제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

SyncML 적합성 시험도구 개발

배장환, 정인혜, 한재일
국민대학교 컴퓨터학부
e-mail: arkray@cs.kookmin.ac.kr, inhae@cs.kookmin.ac.kr,
jhan@kookmin.ac.kr

Development of a SyncML Conformance Test Suite

Jang-Hwan Bae, In-Hae Jung, Jae-Il Han
School of Computer Science, Kookmin University

요 약

모바일 인터넷 환경에서 동일한 데이터가 다양한 단말 장치에 산재되어 사용되는 경우 효과적인 사용과 관리를 위해 산재된 데이터가 서로 동기화(synchronization) 즉 일치되어야 한다. SyncML은 모바일 인터넷 환경에서 다양한 플랫폼을 가진 기기 사이의 데이터 동기화를 위해 제시된 표준 프로토콜로서 많은 국·내외 산업체 및 연구기관이 SyncML 규격을 구현한 제품을 선보이고 있다. 이러한 SyncML 구현 제품이 이질적인 플랫폼에 상관없이 상호운용되기 위해서는 SyncML 규격에 대한 규격 준수 시험이 필요하다. 본 논문은 SyncML 규격의 적합성 시험(conformance test)을 위한 시험도구의 설계와 구현에 대하여 논한다.

1. 서론

SyncML[1,2,3,4,5]은 모바일 인터넷 환경에서 다양한 단말 장치에 산재되어 있는 데이터들을 동기화 시키기 위해 SyncML 컨소시엄에서 제시한 개방형 데이터 동기화 프로토콜이다. SyncML 규격이 발표된 이후 많은 국·내외 연구기관 및 업체가 SyncML 규격을 구현한 제품을 선보이고 있다. SyncML 컨소시엄은 데이터 동기화 프로토콜에 대한 규격만을 정의하고 있을 뿐 구현방법은 각 업체에 위임하고 있으며 적합성(conformance)과 상호운용성(interoperability)에 대한 시험을 수행하여 이들 제품에 대한 SyncML 컨소시엄의 인증을 부여한다. 이렇게 인증을 받은 제품은 서로 다른 플랫폼, 네트워크, 응용서비스에 상관없이 상호 운용됨을 보장받게 된다. 따라서 SyncML 규격에 대한 적합성을 객관적으로 검증할 수 있는 시험도구가 필요하였으나, 이러한 시험도구의 부재로 극히 최근까지도 각 개발자가 SyncML 컨소시엄의 시험지침[6]에 따라 내부적인 시험을 수행하는 상황이었다. SyncML 컨소시엄에서는 SyncML 규격의 적합성을 시험할 수 있는 시험도구 개발을 추진하여 2002년 7월에 SyncML *1.0.1* 버전에 대한 시험도구 SCTS (SyncML Conformance Test Suite) 3.0을, 8월에는 1.1 버전에 대한 시험도구 SCTS 3.1을 개발하였다.

본 논문은 SyncML 규격에 대한 적합성을 시험할 수 있는 시험도구의 설계 및 구현에 대하여 기술한다. 본 논문의 구성은 다음과 같다. 2장은 SyncML 규격 검증을 위해 필요한 시험 요구사항을 살펴보고, 3장에서는 시험도구의 개략적 설계 개념에 대해 논한다. 4장에서는 분산시험사례의 구조 및 프로세스 모델에 대하여 기술하고, 5장은 단순화된 시험사례의 예제 및 시험사례 실행결과를 보인다.

2. SyncML 규격 시험 요구사항

SyncML 규격은 크게 SyncML 표현 프로토콜과, SyncML 동기 프로토콜, 그리고 SyncML HTTP/WSP/OBEX 바인딩으로 나누어져 있다. SyncML 적합성 시험은 위의 규격들의 준수여부에 대한 시험을 대상으로 하기 때문에, 각 규격들을 분석하여 시험 요구사항을 추출해내야 한다. SyncML 규격은 SCR(Static Conformance Requirements)를 사용하여 구현에 대한 강제력을 네가지(MUST, MAY, SHOULD, MUST NOT)로 구분하고 있는데, 본 연구에서는 MUST로 명시되어 있는 규칙들을 제대로 구현했는지에 대해서만 시험요구사항을 추출하여 시험대상으로 삼았다.

