on the contents automatic reuse technology over the concept of creating new wireless contents.

키워드 : 무선 인터넷(Wireless Internet), 컨텐츠(Contents), 무선 인터넷 디바이스(Wireless Internet Device), 인터넷 서비스(Internet Service), XML(eXtensible Markup Language), DTD(Document Type Definition)

1. 서 론

1876년 Alexander Graham Bell에 의하여 전화기가 발명되었다. 이로부터 100년이 훨씬 지난 지금 현재 통신 기술의 발달, 컴퓨터 기술의 발달, 그 밖의 여러 환경들의 발전을 통하여 무선 인터넷 시대에 도달하게 되었다. 무선 인터넷(wireless internet)이란 무선단말기를 통해 무선 상에서 인터넷에 접속하여 데이터 통신이나 인터넷 서비스를 제공받을 수 있는 아키텍처와 기술 및 서비스라고 정의할 수 있다. 이러한 무선 인터넷 기술이 최근 전 세계적으로 기존의 유선 인터넷의 한계를 극복하기 위한 수단으로 제시되고 있다. 좀더 자유롭고, 시간과 장소의 구애를 받지 않고 원하는 정보를 이용하기 위하여 네크워크에 접속할 수 있도록 함으로써 정보에 대한 접근을 용이하게 하고 그의 사용 범위를 확장시키고 있다.

그러나 이러한 무선 인터넷에 대한 사용자 요구의 증가와 사회적 필요에도 불구하고 현재 이의 원활한 사용에 있어서 아직까지는 해결해야할 문제점들이 많이 남아있다. 이는 현재 무선 인터넷을 사용하기 위한 제반 환경이 미약하기 때문인데, 이러한 환경의 구축을 위해서는 몇 가지 개선 사항들이 존재한다.

첫째, 이러한 환경에서 사용되는 디바이스로 하여금 제공되는 서비스의 범위를 수용할 수 있어야 할 것이다. 현재 대부분의 무선 디바이스는 아직까지도 휴대폰에 국한되어 있다. 점차적으로 PDA와 같은 기기들이 보급되고 있기는 하나 아직은 많이 미흡한 실정이다. 그리고, 이와 같은 무선 환경에서 사용할 수 있는 디바이스들은 새로운 제품의 출시

※ 본 연구는 대학 IT 연구센터 육성, 지원사업의 연구결과로 수행되었음.
† 종신회원 : 호서대학교 벤처전문대학원 교수
†† 준 회 원 : (주)HnC테크놀러지 개발부 대리
††† 종신회원 : 선문대학교 컴퓨터정보학부 교수
논문접수 : 2004년 2월 10일, 심사완료 : 2004년 10월 1일

또는 업그레이드가 매우 빠르게 진행되고 있다.

둘째, 유선환경에서 제공되는 컨텐츠나 서비스들을 무선 환경에서 사용하는데 사용자들로 하여금 얼마만큼 그들의 요구에 만족시킬 수 있느냐 하는 문제이다[1]. 이는 물론 첫 번째 문제로 지적된 디바이스의 수용 문제와 결부되어 있지만 인간이 이동하면서 사용할 수 있는 디바이스라는 것은 결국에는 우리가 보편적으로 사무실이나 집에서 사용하고 있는 데스크탑 컴퓨터와 비교하여 보았을 경우 그것에 비하여 제약이 많을 것이다. 그러므로 결국 무선 인터넷을 효과적·효율적으로 사용하기 위해서는 제공되는 컨텐츠의 질적 향상과 더불어 디바이스의 수용 범위를 초과하지 않으면서, 사용자들로 하여금 유선 환경에서의 서비스에 대한 만족감을 무선 환경에서도 가질 수 있도록 적절한 처리가 이루어 져야 한다.

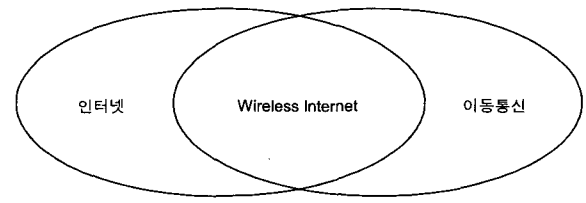
그리고 셋째, 기존의 유선 인터넷을 이용하여 컨텐츠 및 서비스를 제공하고 있는 사업자들이 무선 인터넷 환경으로 사업을 확장·전이할 경우 무선 인터넷 환경 구축에 있어 기존의 유선 인터넷을 활용하지 못하고 새롭게 구축해야 하는 어려움이 있다. 이는 사업을 하는데 있어 비용의 문제도 중요하지만 기존의 유선 인터넷을 이용한 서비스와 무선 인터넷을 이용한 서비스의 동기화 차원에도 많은 어려움을 줄 수 있다[2].

따라서 본 논문에서는 위와 같이 야기되는 문제점들을 개선하기 위하여 디바이스에 종속적이지 않으면서 무선 인터넷에서 사용되는 컨텐츠 또는 서비스를 제공하기 위한 컨텐츠 변환 시스템을 설계 및 구현하였다. 이는 기존의 유선 인터넷 환경에서의 서비스를 쉽고 빠르게 무선 인터넷 환경으로 변환 또는 새롭게 구축할 수 있는 것인데, 이를 위하여 제 2장에서는 관련연구로서 무선인터넷 기술 동향과 DTD 및 서블릿에 대한 최근 동향을 기술하였으며, 제 3장에서는 컨텐츠 변환 시스템에 대한 요구사항 분석과 시스템의 기능, 그리고 내부 시스템을 정의하였으며, 컨텐츠 변환 규칙과 클래스를 설계하였다. 또한, 제 4장에서는 컨텐츠 변환에 대한 규칙을 위한 XML DTD를 정의하였고, 제 5장에서는 Java 기술과 XML 기술을 사용하여 무선 인터넷을 위한 컨텐츠 변환 시스템을 구현하였다. 그리고 제 6장에서는 시험과 평가를 하여 제안 시스템의 타당성을 검증하고 분석 및 고찰을 하였다. 끝으로 제 7장에서는 본 논문의 결론 및 향후과제에 대하여 기술하였다.

2. 관련 연구

2.1 무선 인터넷 기술 동향

무선 인터넷은 무선 고정 인터넷과 무선 이동 인터넷으로 분류할 수 있다. 무선 고정 인터넷은 블루투스(Bluetooth), 무선 LAN(Wireless LAN), B-WLL 등을 이용하는데 이동성이 제한된 반면 전송용량 및 속도에서 우위에 있다. 그리고 무선 멀티미디어 서비스가 제공 가능하다. 반면 휴대폰이나 PDA 등을 통해 인터넷에 접속하는 무선 이동 인터넷은 이동성은 좋으나 전송용량 및 속도에 대한 제약이 있다. 일반적으로 무선 인터넷이라 하면 무선 이동 인터넷을 의미한다. 이러한 무선 인터넷은 또한 (그림 1)과 같이 이동통신과 인터넷의 특성을 동시에 가지는 것으로 이동통신의 이동성, 양방향성, 개인화적 특성과 인터넷의 탈중심적, 개방적, 양방향적인 특성을 모두 가지고 있다[3].



(그림 1) 무선 인터넷의 특성

무선 인터넷은 이동통신망의 발전에 따라 초기 1세대 AMPS망을 이용한 CDPD에서 현재 2세대인 IS-95/GSM을 거쳐 3세대인 IMT-2000으로 발전중이다. 현재 2세대 이동통신망 기반으로 WAP, ME, I-Mode 등의 무선용 프로토콜을 사용하여 무선 인터넷을 서비스 중에 있지만 앞으로 무선 인터넷은 차세대 이동통신망인 All-IP 망을 기반으로 전개될 전망이다.

〈표 1〉 유선 인터넷과 무선 이동 인터넷 비교

구 분	유선 인터넷	무선 이동 인터넷
전송속도	56~수십Mbps	14.4~64Kbps
화 면	640×480pixel 이상	4×16(일반폰), 8×16(스마트폰)
인터페이스	키보드, 마우스, 펜, 모니터 등 다양한 입출력 장치 지원	액정화면, 소프트 버튼
통신 에러율	낮음(100000분의 1)	높음(100분의 1)
휴대성	불 편	편 리
프로토콜	TCP/IP	TCP/IP, WAP
컨텐츠 형태	HTML	c-HTML, s-HTML, WML
접근 형태	양방향	단방향
응용 소프트웨어	다양, 추가 및 변경 쉬움	한정, 추가 및 변경이 어려움
저장성	데이터 저장 쉬움	데이터 저장에 제한

무선 인터넷은 유선 인터넷에 비교하여 통신속도는 낮고 통신 에러율은 높다. 그밖에 사용자 인터페이스의 제약이 높으며 다양한 무선 디바이스로 인하여 다양한 기술 개발이 요구된다. 그러나 이와 같은 사항들 외에 무선 인터넷이 유선 인터넷 보다 우위를 점하는 특성이 있다. 그것은 바로 휴대성인데, 이것은 현재 인터넷 사용자들에게 있어 가장 큰 요

구사항으로 자리잡고 있다. 이와 같은 유선 인터넷과 무선 이동 인터넷의 서로 다른 특성은 아래 <표 1>과 같이 정리해 볼 수 있다[4, 5].

