

# SyncML 서버 응용 개발을 위한 통합 개발 환경

이 지 연<sup>†</sup> · 최 훈<sup>††</sup>

## 요 약

데이터 동기화를 위한 표준 프로토콜인 SyncML은 현재 주소록, 일정 관리와 같은 다양한 응용 서비스 타입에 대한 클라이언트 디바이스와 서버간 동기화를 지원한다. 비록 표준 프로토콜은 제정되었지만, SyncML 응용 서비스 개발자들은 각 응용 서비스의 구체적인 로직을 구현하기 위해서 많은 개발 시간과 노력을 들여야 한다. 본 연구에서는 SyncML 서버 개발자들이 응용 서비스를 신속 정확하게 구현할 수 있도록 SDE(Service Development Environment)라는 통합 개발 환경을 설계하고 구현하였다. SDE는 Sync Library와 SEG(Sync Engine Generator) 툴로 구성된다. 또한 본 연구 결과의 실용성을 입증하기 위해 SDE를 이용하여 SyncML 서버를 구현하였으며, 여러 가지 동기화 시험과 성능 평가를 수행하였다. 본 연구가 SyncML 개발자들이 모바일 응용 서버를 개발하거나 응용 서비스를 추가하는데 보다 쉽고 효과적으로 완수하도록 지원할 것으로 기대된다.

## An Integrated Development Environment for SyncML Server Applications

Jiyeon Lee<sup>†</sup> · Hoon Choi<sup>††</sup>

### ABSTRACT

The SyncML, the standard synchronization protocol, supports the synchronization of various application services between a client and a server such as an address book, a calendar. Even with this standard protocol, SyncML application developers usually spend a long time and efforts implementing service specific logics and databases. This paper designed and implemented the SDE(Service Development Environment) which is an integrated development environment for SyncML server developers to develop an application service rapidly and correctly. The SDE consists of two components i.e., the Sync Library and the SEG(Sync Engine Generator) tool. To prove the applicability of this study we implemented a SyncML server by using the SDE and also carried out the correctness tests and the performance test. We hope this system helps developers implement mobile application services more efficiently.

**키워드 :** SyncML, 데이터 동기화(Data Synchronization), 동기화 응용 서비스(SyncML Application Service), 동기 라이브러리(Sync Library)

### 1. 서 론

무선 인터넷이 발전함에 따라 휴대용 PC(Personal Computer)나 PDA(Personal Data Assistant), 이동 전화기 같은 단말기에서 네트워크 서버의 개인정보관리(PIM : Personal Information Manager) 서비스를 사용하여 주소록 검색, 일정 관리, 전자 메일을 교환하는 것이 개인적인 용도뿐만 아니라 엔터프라이즈용으로 이용되어 비즈니스의 새로운 수단으로 발전하고 있다. 또한 개인이 보유하는 단말기의 수도 증가하여 개인의 동일한 정보가 여러 단말기에 분산되어 존재하는 경우도 많아졌다. 이러한 경우 데이터 동기화(data synchronization) 기능이 필요한데, 이는 다수의 단말

기에 존재하는 동일한 데이터 항목에 대하여 갱신된 내용을 반영하고 데이터간의 버전 차이를 해결하여 가장 최신 값으로 일치시키는 중재 동작을 의미한다.

모바일 산업의 몇몇 선두 업체를 중심으로 하여 SyncML(Synchronization Markup Language) 그룹이 구성되었고 데이터 동기화를 위한 표준 프로토콜인 SyncML이 제정되었다[1, 2]. 현재까지 발표된 SyncML은 동기화 대상으로서 주소록, 일정 관리와 같은 PIM 서비스를 지원하고 있으며, 이 외에도 전자 메일, 텍스트 메모장 등의 다양한 응용 서비스 타입으로 확장될 수 있다. 2002년 11월 기준으로 약 110여 개의 제품이 SyncML 그룹의 적합성 테스트 및 상호연동성 테스트를 통과하였으며, 많은 업체와 기관들이 SyncML 회원으로서 활동하고 있다[1].

전세계적으로 많은 업체와 개발자들이 SyncML에 관심을 가지고 프로토콜 및 응용 서비스 개발을 활발하게 진행

\* 이 연구는 정보통신부 정보통신기초연구사업의 지원을 받아 수행됨.

† 정 회 원 : 한국수자원공사 정보관리실

†† 종 신 회 원 : 충남대학교 전기정보통신공학부 교수

논문접수 : 2003년 7월 7일, 심사완료 : 2003년 12월 15일

하고는 있지만, 프로그램 개발 툴(tool)이나 테스트 기술, 표준 API(Application Program Interface) 정의, 데이터 충돌 시 해결 솔루션 등과 같은 지원 기술 개발은 미약한 실정이다. 현재까지 SyncML 개발을 위한 툴로써 업체에서 개발된 것으로는 Extended Systems사의 XTNDAccess™ SyncML SDK(Software Development Kit)[3]와 Starfish사의 TrueSync® SDK[4]가 있으며, 오픈 프로젝트 그룹에서 개발 중인 것으로는 SyncML SDK 프로젝트[5]와 Lib SyncML 프로젝트[6]가 있다. Extended Systems사의 XTNDAccess™ SyncML SDK는 클라이언트 응용을 개발하기 위한 툴킷이고, TrueSync® SDK는 데스크탑(desktop) 기반의 제품으로서 데이터 동기화 기능을 갖는 모듈을 개발하기 위한 소프트웨어 툴이다. 후자의 SyncML SDK와 LibSyncML 프로젝트는 단지 여러 디바이스 플랫폼에서 PIM 데이터 동기화를 위한 응용 서비스를 개발하기 위해 C와 C++로 스펙의 일부를 구현해 놓은 정도이다. 많은 업체들이 SyncML 개발에 관심을 갖고는 있지만 이에 비해 동기화 서버 또는 클라이언트 개발 툴킷(toolkit)용 솔루션은 매우 적고 다양하지 못하며, 특히 서버 응용 개발을 위한 툴은 더욱 그렇다.

SyncML 규격에 의해 클라이언트와 서버 사이에 동일한 방법으로 데이터 변경 정보를 표현하고 전달할 수 있는 기반이 마련되지만, 각 응용 서비스마다 필요한 동기 시나리오를 구현하거나 룰(rule)에 따라 표준 언어를 사용하여 동기 정보를 기술하는 것은 응용 서비스 개발자가 담당해야 한다. 결국 개발자 입장에서는 하위 계층 프로토콜이 표준화되어도 응용 개발을 위한 환경이 지원되지 않으면 개발에 많은 시간이 소요되며, 구현된 응용 서비스 정보의 호환성이 보장되지 않을 가능성이 있다.

본 논문에서는 이러한 제약을 완화하는데 기여하기 위하여 SyncML 서버 개발자들이 응용 서비스를 신속 정확하게 구현할 수 있도록 SDE(Service Development Environment)라는 통합 개발 환경을 설계하고 구현하였다. 구현된 통합 개발 환경에는 여러 응용 서비스에서 공통적으로 이용되는 데이터 동기 기능을 라이브러리 형태로 제공하고, 응용 서비스의 확장 시에 필요한 많은 작업들을 자동 생성하는 기능의 툴을 제공한다. 본 연구를 위해 우선 이미 구현된 SyncML 서버인 CNU SyncML Server 3.3을 바탕으로 하여, 응용 서비스에 종속적인 부분을 분석한 후 다른 응용 서비스에도 공통적으로 이용되는 부분을 일반화하여 Sync Library로 패키징화 하였으며, 응용 서비스 확장 시 수반되는 많은 일련의 작업들을 정리하고 체계화하여 프로그램 코드를 자동 생성하는 SEG(Sync Engine Generator) 툴을 설계하고 구현하였다. 또한 본 연구 결과의 실용성을 입증하기 위해 SDE를 이용하여 CNU SyncML Server 3.3.1을 구현하였으며, 여러 가지 동기화 시험과 CNU SyncML Server 3.3과의 성능 평가를 수행하였다. 본 연구를 통해

SyncML 개발자들은 응용 서비스 확장 시에 많은 개발 비용을 절감할 수 있을 것으로 기대된다.

본 논문의 2장에서는 관련 연구로서 SyncML과 본 연구팀이 이미 구현한 SyncML 서버인 CNU SyncML Server에 대하여 간단히 소개한다. 3장에서는 CNU SyncML Server 3.3을 바탕으로 응용 서비스 종속적인 부분을 분석하고, SyncML 응용 서비스를 위한 통합 개발 환경인 SDE와 그 구성 요소인 Sync Library와 SEG 툴의 설계 및 구현 내용에 대하여 기술한다. 4장에서는 SDE를 이용하여 개발한 SyncML 동기 서버인 CNU SyncML Server 3.3.1에 대하여 설명하고, 5장에서는 여러 가지 적합성 시험 및 SyncML 클라이언트와의 연동성 시험을 수행함으로써 SDE와 이를 기반으로 구현된 CNU SyncML Server 3.3.1의 검증과 성능 평가 및 분석을 하고 마지막으로 결론을 맺는다.

