# 이동 단말을 위한 SyncML 기반 응용 관리 시스템의 설계 및 구현

## (Design and Implementation of SyncML based Application Management System for Mobile Devices)

박 지 은 † 김 상 욱 ††
(Jieun Park) (Sangwook Kim)

**요 약** 본 논문의 목적은 웹에 게시되는 이동 단말 전용 응용 소프트웨어 정보 검색 과정을 자동화하고, 해당 소프트웨어를 단말에 설치하는 과정을 자동화하여, 이를 한 단계로 통합하여 제공함으로써 단말 사용자들이 보다 쉽게 응용 소프트웨어를 이용할 수 있도록 하는 것이다. 이를 위하여 본 논문에서는 프로그램 자동 배포를 위한 권고안인 OSD(Open Software Description)를 따르는 웹 에이전트를 구현함으로써 웹 모니터링 및 소프트웨어 다운로딩 작업을 자동화하였다. 또한, 단말과 서버간의 데이타 동기화 권고안인 SyncML(Synchronization Markup Languagc)을 기반으로 서버와 단말간 소프트웨어 설치 작업을 자동화하였다.

표준 권고안을 이용한 소프트웨어 자동 관리 방법은 사용자의 개입을 최소화할 뿐 만 아니라, 기존의 다양한 이동 단말, 서로 다른 운영체제, 그리고 다양한 응용 소프트웨어 다운로딩 사이트에 범용적으로 적용할 수 있다는 장점을 제공한다.

**키워드** : 데이타 동기화, 소프트웨어 배포, 이동 단말

***Abstract*** The purpose of this paper is that users can use mobile device applications more easily and efficiently through MoDAM (Mobile Device Application Management) system. MoDAM automates the process of web retrieval for obtaining mobile device applications and the process of installing or updating it on mobile devices and then supply users with these automated services in a step form by combining two separated processes.

We implement a web agent conforming to OSD (Open Software Description) specification for automating software retrieval and downloading and also a mobile device based software management module conforming to SyncML (Synchronization Markup Language) specification for synchronizing common data between server and client.

The method of automated software retrieval and management for mobile devices using these standard specifications minimizes users' interrupts and also can be applied to traditional several different mobile devices, operating systems or downloading web sites without platform dependent considerations.

**Key words** : Data Synchronization, Software Depolyment, Mobile Device, SyncML

## 1. 서 론

### 1.1 연구 배경 및 목적

현재, PDA(Personal Digital Assistant), HPC(Handheld Personal Computer) 등과 같은 이동 정보단말(이하 단말로 명명)은 하드웨어 측면과 소프트웨어 측면을 고려해 볼 때, 일반 사용자의 응용 소프트웨어 활용면에서 문제점을 가지고 있다.

하드웨어 측면에서 응용 소프트웨어 설치를 위해 단말이 유일하게 제공하는 물리적 매체는 네트워크 자체이다. 사용자는 단말의 서버 역할을 하는 데스크 탑 PC에 설치할 소프트웨어를 미리 가져가 놓고, 단말과 서버간의 네트워크 연결을 설정한다. 연결이 이루어지면, 사용자는 서버에 설치된 단말 고유의 응용 소프트웨어 설치 툴을

† 정 회 원 : 한국전자통신연구원 정보가전연구부 연구원
jepark@etri.re.kr
†† 비 회 원 : 경북대학교 컴퓨터과학과 교수
swkim@cs.knu.ac.kr
논문접수 : 2001년 4월 20일
심사완료 : 2001년 10월 10일

이용하여, 실제 응용 소프트웨어를 단말에 설치한다[1, 2]. 이러한 방법은 네트워크를 통한 소프트웨어 설치의 장점인 푸시 패러다임(Push Paradigm) 즉, 사용자의 개입 없는 자동 설치 방법을 이용하는 것이 아니라 기존 CD(Compact Disk)와 같은 물리적 매체를 이용한 소프트웨어 설치 방식인 풀 패러다임(Pull Paradigm) 즉, 단계별 사용자의 작업을 반드시 거쳐서 수행함으로써 네트워크 설치의 고유 장점을 살리지 못할 뿐만 아니라 사용자의 오버헤드가 필연적으로 따르게 된다[3].

소프트웨어 측면에서 고려해보면, 단말 전용 응용 소프트웨어들은 일정한 기간 동안만 유효한 정보 서비스 중심으로 개발되므로 사용자의 빈번한 업그레이드를 필요로 한다[4]. 또한, 단말을 이용하는 실외 업무의 다양화로 인해 실생활에 유효한 정보 서비스 중심으로 개발되므로 특정 응용 영역이 정해져 있지 않고, 수요가 있을 때 곧바로 개발되므로 응용 영역에 대한 예측이 불가능하다. 이와 같은 특성으로 인해 단말 전용 응용 소프트웨어들은 기존 응용 소프트웨어와 달리 대부분 웹 사이트를 통해 게시되며, 현재 매일 수백가지의 이상의 새로운 소프트웨어가 생성 및 수정되어 게시된다.

따라서 사용자는 지속적으로 웹 정보를 모니터링하여 원하는 소프트웨어를 자신의 데스크 탑 PC로 다운로딩하고 이를 다시 단말에 설치하여야 한다. 이는 단말의 다양한 사용자 층을 고려해 볼 때, 컴퓨터에 익숙하지 않는 일반 사용자에게는 사용상의 어려운 점이 있을 뿐만 아니라, 익숙한 사용자에게도 새로운 소프트웨어 정보를 계속해서 모니터링하여 획득 및 활용하여야 하므로 많은 오버헤드가 필연적으로 따르게 된다[4].

본 논문에서는 기존 단말의 응용 소프트웨어 배포 방법의 문제점 즉, 사용자의 지속적인 웹 모니터링, 그리고 세 디바이스 즉, 소프트웨어 다운로딩 웹 사이트, 데스크 탑 PC, 실제 응용 소프트웨어를 설치할 단말을 거쳐서 수행되는 단순 반복적인 소프트웨어 배포 방법을 개선하여 일반 사용자들이 보다 쉽게 응용 소프트웨어를 활용할 수 있도록 하는 것이다.