다음은 SyncML 표현 프로토콜(representation protocol)의 Add 명령어 엘리먼트에 대한 시험 요구사항을 추출한 간단한 예이다.

- Add : 발신자가 자신이 제공하는 데이터 엘리먼트를 수신자의 동기 데이터에 반드시 추가되도록 요구하는 것을 허용. 이 명령어는 반드시 Sync 명령어 내에 명시되어야 한다.

=> 시험요구사항
- 발신자가 수신자에게 Add 명령어의 실행을 요청하였을 때 발신자가 제공하는 데이터 엘리먼트가 수신자의 동기 데이터에 추가되는지의 여부를 확인.
- Add 명령어가 Sync 명령어 내에 명시되었는지 점검.

=> 시험고려사항
- 클라이언트는 Add 명령어의 수신을, 서버는 Add 명령어의 발신과 수신을 반드시 지원하여야 한다.
- 클라이언트의 Add 명령어 발신은 권고사항임.

3. SyncML 적합성 시험도구 설계

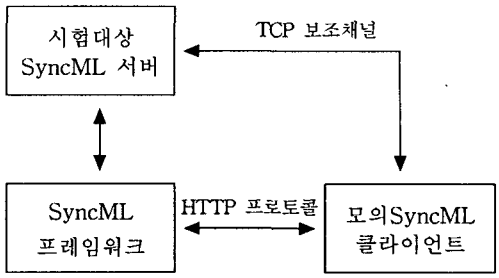
본 절에서는 SyncML 규격 검증 시험도구의 설계 개념을 일반적인 설계 방향, SyncML 표현 프로토콜의 시험방법, SyncML 동기 프로토콜의 시험방법 그리고 SyncML HTTP 바인딩의 시험방법으로 나누어 기술한다.

3.1 개요

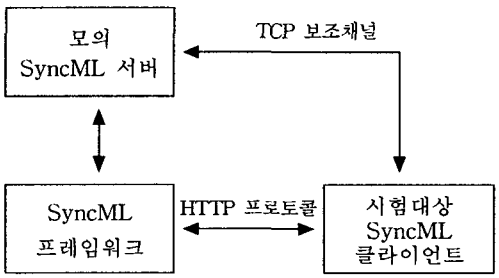
SyncML을 통한 데이터 동기는 SyncML 클라이언트, SyncML 서버, 동기엔진, 동기 클라이언트 대리자(agent), 동기 서버 대리자 등과 같은 여러 구성요소가 참여하며, 이 구성요소의 실행을 위해 SyncML 메시지 발신자 프로세스와 수신자 프로세스를 포함하여 최소 두 개 이상의 프로세스가 필요하다. 따라서 SyncML 적합성 시험을 위한 시험사례는 비동기적으로 통신하는 여러 프로세스로 구성되는 분산 시험사례로 작성되어야 한다.

분산시험사례는 실행 시에 여러 프로세스 (또는 객체) 사이의 동기(synchronization)를 필요로 하며 또한 전체 시스템 작동에 대한 정보를 기록 하여야 한다. 이를 위해 분산시험사례의 개발시 여러 프로세스 사이의 동기 서비스를 제공하는 이벤트 서비스(event service)와 전체 시스템 작동 정보를 기록하는 전역기록 서비스(global logging service)를 라이브러리 형태로 제공한다.

SyncML 클라이언트와 SyncML 서버 사이의 상호운용시험은 SyncML 표현 프로토콜, 동기 프로토콜 그리고 HTTP 바인딩에 대한 시험으로 수행된다. SyncML 적합성 시험도구는 상호운용성을 양방향으로 시험하기 위해 시험대상과 통신하는 SyncML 클라이언트(또는 클라이언트 대리자, SyncML 어댑터, 동기엔진)나 서버(또는 서버 대리자, SyncML 어댑터, 동기엔진)를 시뮬레이션하는 모의 SyncML 클라이언트 또는 서버를 사용한다[그림 1]. 또한 필요한 경우 시험대상 SyncML 시스템 상에서 시험을 위한 SyncML 클라이언트나 서버를 구현하여 이들과 모의 클라이언트/서버 사이의 직접