2.2 컨텐츠 저작 시점에 따른 지원기술

다양한 모바일 단말을 지원하기 위한 웹 컨텐츠의 저작 시점을 기준으로 기술을 분류하는 것은 수동 또는 자동의 기준을 적용하는 것과 유사하다. 즉 단말의 특성에 맞게 각각 컨텐츠를 미리 작성하여 제공하는 방법(device-specific authoring)과 단말의 특성에 맞게 컨텐츠를 자동 재작성하는 방법(automatic re-authoring)으로 구분할 수 있다. 단말에 적합한 별도의 웹 컨텐츠를 미리 만들어 두고 서비스하는 방법과 하나의 HTML 컨텐츠로부터 각 단말용 컨텐츠를 실시간으로 재작성하는 방법으로 대별되는 것이다.

첫 번째 방법은 사용자 만족도가 가장 높은 웹 페이지를 보여줄 수 있지만 단말에 따라 별도의 컨텐츠를 저작해야 하는 수고가 필요하다. 대부분의 웹 컨텐츠가 데스크탑 PC를 대상으로 하여 만들어져 있으므로 모바일 단말용 컨텐츠는 따로 저작해야 하는 어려움이 있고 따라서 모바일 웹 억세스는 상당히 제한된 사이트에서만 지원될 것이다.

두 번째 방법에서는 자동으로 HTML 컨텐츠를 변환 및 재작성하기 위한 다양한 연구들이 진행되어 왔고 상업적인 제품으로 나와 있는 것도 있다. 이 방법은 여러 가지 시도에도 불구하고 여전히 자동 재작성된 HTML 또는 비HTML (HDML, WML, cHTML 등) 컨텐츠의 질이 떨어지는 어려움이 있다. 대략적인 과정은 HTML 문서를 파싱하여 구문 또는 의미 정보를 분석 및 추출하고 이로부터 단말의 환경에 맞게 정보를 변환하고 컨텐츠를 재구성하는 것이다.

2.3 컨텐츠 자동 재작성 기술

이 기술의 대표적인 연구는 Digester 시스템[19]에서 이루어졌는데 3,188개의 웹 페이지를 대상으로 한 분석을 통하여 각종 디자인 휴리스틱스를 얻어내고 재작성 기법들을 개발하였다. Digester 시스템은 크게 3가지의 컨텐츠 재작성 기법을 사용한다. 각각은 다음과 같다.

- Outlining : 웹 페이지가 여러 개의 섹션으로 구성되어 있을 때 각 섹션의 제목만을 따로 모아서 리스트를 만들고 이것을 초기 화면으로 한다. 각 섹션별 구체적인 내용은 리스트의 해당 제목에 링크로 연결된 별도의 웹 페이지로 만들어 사용자의 선택에 따라 억세스된다. 이 방법은 사용자 매뉴얼 등 소책자 형태의 웹 페이지에 적합하고 섹션의 깊이에 따라 어떻게 리스트를 구성할 것인지는 옵션으로 처리된다.
- First Sentence Elision : 내용이 긴 웹 페이지의 경우 섹션의 제목 대신 첫 번째 문장이 그 섹션의 내용을 잘 나타내는 수가 있다. 또 섹션의 제목이 없는 경우에도 첫 번째 문장으로 대신하는 것이 유리하다. 이 때 적용하는 방법은 Outlining과 마찬가지로 첫 번째 문장을 뽑아서 초기 화면을 만들되 섹션의 구체적인 내용은 별도의 페이지로 링크되게 한다.
- Image Reduction and Elision : 데스크탑용의 화려한 웹 페이지는 대부분 많은 수의 이미지를 포함하고 있고 이는 모바일 단말의 소규모 화면에서는 제대로 표현될 수가 없다. 이미지 변환시 고려할 사항은 크기를 줄이더라도 이미지 정보를 유지할 것인가 아니면 전체 이미지를 제거할 것인가 하는 것이다. Digester는 크기를 원하는 만큼(25%, 50%, 75%) 줄여주거나 첫 번째 이미지를 제외한 나머지를 모두 제거하는 방법 등을 제공한다. 이 때 줄어들거나 제거된 이미지의 원본은 링크로 연결된 별도의 페이지로 제공된다.

2.3 DTD(Data Type Definition)

XML은 메타언어로서 다른 마크업 언어를 개발할 수 있는 언어이다. 이는 곧 사용자가 임의대로 태그를 정의하여 사용할 수 있다는 것이다. HTML은 확장성이 없기 때문에 XML과 같이 태그를 정의하여 사용할 수가 없다. 그래서 이러한 것이 HTML의 한계로 여겨지고 있는 것인데, XML에서는 DTD를 이용하여 XML 문서의 구조를 정의하고 이렇게 함으로 서 새로운 마크업 언어를 개발할 수 있는 것이다. 즉, DTD 에서는 XML 문서의 태그나 속성과 같은 구성요소들을 어떻게 사용해야 하는지에 대한 정의가 이루어지며 이러한 구성요소들 간의 관계나 데이터 타입, 출현빈도 등의 규칙들도 정의한다.

DTD는 XML 문서 내에 작성될 수도 있고, 문서 밖의 별도의 파일 형태로 작성될 수도 있다. XML 권고안에서는 DTD를 내부 DTD 서브셋(Subset)과 외부 DTD 서브셋으로 구분한다. 내부 DTD 서브셋은 XML 문서 내에서 작성된 DTD를 말하고, 외부 DTD 서브셋은 별도의 파일로 작성된 DTD를 말하는 것으로 일반적으로 DTD 문서라 하면 외부 DTD 서브셋을 지칭하는 것이다[6].

2.4 서블릿(Servlet)

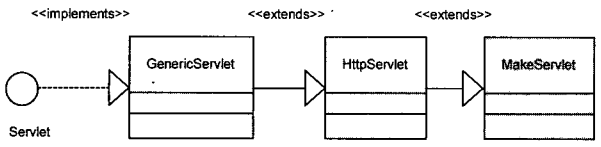
서블릿은 웹 서버에서 수행되는 자바 클래스라 할 수 있다. 서블릿은 서버에서 수행되므로 일반 자바 애플리케이션에서 할 수 있는 모든 일을 수행할 수 있다. 그리고 서버 프로토콜에 구애받지 않고 FTP, DHCP, HTTP등 여러 가지 애플리케이션 레이어의 프로토콜을 사용할 수 있다[7].

자바 서블릿은 클라이언트의 요청에 대해 서블릿 컨테이너에 의해 독립된 스레드기반으로 서비스가 되는 다중 스레드 서비스가 기본적으로 제공된다. 그러므로 프로세스기반의 서비스인 CGI(Common Gateway Interface)에 비해 수행 속

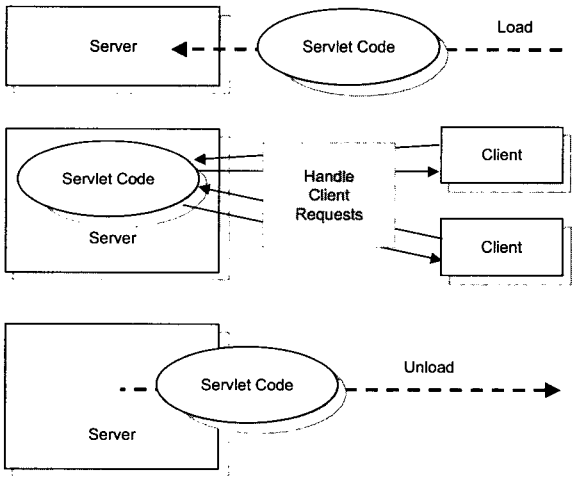
도가 빠르며, 시스템 자원을 효율적으로 활용할 수 있다. 이렇듯 서블릿은 웹 환경에 있어 기존의 CGI Scripts나 다른 웹 기술에 비해 많은 장점을 나타내는데 이러한 장점들은 다음과 같이 정리해 볼 수 있다.

- 클라이언트의 반복되는 요청 시 새로운 프로세스를 생성하지 않고 스레드로 처리한다.
- 스레드 기술을 사용함에 따라 서버의 과부하(Overhead)를 줄일 수 있다.
- 각 요청을 스레드로 처리하기 때문에 멀티테스킹(Multi-Tasking)이 가능하다.
- 서블릿은 모든 요청에 대해 최초 한번의 Memory Loading을 한다.
- 플랫폼(Platform)에 독립적으로 동작한다.
- 보안모델(Security Model)을 사용할 수 있다.

서블릿 프로그램을 작성하기 위해서는 javax.servlet 패키지의 클래스들이 필요하다. 가장 상위에는 Servlet 인터페이스(interface)가 정의되어 있으며, 그 아래 Servlet 인터페이스를 implements한 GenericServlet이 존재하여 프로토콜에 무관한 기본적인 기능을 담당하게 된다. 그리고 HttpServlet 클래스는 HTTP 서비스를 위한 기능들을 제공하기 위하여 GenericServlet 클래스로부터 상속(extends)을 받는다. 결국 서블릿 프로그램이란 이 HttpServlet 클래스로부터 상속받아 클래스를 작성하는 것이다. 이러한 모습은 아래 (그림 2)와 같이 나타내 볼 수 있다[8].