### 1.2 논문의 접근 방향 및 문제 해결 방법

접근 방향은 웹에 게시되는 응용 소프트웨어 정보 검색 과정을 자동화하고, 해당 소프트웨어를 단말에 설치하는 과정을 자동화하여, 이를 한 단계로 통합하여 제공함으로써 일반 사용자들의 개입을 최소화하는 것이다.

본 논문에서 제시하는 문제 해결 방법은 다음과 같이 요약할 수 있다.

첫째, 프로그램 자동 배포를 위한 권고안인 OSD(Open Software Description)를 따르는 웹 에이전트를 구현함으로써 웹 모니터링 및 소프트웨어 다운로딩 작업을 자동화한다.

둘째, 단말과 서버간 데이타 동기화(Data Synchronization)를 위한 권고안인 SyncML(Synchronization Markup Language)을 기반으로 서버와 단말간 소프트웨어 설치 작업을 구현함으로써 사용자의 작업을 최소화 한다.

표준 권고안인 OSD와 SyncML을 결합한 소프트웨어 자동 배포 방법은 사용자의 개입을 최소화할 뿐 만 아니라, 기존의 다양한 개인정보단말, 서로 다른 운영체제, 그리고 다양한 응용 소프트웨어 다운로딩 사이트에 범용적으로 적용할 수 있다는 장점을 제공한다.

본 논문의 구성을 보면, 2장에서는 관련 연구에 관하여 기술하고, 3장에서는 구현하고자 하는 소프트웨어 자동 배포 시스템인 MoDAM(Mobile Device Application Management)의 특성과 기존 시스템과의 비교 그리고 4장에서는 세부 구현, 5장에서는 구현 결과 및 평가에 관하여 기술한다.

## 2. 관련 연구

현재까지 웹에 게시된 응용 소프트웨어를 자동으로 단말에 배포하는 방법에 대한 연구는 거의 없다. 본 장에서는 부분적인 관련 연구 즉, 사용자 개입 없이 웹에 게시된 소프트웨어 정보를 모니터링하고, 해당 소프트웨어를 데스크 탑 PC에 자동으로 갱신 및 설치하는 것과 관련된 연구 그리고, 단말에 응용 소프트웨어를 배포하는 방법과 관련된 연구로 나누어 기술하고자 한다.

### 2.1 웹 모니터링 및 소프트웨어 설치 혹은 갱신 작업의 자동화 관련 연구

소프트웨어 자동 배포와 관련된 연구는 크게 두 형태 즉, 보다 범용의 자동화를 위해 소프트웨어 배포 시 필요한 지식들을 표준화된 언어나 스키마 형태로 기술하는 것에 초점을 맞춘 연구[3, 5]와 소프트웨어를 사용자 시스템에 실제 설치 및 갱신하는 작업을 자동화하는데 초점을 맞춘 연구로 나눌 수 있다[6, 7, 8].

전자에 해당하는 대표적인 연구는 DSD, OSD, MIF, AMS 등을 들 수 있다. 후자에 관한 대표적인 연구는 NSBD, Software Dock 등이 있다.

#### 2.1.1 DSD(Deployable Software Description)

DSD는 콜로라도대학에서 개발한 소프트웨어 시스템 기술 형식(Software System Description Format)으로 Extensible Markup Language(XML) 응용이다[9]. DSD의 특징은 한 소프트웨어의 전체 제품군 즉, 상이한 버전, 다양한 플랫폼에 탑재되는 모든 소프트웨어를

하나의 DSD 형식으로 기술한다. 이는 동일한 제품군에 속하는 특정 버전 혹은 특정 플랫폼에 해당되는 소프트웨어에 대한 접근 및 관리를 보다 용이하게 지원할 수 있다는 장점을 제공한다. 또한, DSD는 소프트웨어가 배포되는 사용자 사이트에 대한 정보 즉, 플랫폼, 운영체제, 기 인스톨된 소프트웨어 정보 등의 기술을 지원하며, 간단한 형태의 수식 언어(Expression Language)가 내재되어 있다. DSD를 이용한 소프트웨어 자동 배포 시스템으로는 Software Dock Framework[7, 9]이 있다.

2.1.2 OSD(Open Software Description)

OSD는 푸시 패러다임을 적용한 1) Hans-free Install, 2) Easy and timely Upgrade, 3) Cross Platform의 특징을 제공하기 위하여 Marimba와 Microsoft가 공동으로 W3C에 제안한 XML 응용이다[3]. 1)은 사용자 개입 없는 소프트웨어 자동 설치를 뜻하며, 2)는 소프트웨어 자동 업그레이드, 3)은 동일한 소프트웨어 버전에 해당되는 상이한 플랫폼 탑재 소프트웨어들의 정보(Single Revision, Multiple Variant)들을 하나의 OSD 형식으로 기술함으로써 소프트웨어 설치 시 플랫폼에 따른 복잡성을 줄이는 것을 뜻한다. OSD는 소프트웨어 패키징을 위한 구성 요소들의 정보 즉 버전, 내부 구조, 요소간 상호 관계를 기술하기 위한 단어들로 구성된다.

2.1.3 MIF(Management Information Format)

MIF는 업계(Industry) 컨소시엄인 Desktop Management Task Force(DMTF)에서 소프트웨어 시스템 기술을 위해 개발한 표준 스키마 이다[10]. MIF의 특징은 단일 소프트웨어 시스템(Single Revision, Single Variant) 명세를 위한 스키마 이며, 스키마 내부에 각 속성(attribute)에 대해 표준 값을 정의하고 있다. MIF는 기존 연구와 달리 웹 표준이나 프로토콜을 사용하지 않는다.

2.1.4 AMS (Application Management Specification)

AMS는 MIF의 확장 버전으로 Tivoli에 의해 개발되었으며, MIF와 동일하게 단일 소프트웨어 기술을 위한 스키마이다[10]. AMS는 소프트웨어 시스템 구성, 제약조건, 상호 의존성, 실행 파일, 지원 기능 등을 기술하는 기본 요소로 구성된다. 또한, AMS는 사용자 사이트에 설치된 소프트웨어 시스템의 구성 정보에 대한 기술 기능을 부가적으로 포함한다.