(1) 시험대상 서버와 모의 클라이언트 사이의 통신 실험



(2) 모의 서버와 시험대상 클라이언트 사이의 통신 실험

[그림 1] SyncML 적합성 시험

통신을 위한 TCP/IP 보조 통신 채널을 구축한다. 이런 보조 통신 수단을 통해 각 엘리먼트에 대한 형, 값, 발생 가능한 예외, 예상 반환 값 등에 대한 정보를 전달한다. 클라이언트나 서버는 SyncML 메시지를 받으면 SyncML 메시지의 엘리먼트를 예상 엘리먼트와 비교 검증하며, 어떤 문제점이 있는 경우 이 사실을 보조 통신 수단을 통해 알린다. 클라이언트나 서버는 문제가 있는 경우 에러 정보가 담긴 SyncML 메시지를 반환하며 그렇지 않으면 메시지 발신자가 요청을 충족시킬 수 있는 SyncML 메시지를 반환한다. 클라이언트나 서버에서 보고되는 문제, 메시지 발신자가 감지하는 반환된 SyncML 메시지의 이상, 그리고 클라이언트나 서버에서 반환되는 SyncML 메시지가 메시지 발신자로 전달되지 못하는 통신 실패 등은 시험의 실패를 의미한다. 에러 정보를 포함한 SyncML 응답메시지가 불법적인 SyncML 요청메시지에 대해 제대로 발생하는지를 시험하기 위해 가능한 경우 불법 SyncML 요청메시지를 만들어 시험한다.

SyncML은 사용자 인터페이스가 명확히 정의되어 있지 않기 때문에 모든 SyncML 구현 시스템에 독립적인 적합성 시험도구를 개발하는 것이 어려우며 SyncML 클라이언트와 SyncML 서버의 기능을 시험할 때 어느 정도의 의존성을 가질 수 밖에 없다. 본 연구는 의존성을 최대한 줄이기 위해 모의 클라이언트/서버와 시험대상 SyncML 시스템 사이에 SyncML 메시지를 담고있는 XML 문서[7]를 직접 HTTP 트랜스포트[3]를 통해 전송하는 방식을 사용한다. 그리고 시험대상 시스템으로부터의 응답

메시지를 받아서 예상되는 기대 값과 비교하는 방법으로 시험을 수행한다.

본 연구의 SyncML 적합성 시험은 크게 표현 프로토콜, 동기 프로토콜, HTTP 바인딩에 대한 시험으로 구성된다. 다음은 각 시험에 대한 간단한 설명이다.

3.1 SyncML 표현 프로토콜 시험

SyncML 표현 프로토콜은 SyncML 메시지 형식(format)과 제한사항을 정의한다. SyncML 메시지의 발신자 역할을 하는 SyncML 클라이언트와 수신자 역할을 하는 SyncML 서버 사이의 통신시험을 통해 SyncML 표현 프로토콜에 대한 시험을 수행하며, 시험대상 서버가 클라이언트가 보낸 SyncML 메시지를 받을 수 있는지와 반대로 시험대상 클라이언트가 서버가 보낸 SyncML 메시지를 받을 수 있는지를 검증한다. SyncML 클라이언트와 SyncML 서버 사이의 통신시험은 클라이언트/서버 대리자와 SyncML 프레임워크 사이의 통신에 사용된 실제 SyncML 메시지의 구문과 의미를 검증하는 데에 초점이 있다. 모의 클라이언트/서버는 SyncML 표현 프로토콜의 모든 요구사항이 제대로 지원되는지 확신하기 위해 SyncML 메시지를 엘리먼트 단위로 철저히 분석한다. 통신시험은 다음과 같은 세부 영역으로 나뉘어 시험된다.