(그림 2) 서블릿 상속 관계



(그림 3) 서블릿 생명 주기

서블릿의 생명 주기(Life Cycle)를 살펴보면 다음과 같다. 처음으로 서블릿이 메모리에 로드되면 init()가 수행되고, 그 다음에 service()의 수행이 이루어진다. 메모리에 로드된 서블릿은 계속 메모리에 유지되며, 클라이언트의 요청 시엔 service()가 병행적으로 수행된다. 서블릿이 더 이상 서비스하지 않으면 destroy()가 수행되며, 서블릿이 종료하게 되는 것이다. 이러한 모습은 (그림 3)과 같이 표현할 수 있다[9].

3. 컨텐츠 변환 시스템의 분석 및 설계

3.1 요구사항 분석

본 논문에서 설계하고자 하는 시스템은 다음과 같은 요구사항 분석을 통하여 구체화되었으며, 이러한 요구사항의 기본 전제는 무선 인터넷의 활용을 위한 새로운 시스템의 구축 또는 기존의 유선으로 사용되어진 인터넷 서비스에 대한 무선으로의 확장이라 할 수 있다. 이와 같은 요구사항은 다음과 같이 정리해 볼 수 있다.

- 기존의 System(운영체제, DB, 응용서비스)과의 손쉬운 Integration이 가능하여 보다 빠르고 손쉽게 무선 환경을 구축하고자 한다.
- 단일 개발을 통하여 무선 환경과 기존의 웹 환경을 동시에 구축할 수 있도록 하고, 이를 통하여 개발에 있어서의 유지보수를 용이하게 하며 전체 시스템의 획일적 관리를 도모한다.
- 개발언어는 독립적으로 하여 인적자원에 대한 활용을 유연하게 하고자 한다.
- 향후 표준의 제정 또는 변경, 새로운 형태의 디바이스에 의한 개발된 시스템의 수정을 최소화하고자 한다.
- 무선 환경의 구축에 있어서 세부적인 기반지식들에 대한 요구를 시스템 자체 내에서 처리하고자 한다.
- 익숙한 사용자 인터페이스를 제공하여 사용의 효율성을 높이고자 한다.

3.2 시스템 기능 정의

실제 위와 같은 요구사항에 대한 분석을 바탕으로 시스템 내부의 기능을 도출해 볼 수 있는데, 기본적으로 웹 환경에서의 서블릿을 기반으로 XML DTD를 정의하여 해당 애플리케이션에 대한 기능을 규정할 수 있다. 이러한 기능은 아래와 같이 크게 7가지로 정의해 볼 수 있다.

- 컨텐츠 실시간 변환 : 해당 컨텐츠의 자료구조에 대한 표준화된 XML DTD를 제공하고, 해당 양식에 따라 개발된 애플리케이션(JSP, ASP, PHP 등)을 일정한 출력형식에 따라 실시간으로 변환한다.
- 다중 컨텐츠 구축 : 기존의 웹 구축과 무선 환경 구축을

단일 애플리케이션으로 구성할 수 있도록 하고, 보다 편리한 사용자 인터페이스 표현을 위하여 애플리케이션의 출력에 HTML과 XML을 모두 포함할 수 있도록 한다.

- 컨텐츠 자동 변환 : 이미 구축된 다량의 웹 페이지가 존재할 경우 모든 페이지를 무선 환경에 맞추어 제작하는 것은 쉽지 않다. 이에 필요한 페이지만 선별하여 처리하고, 일정한 양식의 HTML 페이지의 컨텐츠는 자동으로 변환하도록 한다.
- 멀티미디어 지원 : Image등의 멀티미디어 컨텐츠를 출력 표준에 맞게 실시간으로 처리한다.
- 보안 처리 : 전자금융, 전자상거래 등의 작업을 지원하기 위하여 기본적인 보안에 대한 처리를 제공한다.
- 개발 투명성 제공 : 무선 환경 구축에 있어서 기존의 웹 환경 구축과 같은 개념으로 개발할 수 있게 함으로써 개발자로 하여금 새로운 기술로 인한 어려움을 해결할 수 있다.
- 분산 환경 지원 : 복수 도메인에 위치한 서로 이질적인 환경의 애플리케이션 서버를 지원한다.

3.3 내부 시스템 정의

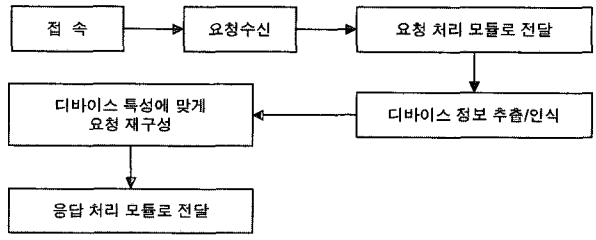
무선 디바이스를 통해 해당 작업을 수행하기 위한 시스템 내부의 구성은 크게 2가지 모듈로 구성될 수 있다. 무선 디바이스는 해당 디바이스에 대한 고유한 데이터를 갖고 있기 때문에 이를 통하여 각각의 환경에 맞게 서비스가 가능하며 이에 대한 처리를 시스템 내부에서 수행하는 것이다. 이러한 2가지 모듈은 아래와 같다.

- 요청에 대한 처리 : 무선 디바이스의 요청을 수신하여 해당 디바이스의 정보를 획득하고, 획득된 정보를 바탕으로 이에 적절한 형태의 데이터를 응답을 처리하는 모듈로 전달한다.
- 응답에 대한 처리 : 애플리케이션의 응답을 수신하여 시스템 내부의 컨텐츠 변환을 통해 무선 디바이스에 맞는 적합한 형태로 데이터를 구성하여 요청에 대한 결과를 사용자에게 전송한다.

3.3.1 요청에 대한 처리 과정

무선 디바이스에서 애플리케이션으로의 요청을 처리하는 과정은 6가지의 단계를 거치면서 수행되며 요청을 시스템 내부적으로 재 전송하게 된다. 이와 같은 단계들은 아래와 같으며 (그림 4)와 같이 나타낼 수 있다.

① 애플리케이션에 할당된 고유한 URL을 통하여 접속을 시도한다.
② 접속이 이루어진 후 무선 디바이스에서의 요청을 수신한다.
③ 수신된 요청을 요청 처리 모듈로 전달한다.
④ 디바이스 인식을 통하여 해당 단말기의 고유 정보를 해석한 후 디바이스 고유 특성에 대한 요청 재구성을 위하여 다음 모듈로 요청을 전달한다.
⑤ 디바이스 인식을 통하여 해석된 무선 디바이스에 대한 고유 정보를 포함하여 요청을 HTTP 프로토콜 규약에 맞춰 재구성한다. 요청되는 URL은 실제 애플리케이션이 위치한 곳으로 한다.
⑥ 재구성된 요청은 요청에 대한 응답을 처리하는 모듈로 전달된다.

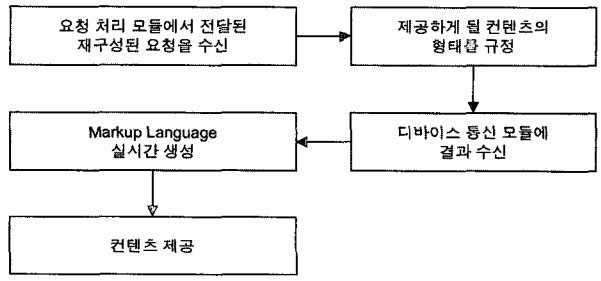


(그림 4) 요청에 대한 처리 과정

3.3.2 응답에 대한 처리 과정

애플리케이션에서 무선 디바이스로 전송되는 응답에 대한 처리 과정은 다음의 5가지의 단계를 거쳐 이루어질 수 있다. 이는 요청에 대한 분석된 정보를 바탕으로 작업이 이루어지며 사용자로 하여금 원하는 결과를 생성하여 서비스하게 된다. 이와 같은 단계들은 아래와 같으며 (그림 5)와 같이 나타낼 수 있다.

① 요청에 대한 처리 과정을 거친 재구성된 요청을 수신한다.
② 재구성된 요청을 통하여 사용자가 원하는 적절한 컨텐츠를 변환/생성하기 위하여 제공하게 될 컨텐츠의 형태를 규정한다.
③ 무선 디바이스에서 수용 가능한 Markup Language를 실시간으로 생성하며, 이를 출력으로 전송한다.
④ 무선 디바이스와의 통신을 위한 모듈에 출력이 수신된다.
⑤ 해당 무선 디바이스에 원하는 형태의 컨텐츠를 제공한다.



(그림 5) 응답에 대한 처리 과정

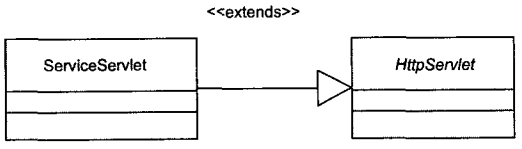
3.4 컨텐츠 변환 규칙

응답에 대한 처리과정 중 사용자에게 제공되는 컨텐츠의 형태에 대한 규정이 적절하게 응답되도록 선택되어지는 컨텐츠 변환 규칙은 다음의 4가지 형태로 정의할 수 있다.