AMS의 개발 목적은 기업 프레임워크 환경 하에서 배포된 소프트웨어 시스템의 상태를 유지 및 관리하기 위함이다.

2.1.5 NSBD(Not-So-Bad Distribution)

Dave Dykstra와 Katherine Lato에 의해 개발된 NSBD는 인터넷 상에서 Free Software를 암호화 기법을 사용하여 안전하게 응용 소프트웨어를 자동으로 배포하기 위한 Unix 시스템이다[6]. NSBD의 자동 배포 방법은 구현 측면에서 푸시 패러다임과 풀 패러다임을 동시에 지원한다. 풀 패러다임은 NSBD 패키지 명세 형식으로 게시된 소프트웨어 정보를 NSBD 클라이언트가 주기적으로 분석하여 변경 유무를 파악하고 해당 소프트웨어를 다운로딩하여 설치하는 형태로 이루어진다. 이는 구현 측면에서는 풀 패러다임이지만, 사용자 입장에서는 결과적으로 푸시 패러다임 형태의 서비스를 받게 된다. 푸시 패러다임은 소프트웨어 변경 시 개발자가 사용자에게 미리 제공한 CGI 스크립트를 NSBD 서버가 직접 호출하는 방법으로 클라이언트의 소프트웨어 변경이 이루어진다.

2.1.6 Software Dock

콜로라도대학에서 개발한 Software Dock은 DSD 형식으로 소프트웨어를 명세하며[7], 소프트웨어 개발 이후의 모든 과정 즉, 설치, 재구성, 갱신, 활성, 비활성, 제거 등을 에이전트를 통해 자동화한 프레임워크이다.

Software Dock은 두개의 주요 구성 요소인 Release Dock과 Field Dock으로 구성되며, 각 Dock은 소프트웨어 배포 작업을 실제 수행할 에이전트들이 상주한다. Release Dock은 서버에 상주하면서 릴리즈된 소프트웨어 정보 게시 및 저장소 기능을 하며, 웹 기반 서비스를 수행한다. Field Dock는 사용자 사이트에 상주하면서 사용자 시스템 정보를 관리한다. Software Dock의 특징은 소프트웨어 공급자뿐 만 아니라 사용자 사이트에 대해서도 표준화된 방법으로 정보를 기술하는 DSD를 이용함으로써 소프트웨어의 배포를 자동화하였다.

### 2.2 개인용 정보 단말의 응용 소프트웨어 배포 관련 연구

본 절에서는 대표적인 개인용 정보 단말 OS인 Palm과 WinCE 계열에서 지원하는 응용 소프트웨어 배포 방식에 관하여 기술한다.

2.2.1 Palm 계열의 응용 소프트웨어 배포

Palm 계열 단말의 응용 소프트웨어 배포는 데스크탑과 단말간의 데이타 동기화를 위한 소프트웨어인 HotSync[1]에 의해 이루어진다. 먼저, 사용자는 설치하고자 하는 소프트웨어를 데스크 탑 PC의 특정 위치에 가져가 놓은 후, 데스크 탑 PC에 설치된 HotSync를 실행하여 설치할 소프트웨어를 선택한다. 다음, 단말을 거치대(Cradle)에 탑재하고, 데스크 탑과 연결 설정을 한 후, 거치대에 있는 HotSync 버튼을 누름으로써 소프트웨어 설치가 이루어진다. HotSync는 데스크 탑 PC와

단말이 공통으로 사용하는 응용 소프트웨어의 데이타 동기화 서비스와 데스크 탑 PC과 단말간의 파일 전송 기능을 수행한다. 응용 소프트웨어 설치는 HotSync의 파일 전송 기능을 통해 이루어진다.

2.2.2 WinCE 계열의 응용 소프트웨어 배포

WinCE 계열 단말의 응용 소프트웨어 배포는 Palm과 같이 동기화 툴을 이용하지 않고, 단말도 데스크 탑에 연결된 하나의 물리적 장치로 간주하여 기존의 Window 응용 프로그램의 설치 방법과 동일하게 응용 소프트웨어를 설치한다[2]. 실제 WinCE 응용 프로그램 설치 작업은 데스크 탑 PC에서 수행되는 WinCE 용 Application Manager인 CEAppMgr에 의해 이루어진다. CEAppMgr은 먼저, 설치하고자 하는 소프트웨어의 사양을 명세한 INI파일을 참조하여 필요한 정보를 Window Registry Table에 등록하면서 해당 응용 소프트웨어를 단말로 다운로딩하고 설치한다. INI 파일은 응용 소프트웨어의 버전, 아이콘 파일, CAB 파일 등의 정보를 포함한다.

## 3. MoDAM 시스템의 개요

본 장에서는 MoDAM 시스템의 개념 모델과 특징 그리고 기존 시스템과의 차이점에 관하여 기술한다.

### 3.1 MoDAM 시스템의 정의와 개념 모델

MoDAM 시스템은 1) 사용자 개입 없이 OSD 형식으로 게시된 단말 전용 응용 소프트웨어 웹 사이트를 주기적으로 모니터링하고, 2) 현재 사용자의 단말에 설치되지 않거나 변경된 소프트웨어 정보가 있는 경우, 이를 SyncML을 기반으로 단말에 전달하며, 3) 사용자가 원하는 경우, 자동으로 소프트웨어를 설치하거나 갱신하는 단말 전용 응용 소프트웨어 자동 배포 시스템으로 정의할 수 있다. MoDAM의 개념 모델은 그림 1과 같다.

MoDAM 시스템은 데스크 탑 PC에 탑재되는 MoDAM 서버와 단말에 탑재되는 MoDAM 클라이언트로 구성된다. 사용자는 MoDAM 서버를 통해 단말 전용 응용 소프트웨어 정보를 게시하는 여러 웹 사이트를 등록한다. 이때, 등록된 웹 사이트는 OSD 형식으로 소프트웨어 정보를 게시하여야 한다.