- SyncML 메시지의 정확한(correct) 수신 및 생성
- SyncML 메시지의 XML 문법준수에 대한 정형성(well-formedness)
- SyncML 표현 프로토콜의 일반 요구사항 만족
- SyncML 표현 프로토콜의 엘리먼트 타입에 대한 제한사항 만족

3.2 SyncML 동기 프로토콜 시험

SyncML 동기 프로토콜은 SyncML 클라이언트와 SyncML 서버 사이에서 데이터 동기를 위한 일반 시스템 요구사항, 사용자 인증(authentication) 및 권한설정(authorization), 각 동기타입을 위한 특정 실행환경의 초기화(initialization), SyncML 표현 프로토콜의 SyncML 메시지를 이용한 7개의 동기타입(synchronization type)을 정의한다. 본 연구는 통신 프로토콜 시험에서 일반적으로 사용되는 FSM (Finite State Model) 블랙박스 시험기술[8, 9, 10, 11]을 동기타입 시험에 사용하며 다음과 같은 세부 영역으로 나누어 시험한다.

- 데이터 동기를 위한 일반 시스템 요구사항
- 사용자 인증 및 권한설정(authorization)
- 동기타입에 따른 SyncML 클라이언트와 SyncML 서버의 실행환경 초기화
- 동기타입의 MSC에 따른 SyncML 메시지 교환 및 오류처리
- 동기엔진의 SyncML 명령어 실행 및 제한조건 검증
- 다중 메시지(multiple-message) 패키지 수신 및 실행

SyncML 동기 프로토콜은 SyncML 적합성 시험(conformance testing)에 명시된 시험사례를 중심으로 시험하며 데이터 동기 시험에 필요한 부수적인 시험을 추가한다. 또한 실행 중에 시험되어야 하는 SyncML 명령어와 데이터 형식 등에 대한 시험이 동기시험 중간에 포함될 수 있다.

3.3 HTTP 바인딩 시험

SyncML HTTP 바인딩은 SyncML 클라이언트가 HTTP 트랜스포트를 이용한 SyncML 서버로의 연결과 메시지 전송에 대하여 정의한다. 본 연구는 SyncML 프레임워크의 SyncML 어댑터를 시뮬레이션하는 모의 SyncML 어댑터를 사용하여 시험대상 SyncML 시스템의 연결과 메시지 전송 기능을 시험한다. HTTP 바인딩 시험은 다음과 같은 세부 영역으로 나뉘어 시험된다.

- TCP 전송(transport) 서비스를 이용한 SyncML 클라이언트의 TCP 연결과 HTTP 연결 기능
- HTTP 연결을 통한 SyncML 메시지의 교환 기능과 패키지 지원
- HTTP 전송 명령어 및 HTTP 헤더 지원

4. 분산시험사례 구조 및 프로세스 모델

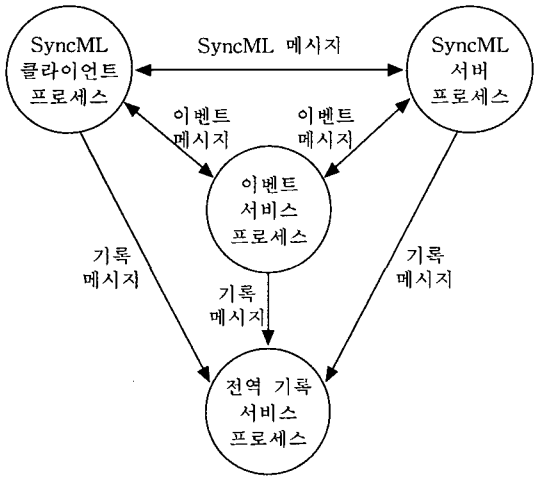
분산시험사례는 요구사항에 따른 세부 시험을 수행하기 위해 여러 개의 시험항목으로 나뉘어 실행되며 각 시험 항목은 대체로 하나의 메쏘드와 대응된다. 본 연구는 시험사례의 자동 실행을 대비하여 각 시험사례를 아래와 같이 세부분으로 구성한다 [12].

- 시험사례 실행에 필요한 환경의 초기화 부분
- 시험사례를 구성하는 여러 개의 세부 시험항목(test item) 부분
- 시험사례 실행 중에 임시로 생성된 파일 등을 삭제하는 정화(cleanup) 부분

시험사례를 구성하는 각 시험항목도 다음과 같이 세 부분으로 구성한다.