- 지정된 변환 규칙 : 애플리케이션의 출력이 순수하게 내부 시스템에서 정의된 XML DTD로만 구성되어진 경우로 무선 디바이스 전용 페이지를 제작할 때 효율적으로 사용될 수 있다.
- 다중 변환 규칙 : 애플리케이션의 출력이 HTML과 내부 시스템에서 정의된 XML DTD의 혼합으로 구성되어진 경우로 기존의 웹 환경과 무선 디바이스를 모두 지원하고자 할 때 사용될 수 있다.
- 자동 변환 규칙 : 애플리케이션의 출력이 HTML로만 구성되어진 경우로 이미 구축되어진 사이트에 무선 디바이스를 위한 환경을 추가하고자 할 때 사용될 수 있다.
- 멀티미디어 변환 규칙 : 애플리케이션의 출력이 Image 및 Multimedia 형태일 경우 이를 무선 디바이스가 실행할 수 있는 wbmp, xbmp, nbmp, bmp 등의 형태로 변환될 수 있다.

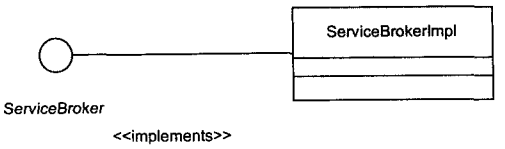
3.5 Class 설계

기본적으로 컨텐츠 변환 시스템은 서블릿 기술을 기반으로 서비스된다. (그림 6)에서와 같이 최초의 요청이 있을 때 그 요청을 수신하기 위하여 HttpServlet 클래스를 상속하는 서블릿인 ServiceServlet을 정의한다[10]. HttpServlet 클래스가 추상 클래스(Abstract Class)이기 때문에 HttpServlet 클래스를 상속하는 ServiceServlet 클래스는 HttpServlet 클래스에서 정의된 메소드를 재정의하여 사용하여야 한다. 그에 따라 ServiceServlet 클래스는 서블릿 클래스로서의 기능을 담당하게 된다.



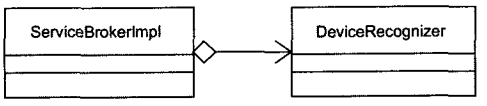
(그림 6) 서블릿 클래스

서블릿이 요청을 받으면 실제 해당하는 요청에 대한 작업은 ServiceBrokerImpl 클래스에서 수행하게 된다. ServiceBroker 인터페이스에는 시스템 내부에서 동작하게 되는 메소드들이 선언되어 있다. (그림 7)과 같이 ServiceBrokerImpl 클래스는 ServiceBroker 인터페이스를 implements하여 ServiceBrokerImpl에 선언된 메소드들을 작성하게 된다. 이에 따라 서블릿에서는 ServiceBroker의 인스턴스를 호출하여 작업을 처리하게 된다.



(그림 7) 작업 수행을 위한 핵심 클래스

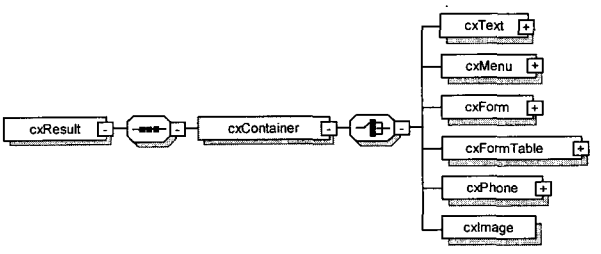
사용자가 사용하는 무선 디바이스에 대한 정보를 담당하는 클래스는 다음과 같이 설계할 수 있다. 무선 디바이스를 인식하는 클래스는 실제 요청과 응답에 대한 처리를 담당하는 클래스에 호출되어 사용되게 된다. 무선 디바이스의 인식을 위해서는 외부 설정파일을 사용한다. 이는 현재 새로운 무선 디바이스의 보급이 매우 빠른 속도로 이루어지는 것을 감안한 것으로 시스템이 구축되고 난 뒤 유지보수 작업 시에 좋은 효과를 볼 수 있다. 이러한 무선 디바이스에 대한 클래스는 다음의 (그림 8)과 같이 나타낼 수 있다.



(그림 8) 무선 디바이스 인식 클래스

4. XML DTD의 정의

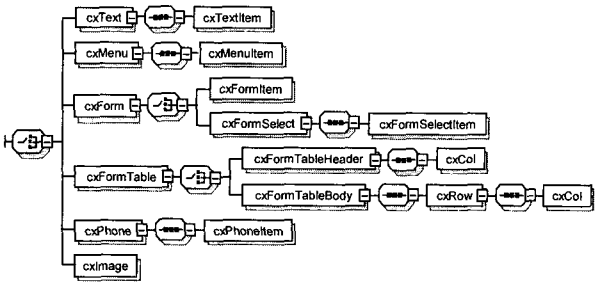
무선 디바이스의 제한적인 환경과 입·출력에 대한 데이터 구조를 규정하기 위하여 XML DTD를 정의하여야 한다. 최상위 노드에는 페이지의 시작을 알리는 메타 엘리먼트가 존재하게 되며 그의 하부 노드에서부터 실제 페이지를 표현하는 엘리먼트들이 존재하게 된다. 일반적으로 웹 환경에서 이루어지는 작업들을 고려하여 DTD를 정의하였으며 필요에 따라서 추가 및 삭제가 가능하다. 또한, 현재 무선 디바이스의 대부분을 차지하는 모바일폰의 사용을 감안하여 전화번호에 대한 특성을 나타낼 수 있는 엘리먼트를 정의하였다.[11] 이와 같이 정의한 DTD의 구조는 다음의 (그림 9)와 (그림 10)과 같은 형태로 표현할 수 있다.



(그림 9) XML DTD 상층부

최상위 노드에는 요청에 대한 애플리케이션의 응답의 시작을 알리는 메타 엘리먼트로 cxResult 엘리먼트가 존재한다.

그 하위로는 cxContainer 엘리먼트가 존재하는데 이는 실제 페이지를 표현하는 엘리먼트의 최상위 엘리먼트로써 향후의 확장을 위한 메타 엘리먼트로서 엘리먼트를 그룹핑하는 역할을 한다. 그리고 cxContainer 엘리먼트의 하위에는 일반적으로 웹에서 사용되는 여러 가지 요소들을 고려하여 그것의 지원을 위한 엘리먼트들이 존재하게 된다[12-14]. 이러한 DTD Set에 대한 자세한 설명은 아래와 같이 나타낼 수 있다.



(그림 10) XML DTD 하층부

- cxResult 엘리먼트 : cxResult 엘리먼트는 요청에 대한 애플리케이션의 응답의 시작을 알리는 메타 엘리먼트로 애플리케이션의 시작은 반드시 cxResult로 시작되어야 한다.
- cxContainer 엘리먼트 : cxContainer 엘리먼트는 실제 페이지를 표현하는 엘리먼트의 최상위 엘리먼트이다. 향후의 확장을 위한 메타 엘리먼트로서 엘리먼트를 그룹핑하는 역할을 한다.
- cxText 엘리먼트 : cxText 엘리먼트는 텍스트 문단을 표현하기 위해 사용된다. 텍스트 문단의 시작과 끝을 나타내는 범위 지정 엘리먼트이고, 실제 문단에 대한 자료는 자식 엘리먼트인 cxTextItem에 의해 표현된다. 그리고 title 속성을 통하여 텍스트 문단의 타이틀을 지정할 수 있다.
- cxTextItem 엘리먼트 : cxTextItem 엘리먼트는 cxText의 자식 엘리먼트로서 실제 한 문단을 표현하며, 복수 사용이 가능하다. 복수의 cxTextItem이 사용되었을 경우 문단의 구분은 디바이스의 특성에 따라 다르게 표현될 수 있다.
- cxMenu 엘리먼트 : cxMenu 엘리먼트는 선택 가능한 목록의 표현에 사용된다. cxMenu는 실제 자식 엘리먼트인 cxMenuItem에 의해 각 항목이 표현되며, 표현시의 각 목록은 일반적으로 무선 디바이스의 숫자 키패드와 1대 1로 매치 되어 표현된다.
- cxMenuItem 엘리먼트 : cxMenuItem은 cxMenu의 목록을 표현하는데 사용된다. 각 목록은 일반적으로 무선 디바이스의 숫자 키패드와 1대 1로 매치되어 표현된다. 그리고 target 속성을 통하여 해당 목록이 선택되었을 때 이동할 URL을 지정해 줄 수 있다.
- cxForm 엘리먼트 : cxForm 엘리먼트는 애플리케이션에 무선 디바이스에서 입력한 자료를 전송하기 위한 요소들을 나타낸다. 이러한 전송은 기본적으로 POST 방식을 지원한다. 자식 엘리먼트로 cxFormItem을 가지고 있으며, title, target 속성을 통하여 입력 폼의 이름을 지정하고 폼의 입력 값이 전달될 애플리케이션의 URL을 지정할 수 있다.
  - cxFormItem 엘리먼트 : cxForm 내의 입력 아이템을 표현한다. cxFormItem은 cxForm 안에 복수의 사용이 가능하다. 속성으로는 text 혹은 password를 구분해 주는 type, 입력 변수명을 지정하는 name, 변수에 대한 기본 설정값을 나타내는 value, 화면에 표시될 표기 문자열을 나타내는 title 등이 있다.
  - cxPhone 엘리먼트 : cxPhome 엘리먼트는 전화번호를 표현하는데 사용된다. 무선 디바이스가 지원하는 경우 cxPhone으로 표현된 전화번호는 해당 번호로 자동으로 연결될 수 있다. 자식 엘리먼트로 cxPhoneItem을 가지고 있다.
  - cxPhoneItem 엘리먼트 : cxPhoneItem 엘리먼트는 한 개의 전화번호에 대한 연결을 표현한다. 연결할 번호에 대해서는 target 속성을 이용하여 나타낸다.
  - cxImage 엘리먼트 : cxImage 엘리먼트는 image data에 대한 표현을 제공한다. target 속성을 이용하여 확장자를 제외한 이미지 파일명을 지정하고, 이미지가 위치한 곳에 같은 이미지를 wbmp, xbmp, nbmp, bmp 포맷으로 변환하여 위치시켜 놓는다. 그리고 alt 속성을 이용하여 이미지에 대한 텍스트 설명을 추가할 수 있다.