MoDAM 서버는 개념상 WIM(Web Information Manager)과 SyncML 서버로 구성된다. WIM은 사용자가 등록한 웹 사이트에 게시된 소프트웨어 정보를 주기적으로 분석하고 이의 결과를 데이타베이스에 저장한다. WIM 내부의 데이타베이스는 아직 사용자에게 알리지 않은 웹 게시 소프트웨어 정보와 단말에 이미 탑재된 응용 소프트웨어 정보를 저장하고 있다.

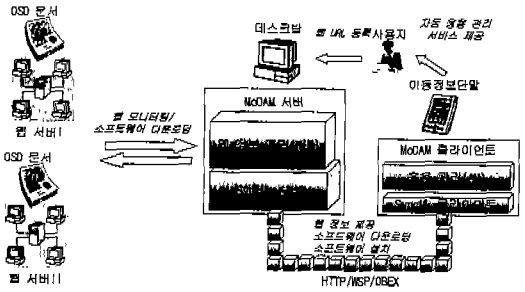



그림 1 MoDAM의 개념 모델

SyncML 서버는 서버와 클라이언트간 응용 소프트웨어 정보에 대한 동기화 요구가 있는 경우 즉, 단말에 설치되지 않은 소프트웨어 정보가 있거나, 혹은 사용자가 MoDAM 서비스를 통하지 않고 매뉴얼 방식으로 이미 단말에 설치한 소프트웨어 정보가 있는 경우, SyncML 명세 프로토콜(SyncML Representation Protocol)[11]을 기반으로 SyncML 문서를 생성하며, SyncML 동기화 프로토콜(SyncML Synchronization Protocol)[12]에 따라 MoDAM 클라이언트로 정보를 전달한다.

MoDAM 클라이언트는 응용 관리 부분(AM: Application Manager)과 SyncML 클라이언트로 구성된다. AM은 현재, 매뉴얼 혹은 자동화 방식으로 단말에 설치된 응용 소프트웨어에 대한 정보를 저장 및 관리하는 기능을 한다. SyncML 클라이언트는 SyncML 서버와 대응 관계에 있는 모듈로 SyncML 권고안에 따라 서버와 클라이언트가 공유하는 데이타에 대한 동기화 서비스를 수행한다.

### 3.2 OSD와 SyncML

기존 이동 단말의 응용 소프트웨어 관리 방법과 비교해 볼 때 MoDAM 시스템의 가장 큰 차이점은 "***플랫폼 독립적인 자동화된 휴대 단말의 응용 소프트웨어의 관리 방법***"으로 요약할 수 있다. "***자동화***"와 "***플랫폼 독립적***"이라는 두 가지 특징을 위해 MoDAM은 OSD와 SyncML이라는 권고안을 기반으로 시스템을 설계하였다. MoDAM에서 OSD를 채택한 이유는 1) 소규모 단어 집합 구성, 2) XML 기반의 응용, 3) 동일 버전의 상이한 플랫폼(Single Revision, Multiple Variant) 탑재 소프트웨어 정보 표현 기능으로 요약할 수 있다. 1)은 단말 전용 소프트웨어의 규모와 특징을 고려할 때, DSD, MIF, AMS와 같은 자세한 소프트웨어 표현 기능은 불필요하다. 2)는 웹 기반 응용을 고려할 때 선행되어야 하는 기능으로 MIF나 AMS에서 지원되지 않은 기능이며, 3)은 다양한 플랫폼상에서 정보 관리 기능이

주요 특징인 단말 전용 응용 소프트웨어를 고려할 때 가장 적합한 기능이다.

SyncML의 주요 내용은 명세 프로토콜(SyncML Representation Protocol)과 동기화 프로토콜(SyncML Synchronization Protocol)로 요약할 수 있다. 전자는 XML 응용으로 데이타 동기화를 위한 메시지 내용들을 SyncML 문서 형식으로 표현하기 위한 권고안이며[12], 후자는 서버와 클라이언트간 데이타 동기화를 위해 수행하여야 하는 절차를 명세한 권고안이다[11]. MoDAM은 SyncML을 이용하여 단말과 서버간의 응용 소프트웨어에 대한 정보를 동기화 하고 있다. 이의 장점은 SyncML 자체가 가지는 상호 연동성이며, 기존의 응용 소프트웨어 배포 서비스와 달리 플랫폼 독립적인 응용 소프트웨어 배포가 가능하다. 또한 단말의 시스템 오류 시 응용 소프트웨어 복구 및 관리가 용이하다. 결론적으로 MoDAM은 앞서 기술한 SyncML의 중요한 응용인 단말 관리 시스템의 한 예로 적용될 수 있다.

### 3.3 기 개발된 시스템과의 차이점

본 절에서는 MoDAM과 직접적인 관련이 있는 시스템 즉, 단말의 응용 소프트웨어 배포 시스템인 Instant Update[4] 그리고 데이타 동기화 메커니즘을 이용한 소프트웨어 배포 시스템인 HotSync로 범위를 좁혀 이들 간의 차이점에 관하여 기술한다.

InstantUpdate 시스템은 동일한 목적을 가지고 구현된 MoDAM의 선행 시스템이다. InstantUpdate는 멀티 에이전트 기반 구조인 OAA(Open 에이전트 Architecture) 상에서 구현된 응용으로 웹에 게시된 소프트웨어 정보 분석을 위한 웹 에이전트들의 집합 그리고, 데스크탑에서 웹 에이전트들로부터 필요한 정보를 전달 받는 데스크 탑 에이전트 그리고 단말에 상주하면서 데스크 탑 에이전트로부터 전달된 응용 소프트웨어를 설치 및 관리하는 PDA 에이전트로 구성된다. InstantUpdate 시스템은 MoDAM과 비교할 때 자동화라는 특징은 동일하게 제공하지만, 플랫폼 독립적인 소프트웨어 배포 서비스는 수행할 수 없다. 즉, OAA에 탑재되는 에이전트들의 상호 작용은 표준화된 권고안에 따라 수행되지 않으며, 또한 웹에 게시되는 응용 소프트웨어 기술 방법도 표준화된 소프트웨어 명세 방법이 아니므로 특정 웹 사이트와 특정 단말 상에서만 소프트웨어 배포 서비스가 운영될 수 있는 단점이 있다.