## 2. 관련 연구

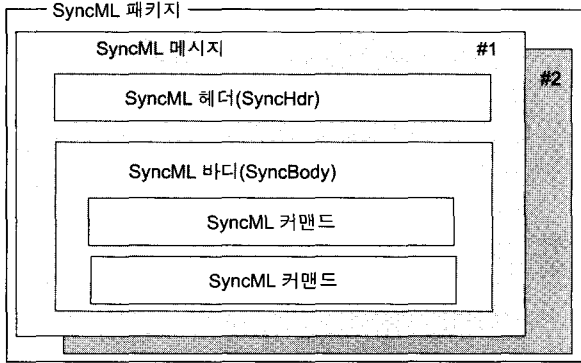
관련 연구로서 데이터 동기화의 표준 프로토콜인 SyncML(Synchronization Markup Language)과, 본 논문의 응용 서비스를 위한 분석 대상 및 성능 평가의 대상이 되며 SyncML 규격을 기반으로 이미 구현한 SyncML 동기 서버인 CNU SyncML Server에 대하여 기술한다.

### 2.1 SyncML

데이터 동기화는 서로 다른 디바이스 및 어플리케이션 간의 동일한 데이터 항목에 대하여 갱신된 내용을 반영하고 데이터간의 버전 차이를 해결하여 가장 최신 값으로 일치시키는 중재 동작을 의미한다. 2000년 2월에 IBM, Lotus, Motorola, Nokia, Palm, Psion, Starfish Software 등의 전 세계 모바일 관련 업체를 중심으로 SyncML 그룹이 구성되었어 데이터 동기화를 위한 표준 프로토콜인 SyncML 규격을 제정하였다[1, 2]. SyncML이란 임의의 네트워크 상에서 서로 다른 디바이스 및 응용 서비스 간에 데이터를 일치시켜 주기 위한 국제 표준 언어이며, XML(Extensible Markup Language) 기반으로 표현된 메시지를 서버와 클라이언트간에 주고 받음으로써 데이터 동기화를 수행하는데 표현 프로토콜(Representation Protocol)[7], 동기화 프로토콜(Synchronization Protocol)[8] 그리고 전송 프로토콜 바인딩(Transport Protocol Binding)[9]으로 구성된다.

SyncML 데이터 표현 프로토콜은 동기화를 위한 XML DTD(Document Type Definition)를 정의하고 있는 SyncML 메시지에 대한 구조체 규약이다((그림 1))[7, 10]. 메시지는 헤더 요소 타입(SyncHdr)과 바디 요소 타입(SyncBody)으로 정의된다[11]. SyncML 헤더에는 SyncML 버전 정보, 인증 정보 및 SyncML 메시지가 전송될 타겟 주소 및 전송을 요구한 소스 주소가 포함된다. SyncML 바디는 하나 또는 그 이상의 SyncML 커맨드를 포함하는 컨테

이러한 역할을 한다. 동기화를 수행하기 위하여 SyncML 메시지는 하나 혹은 그 이상으로 구성될 수 있으며, 이를 SyncML 패키지라고 하는 논리적인 개념으로 정의한다.



(그림 1) SyncML 메시지 구조

SyncML은 데이터 동기화를 위하여 표현 프로토콜에 다양한 커맨드를 정의하고 있다. 크게 동기화 요청을 위한 커맨드와 동기화 처리에 대한 응답 커맨드로 구성된다. 다음의 <표 1>은 SyncML 커맨드의 종류를 기능별로 분류한 것이다.

<표 1> SyncML 커맨드 종류

종류	역할	커맨드
요청 커맨드	데이터 동기화	ADD, REPLACE, DELETE, COPY, MAP, EXEC, SEARCH
	초기화	ALERT
	디바이스 정보	GET, PUT
	컨테이너	ATOMIC, SEQUENCE, SYNC
응답 커맨드		STATUS, RESULTS

SyncML 동기화 프로토콜은 클라이언트와 서버 간의 SyncML 메시지 교환 방법 및 절차, 동기화 수행 방법 그리고 동기화 타입을 정의한다[8]. 동기화 수행을 위하여 변경 기록 정보, Sync Anchor 정보, 데이터 아이템에 대한 식별자 맵핑(mapping) 정보가 중요한 역할을 한다. 각 디바이스는 자신의 데이터베이스 내에서의 변경 사항 정보를 유지함으로써 동기화 수행을 추적할 수 있다. 변경 기록 정보(Change Log Information)는 데이터의 추가, 수정, 삭제를 구분할 수 있는 변경된 타입에 대한 정보를 포함한다. 클라이언트나 서버는 데이터를 동기화 할 때 바로 이전의 마지막 동기 시점을 확인하는 작업이 필요하며, 이를 위해 SyncML은 Sync Anchor 정보를 사용한다. Sync Anchor는 가장 마지막으로 동기화를 마친 시점을 기록하는 'Last' Sync Anchor와 현재 진행 중인 동기화 시점을 알려주는 'Next' Sync Anchor 두 가지가 있다.

SyncML 클라이언트와 서버는 시스템 자원 및 성능의

차이로 동일 데이터 아이템에 대하여 부여하는 식별자(identifier)의 범위가 다를 수 있다. 서버측에서 데이터 아이템에 대하여 사용하는 식별자를 GUID(Global Unique Identifier)라 하고, 클라이언트측에서 사용하는 식별자를 LUID(Locally Unique Identifier)라고 한다. 서버는 동일한 데이터 아이템임을 알기 위하여 각 GUID와 LUID를 맵핑시키기 위한 테이블을 유지해야 한다.

SyncML 동기화 프로토콜은 7가지 동기화 모드를 정의한다. *Two-way Sync* 모드는 서버와 클라이언트가 변경된 정보를 서로 교환하는 일반적인 동기화 종류이며, *Slow Sync* 모드는 *Two-way Sync* 방식의 일종으로서 하나 이상의 데이터베이스 내에 있는 모든 데이터 아이템들은 항목별로 서로 비교한다[9]. 주로 클라이언트나 서버가 변경 기록 정보를 잃어버렸거나 마지막 동기화 시점이 서로 일치하지 않을 경우에 사용된다. *One-way Sync from Client Only Sync* 모드는 클라이언트만 자신의 변경 사항을 서버에게 보내고 서버는 자신의 변경 사항을 클라이언트에게 알리지 않는 동기화 종류이며, *Refresh Sync from Client Only Sync* 모드는 *One-way Sync from Client Only Sync*의 특별한 경우로서 클라이언트에 있는 모든 데이터 아이템을 서버에게 보내는 방식이다. *One-way Sync from Server Only Sync* 모드는 반대로 서버만 자신의 변경 사항을 클라이언트에게 보내고 클라이언트는 자신의 변경 사항을 서버에게 알리지 않는 동기화 종류이며, *Refresh Sync from Server Only Sync* 모드는 *One-way Sync from Server Only Sync*의 특별한 경우로서 서버에 있는 모든 데이터 아이템을 클라이언트에게 보내는 방식이다. *Server Alerted Sync* 모드는 동기화를 수행하기 위해 서버가 먼저 클라이언트에게 동기화를 요청하는 방법이다.

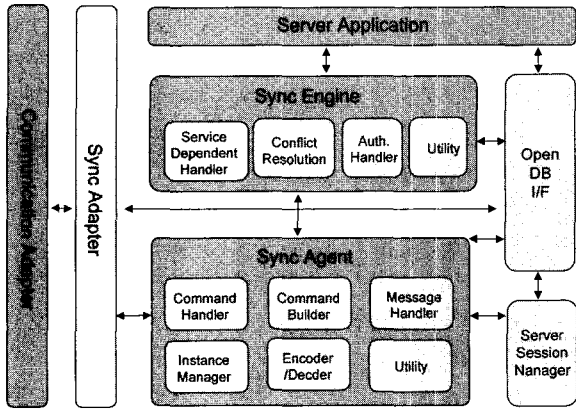
SyncML 전송 프로토콜 바인딩은 메시지를 전송을 위하여 HTTP, WSP(Wireless Session Protocol), OBEX(Object EXchange Protocol) 등과 같은 기존의 전송 프로토콜들을 이용할 수 있도록 바인딩 규칙을 정의한 것이다. 데이터 표현 프로토콜과 동기화 프로토콜이 전송 프로토콜에 독립적이므로 향후 다른 전송 프로토콜과의 바인딩이 가능하다 [9-11].

### 2.2 CNU SyncML Server

본 연구팀은 별도의 연구에서 SyncML 규격을 따르며 다수의 단말기로부터의 다양한 동기화 요구를 처리하고 서버의 변경 사항을 반영할 수 있는 SyncML 서버(CNU SyncML Server)를 개발한 바 있다. CNU SyncML Server는 구조적인 특징에 따라 버전이 2.0부터 3.3까지 몇가지 버전이 구현되었는데 그 중 비교 대상이 되는 CNU SyncML Server 3.3의 구조와 특징에 대하여 설명하도록 한다.

CNU SyncML Server 3.3은 (그림 2)와 같이 총 7개의 프레임으로 구성되어 있는데, Session Manager와 Commu-

nication Adapter는 독립적인 프로세스 형태로 구현되었으며, 나머지 각 프레임은 DLL(Dynamic Link Library)로 구현되었다. SyncML 클라이언트와 서버의 통신은 HTTP(Hyper Text Transfer Protocol) 프로토콜[9]을 이용하였다.