5. 컨텐츠 변환 시스템의 구현

5.1 개발 환경

본 시스템은 분산환경에서의 멀티 도메인을 지원한다. 이는 곧 다양한 플랫폼의 지원을 기본 전제로 하는 것인데 이를 위해 개발 언어를 자바로 선택하였다. 자바 언어는 특정 플랫폼에 종속적이지 않고 JVM(Java Virtual Machine)상에서 동작하기 때문에 현재 개발하는 상황에 적합하다 할 수 있다. J2SDK 는 1.3.1_09 버전을 사용하였고, 서블릿을 위한 J2EE 는 1.3.1 버전을 사용하였다. 그리고 웹서버와 서블릿 컨테이너의 기능으로 Apache Tomcat 4.1을 사용하였다. 그리고 개발 당시 운영체제는 Windows 2000 Professional을 사용하였다.

XML DTD를 설계하는데는 Tibco사(http://www.tibco.com)의 Turbo XML Version 2.4.1 도구를 사용하였다. 이와 같은 내용들을 정리하면 다음 <표 2>와 같다.

〈표 2〉 개발 환경

개발 플랫폼	Windows 2000 Professional
개발언어	J2SDK 1.3.1_09, J2EE 1.3.1
웹서버, 서블릿 컨테이너	Apache Tomcat 4.1
XML DTD 설계	Turbo XML Version 2.4.1

5.2 외부 설정 요소

본 시스템을 구현하기 위하여 3가지 외부 설정 요소를 규정하였다. 이는 설정의 변경 시 효과적으로 대처하기 위한 것으로 시스템 유지보수에 신속하게 대처할 수 있는 장점이 있다. 이러한 외부 설정 요소에는 요청과 응답을 요구하는 디바이스에 대한 정보와 애플리케이션을 서비스하는 애플리케이션 서버에 대한 정보, 마지막으로 애플리케이션 서버와 디바이스를 중계시켜주는 게이트웨이 서버에 대한 정보로 구분된다.

(그림 11)은 무선 디바이스를 설정한 일부분으로 SK 텔레콤의 SKY 단말기에 대한 정보를 나타내고 있다. 이렇게 미리 외부 설정 파일을 이용하여 사용될 수 있는 무선 디바이스에 대한 정보를 시스템 내부에 전달하여 새로운 무선 디바이스의 출시나 기존 디바이스의 정보 변경에 대하여 신속히 대처할 수 있다[15].

```
[SKT_SKY]
10.DeviceName = SKT
10.Accept = WML_1.1
10.MIME = text/vnd.wap.wml;charset=euc-kr
10.ImageMIME = image/vnd.wap.wbmp
10.Allow = 255.255.255.255
10.UserAgent = SKT
10.Transformer = wml-1.1.xsl
10.UniqueReference = Cookie
10.XLength = 20
10.YLength = 8
10.BufferSize = 200
```

(그림 11) 무선 디바이스 설정

(그림 12)는 애플리케이션 서버에 대한 정보를 설정한 모습이다. 그림에서와 같이 서버의 프로토콜, IP 주소, Port 번호, 서버 내부에서의 경로, 접속 페이지 등을 나타내고 있다.

```
[selab]
selab.Protocol = http
selab.Hostname = 211.192.249.237
selab.Port = 8000
selab.BasePath = /
selab.DirectoryIndex = index.jsp
selab.TransformerID = m-xsl
```

(그림 12) 애플리케이션 서버 설정

(그림 13)은 애플리케이션 서버와 컨텐츠 변환 시스템과의 연결 설정 모습을 나타낸다. 컨텐츠 변환 시스템에서의 Home 디렉토리를 설정하고, 컨텐츠 변환 시스템의 IP 주소와 위에서 설정한 애플리케이션 서버의 이름을 연결시킴으로 하여 애플리케이션 서버와 컨텐츠 변환 시스템을 서로 연동시킬 수 있다.

```
HIDDENBROKER_HOME= C:/Program Files/Apache Group/Tomcat 4.1

211.192.249.246 = selab
```

(그림 13) 컨텐츠 변환 시스템 설정

5.3 디바이스 인식

디바이스를 인식하기 위한 구현은 외부에서 설정한 설정파일을 읽어들여 해당 정보의 분류 작업을 통하여 HttpSession에 세션(Session) 값을 설정해 주면 된다. 아래의 (그림 14)를 보면 FileInputStream을 이용하여 agent.properties 파일의 정보를 입력하는 것을 볼 수 있다. 이렇게 입력된 FileInputStream 객체는 Properties 객체를 생성하여 load()의 파라메터(Parameter)로 입력하여 외부 설정 파일에서 정의한 모든 디바이스에 대한 정보가 인식된다[16].

```
public void recognize (HttpServletRequest req, boolean debug_flag)
    throw IOException
{
    String ls_unique_id, ls_user_agent, ls_uniquereference ;
    User user = new User( ) ;
    ls_user_agent = req.getHeader ("user-agent") ;

    String SYSTEM_HOME = "C:/Program Files/Apache Group/ Tomcat 4.1" ;
    FileInputStream is = new FileInputStream(
                                SYSTEM_HOME + "conf/agent.properties") ;
    Properties agentProps = new Properties( ) ;
    try {
        agentProps.load(is) ;
    }
    catch(Exception e) {
        L.out("Can't read the properties file." +
            "Make sure properties is in the CLASSPATH") ;
            throw new IOException(e.toString( )) ;
}
```

(그림 14) 디바이스 설정 파일 입력

```
int li_num = 10 ;
String ls_tmp = "" ;
while((ls_tmp = agentProps.getProperty(li_num + ".UserAgent"))!= null) {
    if(ls_user_agent.startsWith(ls_tmp)) {
        break ;
    }
    li_num += 10 ;
}
if(ls_tmp != null) {
    ls_uniquereference = agentProps.getProperty(li_num + ".UniqueReference") ;
    if(! ls_uniquereference.equals("")) {
        ls_unique_id = req.getHeader(ls_uniquereference) ;
        user.setUnique_ID(String.valueOf(ls_unique_id.hashCode( ))) ;
    }
    else {
        user.setUnique_ID(String.valueOf(ls_user_agent.hashCode( ))) ;
    }

    user.setDevice_Type(agentProps.getProperty(li_num + ".DeviceName")) ;
    user.setTransFormer(agentProps.getProperty(li_num + ".Transformer")) ;
    user.setBufferSize(agentProps.getProperty(li_num + ".BufferSize")) ;
    user.setXLength(agentProps.getProperty(li_num + ".XLength")) ;
    user.setYLength(agentProps.getProperty(li_num + ".YLength")) ;
    user.setMIME(agentProps.getProperty(li_num + ".MIME")) ;
}
```

(그림 15) 요청된 디바이스 인식

(그림 15)에서는 사용자로부터 요청이 인식된 뒤 사용자

의 디바이스 정보를 분석하는 것으로 미리 정해놓은 외부 설정 파일에서 각 디바이스의 UserAgent 항목을 사용자의 디바이스 정보와 비교하여 일치 여부를 판별하는 것으로 일치하는 값이 존재하면 그 값을 설정하게 되는 것이다. 이러한 항목들은 위의 (그림 11)을 살펴보면 알 수 있다.