HotSync는 소프트웨어 배포 부분에 대해 자동화적인 측면과 플랫폼 독립적인 측면으로 나누어 MoDAM과의 특징을 비교한다. 먼저, 자동화적인 측면은 사용자 개입 없이 웹 게시 정보를 분석하는 부분과 사용자의 작업 즉, 설치할 소프트웨어 지정이 이루어지면, 데이타 동기화 타임에 맞춰 단말로 소프트웨어 자동설치가 이루어지는 부분으로 나누어 비교할 수 있는데, HotSync는 전자 의미의 자동화는 이루어지지 않고 있다. 또한 플랫폼 독립적인 측면에서 HotSync에서 수행되는 소프트웨어 자동 설치는 HotSync 고유의 데이타 동기화 메커니즘을 사용하여 수행되므로 특정 기기, 특정 응용에 제한적으로 운영될 수 있는 단점이 있다. 따라서 MoDAM은 기존 시스템에 비해 자동화와 플랫폼 독립적이라는 장점을 지닌다.

## 4. MoDAM 시스템의 구조

본 장에서는 MoDAM 시스템의 전체 구조 및 세부 설계에 관하여 기술한다.

### 4.1 전체 구성

MoDAM은 그림 2와 같이 클라이언트/서버 모델로 단말에는 MoDAM 클라이언트 패키지, 서버에는 MoDAM 서버 패키지가 탑재한다.

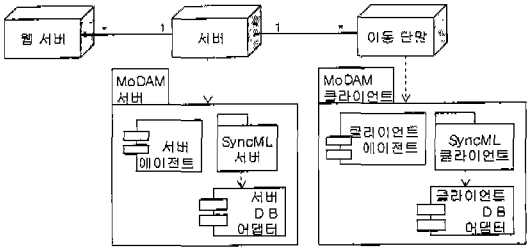



그림 2 MoDAM의 구성

웹 서버는 단말 전용 소프트웨어 정보를 OSD 형식으로 관리하는 소프트웨어 다운로딩 사이트이며, 서버와 N:1의 관계를 가진다. 사용자는 MoDAM 시스템을 통해 하나 이상의 소프트웨어 다운로딩 사이트를 등록할 수 있으며, MoDAM 서버는 이를 기준으로 웹 검색 서비스를 수행한다.

서버와 이동 단말은 1:N의 관계를 가진다. MoDAM 서버는 연결된 하나 이상의 이동 단말에게 웹에 새롭게 게시된 소프트웨어 정보를 알려주거나, 새로운 소프트웨어를 설치 혹은 갱신 작업을 수행한다. MoDAM 서버 패키지는 서버 에이전트, SyncML 서버, 서버 DB 어댑터로 구성되며, MoDAM 클라이언트 패키지는 클라이언트 에이전트, SyncML 클라이언트, 클라이언트 DB 어댑터로 구성된다.

서버 에이전트와 클라이언트 에이전트는 그림 1의 개

념 모델에서 각각 WIM과 AM에 해당하며, 서버 DB 어댑터와 클라이언트 DB 어댑터는 소프트웨어 정보 관리를 위해 필요한 서버와 클라이언트 데이타 베이스를 접근 및 갱신하는 기능을 한다.

### 4.2 구성 요소간 상호 작용

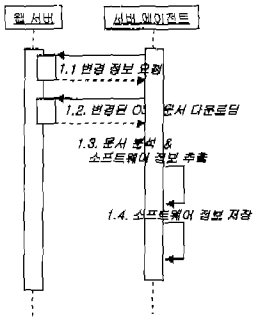
구성 요소간 상호 작용은 그림 3과 같이 두 단계 즉, 웹으로부터 소프트웨어 정보를 검색하는 단계와 이동 컴퓨터와 서버간 데이타 동기화를 통해 소프트웨어를 획득하는 단계로 나누어 진행된다.

소프트웨어 정보 검색을 위한 과정은 다음과 같다. 서버 에이전트는 먼저, 웹 서버의 변경 정보를 요구한다(1.1). 웹 서버의 변경이 이루어진 경우, 서버 에이전트는 해당 문서를 서버로 다운로딩 한다(1.2). 다음, 다운로딩한 문서를 파싱하여, 정보를 추출하고(1.3), 이를 서버 데이타베이스에 저장한다(1.4).

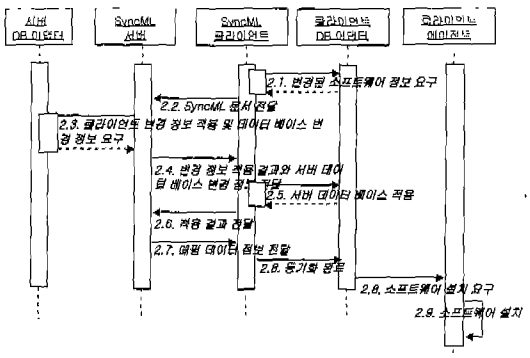
소프트웨어 획득을 위한 과정은 다음과 같다. SyncML 클라이언트는 클라이언트 DB 어댑터에게 이전 동기화 이후 변경된 데이타베이스 정보를 요구한다(2.1). SyncML 클라이언트는 변경 정보를 SyncML 문서로 변환하여 SyncML 서버로 전달한다(2.2). SyncML 서버는 서버 DB 어댑터에게 클라이언트의 변경된 데이타 정보를 서버 데이타베이스에 적용할 것과 동기화 이전에 발생한 서버 데이타베이스의 변경 정보를 요구한다(2.3).

SyncML 서버는 수신한 변경 정보를 2.3의 결과와 서버 데이타 베이스의 변경 정보를 SyncML 문서로 변환하여 SyncML 클라이언트에게 전달한다(2.4).