(그림 2) CNU SyncML Server 3.3의 구조

Server Application은 서버측에서 서버 데이터베이스를 접근하여 직접 데이터베이스의 데이터를 변경하거나 서버 내의 정보를 검색하는 등의 작업을 용이하게 하기 위한 GUI(Graphic User Interface) 틀이다. 실제 클라이언트와의 데이터 동기화를 수행하는 Sync Agent 및 Engine 등의 모듈과는 독립적이다.

Communication Adapter 프레임은 클라이언트와 서버간의 전송 바인딩을 지원[10, 12]하는 프레임이다.

Sync Adapter 프레임은 Communication Adapter로부터 SyncML 메시지를 전송 받으며, 메시지 처리를 위한 작업 공간 및 인스턴스를 생성하고 Sync Agent를 호출한다.

Sync Agent 프레임은 SyncML 프로토콜을 구현한 것이며 응용 서비스에 비종속적이다. Sync Agent는 Command Handler, Command Builder, Message Handler, Instance Manager, Encoder/Decoder, Utility 모듈로 구성된다. Command Handler 모듈은 실제적으로 동기화를 수행하는데 있어서의 동작 흐름을 제어하고 있는 모듈로서, 전체적인 동기화 메커니즘을 실행한다. Sync Engine에게 관련 데이터 정보를 넘겨주고 해당 데이터베이스에 데이터를 반영하도록 지시한다. 또한 Session Manager에 커맨드와 처리 상태 정보를 등록하는 기능을 한다. Command Builder 모듈은 STATUS 커맨드와 서버 내의 변경 사항을 알리기 위해 클라이언트에게 보낼 커맨드를 생성한다. Message Handler 모듈은 Command Handler로부터 요구되어진 커맨드에 대한 리스트 관리와 이에 대한 상태 정보 유지를 한다. Instance Manager 모듈은 인스턴스의 생성, 종료, 관리를 담당하며, 다른 프레임들이 인스턴스의 버퍼를 접근하여 검색할 수 있는 인터페이스를 제공한다. Encoder/Decoder 모듈은 클라이언트로부터 받은 메시지를 커맨드 별로 파싱하는

기능과 보낼 메시지를 Representation 프로토콜에 맞게 조립하는 역할을 담당하며, Utility 모듈은 SyncML 메시지 타입의 변환 및 SyncML DTD 구조체에 대한 메모리 해제 등의 유틸리티를 제공한다.

Sync Engine 프레임은 서비스 정책에 종속적이며, Service Dependent Handler, Conflict Resolution, Authentication Handler, Utility 모듈로 구성된다[13]. Service Dependent Handler 모듈은 데이터 동기화 시에 응용 서비스 종속적인 부분을 처리하는 모듈로서 서비스의 확장성을 고려하여 설계되었다. Conflict Resolution 모듈은 동기화를 요청한 디바이스로부터 온 데이터가 서버에 있는 데이터와 충돌할 경우, 이를 해결하기 위한 정책을 구현한다. Authentication Handler 모듈은 동기화를 요청한 디바이스가 서버에 등록되어 있는 정당한 사용자인지를 검증하는 기능을 제공하며, Utility 모듈은 문자열 생성 및 제어 기능, 데이터 토근화 기능과 기타 공통적으로 이용 가능한 유틸리티를 제공하는 모듈이다.

ODBI(Open Database Interface) 프레임은 데이터를 저장하기 위한 데이터베이스와의 인터페이스 역할을 한다[14].

Session Manager 프레임은 동기화를 위한 세션을 유지하고 관리하는 기능을 담당한다. SyncML 서버는 동기화를 위하여 세션이 시작되어서 종료할 때까지 필요한 정보를 유지하고 있어야 하는데 Session Manager 프레임이 이런 역할을 담당한다. Session Manager 프레임은 독립된 프로세스로 구성되며 서버에 데몬(daemon)처럼 항상 실행된 상태로 되어 있다. 또한 동기화 수행에 필요한 세션 정보를 메모리에서 직접 관리하게 된다[15]. 즉, SyncML 서버는 클라이언트로부터 요청 메시지를 받으면 세션 매니저에 새로운 세션을 생성하여 클라이언트로부터 받은 요청 메시지에서 필요한 정보를 등록하고, 서버가 클라이언트에게 보낼 응답 메시지 생성에 필요한 정보들도 관리한다. 클라이언트와 서버 간에 동기화 작업에 필요한 데이터는 세션 매니저로부터 얻을 수 있으며, 모든 동기화 작업이 끝날 때 세션 매니저에 있는 해당 세션의 정보를 삭제하고 세션 종료를 하게 된다.

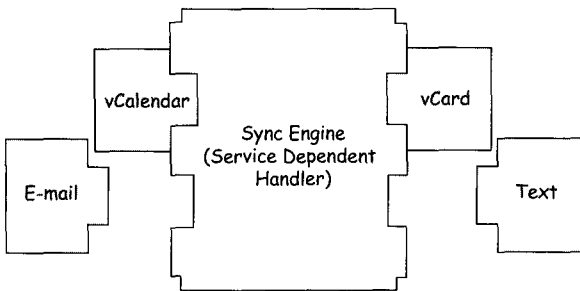
### 3. SDE의 설계 및 구현

서론에서 언급하였듯이 많은 업체나 개발자들이 활발히 SyncML 응용 서비스 개발을 진행하고는 있지만, 프로그램 개발 틀이나 테스트 기술, 표준 API 정의, 데이터 충돌 시 해결 솔루션 등과 같은 기반 기술의 개발은 미약한 실정이다. 이에 본 논문에서는 SyncML 응용 서비스를 신속 정확하게 구현할 수 있는 통합 개발 환경을 설계 및 구현하였다.

#### 3.1 서버의 응용 서비스 종속성 분석

SyncML은 일정 관리나 주소록 서비스 이외에도 전자

메일, 메시지, 텍스트 메모장 등 여러 가지 응용 서비스에 적용할 수 있다. 2장에서 살펴본 CNU SyncML Server 3.3은 vCard 포맷의 주소록 서비스와 vCalendar 포맷의 일정 관리 서비스에 대하여 동기화를 지원하며, 새로운 응용 서비스를 추가하려 할 때 응용 서비스 종속적인 부분을 Sync Engine 프레임의 Service Dependent Handler 모듈에 추가하면 되도록 설계되었다. 즉 SyncML 프로토콜을 구현한 Sync Agent 프레임과는 무관하다. 이는 응용 서비스의 확장성을 고려한 것이며, 적은 비용으로 신속하게 서비스를 적용할 수 있도록 설계된 것이다[13].



(그림 3) 응용 서비스 확장성을 고려한 Sync Engine 프레임

이 구조가 응용 서비스에 대하여 최대한으로 확장성은 고려되어 있지만, 개발자는 서비스를 확장할 때마다 반복적인 일련의 작업을 거쳐야 한다. 예를 들면, 응용 서비스에 대한 데이터베이스 테이블 추가, Sync Engine 프레임과 데이터베이스와 인터페이스하기 위한 API 추가, Sync Agent 프레임과 Sync Engine 프레임의 인터페이스를 위한 API 추가 등이다. 응용 서비스와 관련된 인터페이스 API들은 서비스 종류에 따라 구체적으로 입출력 파라미터, 데이터 구조체, 데이터베이스 테이블의 속성(attribute)들은 다르지만, 궁극적으로 응용 서비스마다 필요한 API들의 기능과 결과는 동일하다. 따라서 각 API들을 기능별로 분류하고 다른 응용 서비스에도 이용 가능하도록 일반화하여 공통적인 기능을 라이브러리화할 수 있다.

CNU SyncML Server 3.3에서 응용 서비스에 영향을 받는 영역을 크게 데이터베이스와 동기를 위한 프로그램 코드 두 가지로 분류할 수 있다. 전자의 경우 “ADDRESS-BOOK”, “CALENDAR” 테이블과 같이 각 응용 서비스의 콘텐츠(contents) 저장에 위한 데이터베이스 테이블과 각 응용 서비스 식별자와 테이블 이름을 데이터로 포함하는 나머지 테이블들이다. 후자의 경우 응용 서비스의 데이터 구조체, 콘텐츠를 데이터베이스 테이블에 저장하기 위한 인터페이스인 ODBI API, 응용 서비스에 대한 커맨드 처리를 위한 Sync Engine 프레임의 Service Dependent Handler 모듈 그리고 기타 유틸리티를 위한 Sync Engine 프레임의 Utility 모듈이다.

다음 (그림 4)는 CNU SyncML Server 3.3의 경우 응용

서비스 종속적인 프로그램 코드의 일부이다. Service Dependent Handler 모듈의 응용 서비스별 데이터를 검색하기 위한 sdhFindEntry() 함수인데, 점선으로 표시된 부분이 바로 응용 서비스 종속적인 코드이다. dbID가 “C”인 경우, 즉 응용 서비스가 일정 관리 서비스일 때 DbCalendarEntry\_t 타입의 데이터 구조체 크기만큼 메모리를 할당하고(9라인), odbiCalFindEntry() 함수를 호출하여 검색된 데이터를 얻고(11라인), pObj 구조체에 응용 서비스의 속성을 지정한다(15라인~17라인). 응용 서비스가 주소록 서비스인 경우에도 데이터 구조체, 호출하는 ODBI API, pObj의 속성 값은 다르지만 같은 방식으로 데이터를 검색한다(28라인~36라인). 따라서 데이터 구조체, ODBI API, pObj의 속성 값을 파라미터화하면 두 응용 서비스 모두 sdhFindEntry() 함수의 기능을 공통적으로 이용할 수 있다.