(그림 16)은 위와 같이 인식된 디바이스 정보에 대하여 로그 정보를 출력하기 위한 코드이다. 로그 파일의 작성은 L 클래스의 out()를 이용하며 이를 통해 생성된 로그 정보는 (그림 17)과 같다.

```
L.out(   "\n"
    + " -- DeviceRecognizer.java --      \n"
    + " -- recognize() --                \n"
    + " -- request parameters --         \n"
    + ls_str+                            "\n"
    + " user.getUnique_ID( ) :           \n"
    + user.getUnique_ID( )+              "\n"
    + " user.getDevice_Type( ) :         \n"
    + " "+user.getDevice_Type( ) +       "\n"
    + " user.getTransFormer( ) :         \n"
    + " "+user.getTransFormer( )+        "\n"
    + " user.getBufferSize( ) :          \n"
    + " "+user.getBufferSize( )+         "\n"
    + " user.get.MIME( ) :               \n"
    + " "+user.getMIME( ) +              "\n"
    + " -- end --            ");
```

(그림 16) Log 파일 생성

```
[ Thu Jan 29 05 : 32 : 21 KST 2004 ]
 -- DeviceRecognizer.java --
 -- recognize() --
 -- request parameters --

 HeaderName : accept
 Value : */*
 HeaderName : accept-language
 Value : ko
 HeaderName : accept-encoding
 Value : gzip, deflate
 HeaderName : user-agent
 Value : Mozilla/4.0 (compatible ; MSIE 6.0 ; Window NT 5.1)
 HeaderName : host
 Value : 220.121.109.149
 HeaderName : connection
 Value : Keep-Alive
 user.genUnique_ID( ) :
-1334062217
 user.getDevice_Type( ) :
 Desktop_Browser
 user.getTransFormer( ) :
 html.xsl
 user.getBufferSize( ) :
 1000
 user.getMIME( ) :
 text/html ; charset = euc-kr
 -- end --
```

(그림 17) 로그 정보

5.4 요청과 응답에 대한 처리 구현

요청에 대한 처리 구현은 아래 (그림 18)의 ServiceServlet 클래스의 doPost()에 의해 최초로 접수된다.

```
public void doPost(HttpServiceRequest req, HttpServletRequese res)
    throws ServletException, IOException
{
    try {
        ServiceBrokerFactory sbfactory =
                        ServiceBrockerFactory.getInstance() ;
        ServiceBroker sbroker
            = sbfactory.newServiceBroker(req, res, debug) ;
        sbroker.process( ) ;
    } catch(BrokerException e) {
        L.out(e.toString( )) ;
        throw new ServletException(e) ;
    }
}

public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
{
    doPost(req, res) ;
}
```

(그림 18) 사용자 요청 수신

doPost()는 ServiceBrokerFactory 클래스의 인스턴스를 생성한 후 ServiceBroker 클래스의 process()를 호출한다. 실제로 ServiceBroker 클래스의 생성에는 ServiceBroker 클래스를 상속한 ServiceBrokerImpl 클래스의 객체가 반환되며 이로서 ServiceBrokerImpl 클래스의 process()가 호출되는 것이다. 이와 같은 모습은 다음의 (그림 19)과 같다.

```
public ServiceBroker newServiceBroker (HttpServletRequest req,
        HttpServletResponse res, boolean debug)
    throw IOException
{
    return new ServiceBrokerImpl (req, res, debug) ;
}
```

(그림 19) ServiceBrokerImpl 클래스 객체 반환

ServiceBrokerImpl 클래스 객체가 생성될 때 ServiceBrokerImpl 클래스의 생성자 메소드에 내에서는 ServiceBrokerImpl 클래스 내부의 init()를 호출한다. init()는 외부 설정 파일의 데이터를 입력받아 요청에 대한 내부 시스템 처리를 위한 준비 작업을 수행한다. 다음의 (그림 20)에서 보는 바와 같이 애플리케이션 서버에 대한 외부 설정 파일과 애플리케이션 서버와 컨텐츠 변환 시스템과의 맵핑을 위한 외부 설정 파일을 FileInputStream 객체로 받아오는 모습을 확인할 수 있다[17].

```
public void init( ) throws IOException {
    String SYSTEM_HOME = "C:/Program Files/Apache Group/Tomcat 4.1" ;
```

```
    FileInputStream is_host = new FileInputStream(
                        SYSTEM_HOME+"/conf/host.properties") ;
    FileInputStream is_service = new FileInputStream(
                        SYSTEM_HOME+"/conf/service.properties") ;
    hostProps = new Properties( ) ;
    serviceProps = new Properties( ) ;
    try {
        hostProps.load(is_host) ;
        serviceProps.load(is_service) ;
    } catch (Exception 2) {
       L.out("Can't read the properties file." +
           "Make sure properties is in the CLASSPATH") ;
           throw new IOException (e.toString( )) ;
    }
}
```

(그림 20) init 메소드

ServiceBrokerImpl 클래스의 process()에서는 Service-BrokerImpl 클래스 내부의 request() 와 reponse()를 호출한다. request()는 요청을 수신하여 시스템 내부적으로 처리할 수 있도록 요청에 대한 분석작업이 이루어지며, request()의 작업이 처리된 후 요청에 대한 응답을 출력하기 위하여 reponse()가 호출된다. 이러한 request()와 reponse()의 모습은 다음의 (그림 21)과 (그림 22)와 같다.

```
public void request ( ) throws BrokerException {
    try {
        DeviceRecognizer devicerecognizer =
                        DeviceRecognizer.getInstance ( ) ;
        devicerecognizer.recognize (req, debug) ;
        user = (User)httpsession.getAttribute("user") ;
        requestGenerate( ) ;
        requestForward( ) ;
    } catch (IOException e) ;
        throw new BrokerException (e.toString( )) ;
    }
}
```

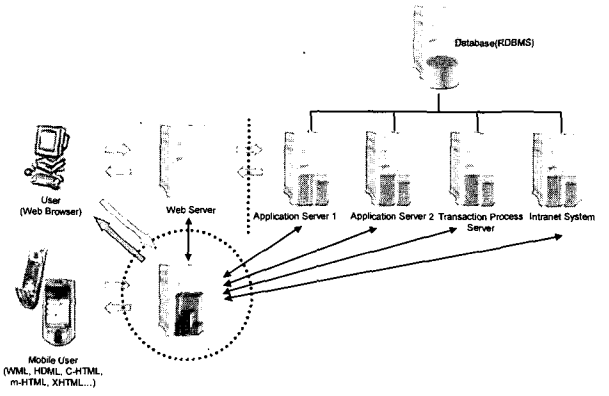
(그림 21) request 메소드

```
public void response ( ) throws BrokerException {
    String Is_PathInfo = req.getPathInfo( ) ;
    StringBuffer lsb_Result = new StringBuffer( ) ;
    InputStream in = null ;
    try {
        if(Is_PathInfo.endsWith("wbmp")) {
            res.setContentType("image/vnd.wap.wbmp") ;
            out = res.getWriter( ) ;
            in = user.getInputStream( ) ;
            int chr = in.read( ) ;
            while(chr != -1)  {
                lsb_Result.append(String.valueOf((char)chr)) ;
                chr = in.read( ) ;
            }
            out.println(lsb_Result.toString()) ;
            if(debug)  {
               L.out (                                    "\n"
                  + "-- ServiceBrokerimpl.java --      \n"
                  + "-- response ( ) --                \n"
                  + "-- Result Document wbmp --    \n"
                  + lsb_Result.toString( )+          "\n"
                  + "-- end --              " ;
                      :
        } else  {
            res.setContentType(user.getMIME( )) ;
            out = res.getWriter( ) ;
            transform(parseDocument( )) ;
        }
    } catch (IOException e) {
        throw new BrokerException(e.toString( )) ;
    }
}
```

(그림 22) reponse 메소드

6. 시험 및 평가

본 논문에서 제안하는 무선 인터넷을 위한 컨텐츠 변환 시스템을 시험하기 위해서 필요한 자원은 다음과 같다. 첫째, 서비스하는 애플리케이션이 위치하게 되는 애플리케이션 서버가 있다. 둘째, 서비스를 요청하는 사용자의 무선 디바이스가 있다. 그리고 마지막으로 애플리케이션 서버에 위치한 애플리케이션을 무선 디바이스에 맞게 변환시키기 위한 컨텐츠 변환 시스템이 위치하는 서버가 존재한다. 이것을 그림으로 표현하면 다음의 (그림 23)과 같다.



(그림 23) 전체 하드웨어 구성

6.1 무선 컨텐츠 구축과 시험

시험 및 평가를 위하여 먼저 애플리케이션 서버에 위치할 애플리케이션을 구성하였다. 이는 테스트를 위하여 본 논문에서 정의한 XML DTD만을 사용하여 새롭게 애플리케이션을 구축한 것이다. 이를 위해 작성한 XML 형식의 문서는 다음의 (그림 24), (그림 25), (그림 26), (그림 27)과 같다.