- 시작할 때 시험항목의 목적에 대한 메시지 출력
- 진단 메시지 출력문을 포함한 본 시험 코드
- 시험결과 코드의 출력

각 시험사례는 SyncML 클라이언트 클래스, SyncML 서버 클래스, 이벤트 서비스 클래스, 전역 기록 서비스 클래스 그리고 시험사례의 실행에 필요한 부수적인 클래스로 구성된다. SyncML 클라이언트 클래스와 SyncML 서버 클래스는 하나의 시험사



[그림 2] 프로세스 모델

래에서 서로 시험자와 시험대상자로서의 역할을 수행하며, 지역이나 원격 컴퓨터에서 서로 독립적인 프로세스로 실행된다. 따라서 본 연구의 분산시험사례는 SyncML 클라이언트 프로세스, SyncML 서버 프로세스, 이벤트 서비스 디몬(daemon) 프로세스 그리고 전역 기록 서비스 디몬 프로세스를 포함한 최소 4개의 프로세스로 실행되며, [그림 2]는 실행시의 프로세스 모델을 보이고 있다.

5. 시험사례 및 실행결과

다음은 표현 프로토콜의 Add 명령어 엘리먼트를 모의 클라이언트가 SyncML 서버에 대하여 시험하는 단순화된 시험사례의 예로서 크게 두 단계로 실행된다. [그림 3]은 이 시험사례의 수행결과를 보이고 있다.

1. 시험대상 서버로 요청메시지 전달
   - 보낼 메시지 생성
   - TCP 연결
   - HTTP 연결
   - 메시지 전송
2. 실험대상 서버로부터의 응답메시지 확인
   - 응답메시지가 정형(well-formed) XML 문서인지 확인
   - 규격에 명시된 제한사항을 만족하는지 확인
   - 시험요구사항에 따른 예상결과와 같은지 확인
   - 연결 종료

6. 결론 및 향후 과제

본 논문은 SyncML 적합성 시험도구의 개발을 위하여 SyncML 표현 프로토콜, SyncML 동기 프로토콜, SyncML HTTP 바인딩, SyncML 메타정보 DTD, SyncML 기기정보 DTD 그리고 SyncML 시

```
Test-Case Name : AddCommandElement
-----------------------------------
Test-Case started : 14:40:20
[g1t1] Well-formedness: pass
[g2t1] Imposed Restrictions: pass
[g2t3] Command Execution: pass
Test-Case ended  : 14:40:23
```

[그림 3] 시험결과

험지침에 대한 분석을 바탕으로 SyncML 적합성 시험도구의 개발을 위한 시험요구사항의 추출, 시험도구의 설계와 구현, 그리고 분산시험사례의 형식 및 시험사례 결과에 대하여 기술하였다. SyncML 적합성 시험은 매우 방대한 양의 시험사례를 요구하며, 각 시험사례가 분산환경에서 실행되어 상당한 고난도의 시험기술을 요구한다. 본 연구는 SyncML 규격 1.0.1 버전을 대상으로 하였으나 1.1 버전으로 쉽게 확장할 수 있으며, 앞으로 시험사례의 자동실행이 가능하도록 시험도구 기능을 확장할 예정이다.

참고문헌

[1] SyncML Representation Protocol, version 1.0.1
[2] SyncML Sync Protocol, version 1.0.1
[3] SyncML HTTP Binding, version 1.0.1
[4] SyncML Meta-Information DTD, version 1.0.1
[5] SyncML Device Information DTD, version 1.0.1
[6] SyncML Manual Test Cases, version 1.1
[7] Extensible Markup Language (XML) 1.0, W3C
[8] B. Beizer, Black-Box Testing, John Wiley & Sons, 1995
[9] G. V. Bochmann and A. Petrenko, "Protocol testing: Review of Methods and Relevance for Software Testing,"Proceedings of the 1994 International Symposium on Software Testing and Analysis, August 1994, pp. 109-124
[10] C. H. Chow and S. S. Lam, "Prospec: An Interactive Programming Environment for Designing and Verifying Communication Protocols," IEEE Transactions on Software Engineering, Vol. 14, No. 3, 1998, pp. 327-338
[11] D. P. Sidhu and T. Leung, "Formal Methods for Protocol Testing: A Detailed Study," IEEE Transactions on Software Engineering, Vol 15, No. 4, 1989, pp. 413-423
[12] R. M. Poston, Automating Specification-based Software Testing, IEEE Computer Society Press, 1996