이러한 방식으로 Sync Engine 프레임의 Service Dependent Handler 모듈과 Utility 모듈을 여러 응용 서비스에서 이용 가능하도록 일반화하고 공통되는 기능을 라이브러리화 할 수 있으며, 응용 서비스 관련된 데이터베이스 테이블, 데이터 구조체 그리고 ODBI API들은 개발자로부터 프로그램 코드 및 테이블 생성에 필요한 입력을 받아 자동 생성할 수 있다.

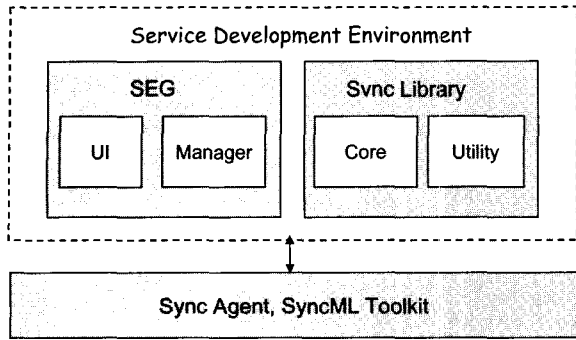
```

1 // This is used to retrieve the data from an application service
2
3 Ret_t ServiceDependentHandler::sdhFindEntry
4 (ODBC_hdbc, HENV henv, String_t UserID, String_t dbID, Short_t deviceID, Long_t guidNum, DataObjectPtr_t pObj)
5 {
6     /* ..... */
7     if (!strcmp ( dbID, "C" ))
8     {
9         DbCalendarEntryPtr_t pCalendarEntry = (DbCalendarEntryPtr_t) malloc (sizeof (DbCalendarEntry_t));
10        rc = odbiCalFindEntry ( hdbc, henv, UserID, guidNum, pCalendarEntry );
11        /* ..... */
12        if (rc == ODBI_OK)
13        {
14            pObj->objectType = OBJECT_SDA_CALENDAR;
15            pObj->length = sizeof (pCalendarEntry);
16            pObj->pContent = (void*) pCalendarEntry;
17        }
18        ret = SML_RESP_OK;
19    }
20    else
21    {
22        /* ..... */
23        if (!strcmp ( dbID, "A" ))
24        {
25            DbAddressEntryPtr_t pAddressEntry = (DbAddressEntryPtr_t) malloc (sizeof (DbAddressEntry_t));
26            rc = odbiAddressFindEntry ( hdbc, henv, UserID, guidNum, pAddressEntry );
27            /* ..... */
28            if (rc == ODBI_OK)
29            {
30                pObj->objectType = OBJECT_SDA_ADDRESS;
31                pObj->length = sizeof (pAddressEntry);
32                pObj->pContent = (void*) pAddressEntry;
33            }
34            ret = SML_RESP_OK;
35        }
36        else
37        {
38            /* ..... */
39        }
40    }
41    return ret;
42 }
43
44
45
46
47
    
```

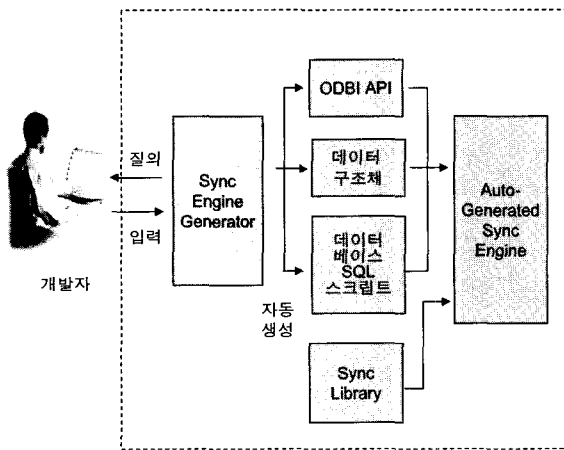
(그림 4) 응용 서비스 종속적인 프로그램 코드(일부)

분석된 결과를 바탕으로 (그림 5)와 같은 구조를 갖는 SDE(Service Development Environment)를 설계하였다. SDE는 SyncML 서버 개발자가 응용 서비스 확장을 위해 필요한 통합 개발 환경이며, 다양한 응용 서비스의 확장에 공통적으로 이용될 수 있는 Sync Library와 코드 및 환경을 자동으로 생성하는 툴인 SEG(Sync Engine Generator)로 구성된다. SEG 툴을 이용하여 코드 생성에 필요한 최소한의 입력을 하면 ODBI API, 데이터 저장 구조체, 데이터

베이스 테이블 관련 SQL 스크립트가 생성된다. 개발자는 생성된 결과와 함께 Sync Library의 API를 이용하여 편리하고 신속하게 Sync Engine을 생성할 수 있다(그림 6).



(그림 5) SDE의 구조



(그림 6) SDE를 이용한 Sync Engine 개발

### 3.2 Sync Library

Sync Library는 여러 응용 서비스에서 공통적으로 이용되는 기능을 (그림 4)에서 설명한 방법으로 일반화하여 라이브러리 형태로 설계하였으며, 기능적으로 동기화를 위한 커맨드 및 데이터 처리를 위한 Core 모듈과 응용 서비스 처리 및 기타 범용적으로 이용 가능한 기능의 Utility 모듈로 구성된다.

Core 모듈은 Sync Engine 프레임의 Service Dependent Handler 모듈의 기능을 일반적 API로 확장한 것이며, 주된 기능은 커맨드 처리(command processing)와 데이터 처리(data processing)이다. 커맨드 처리는 응용 서비스의 동기요구 메시지 중에서 ADD, REPLACE, DELETE, COPY 커맨드에 대한 처리 및 해당 콘텐츠를 검색하는 기능을 한다. 데이터 처리는 vCard, vCalendar 등의 응용 서비스 포맷을 서버가 처리 가능한 데이터 포맷으로 디코딩(decoding)하고 반대의 경우에 인코딩(encoding) 및 응용 서비스 콘텐츠에 대해 항목별로 비교하는 기능을 한다. <표 2>는 Core 클레

스의 주요 메소드와 기능에 대하여 기술한 것이다.

<표 2> Core 클래스의 메소드

메 소 드	설 명
slibInit	데이터베이스 핸들 값, 사용자 ID, 디바이스 ID값을 초기화
cmdFindEntry	응용 서비스 타입의 GUID에 해당하는 엔트리를 검색
cmdAddEntry	응용 서비스 타입에 따라 새로운GUID 번호를 얻어 데이터베이스 테이블에 해당 데이터를 추가 (ADD 커맨드의 처리)
cmdReplaceEntry	응용 서비스 타입에 따라 데이터를 변경 (REPLACE 커맨드의 처리)
cmdDeleteEntry	응용 서비스 타입에 따라 데이터를 삭제 (DELETE 커맨드의 처리)
cmdCopyEntry	응용 서비스 타입에 따라 원본 GUID에 해당하는 데이터 아이템을 새로운 데이터 아이템으로 복사 (COPY 커맨드의 처리)
datEncode	내부 구조체 포맷의 콘텐츠를 응용 서비스 타입 관련 포맷으로 인코딩
datDecode	응용 서비스 타입 관련 포맷으로 인코딩되어 있는 콘텐츠를 내부 구조체 포맷으로 디코딩
datCompare	응용 서비스 타입별 원본 엔트리와 대상 엔트리를 각 항목별로 비교

Utility 모듈은 문자열 생성 및 제어 기능, 응용 서비스별 콘텐츠의 토큰(token)화 기능 및 기타 서버의 다른 프레임에서 범용적으로 이용 가능한 유틸리티 기능을 제공한다. <표 3>은 Utility 클래스의 주요 메소드와 기능에 대하여 기술한 것이다.

<표 3> Utility 클래스의 메소드

메 소 드	설 명
utParseURI	절대 또는 상대 경로로 된 URI로부터 서비스 타입과 LUID값을 얻음
utBuildURI	서비스 타입과 LUID값으로 모드에 적합한 URI를 생성
utGetdbIDfromGuid	소스 주소 값으로부터 GUID값과 서비스 식별자를 분리
xutMakeUTC	로컬 현재 시간을 UTC 타임 포맷으로 변환
utChopNull	문자열에서 널(Null) 문자 제거
utGetToken	주어진 문자열로부터 구분자와 토큰 값을 얻음
utTokenize	서비스 타입의 데이터를 토큰화

Sync Library는 Windows 2000 운영 체제와 C++ 언어와 Microsoft Visual Studio C++ 개발 툴을 이용하여 구현하였다. 정적 라이브러리(static library)로 형태로 구현되었고, SyncML 서버 개발자는 링크하여 쉽게 라이브러리의 API를 이용할 수 있다. Sync Library는 Core 클래스와 Utility 클래스로 구성된다.