(그림 24)는 초기화면으로 각각의 세부 메뉴로 이동할 수 있는 메뉴들로 구성되어 있는 것을 볼 수 있다. 다음 화면으로의 이동은 아래의 그림에서와 같이 target Attribute를 입력하여 이루어지게 된다.

```
<cxResult>
<cxContainer>
<cxMenu>
<cxImage target = "/images/selab" alt = "image" />
<cxMenuItem target = "<% = response.encodeURL("/info/index.jsp")%>">
SELab 소식 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/index.jsp")%>">
연구원 주소록 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/class/index.jsp")%>">
강의정보 </cxMenuItem>
</cxMenu>
</cxContainer>
</cxResult>
```

(그림 24) 첫 번째 화면의 소스코드

(그림 25)는 위의 문서에서 '연구소주소록' 부분을 선택하였을 때 나타나는 페이지의 소스코드이다. 이 문서 또한 위와 같이 다음 화면으로 이동할 수 있도록 URL을 명시하고 있다.

```
<cxResult>
<cxContainer>
<cxText>
<cxTextItem> *연구원주소록* </cxTextItem>
</cxText>
<cxMenu>
<cxMenuItem target = "<% = response.encodeURL ("/address/yang.jsp")%>">
양해술 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/lee.jsp")%>">
이하용 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/choi.jsp")%>">
최민용 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/hong.jsp")%>">
홍정훈 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/kim.jsp")%>">
김형기 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("../index.jsp")%>"> 뒤로
</cxMenuItem>
</cxMenu>
</cxContainer>
```

(그림 25) 두 번째 화면의 소스코드

(그림 26)은 위에서 '최민용'이란 항목을 선택하였을 때 나타나는 화면의 소스코드이다. 몇 개의 텍스트 정보와 다음 화면으로 이동할 수 있는 메뉴로 구성되어 있는 것을 확인할 수 있다.

```
<cxResult>
<cxContainer>
<cxText>
<cxTextItem> *최민용* </cxTextItem>
<cxTextItem> 집주소 : 인천광역시 연수구 연수1동 XXX </cxTextItem>
<cxTextItem> 집전화 : 032-817-4079 </cxTextItem>
<cxTextItem> 핸드폰 : 011-799-6353 </cxTextItem>
<cxTextItem> 이메일 : sdev@hotmail.com </cxTextItem>
</cxText>
<cxMenu>
<cxMenuItem target = "<% = response.encodeURL ("/address/call.jsp")%>">
통화연결 </cxMenuItem>
<cxMenuItem target = "<% = response.encodeURL("/address/index.jsp")%">
뒤로 </cxMenuItem>
</cxMenu>
</cxContainer>
</csResult>
```

(그림 26) 세 번째 화면의 소스코드

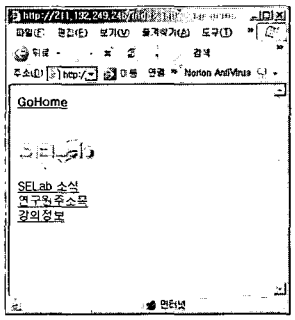
(그림 27)은 위의 (그림 26)에서 '통화연결' 메뉴를 선택했을 때의 소스코드이다. 여기서는 전화번호에 대한 정보를 갖고있는 target Attribute가 존재하는데, 이 부분을 선택하면 선택된 번호와 직접 통화가 가능하게 된다.

```
<cxResult>
<cxContainer>
<cxText>
<cxTextItem> *최민용* </cxTextItem>
</cxText>
<cxPhone>
<cxPhoneItem target = "0328174097"> 집
</cxPhoneItem>
<cxPhoneItem target = "0117996353"> 핸드폰
</cxPhoneItem>
</cxPhone>
<cxMenu>
<cxMenuItem target = "<% = response.encodeURL("/address/choi.jep")%>">
뒤로 </cxMenuItem>
</cxMenu>
</cxContainer>
</cxResult>
```

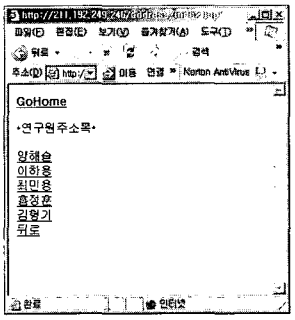
(그림 27) 네 번째 화면의 소스코드

위와 같이 작성된 문서는 무선 디바이스 사용자가 직접 접근하게 되는 컨텐츠 변환 시스템에 의하여 디바이스의 특성을 고려하여 서비스를 제공하게 된다. 다음의 (그림 28), (그림 29), (그림 30), (그림 31)은 위에서 작성된 문서를 실제 사용자가 서비스 받는 모습인데, 먼저 유선인터넷 환경에서 제대로 동작하는지를 판별하기 위하여 인터넷 익스플로러 브라우저를 대상으로 테스트해 보았다. 익스플로러 브라우저는 gif 형태의 이미지를 지원하기 때문에 gif로 작성된 이미지가 출력되는 것을 확인할 수 있다. 첫 번째 화면이 초기화면으로 세 가지 메뉴가 구성되어 있는 것을 볼 수 있다. 첫 번째 화면에서 두 번째 줄의 메뉴를 선택하면 두 번째 화면으로 이동하게 된다. 그리고 두 번째 화면에서 세 번째 줄의 메뉴를 선택하게 되면 세 번째 화면으로 이동하는 모습을 볼 수 있다. 그리고, 세 번째 화면에서 '통화연결'을 선택하면 네 번째 화면으로 이동하고 화면에 나타나 있는 번호와 통화할 수 있다. 그러나 현재 테스트하고 있는 인터넷 브라우저라는 디바이스 특성상 전화통화가 불가능하므로 마지막 네 번째 화면이 비정상적으로 출력된 모습을 볼 수 있다. 이와 같이 디바이스의 특성상 약간의 문제는 있지만 인

터넷 환경에서 익스플로러 브라우저를 통하여 테스트해 본 결과 무리 없이 애플리케이션이 서비스되는 모습을 확인할 수 있다.

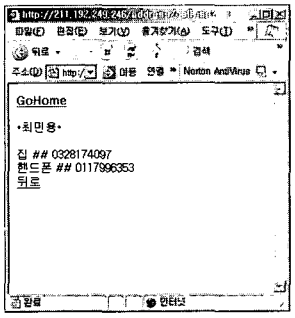


(그림 28) IE Test 1



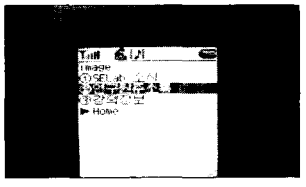
(그림 29) IE Test 2

(그림 30) IE Test 3

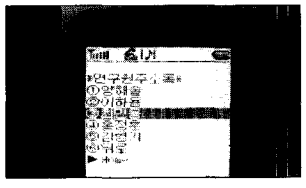


(그림 31) IE Test 4

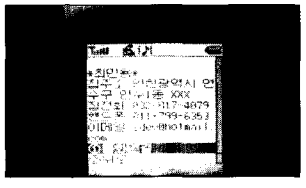
다음은 무선 인터넷 환경에서 무선 디바이스를 대상을 테스트한 모습이다. 테스트에 사용된 디바이스는 SK 텔레콤의 SKY-6200과 HP의 IPAQ 5450이다. 이는 현재 보편적으로 널리 사용되는 핸드폰과 PDA를 대상으로 한 것이다. 위에서 작성된 네 개의 소스코드에 대하여 (그림 32), (그림 33), (그림 34), (그림 35)는 SKY-6200, (그림 36), (그림 37), (그림 38), (그림 39)는 IPAQ 5450을 테스트 한 모습을 나타내고 있다.



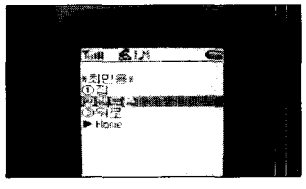
(그림 32) SKY-6200 Test



(그림 33) SKY-6200 Test

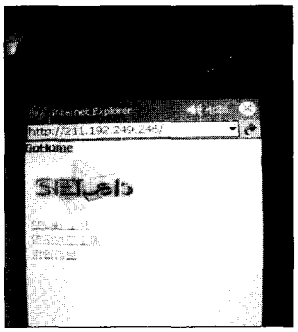


(그림 34) SKY-6200 Test 3

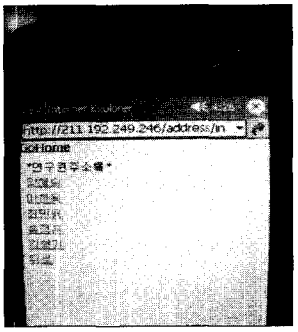


(그림 35) SKY-6200 Test 4

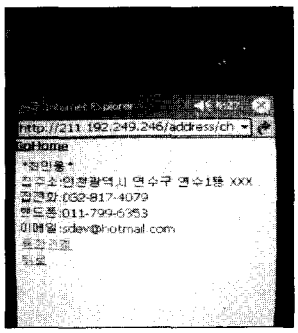
위에서 인터넷 익스플로러를 가지고 유선환경에서 테스트한 결과와는 다르게 네 번째 화면에서 해당 전화번호로 정상적으로 통화할 수 있도록 나타난 모습을 확인할 수 있다. 그리고 핸드폰이라는 디바이스 특성상 위에서의 인터넷 익스플로러의 모습과는 약간 다른 모습을 볼 수 있다.



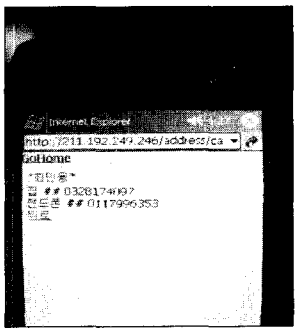
(그림 36) ipaq5450 Test 1



(그림 37) ipaq5450 Test 2



(그림 38) ipaq5450 Test 3



(그림 39) ipaq5450 Test 4

IPAQ 5450을 가지고 테스트한 결과 나타난 모습이 인터넷 익스플로러를 가지고 테스트한 모습과 거의 흡사하다. 그리고 이것 또한 전화통화의 기능이 없으므로 네 번째 화면에서 비정상적으로 출력된 모습을 확인할 수 있다.