SyncML 클라이언트는 클라이언트 DB 어댑터에게 서버 데이타의 변경 정보를 데이타베이스에 적용할 것을 요구하며(2.5), 이의 결과를 SyncML 문서 형식으로 변환하여 SyncML 서버로 전달한다(2.6). SyncML 서버는 클라이언트와 공통으로 가지는 데이타에 대한 매핑 정보와 동기화 완료를 알리는 메시지를 보낸다(2.7). 이상의 작업이 완료되면, 웹 서버에 게시된 새로운 소프트웨어 정보는 단말로 전달된다. 다음, 획득한 소프트웨어 정보를 단말에 적용하기 위해 SyncML 클라이언트는 동기화가 완료되었음을 클라이언트 DB 어댑터(2.8)에게 알린다. 클라이언트 DB 어댑터는 클라이언트 에이전트에게 서버에서 획득한 새로운 소프트웨어 정보를 사용자에게 알리고, 사용자가 원하는 소프트웨어를 단말에 설치 한다(2.9).




(a) 소프트웨어 정보 검색을 위한 상호 작용




(b) 동기화를 통한 소프트웨어 획득을 위한 상호 작용

그림 3 MoDAM의 구성 요소간 상호 작용

### 4.3 MoDAM 설계

MoDAM 시스템의 전체 설계도는 그림 4와 같다. MoDAM 서버 패키지 내부는 동기화 게이트웨이, 동기화 어댑터, 동기화 엔진, SyncML 처리로 구성되는 SyncML 서버와 서버 에이전트 그리고 서버 DB 어댑터로 이루어진다. 서버 에이전트는 소프트웨어 검색 기능을 수행하기 위한 패키지이며, 서버 DB 어댑터는 서버 데이타베이스의 접근 및 갱신 기능을 수행하기 위한 패키지이다. 동기화 게이트웨이는 SyncML 클라이언트와의 네트워크 송수신을 담당하는 패키지로 SyncML 스펙에 따라 근거리는 OBEX, 원거리 통신은 WSP, HTTP 프로토콜을 이용하여 SyncML 문서를 송수신한다. 동기화 어댑터는 수신한 SyncML 명령을 처리하고, 송신할 SyncML 문서를 생성하기 위하여 동기화 엔진과 SyncML 처리를 함께 결합하는 기능을 한다. 동기화 엔진은 수신한 SyncML 명령들을 수행하며, 실제 서버 데이타 베이스의 추가 및 갱신, 변경된 데이타 질의를 위해 서버 DB 어댑터를 호출한다. SyncML 처리는 수신한 SyncML 문서를 파싱하여 SyncML 명령별 구조체로 변경하는 기능과 데이타베이스의 변경된 정보를 클라이언트로 전달하기 위하여 SyncML 명령별 구조체 내용들을 SyncML 문서로 변환하기 위한 기능을 한다.

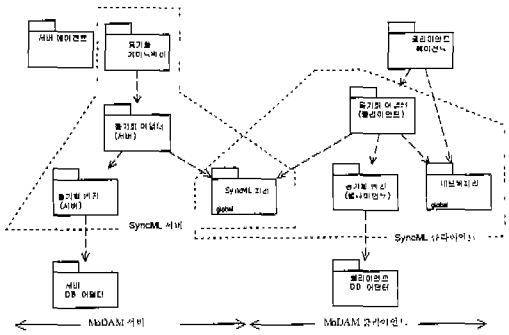


그림 4 MoDAM의 설계

MoDAM 클라이언트 패키지 내부는 동기화 어댑터, 동기화 엔진, SyncML 처리, 네트웍 처리로 구성되는 SyncML 클라이언트와 클라이언트 에이전트, 클라이언트 DB 어댑터로 구성된다. 클라이언트 에이전트는 SyncML 클라이언트로 동기화를 요구하거나, 변경된 소프트웨어 정보를 사용자에게 전달하는 기능을 수행하기 위한 패키지이다. 클라이언트 DB 어댑터는 클라이언트 데이타베이스의 접근 및 갱신 기능을 수행하기 위한 패키지이다. 네트웍 처리는 네트워크 환경에 따라 SyncML 문서를 서버로 송수신하기 위한 각 프로토콜을 지원하기 위해 필요한 함수들의 패키지이다. 동기화 어댑터, 동기화 엔진은 동일한 이름으로 존재하는 서버 패키지와 서로 대응관계를 가지며, 그 기능은 동일하다.

## 5. MoDAM 시스템의 구현

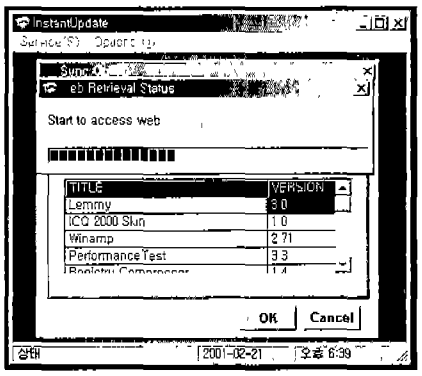
본 장에서는 MoDAM 시스템의 구현 결과 및 평가에 관하여 기술한다.

### 5.1 구현 결과

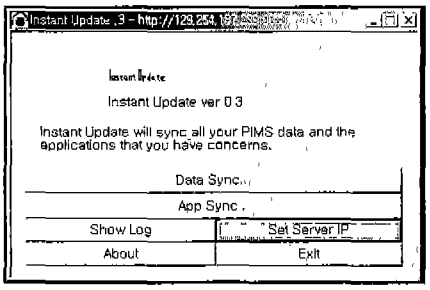
MoDAM 서버와 서버 에이전트는 Visual Basic 6.0으로, SyncML 서버는 SyncML 스펙 1.0을 기준으로 Visual C++로 구현하였다. SyncML 서버와 SyncML 클라이언트간 통신은 근거리인 경우에는 OBEX, 원거리인 경우 HTTP를 프로토콜을 지원한다. 서버 DB Adapter는 Microsoft Jet Engine을 이용하여 Visual C++로 구현하였다. MoDAM 클라이언트는 이식성을 고려하여 표준 C 언어로 구현하였으며, 현재 상품화된 PDA로 이식 작업을 진행 중이다.