다음 (그림 7)은 Core 클래스의 일부 코드이며, 응용 서비스의 데이터 아이템을 검색하기 위한 cmdFindEntry( )

함수이다. getServiceType( ) 함수를 이용하여 정의된 응용 서비스 코드 타입을 얻은(8라인) 후, 서버 개발자에 의해 등록된 해당 응용 서비스의 콜백(callback) 함수를 호출한다(15라인). 그리고 DataObject\_t 타입의 pObj 구조체에 응용 서비스의 속성 값을 지정하는 기능은 호출되어진 ODBI API에서 하도록 하여 특정 응용 서비스의 구조체를 사용하지 않도록 구현하였다. Sync Library의 cmdFindEntry( ) 함수에서는 findFunc 함수 포인터를 사용하여 해당 응용 서비스의 콜백 함수를 호출하기 때문에 특정 응용 서비스 타입에 종속적이지 않다. 또한 ODBI API 함수 인자에 검색된 데이터를 가져오기 위한 특정 응용 서비스 타입의 구조체를 넘기지 않고, DataObject\_t 타입의 pObj를 사용한다. DataObject\_t 타입의 구조체는 응용 서비스 타입, 콘텐츠 크기 그리고 콘텐츠 내용을 멤버 변수로 취하므로 응용 서비스가 주소록이든 일정이든 상관 없이 사용할 수 있다. ADD, REPLACE, DELETE, COPY 커맨드 그리고 데이터 아이템 검색을 위한 처리도 이와 같은 방식으로 특정 응용 서비스 타입에 국한되지 않도록 구현하여 여러 응용 서비스에서 공통적으로 이용 가능하도록 하였다.

```

1 // This is used to retrieve the data item of an application service
2
3 Ret_t CCore::cmdFindEntry(String_t dbID, Long_t guidNum, DataObjectPtr_t pObj)
4 {
5     Ret_t ret = SML_RESP_INIT;
6     Ret_t rc = SML_RESP_INIT;
7
8     slibServiceType_t sType = getServiceType(dbID);
9
10    if(sType == SLIB_UNKNOWN)
11    {
12        return SML_RESP_UNSUPPORTED_MEDIA;
13    }
14
15    rc = pODBI(sType).findFunc(usrInfo.hdc, usrInfo.henv, usrInfo.usrid, guidNum, pObj);
16    if(rc == ODBI_OK)
17    {
18        ret = SML_RESP_OK;
19    }
20    else
21    {
22        pObj = NULL;
23        if(rc == ODBI_NO_ENTRY)
24            ret = SML_RESP_NO_ENTRY;
25        else
26            ret = SML_RESP_COMMAND_FAILED;
27    }
28    return ret;
29 }
30
31 // This is used to get the service type of an application service
32 slibServiceType_t CCore::getServiceType(String_t dbID)
33 {
34     if (!strcmp(dbID, "CALENDAR"))
35         return SLIB_VCALENDAR;
36     else if (!strcmp(dbID, "CARD"))
37         return SLIB_VCARD;
38     else
39         return SLIB_UNKNOWN;
40 }

```

(그림 7) Core 클래스의 구현 코드(일부)

### 3.3 SEG

SEG 툴은 툴의 사용자인 SyncML 개발자로부터 GUI를 통해 최소한의 필요한 사항만 입력 받아 데이터 구조체 및 데이터베이스 테이블 그리고 ODBI API 코드를 생성함으로써 개발자가 SyncML 응용 서비스 프로그램을 빠른 시간 내에 편리하게 개발할 수 있도록 지원한다. 크게 데이터 구조체 및 데이터베이스 테이블 생성 기능과 ODBI API 생성 기능으로 나뉜다.

<표 4>는 데이터 구조체 및 데이터베이스 테이블 생성을 위해 개발자로부터 입력을 요구하는 최소한의 항목으로서 응용 서비스에 종속적인 사항들이다. 응용 서비스의 이

름과 식별자는 공통적으로 필요한 항목이고, 데이터 구조체의 이름 및 구조체 멤버의 속성 그리고 데이터베이스 테이블 이름 및 테이블 속성 정보를 입력받는다.

<표 4> 데이터 구조체 및 데이터베이스 테이블 생성을 위한 요구 항목

요구 항목	데이터 구조체	데이터베이스 테이블
응용 서비스 이름	공통	공통
응용 서비스 식별자	공통	공통
데이터 구조체 이름	○	
데이터 구조체 포인터 여부	○	
구조체 멤버 변수 이름	○	
구조체 멤버 변수 타입	○	
구조체 멤버 변수 길이	○	
테이블 이름		○
테이블 컬럼 이름		○
테이블 컬럼 타입		○
테이블 컬럼 길이		○
기본 값		○
Primary Key 여부		○
Not Null 여부		○

요구 항목에 대한 입력 후 다음 <표 5>의 결과물이 생성된다.

<표 5> 데이터 구조체 및 데이터베이스 테이블 생성 결과물

생성 항목	데이터 구조체	데이터베이스 테이블
로그 파일(.log)	공통	공통
데이터 구조체 정의 파일(.h)	○	
SQL 스크립트 파일(.sql)		○

다음의 <표 6>과 <표 7>은 ODBI API 생성을 위해 필요한 최소한의 입력 요구 항목과 생성 결과물이다. 앞 절의 데이터 구조체 및 데이터베이스 테이블 생성을 위해 입력된 정보를 함께 이용하여 응용 서비스 콘텐츠 정보를 추가, 변경, 삭제, 검색을 위해 필요한 C 언어 기반의 ODBI API 소스 파일과 헤더 파일을 생성한다.

<표 6> ODBI API 생성을 위한 요구 항목

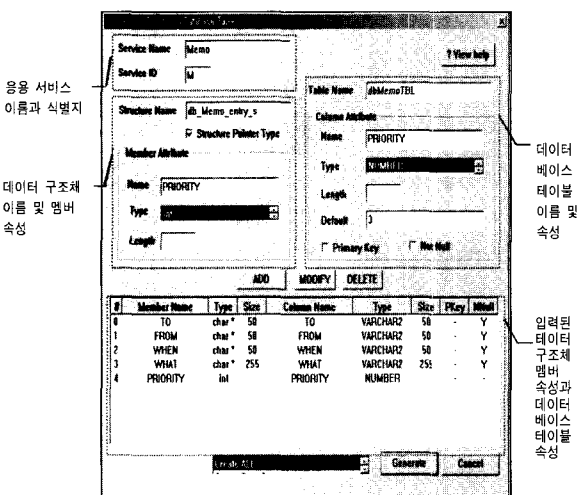
요구 항목
응용 서비스 이름
응용 서비스 식별자
콘텐츠 추가를 위한 함수 생성 여부
콘텐츠 변경을 위한 함수 생성 여부
콘텐츠 삭제를 위한 함수 생성 여부
콘텐츠 검색을 위한 함수 생성 여부

〈표 7〉 ODBI API 생성 결과물

생성 항목
로그 파일(.log)
ODBI API 정의 파일 (.h)
ODBI API 소스 파일 (.c)

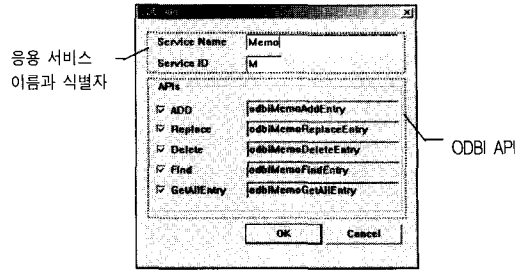
SEG 툴은 Windows 2000 운영 체제와 MFC(Microsoft Foundation Class library) 기반의 C++ 개발 언어와 Microsoft Visual Studio C++ 개발 툴을 이용하여 구현하였다.

다음의 (그림 8)은 데이터 구조체 및 데이터베이스 테이블 생성을 위한 입력 화면이다. 응용 서비스의 이름과 식별자, 데이터 구조체의 이름과 멤버의 속성, 데이터베이스 테이블 이름과 속성을 입력한 후 ADD 버튼을 누르면 아래 리스트로 추가되고, MODIFY 버튼과 DELETE 버튼을 누르면 각각의 속성을 변경하고 삭제할 수 있다. 데이터 구조체 멤버의 속성은 멤버의 이름, 변수의 타입, 변수의 길이로 구성되며, 테이블 속성은 컬럼의 이름, 컬럼의 타입, 컬럼의 길이, 기본 값, 기본 키 지정, Not Null 지정으로 구성된다. 사용자의 편리성을 위해서 주어진 명명 규칙(naming convention)에 따라 응용 서비스 이름을 입력하면 응용 서비스 식별자, 구조체 이름, 데이터베이스 테이블 이름을 자동 생성하여 기본 값으로 보여주고, 데이터 구조체의 멤버 속성을 입력하면 자동으로 상응하는 테이블 속성 값을 생성하여 기본 값으로 보여준다.



(그림 8) 데이터 구조체 및 데이터베이스 테이블 생성을 위한 입력화

ODBI API 생성을 위한 입력 화면은 (그림 9)과 같다. 응용 서비스의 이름(Service Name 필드), 서비스 식별자(Service ID 필드), 각 ODBI API의 생성 여부를 입력 받고 명명 규칙(naming convention)에 따라 각 ODBI API의 이름을 자동 생성하여 기본 값으로 보여준다.