6.2 분석 및 고찰

위와 같이 시험한 결과 구축된 애플리케이션에 대하여 모든 디바이스에서 비슷한 결과가 나타나는 것을 확인할 수 있다. SKY-6200의 경우 이미지 파일이 출력되지 않았지만, 이는 해당 디바이스에서 인식할 수 있는 이미지 형식인 wbmp 파일이 존재하지 않기 때문이므로 구현에 대한 오류라고 볼 수는 없다[18]. 그리고 디바이스 고유 특성에 따르는 기능은 디바이스에서 처리 가능하지 않을 경우 비정상적으로 나타나는 것을 확인할 수 있다. 이러한 점을 제외하고는 모든 데이터가 올바르게 출력되는 모습을 확인할 수 있다. 이는 하나의 애플리케이션을 구축하여 다양한 종류의 디바이스를 대상으로 서비스를 제공할 수 있으므로 디바이스에 독립적인 애플리케이션 구축을 나타내는 것이다. 그러나 위와 같은 서비스를 제공함에 있어 문제점은 화면을 구

성하는데 있어 많은 제약이 있다는 것이다. 이는 무선 디바이스 특유의 제한적인 하드웨어 환경으로 인한 것인데, 이를 위해서는 제한적인 화면에서 효과적으로 정보를 제공할 수 있는 기술이 요구된다. 그리고 현재 고사양의 디바이스들이 계속 출시되고 있으므로 앞으로는 이에 대한 제약에서 자유로울 것이다[5].

평가를 위하여 테스트에 사용된 애플리케이션이 매우 간단하여 전체적인 평가는 이루어졌다고 볼 수는 없지만, 다양한 기능의 애플리케이션이라 할지라도 테스트에 사용된 애플리케이션의 확장의 개념으로 비슷한 결과를 유추하여 볼 수 있다. 그러나 현재 텍스트 위주의 테스트에서 벗어나 최근 그의 수요가 많이 요구되고 있는 멀티미디어 환경에 대한 테스트도 병행적으로 이루어져야 할 것이다.

7. 결론 및 향후 연구과제

오늘날 기술의 발전과 변화는 1990년대 들어 인터넷이라는 새로운 개념의 컴퓨팅 환경의 도입으로 인하여 빠른 속도로 전개되고 있다. 유선 인터넷의 보급으로 인한 정보화 사회의 기술 발전은 무선 인터넷의 보급으로 인하여 더욱더 가속화되고 있다. 이러한 무선 인터넷 환경의 확산을 뒷받침 해줄 수 있는 것 중에 하나가 바로 무선 인터넷상에서 사용자들이 이용하는 데이터, 서비스, 컨텐츠, 애플리케이션과 같은 것이다. 이러한 것들은 기존의 유선 인터넷 환경에서 이루어 졌던 것들과 서로 상호보완적으로 조화를 이루어 발전해 나가야 한다. 즉, 유선 환경의 인터넷과 무선 환경의 인터넷이 서로 혼용되어야하며 둘 사이에는 항상 동기화가 이루어져야 한다. 그러한 의미에서 본 논문은 유선의 환경을 무선의 환경으로 확장·전이 시에 유용하게 활용될 수 있다. 즉, 유·무선 통합 환경의 구축을 용이하게 할 수 있을 뿐만 아니라, XML 변환 기법을 이용하여 다양한 디바이스에 유연하게 반응하고, 새로운 디바이스에 대한 대응도 신속히 할 수 있다. 그리고, 개발에 있어서 기술 종속적이지 않은 시스템을 설계, 구현할 수 있으며, 개발 후의 유지보수도 용이하다.

따라서 본 논문에서는 위와 같은 시스템을 개발하기 위하여 디바이스에 종속적이지 않으면서 무선 인터넷에서 사용되는 컨텐츠 또는 서비스를 제공하기 위한 컨텐츠 변환 시스템을 설계 및 구현하였다. 이는 기존의 유선 인터넷 환경에서의 서비스를 쉽고 빠르게 무선 인터넷 환경으로 변환 또는 새롭게 구축할 수 있는 것인데, 이를 위하여 컨텐츠 변환에 대한 규칙을 위한 XML DTD를 정의하였고, 새로운 무선 디바이스에 대한 빠른 인식을 위하여 외부 설정파일을 사용하였다. 따라서 다양하고 이질적인 환경의 무선 인터넷 디바이스들을 수용하고 기존의 유선 인터넷 기반의 컨텐츠나 기타 서비스들을 무선 환경으로 확장, 혼용을 손쉽게 할 수 있게 되었다.

이로 인하여 언제 어디서나 유선과 무선에 상관없이 동일한 정보를 이용할 수 있으며 이를 통한 양질의 인터넷 서비스를 제공받을 수 있었으나, 앞으로 무선 인터넷이 가지고 있는 디바이스에 대한 제약, 컨텐츠에 대한 고급화 그리고 멀티미디어 서비스를 활용할 수 있는 기반 시설의 확충 등 보다 효율적이고 호환성 높은 표준의 제정과 함께 향후 기술적인 진보와 더불어 사용성의 측면에서의 양적·질적 성장이 이루어져야 할 것이다.

참 고 문 헌

[1] 배석희, 모바일 플랫폼 표준화 동향 및 향후 발전방향, TTA 저널, 제82호, July, Aug., 2002.

[2] 김윤호, 황홍선, 박준호, "모바일 콘텐츠 비즈니스로 가는 성공 로드맵", 도서출판 비비컴, p.34, Apr., 2003.

[3] 이기혁, 배석희. 이근호, "차세대 무선인터넷 기술", 진한도서, p.3, Feb., 2003.

[4] 이영곤, "이것이 모바일 비즈니스다", 도서출판 비비컴, p.46, Jan., 2002.

[5] 한국무선인터넷표준화 포럼, http://www.kwisforum.org.

[6] 김찬웅, 이명진, "자바 개발자를 위한 XML 프로그래밍", Oct., 2003.

[7] http://java.sun.com/products/servlet/whitepaper.html.

[8] Craig Larman, "Applying UML and Patterns," 2002.

[9] http://www.javaworld.com/jw-12-1998/jw-12-servletapi.html.

[10] http://java.sun.com/developer/technicalArticles/Servlets/JavaServerTech1/index.html.

[11] http://www.ibm.com/developer/xml.

[12] http://www.microsoft.com/windowsmobile/products/smartphone/default.mspx.

[13] http://www.microsoft.com/presspass/features/2000/aug00/08-10mobile.asp.

[14] http://www.w3c.org/TR/WD-xml-lang.

[15] 일본 멀티미디어 통신연구회, "모바일 컴퓨팅", 교보문고, p. 31, Aug., 2001.

[16] http://www-106.ibm.com/developerworks/java/library/j-pj2ee10.html.

[17] Sun Microsystems, "Java Technology & XML," http://developer.java.sun.com/developer/technicalArticles/xml/JavaTechandXML/, Nov., 2001.

[18] Lloyd Rutledge, "SMIL 2.0-XML for Web Multimedia," Internet Computing, IEEE, July, 2001.

[19] Bickmore T. and Schilit W., Digester : device- independent access to the world wide web, In Computer Networks and ISDN Systems, Vol.29, No.8, pp.1075-1082, 1997.

양 해 술

e-mail : hsyang@office.hoseo.ac.kr

1975년 홍익대학교 전기공학과(학사)

1878년 성균관대학교 정보처리학과(석사)

1991년 日本 오사카대학 정보공학과 S/W 공학전공(공학박사)

1975년~1979년 육군중앙경리단 전산실 시스템분석 장교

1980년~1995년 강원대학교 전자계산학과 교수

1986년~1987년 日本 오사카대학교 객원연구원

1994년~1995년 한국정보처리학회 총무이사, 논문지편집위원장

1995년~2002년 한국S/W품질연구소 소장

1999년~현재 호서대학교 벤처전문대학원 교수

2001년~현재 한국정보처리학회 부회장

2003년~현재 미국 ACIS 학회 Vice President

관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 품질감리와 컨설팅, OOA/OOD/OOP, CASE, SI), 프로젝트 관리, CBD기반기술, IT품질경영

최 민 용

e-mail : sdev@hotmail.com

1995년~2002년 호서대학교 컴퓨터공학과 (학사)

2002년~2004년 호서대학교 벤처전문대학원 석사과정(공학석사)

2004년~현재 (주)HnC테크놀러지 개발부 대리

관심분야 : 소프트웨어공학(S/W품질평가, 개발 방법론, 인공지능), XML, 임베디드 시스템, Wireless Technology, Mobile Game

황 석 형

e-mail : shwang@sunmoon.ac.kr

1991년 강원대학교 전자계산학과 조기졸업 (이학사)

1994년 일본 오사카대학 대학원 정보공학과 (공학석사)

1997년 일본 오사카대학 대학원 정보공학과 (공학박사)

1997년~현재 선문대학교 컴퓨터정보학부 부교수

2001년~현재 국방대학원 국방정보화사업관리과정 외래강사

2001년~현재 일본 OGIS-RI Co. LTD. Certified UML Engineer

관심분야 : 객체지향 소프트웨어 시스템의 재구성 및 재이용, UML, Design Pattern, Adaptive Programming 기법, Formal Method 등