논문 2009-46CI-1-11

SCA에서 적응형 도메인 프로파일 파서의 구축 방법

(Construction of a Adaptive Domain Profile Parser in the SCA)

배 명 남*, 이 병 복*, 박 애 순**, 이 인 환*, 김 내 수*

(Myungnam Bae, Byungbog Lee, Aesoon Park, Inhwan Lee, and Naesoo Kim)

요 약

SCA에서 코어 프레임워크는 이동단말 플랫폼의 시동, 무선의 초기화 등의 시점에 도메인 프로파일을 파싱하고, 이에 따라 플랫폼을 재구성하는 일종의 미들웨어이다. 도메인 프로파일은 XML로 작성되며, 소프트웨어 컴포넌트와 하드웨어 장치에 대한 고유 특성들을 포함하고 있다. 기본적으로, 코어 프레임워크는 도메인 프로파일을 파싱하기 위해 도메인 프로파일 파서를 포함해야 한다. 본 논문은 도메인 프로파일 파서를 구축하는 방법에 있어서, 이동단말과 같은 제한된 환경에도 적용할 수 있도록 도메인 프로파일 파서를 경량화하고, XML 파서 벤더에 대한 독립성을 강화하는 방법을 제안하고자 한다. 이를 통해, 도메인 프로파일 파서는 도메인 프로파일의 반복적인 파싱으로 인한 DOM 트리 생성에 대한 오버헤드 문제, 특정 XML 파서 벤더에 의한 호환성 저하 문제, 도메인 프로파일 기술 방식에 대한 의존성 문제 등을 해결할 수 있다.

Abstract

In SCA, the core framework must include the domain parser to parse the domain profile and thus reconstructs the platform on the time including the starting of the platform, the initialization of the new radio, and etc. The domain profile is described in XML and it includes the characteristics about the software component or the hardware device in a platform. Elementarily, the core framework has to have within the domain profile parser in order to parse the domain profile. In this paper, in order to apply to the limited environment like the mobile terminal, we propose the method for reducing the size of the domain profile parser and for strengthening the independency of the XML parser vendor to have with the domain profile parser. Therefore, domain profile parser can be solve the problem like the overhead about the DOM tree creation due to the repetitive parsing of the domain profile, the compatibility degradation by the specific XML parser vender, the dependency about the domain profile technique, and etc.

Keywords: Domain Profile, Domain Profile Parser, SCA, SDR, XML

Ⅰ. 서 론

SCA(Software Communication Architecture)는 JTRS (Joint Tactical Radio System) JPEO(Joint Program Executive Office)에 의해 단일 플랫폼에서 여러 waveform간의 상호 연동을 제공하는 유연한 소프트웨어 구조 개발을 목표로 제안되었다[1]. 이를 통해, 새로운 무선 환경의 확장과 추가 시에 비용과 시간을 절감할 수 있고, 새로운 기술과 쉽게 융합할 수 있어 진화된 시스템과의 호환성을 제공할 수 있다. 이러한 특성은 SCA의 미들웨어 계층과 코어 프레임워크 계층을 통해 플랫폼과 waveform간의 의존성 제거를 통해 달성될 수 있다.

미들웨어 계층은 상업적으로 제공되는 CORBA (Common Object Request Broker Architecture)[2], XML (eXtensible Markup Language)[3], 그리고 POSIX (Portable Operating System Interface)[4]를 기본적으로

* 정회원, 한국전자통신연구원 USN전송기술연구팀 (USN Transmission Technology Research Team, ETRI)

** 정회원, 한국전자통신연구원 차세대이동단말연구팀 (Next Generation Mobile Terminal Research Team, ETRI)

※ 본 논문은 지식경제부 및 정보통신연구진흥원의 IT 신성장동력핵심기술개발사업(2005-S-404-33), 국토해양 지능형국토정보기술혁신사업(06국토정보C01)의 연구비지원에 의해 수행되었습니다.

접수일자: 2008년9월7일, 수정완료일: 2009년1월5일

사용한다. 코어 프레임워크는 미들웨어 계층을 바탕으로 잘 통제되고 안정된 방법으로 여러 waveform을 이동단말 플랫폼에 적재하고 운용하는 것을 목적으로 한다. 이를 통해, 주파수 밴드 변경, 진화된 통신 프로토콜 및 무선체계간 절체를 달성할 수 있다. 코어 프레임워크와 waveform은 플랫폼을 구성하는 하드웨어와 소프트웨어의 구성 정보를 기술하고 있는 다수의 도메인 프로파일을 포함한다.

도메인 프로파일은 플랫폼의 시동이나 무선의 초기화 시점 혹은 무선 영역에 어플리케이션이 새로이 설치되는 시점에 파싱되고, 코어 프레임워크에 의해 플랫폼을 재구성하는데 사용된다.