(그림 9) ODBI API 생성을 위한 입력 화면

다음은 메모장 서비스를 위해 SEG 툴을 사용하여 생성된 파일의 내용이다. (그림 10)의 데이터 구조체 정의 파일과 (그림 11)의 SQL 스크립트 파일은 SEG 툴의 데이터 구조체 및 데이터베이스 테이블 생성 기능을 이용한 결과이고, (그림 12)의 ODBI API 소스 파일은 ODBI API 생성 기능을 이용한 결과이다.

```

1 /*
2  Structure Name : db_Memo_entry_s
3  Description   : MEMO service
4  .....*/
5
6 typedef struct db_Memo_entry_s{
7  unsigned long int guidNum;
8  char * TO;
9  char * FROM;
10 char * WHEN;
11 char * WHAT;
12 int PRIORITY;
13 struct db_Memo_entry_s *next;
14 } *DbMemoEntryPtr_t, DbMemoEntry_t;
15
    
```

(그림 10) 생성된 데이터 구조체 정의 파일

```

1 /*
2  TABLE Name: dbMemoTbl
3  .....*/
4
5 DROP TABLE dbMemoTbl CASCADE CONSTRAINTS;
6
7 CREATE TABLE dbMemoTbl (
8  usrID VARCHAR2(20) NOT NULL,
9  guidNum NUMBER NOT NULL,
10 TO VARCHAR2(50) NOT NULL,
11 FROM VARCHAR2(50) NOT NULL,
12 WHEN VARCHAR2(50) NOT NULL,
13 WHAT VARCHAR2(255) NOT NULL,
14 PRIORITY NUMBER,
15
16 CONSTRAINT IDX_dbMemoTbl
17 PRIMARY KEY (usrID, guidNum)
18 );
    
```

(그림 11) 생성된 SQL 스크립트 파일

```

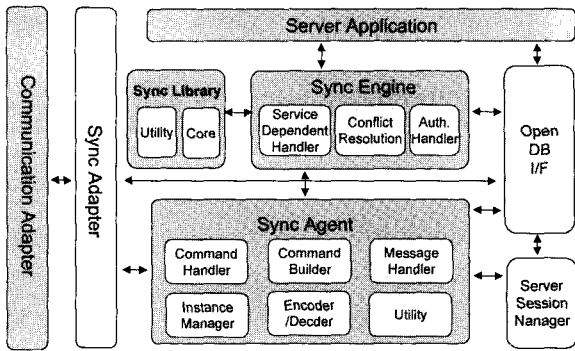
16 #include <stdio.h>
17
18 .....*/
19 // Import low soft MemodEntry
20 // Required low Add Entry to Mem
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2
```



#### 4. CNU SyncML Server 3.3.1의 설계 및 구현

구현된 SDE의 기능 검증을 위하여 우선 Sync Library와 SEG 툴을 이용하여 동기 서버인 CNU SyncML Server 3.3.1을 구현하였다. CNU SyncML Server 3.3.1의 구조는 CNU SyncML Server 3.3과 유사하며 Sync Engine 프레임의 Service Dependent Handler 모듈에서 응용 서비스 중속적인 처리를 위하여 Sync Library의 API를 이용하고, 기존의 Sync Engine 프레임의 Utility 모듈은 Sync Library의 Utility 모듈의 API를 이용한다.

CNU SyncML Server 3.3.1은 Windows 2000 운영 체제, C와 C++ 언어, Microsoft Visual Studio C++ 개발 툴, Oracle 8i 데이터베이스를 이용하여 개발되었다.



(그림 13) CNU SyncML Server 3.3.1의 구조

CNU SyncML Server 3.3.1에서 제공하는 응용 서비스는 버전 3.3 서버와 동일한 주소록과 일정 관리 서비스이다. (그림 8)의 데이터 구조체 및 데이터베이스 테이블 생성을 위한 입력 화면과 (그림 9)의 ODBI API 생성을 위한 입력 화면에서 두 응용 서비스에 대한 정보를 입력한다. 응용 서비스 이름과 식별자에 대하여 주소록 서비스는 "ADDRESS-BOOK", "A"를 입력하고, 일정 관리 서비스는 "CALENDAR", "C"로 입력한다.

개발자는 Sync Engine 프레임에서 Sync Library의 API를 사용하기 전에 먼저 자신의 동기 서버의 ODBI API 중 변경 기록을 위한 것과 응용 서비스에 관련된 것들을 Sync Library의 Core 모듈에 콜백 함수(callback function)로 등록해 주어야 한다. Sync Library는 특정 응용 어플리케이션의 구조에 영향 받지 않으며 일반적으로 사용 가능한 라이브러리므로, 이 절차는 해당 처리를 위해 자신의 동기 서버의 ODBI API가 호출될 수 있도록 하기 위한 것이다. (그림 14)는 Sync Engine 프레임의 Service Dependent Handler 모듈 중 sdhLibRegister() 함수의 일부 코드이다. sdhLibRegister() 함수는 ServiceDependentHandler 클래스의 생성자 함수에서 호출되며, 자신의 동기 서버의 변경 기록을 위한 ODBI API와 주소록과 일정 관리 서비스의 ODBI API를 콜백 함수로 등록한다.

```

1 // This is used to register the ODBI APIs at Core class
2
3 void ServiceDependentHandler::sdhLibRegister()
4 {
5     // To register the ODBI APIs for ChangeLog
6
7     core.pComODBI.ciGetNewIdFunc = &odbiGetNewGuidNum;
8     core.pComODBI.ciFindFunc = &odbiChgLogFindEntry;
9     core.pComODBI.ciAddFunc = &odbiChgLogAddEntry;
10    core.pComODBI.ciUpdateFunc = &odbiChgLogUpdateEntry;
11    core.pComODBI.ciDeleteFunc = &odbiChgLogDeleteEntry;
12    core.pComODBI.ciGetAllFunc = &odbiChgLogGetAllEntry;
13    core.pComODBI.ciGetAll4DBFunc = &odbiChgLogGetAll4DBEntry;
14    core.pComODBI.ciGetChangedFunc = &odbiChgLogGetChanged;
15    core.pComODBI.ciDeleteAllFunc = &odbiChgLogDeleteAllEntry;
16    core.pComODBI.ciDeleteAll4DBFunc = &odbiChgLogDeleteAll4DBEntry;
17    core.pComODBI.ciFindOriginIdFunc = &odbiChgLogFindbyOriginGuidEntry;
18
19    // To register the ODBI APIs for AddressBook service
20
21    core.pODBI[SLIB_VCARD].findFunc = &odbiAddressFindEntry;
22    core.pODBI[SLIB_VCARD].getAllFunc = &odbiAddressGetAllEntry;
23    core.pODBI[SLIB_VCARD].addFunc = &odbiAddressAddEntry;
24    core.pODBI[SLIB_VCARD].replaceFunc = &odbiAddressUpdateEntry;
25    core.pODBI[SLIB_VCARD].deleteFunc = &odbiAddressDeleteEntry;
26
27
28    // To register the ODBI APIs for Calendar service
29
30    core.pODBI[SLIB_VCALENDAR].findFunc = &odbiCalFindEntry;
31    core.pODBI[SLIB_VCALENDAR].getAllFunc = &odbiCalGetAllEntry;
32    core.pODBI[SLIB_VCALENDAR].addFunc = &odbiCalAddEntry;
33    core.pODBI[SLIB_VCALENDAR].replaceFunc = &odbiCalUpdateEntry;
34    core.pODBI[SLIB_VCALENDAR].deleteFunc = &odbiCalDeleteEntry;
35 }
    
```

(그림 14) Core 모듈에 동기 서버의 ODBI API 등록

다음 (그림 15)는 Service Dependent Handler 모듈에서 응용 서비스의 커맨드별 처리를 위해 Sync Library의 Core 모듈의 API를 호출한 코드의 일부이다. ADD 커맨드에 대한 처리를 위한 sdhAddEntry() 함수를 보면, 데이터베이스 핸들 값, 사용자 ID, 디바이스 ID 정보로 우선 초기화(8라인)한 후 Core 클래스의 cmdAddEntry() 함수를 호출하여(9라인) 응용 서비스 컨텐츠의 추가 후 처리 결과 값을 받기만 하면 된다. 실제적으로 Core 클래스의 cmdAddEntry() 함수는 컨텐츠를 응용 서비스의 데이터베이스 테이블에 추가하고, 변경 기록을 위하여 이미 콜백 함수로 등록된 ODBI API를 실행함으로써 ADD 커맨드의 처리를 수행한다.