본 절에서는 Windows2000 환경 하에서 테스트된 MoDAM 클라이언트의 실행 예를 기술한다. 따라서 MoDAM 클라이언트의 사용자 인터페이스 부분은 Microsoft Win32 API를 이용하여 추가 구현하였다.

그림 5의 (a)은 MoDAM 서버의 실행 예로 사용자가 지정한 일정 시간 간격으로 웹 정보를 추출하는 과정과 현재 저장하고 있는 데이타베이스에 대한 정보를 보여주는 화면이다 MoDAM 서버는 사용자의 데스크 탑에 데몬 형태로 실행되며, 주요 기능으로는 웹 검색 시간 지정 기능, 소프트웨어 정보를 보여주는 기능, 검색할 웹 사이트 등록 기능을 제공한다.




(a) MoDAM 서버




(b) MoDAM 클라이언트

그림 5 MoDAM 시스템의 실행 예

(b)는 MoDAM 클라이언트의 실행 예로 사용자는 "App Sync..." 버튼을 눌러 매뉴얼 형태로 MoDAM 서비스를 신청할 수도 있고, 서버로부터의 동기화 이벤트를 받은 경우, 새로운 소프트웨어 정보를 전달 받거나 소프트웨어 설치를 명할 수 있다.

### 5.2 평 가

이동 단말 사용자가 웹 서버로부터 원하는 소프트웨어를 획득하는 유형은 그림 6과 같이 세가지로 요약할 수 있다. 1-Type은 단말에 장착된 네트워크 디바이스를 통해 직접 n개의 웹 페이지를 방문하여 사용자가 원하는 m개의 소프트웨어를 다운로딩하고 설치하는 방법이

2-Type은 서버에서 n개의 웹 페이지를 방문하여, 의 원하는 소프트웨어를 서버로 다운로딩 한 다음, 기기용 네트워크 디바이스를 이용하여 다시 m개의 프트웨어를 이동 단말에 다운로딩 하여 설치하는 방이다. 3-Type은 MoDAM 방법에 해당한다. 이때, k n개의 Page에 포함된 새로운 소프트웨어의 총 개수 나타내며, SyncML 프로토콜은 SyncML 문서 형태 전달되는 소프트웨어 정보를 나타낸다.

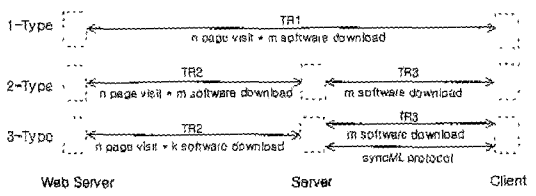



그림 6 단말 소프트웨어 설치 유형

본 평가에서는 일반적인 네트워크 환경을 고려하여 1-Type의 네트워크 디바이스는 평균 전송률 TR1이 56.6kbps 모뎀, 2-Type은 TR2가 150kbps인 10BASE T, 그리고 3-Type은 TR3가 115.2kbps인 시리얼 케이블을 사용하였다. 다음은 각 유형별로 사용자가 원하는 소프트웨어를 단말에 설치하기까지 걸리는 시간 T를 산출하기 위한 모델이다. 이때, $S(NP_i)$는 사용자가 서버 혹은 단말에 로딩하는 웹 페이지의 크기를 나타내며, $T(RP_i)$는 사용자가 원하는 소프트웨어를 찾기 위해 다운로딩한 웹 페이지를 분석하는 데 걸리는 시간을 나타낸다. $S(SD_j)$는 다운로딩할 소프트웨어 사이즈를 나타낸다. S(SyncDoc)는 소프트웨어 정보 전달을 위해 교환되는 SyncML 패키지의 크기를 나타낸다.

$$T(1-type) = \sum_{i=1}^{n} S(NP_i)/TR1 + \sum_{i=1}^{n} T(RP_i) + \sum_{j=1}^{m} S(SD_j)/TR1$$

$$T(2-type) = \sum_{i=1}^{n} S(NP_i)/TR2 + \sum_{i=1}^{n} T(RP_i) + \sum_{j=1}^{m} S(SD_j)/TR2 + \sum_{j=1}^{m} S(SD_j)/TR3$$

$$T(3-type) = \sum_{i=1}^{n} S(NP_i)/TR2 + \sum_{i=1}^{k} S(SD_i)/TR2 + S(SyncDoc)/TR3 + \sum_{j=1}^{m} S(SD_j)/TR3$$

먼저, T(1-type)과 T(2-Type)을 비교하면, TR2≒TR1*3, TR3≒TR1*2이므로 동일항을 제거한 후 T(1-type)은 $1/TR1*(\sum_{i=1}^{n} S(NP_i) + \sum_{j=1}^{m} S(SD_j))$, T(2-type)은 $1/TR1*(1/3 \sum_{i=1}^{n} S(NP_i) + 5/6 \sum_{j=1}^{m} S(SD_j))$이므로 T(1-type)>T(2-type)이다. T(2-type)과 T(3-type)에서 동일항을 제거하면, T(2-type)은 $\sum_{i=1}^{n} T(RP_i) + \sum_{j=1}^{m} S(SD_j)/3TR1$이고, T(3-type)은 $\sum_{i=1}^{k} S(SD_i)/3 * TR1 + S(SyncDoc)/2*TR1$이다.

이상의 결과에 현재 상용화된 웹 사이트(download.com)에 게시된 단말 소프트웨어 정보를 샘플로 대입하여 결과 값을 비교하고자 한다. 그림 7은 샘플 웹 사이트를 나타낸다.

사용자는 R0 페이지를 루트로 하는 웹 사이트에 접근하여, 새롭게 변경된 소프트웨어 정보를 포함하고 있는 D0, D1, D2 웹 페이지를 방문한다. D0는 새로운 소프트웨어 S0, S1, S2, S3를 게시하고 있으며, D1은 S4, D2는 S5, S6, S7, S8, S9를 각각 게시한다. 각 노드에 나타난 숫자는 페이지 혹은 소프트웨어의 사이즈를 나타낸다. 사용자는 최종적으로 S0, S4, S7, S9 소프트웨어를 단말에 설치한다.