응용 서비스의 컨텐츠 데이터를 검색하기 위한 함수인 sdhFindEntry() 함수를 보면, 사용자 디바이스의 정보를 초기화하고(33라인), Core 클래스의 cmdFindEntry() 함수를 호출하여(34라인) 처리 결과 값을 받기만 하면 된다. 이는 개발자가 각 응용 서비스에 대하여 직접 sdhFindEntry() 함수를 구현한 (그림 4) 비교할 때 코드가 매우 간소화 되었음을 알 수 있다.

```

1 // This is used to add the contents data
2 Ret_t ServiceDependentHandler::sdhAddEntry
3 (HDBC hdbc, HENV henv, String_t UserID, String_t dbID, Short_t deviceID, DataObjectPtr_t pObj, Long_t *guidNum)
4 {
5     Ret_t ret = SML_RESP_INIT;
6     core.slibInit(hdbc, henv, UserID, deviceID);
7     ret = core.cmdAddEntry(dbID, guidNum, pObj);
8     return ret;
9 }
10
11 // This is used to register the contents data
12 Ret_t ServiceDependentHandler::sdhReplaceEntry
13 (HDBC hdbc, HENV henv, String_t UserID, Long_t guidNum, Short_t deviceID, String_t dbID, DataObjectPtr_t pObj)
14 {
15     Ret_t ret = SML_RESP_INIT;
16     core.slibInit(hdbc, henv, UserID, deviceID);
17     ret = core.cmdReplaceEntry(dbID, guidNum, pObj);
18     return ret;
19 }
20
21 // This is used to retrieve the entry
22 Ret_t ServiceDependentHandler::sdhFindEntry
23 (HDBC hdbc, HENV henv, String_t UserID, String_t dbID, Short_t deviceID, Long_t guidNum, DataObjectPtr_t pObj)
24 {
25     Ret_t ret = SML_RESP_INIT;
26     core.slibInit(hdbc, henv, UserID, deviceID);
27     ret = core.cmdFindEntry(dbID, pObj);
28     return ret;
29 }
30
31 // This is used to compare the contents data of two entries field by field
32 Ret_t ServiceDependentHandler::sdhCompare
33 (DataObjectPtr_t pObjEntry, DataObjectPtr_t pObjCompareEntry, String_t dbID)
34 {
35     Ret_t ret = SML_RESP_INIT;
36     ret = core.dtcCompare(dbID, pObjEntry, pObjCompareEntry);
37     return ret;
38 }
    
```

(그림 15) Sync Library의 API를 이용한 Service Dependent Handler 모듈

### 5. 검증 및 성능 분석

SDE에 대한 검증을 위하여 이를 이용하여 개발한 CNU SyncML Server 3.3.1의 SyncML 적합성 테스트(conformance test) 및 SyncML 클라이언트와 상호 연동성 테스트(interoperability test)를 수행하였다. 그리고 SDE를 사용하지 않고 구현한 동일 기능의 서버 3.3과 기능 및 성능을 비교 평가 하였다.

#### 5.1 SDE의 검증

검증은 크게 적합성 시험 그리고 SyncML 클라이언트와의 연동 시험으로 진행되었다. SEG 틀을 이용하여 추가하려는 응용 서비스 관련된 ODBI API, 데이터 구조체, 데이터베이스 테이블을 생성한 후 결과물들과 Sync Library를 이용하여 Sync Engine 프레임의 Service Dependent Handler 모듈을 완성한다. 검증 단계 1은 구현된 CNU SyncML Server 3.3.1에 대하여 SyncML의 적합성 시험을 하는 단계이다. 적합성 시험은 구현된 SyncML 서버가 올바르게 동작하는지를 확인하기 위한 작업으로서, SyncML 표준 규격을 바탕으로 구현한 시스템을 대상으로 하여 SyncML 규격의 일부인 SIC(SyncML Interoperability Committee)에서 제시하는 테스트 절차를 시험 시스템에 적용하여 그 결과를 예측된 결과와 비교 분석하여 프로토콜 구현의 적합성 여부를 판별하는 것이다[16].

검증 단계 2는 실제 SyncML 클라이언트와 구현한 SyncML 서버를 동기화 시나리오에 따라 연동 시험을 하는 단계이다. 연동 대상으로서 선택한 SyncML 클라이언트는, 본 연구팀이 이미 구현하여 SyncML 적합성 시험과 CNU SyncML Server 3.3과의 연동 시험을 통과한 시스템이다. CNU SyncML Server 3.3.1과 SyncML 클라이언트와의 동기화 시험은 Server Alerted Sync를 제외한 6가지 동기화 모드와 다양한 동기화 커맨드를 포함하는 시나리오를 가지고 수행되었다.

다음의 <표 8>은 검증 단계 1의 적합성 시험 결과에 대한 표이다. SyncML에서 제시하는 적합성 시험 절차[16, 17]대로 12가지 테스트를 수행하였으며, 서버에 해당하는 항목에 대하여 모두 적합성 통과 기준과 일치하는 결과를 얻었다.

CNU SyncML Server 3.3.1에 대하여 적합성 시험과 동기화 연동 시험을 수행하여 정상적이며 성공적인 결과를 확인하였다. CNU SyncML Server 3.3.1이 SDE를 이용하여 개발되었으므로, 이 검증은 SDE에 대한 검증 결과라고 할 수 있다.

<표 8> 적합성 시험의 결과

테스트 내용	결과	비고
① Two-way with Client & Empty Server	Pass	
② Two-way with Client Add & Server Add	Pass	
③ Two-way with Client Replace & Server Replace	Pass	
④ Two-way with Client delete & Server Delete	Pass	
⑤ Two-way with Client Add (shows empty server sync data)	Pass	
⑥ Two-way with Server Add (shows empty client sync data)	Pass	
⑦ Two-way with Server with large amount of data(shows multiple message)	Pass	
⑧ Two-way with Client with large amount of data(shows multiple message)	Pass	
⑨ Two-way with Server responding busy	Pass	
⑩ Two-way with Server not responding	Pass	클라이언트 해당
⑪ Two-way with communication broken during sync	Pass	
⑫ Two-way Slow Sync	pass	

#### 5.2 성능 평가 및 분석

CNU SyncML Server 3.3.1(그림 13)은 CNU SyncML Server 3.3(그림 2)의 기본 구조를 바탕으로 하였고, SDE를 이용하여 구현되었다. 서버를 구성하는 7개의 프레임 중에서 Sync Engine만 SDE의 Sync Library를 이용하여 구현되었으므로 인터페이스가 CNU SyncML Server 3.3와 약간 다르며, SEG틀을 이용하여 자동 생성된 소스 코드는 동일하다. 따라서, 본 논문에서는 SyncML 서버를 구현하는데 있어서 SDE를 이용하는 경우의 시간적·경제적 효율성을 입증하기 위하여, SyncML Server 3.3.1과 동일한 성능 및 기능을 갖는 CNU SyncML Server 3.3을 비교 대상으로 하였다.

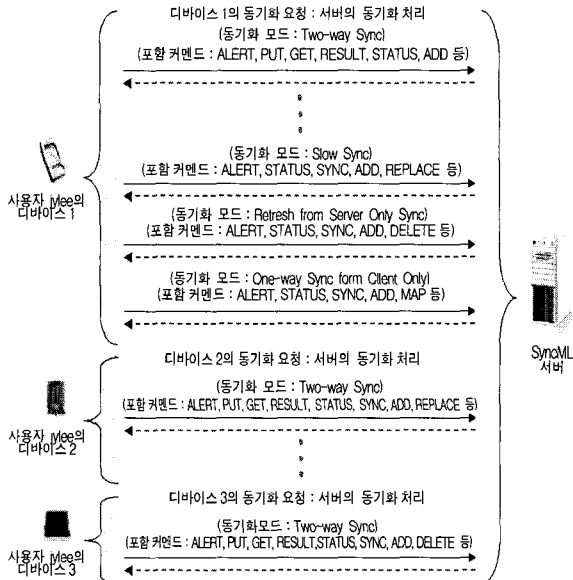
두 동기 서버의 성능 측정은 동일한 테스트 환경 하에서 동일한 동기화 시나리오로 수행하였으며, 동기화 처리를 위한 서버의 평균 수행 처리 속도에 대하여 측정하였다. 이때, 처리 시간에 대한 측정은 SDE를 이용하여 구현된 Sync Engine 프레임과 ODBI 프레임을 포함해야 한다. 따라서, Sync Agent 프레임에서 응용 서비스와 관련된 처리를 위하여 Sync Engine 프레임과 ODBI 프레임을 호출하기 전부터 동기화 처리 후까지를 측정 범위로 하였다.

테스트는 Pentium 4 1.7GHz의 CPU, 256MB 메모리, 60GB 하드 디스크 사양의 시스템에서 수행되었다. <표 9>는 동기화 시나리오의 파라미터들을 기술한 것이다. 동기화 시나리오는 6가지 동기화 모드를 포함하며 동기화 메시지 안에는 ADD, REPLACE, DELETE 커맨드들과 해당 디바이스의 최초 접근 시 수행되는 PUT, GET 커맨드와 기타 커맨드 등을 골고루 포함한다.

<표 9> 성능 측정을 위한 동기화 시나리오의 파라미터

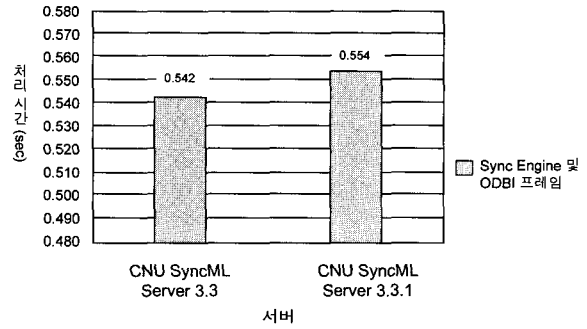
파라미터	내 용
사용자 수	10명
디바이스 수	사용자당 2개 이상
동기화 횟수	3000번
동기화 모드	Two-way Sync Slow Sync One-way Sync from Client Only Sync One-way Sync from Server Only Sync Refresh Sync from Client Only Sync Refresh Sync from Server Only Sync

(그림 16)은 수행된 동기화 시나리오의 예제를 도시화한 것이다. "jylee" 사용자가 디바이스 세 개를 가지고 SyncML 서버와 동기화를 수행하는 경우이다.



(그림 16) 동기화 시나리오 예제

두 동기 서버에 대하여 동기화 처리에 대한 평균 수행 속도를 측정하여 다음 (그림 17)의 그래프와 같은 결과를 얻었다. CNU SyncML Server 3.3.1은 응용 서비스의 동기 처리에 대하여 평균 0.554초의 수행 시간이 소요되며, CNU SyncML Server 3.3에 비하여 2%의 처리 시간이 더 소요된다. 이것은 CNU SyncML Server 3.3에 비해서 서버 3.3.1의 경우 Sync Engine 프레임에서 Sync Library의 Core 모듈, Utility 모듈과 추가적인 호출이 필요하기 때문에 발생한 결과이다. 즉 Sync Library의 모듈들과의 추가적인 인터페이스로 인한 오버헤드(overhead)이다. 그러나, 이 결과는 동기화 수행 처리 속도에 있어서 미소한 차이이기 때문에 CNU SyncML Server 3.3.1과 CNU SyncML Server 3.3은 성능의 차이가 거의 없다고 간주할 수 있다.



(그림 17) CNU SyncML Server의 성능 측정

두 동기 서버의 응용 서비스 확장을 위해 소요되는 개발 비용을 해당 소스 코드의 크기를 통해 비교해 보았다.

다음의 <표 10>은 두 서버의 경우 주소록, 일정 관리 서비스를 구현하기 위하여 서버 개발자가 직접 구현해야 하는 소스 코드의 크기를 비교한 것이다. CNU SyncML Server 3.3의 경우에 두 응용 서비스의 구현을 위한 Service Dependent Handler 모듈의 소스 크기는 총 920라인이며, CNU SyncML Server 3.3.1은 이보다 약 1/3정도로 총 290라인이다. 이는 많은 부분이 Sync Library의 API를 단지 호출하는 것만으로 해결되기 때문이다. ODBI 프레임의 API를 구현하기 위해 두 서버 모두 총 2,070라인이 필요하지만, CNU SyncML Server 3.3.1은 SEG 툴을 이용하여 약간의 정보 입력만으로 소스가 자동 생성되므로 실제적으로 구현을 위한 개발자의 노력은 많이 줄어든다. SEG 툴을 이용하여 정보를 입력해야 하는 노력을 Service Dependent Handler 모듈을 구현하는 노력과 같은 비율로 보고 작업량을 비교해 보면, CNU SyncML Server 3.3은 총 2,990라인이고, CNU SyncML Server 3.3.1은 총 580라인이므로 약 80%의 작업량이 감소된 것이다.

<표 10> 소스 코드 크기의 비교

(단위 : 개발자가 직접 구현해야 하는 라인(line) 수)

개발 항목	서버	CNU SyncML Server 3.3	CNU SyncML Server 3.3.1
Sync Engine 프레임의 Service Dependent Handler 모듈		920	290
ODBI 프레임의 API		2,070	2,070(자동 생성)

## 6. 결 론

전세계적으로 SyncML 프로토콜 개발은 활발히 진행되고 있는데 비하여, 개발을 위한 기반 기술은 미약한 환경이다. 그리고 SyncML이 현재까지는 주로 개인정보관리 서비스를 대상으로 하고 있지만, 앞으로 개인 및 기업의 정보를 위한 다양한 응용 서비스로 그 대상이 확장될 것이다.

본 논문은 SyncML 서버 개발자들이 응용 서비스를 신속 정확하게 구현할 수 있도록 통합 개발 환경인 SDE를 설계하고 구현하였다. 이 통합 개발 환경은 여러 응용 서비스에 공통적인 기능을 일반화하여 라이브러리로 구현한 Sync Library와 응용 서비스 확장에 필요한 코드 및 환경을 자동으로 생성하는 SEG 툴로 구성된다. 구현된 통합 개발 환경인 SDE를 이용하여 SyncML 동기 서버인 CNU SyncML Server 3.3.1을 개발하여 SyncML 적합성 시험과 SyncML 클라이언트와 동기화 연동 시험을 수행함으로써, CNU SyncML Server 3.3.1의 검증과 더불어 SDE에 대한 검증도 확인하였다. 검증된 CNU SyncML Server 3.3.1의 성능 평가를 위하여 비슷한 구조의 동기 서버인 CNU SyncML Server 3.3과 동일한 환경 하에서 주어진 시나리오로 응용 서비스의 동기화 수행에 소요되는 평균 처리 시간을 측정 한 결과, CNU SyncML Server 3.3.1이 약 2%의 수행 처리 속도가 더 소요되었다. 그러나, 이는 매우 미소한 차이이므로 두 동기 서버의 성능은 크게 차이가 없다고 간주할 수 있다. SyncML 서버 개발자가 응용 서비스 확장을 위하여 SDE를 이용하여 개발하는 경우, 성능면에서는 동일 시나리오에 대하여 약간의 수행 속도 저하가 있지만 이는 매우 근소한 차이이므로 성능에 영향을 주지 않는다고 볼 수 있다.

또한 개발에 있어 많은 작업이 자동 생성되고 편리하게 Sync Library의 API를 이용할 수 있으므로, 시간적·경제적으로 많은 개발 비용을 절감할 수 있으며, 편리하고 프로그래밍 오류를 줄일 수 있어서 매우 효율적이다.

향후 연구로서 SDE를 이용하여 개발된 동기 서버의 성능을 최적화할 수 있는 방안을 연구하고, SyncML 개발자에게 보다 큰 편리성을 제공할 수 있도록 자동 생성 툴인 SEG의 기능 보강을 고려할 수 있다.

**참 고 문 헌**

[1] SyncML Initiative, <http://www.syncml.org>  
 [2] SyncML Initiative, Building an Industry-Wide Mobile Data Synchronization Protocol, SyncML White Paper, Mar., 2000.  
 [3] Extended Systems, <http://www.extendedsystems.com>.  
 [4] Starfish, <http://www.starfish.com>.  
 [5] SOURCEFORGE, <http://sourceforge.net/projects/syncmlsdk>.  
 [6] SOURCEFORGE, <http://sourceforge.net/projects/libsyncml>.  
 [7] SyncML Initiative, SyncML Representation Protocol, version 1.0.1, Jun., 2001.  
 [8] SyncML Initiative, SyncML Synchronization Protocol, version 1.0.1, Jun., 2001.  
 [9] SyncML Initiative, SyncML HTTP Binding, version 1.0.1,

Jun., 2001. SyncML Initiative, SyncML Architecture, version 0.2, May., 2000.

[10] SyncML Initiative, SyncML Architecture, version 0.2, May, 2000. SyncML Initiative, SyncML HTTP Binding, version 1.0.1, Jun., 2001.  
 [11] 이병윤, 이길행, 조진현, 류수희, 최 훈, “세션매니저를 이용한 SyncML 동기화 시스템 설계 및 구현”, 한국정보과학회논문지, 제8권 제6호, pp.647-656, Dec., 2002.  
 [12] SyncML Initiative, SDA2 Specification, version 0.2, Aug., 2000.  
 [13] 조진현, 최 훈, 김경용, “SyncML 서버 데이터 동기엔진의 설계 및 구현”, 한국정보과학회 가을학술발표논문집, 제 28권 제2호, pp.580-582, Oct., 2001.  
 [14] 이지연, 조진현, 최 훈, “SyncML 데이터 동기를 위한 데이터 베이스 설계 및 구현”, 한국정보처리학회 추계 학술발표논문집, 제8권 제2호, pp.1343-1346, Oct., 2001.  
 [15] JiYeon Lee, ChangHoe Kim, Hoon Choi, “Implementation of the Session Manager for a Stateful Server,” IEEE TENCON, Beijing, China, Oct., 2002.  
 [16] SyncML Initiative, SyncML Conformance Test Suite version 0.2, Jan., 2001.  
 [17] SyncML Initiative, SyncML Manual Test Cases version 0.3, Feb., 2001.

**이 지 연**



e-mail : eunbi@kowaco.or.kr  
 1998년 충남대학교 컴퓨터공학과 학사  
 2003년 충남대학교 대학원 컴퓨터공학과 석사  
 1998년~2001년 삼성전자 반도체 총괄 근무

2003년~현재 한국수자원공사 정보관리실 근무  
 관심분야 : SyncML, 모바일 컴퓨팅, 분산 시스템, 웹 서비스, 데이터베이스 등

**최 훈**



e-mail : hc@cnu.ac.kr  
 1983년 서울대학교 전자계산기공학과 학사  
 1990년 Duke University 전산학과 석사  
 1993년 Duke University 전산학과 박사  
 1983년~1996년 한국전자통신연구원 선임 연구원

1996년~현재 충남대학교 전기정보통신공학부 컴퓨터전공 부교수  
 2000년 미국 NIST(National Institute of Standards and Technologies) 객원연구원  
 관심분야 : 모바일 컴퓨팅, 분산 시스템, Fault-tolerant 시스템