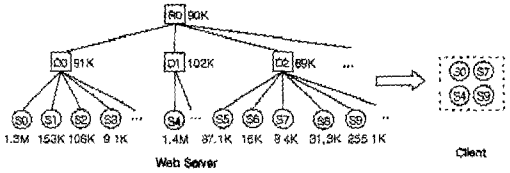



그림 7 유형별 비교를 위한 샘플 값

$\sum_{j=1}^{m} S(SD_j)$는 2,963KB, $\sum_{i=1}^{k} S(SD_i)$는 3,366KB이고, S(SyncDoc)는 총 전달된 SyncML 패키지의 평균이 3 KB이므로, T(2-type)은 $\sum_{i=1}^{n} T(RP_i)$+988/TR1, T(3-type)은 1,123.5/TR1이다. 이때, 988/TR1과 1123.5/TR1은 약 19초의 차이가 나므로 T(2-type)은 $\sum_{i=1}^{n} T(RP_i)$+T(3-type)-19이다. 따라서 T(2-type)은 $\sum_{i=1}^{n} T(RP_i)$이 19초 이하인 경우 즉, 사용자가 원하는 소프트웨어를 찾기 위해 다운로딩한 n개의 페이지를 분석하는데 걸리는 시간이 19초 이하인 경우를 제외하면, T(2-type)>T(3-Type) 이다.

결과적으로 MoDAM은 기존 형태에 비해 최소 시간에 원하는 소프트웨어를 단말에 설치할 수 있을 뿐만 아니라, 사용자의 개입 없이 자동화된 시스템을 통해 이루어진 결과 값이므로 기존 소프트웨어 배포 방법보다 비용 및 사용자의 편의성에 있어서 효과적이다.

## 6. 결 론

MoDAM 시스템이 주는 이점은 다음과 같이 요약할

수 있다. 첫째, MoDAM은 일반 사용자에게 자신의 이동 단말에 대한 응용 소프트웨어 관리 즉, 원하는 소프트웨어의 검색 및 설치 그리고 버전 관리 등에 대한 부담감을 최소화 할 수 있는 장점을 제공한다. 둘째, SyncML과 OSD와 같은 표준화된 권고안에 따라 시스템을 개발하였으므로 특정 시스템 혹은 특정 웹 사이트 정보 제공에 국한되지 않고 보다 다양한 형태의 서비스를 제공할 수 있다. 셋째, 플랫폼 의존적인 부분을 최소화하는 형태로 설계하였으므로 쉽게 타 시스템으로의 이식될 수 있다. 현재, MoDAM의 하부 구조인 SyncML 부분은 SyncML 단체에서 시행하는 권고안 준수 테스트(Conformance Test)와 타 단말간 상호 운영 테스트(Interoperability Test)를 위해 기능 보완 작업을 진행 중이다.

MoDAM의 향후 계획은 이동 단말의 응용 소프트웨어 관리 기능뿐만 아니라, 이동 단말 시스템 전반에 걸친 관리 기능을 수행할 수 있도록 기능 확장 작업과 사용자의 기호 정보 등 보다 지능적인 서비스 수행이 가능하도록 확장 보완할 계획이다.

## 참 고 문 헌


[1] David L. Wilson, Creating a Conduit for Macintosh, http://www.palm.com/devzone/palmsource/1999/data/track1/182_CreatingAConduit4Mac_v1.pdf, PalmSource 99.

[2] Microsoft, Installing Applications, http://msdn.microsoft.com/library/wcedoc/wcesetup/instapps.htm, 2000.

[3] Arthur van Hoff, Hadi Partovi, Tom Thai, The Open Software Description Format(OSD), Microsoft Corp. and Marimba, Inc., 1997. http://www.w3.org/TR/NOTE-OSD.html

[4] 박지은, 한동원, 황승구, 사공준, 김상욱, 휴대정보단말을 위한 에이전트 기반의 소프트웨어 자동 설치 시스템의 구현, 정보과학회논문지, Vol.27, No.12, pp.1183-1192, December 2000.

[5] Richard S.Hall, Dennis M. Heimbigner, Alexander L. Wolf, Evaluating Software Deployment Language and Schema, Proceedings of the 1998 International Conference on Software Maintenance, IEEE Computing Society, p.177~185, Nov. 19998.

[6] Dave Dykstra, Katherine Lato, Up to Date Software with the Not-So-Bad-Distribution Program, http://www.bell-labs.com/project/nsbd/nsbdpaper1.html, April, 21, 1998.

[7] Rechard S.Hall, Dennis Heimbigner, Alexander L. Wolf, A Cooperative Approach to Support Software Deployment Using Software Dock, ICSE'99, pp.174-183.

[8] Desktop Management Task Force, Inc. Desktop Management Interface Specification, Version 2.0, June 24, 1998.

[9] Richard Scott Hall, Agent-based Software Configuration and Deployment, Doctoral Thesis, University of Colorado, 1999.

[10] Tivoli Systems, Application Management Design, http://www.tivoli.com/services/education/courses/bodv_appdesign.html, 2001.

[11] SyncML Synchronization Protocol Specification, http://www.syncml.org/docs, Ver 1.0.

[12] SyncML Representation Protocol Specification, http://www.syncml.org/docs, Ver 1.0.



박 지 은

1991년 안동대학교 전산통계학과(학사). 1993년 경북대학교 전자계산학과(석사). 2001년 경북대학교 컴퓨터과학과(박사). 1993년 ~ 현재 한국전자통신연구원 정보가전연구부 휴대클라이언트팀 선임연구원. 관심분야는 HCI, 멀티 에이전트 시스템, 모빌 컴퓨팅 응용, 시각언어



김 상 욱

1979년 경북대학교에서 컴퓨터공학으로 학사 학위를 취득. 1981년 서울대학교에서 컴퓨터과학으로 석사 학위를 취득. 1989년 서울대학교에서 컴퓨터과학으로 박사 학위를 취득. 1988년 ~ 현재 경북대학교에서 컴퓨터과학과 교수로 재직중. 관심분야는 컴퓨터 언어, 객체중심 컴퓨팅, 시각언어, 멀티미디어와 지식처리임