현재, SCA는 코어 프레임워크와 도메인 프로파일과의 체계적인 인터페이스를 정희하지 않고 있다. 그 결과, 도메인 프로파일 파싱 과정에 XML 파싱 API를 직접 사용해 코딩되고 있다. 또한, 서비스에 따라 도메인 프로파일에 접근 요구가 제기되고 있다.

본 논문은 이 과정에서 발생하는 프로파일에 대한 반복적인 파싱을 방지하며, 특정 XML 파서에 대한 코어 프레임워크의 독립성을 보장하기 위한 방법에 대해 기술한다.

II. 관련 연구

이 장에서는 소프트웨어 통신 구조인 SCA와 참조모델, 도메인 프로파일과 도메인 프로파일 파서의 역할, 그리고, 기존 도메인 프로파일 파서 구성 방식의 문제점에 대해 기술한다.

1. 소프트웨어 통신 구조

SCA는 미국방성 산하 JTRS JPEO에서 제안되었으며, SDR(Software Defined Radio)을 지원하기 위해 제안된 통신용 소프트웨어 구조이다. 현재 실질적으로 SDR 시스템의 소프트웨어 구조용 표준으로 인정받고 있다.

SCA는 다음과 같이 waveform의 이식성을 극대화하기 위해 표준화된 구조를 제공한다. SCA는 단말에 개별 무선통신 기능을 구현하는 waveform 응용 계층과 이들 waveform의 재구성성을 보장하기 위한 코어 프레임워크 계층, 그리고 단말 환경간의 의존성을 제거하기 위한 미들웨어 계층으로 구성된다. 이러한 계층적 구분으로, SCA 기반 waveform은 SCA 표준을 만족시키는 어떠한 SDR 플랫폼에서도 설치 운용이 가능하다[1].

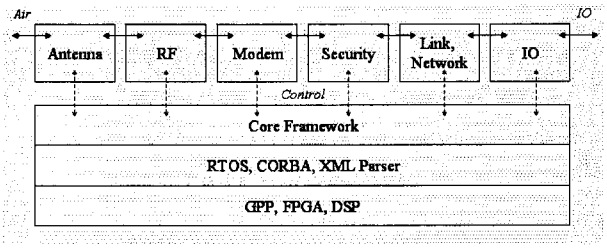


그림 1. SCA 참조 모델
Fig. 1. SCA Reference Model.

응용 계층은 안테나, RF, 모뎀 수준에서의 디지털 신호 처리, 링크/네트워크 수준에서의 프로토콜 처리, 보안, 외부 입출력 등을 컴포넌트로 포함한다[6]. 코어 프레임워크는 분산 환경에서 이들 개별 컴포넌트들을 배치하고 상호 연결하여 waveform의 동적 재구성을 지원한다. 코어 프레임워크는 이 과정에 필요한 컴포넌트들의 구성 정보를 얻기 위해, 도메인 프로파일을 해석하고 정보를 추출하기 위해 도메인 프로파일 파서를 사용한다. 미들웨어 계층은 컴포넌트들이 플랫폼과 독립적으로 수행될 수 있는 환경을 제공한다. 궁극적으로 플랫폼 독립성을 확보하기 위해 POSIX, CORBA, XML 요소들로 구성된다. 플랫폼은 여러 프로세서(예, CPU, FPGA, DSP)로 구성된 분산 형태의 시스템으로 설계된다. 따라서, SCA기반 시스템은 분산된 다중 프로세서들로 이루어진 분산환경과 다양한 하드웨어 자원을 관리하고 추상화하기 위해 CORBA를 기반으로 한다[5].

2. 도메인 프로파일 파서

SCA는 모든 컴포넌트에 대해 도메인 프로파일(Domain Profile)을 정의하도록 규격화하고 있다. 이는 코어 프레임워크가 무선체계의 재구성을 위해, 컴포넌트가 갖는 플랫폼 의존적인 특성들을 구현 코드(컴포넌트)와 분리하여 플랫폼 독립성을 보장해야 하기 때문이다. 도메인 프로파일의 각 섹션은 임의 특정 플랫폼의 종속성을 감소시키는 일종의 지시어를 사용함으로서 선택적으로 적용할 수 있다. 예를 들어, 도메인 프로파일은 CPU 사용량, 메모리 용량과 같은 플랫폼 요구사항과 해당 컴포넌트의 운용에 요구되는 또 다른 요소 컴포넌트의 배치와 연결 정보를 기술하고 있다.

그림 2는 SCA에서 정의한 도메인 프로파일 관계도이다.

도메인 프로파일 구조에서, SPD(Software Package Description)는 단위 컴포넌트(Resource 혹은 이의 구체

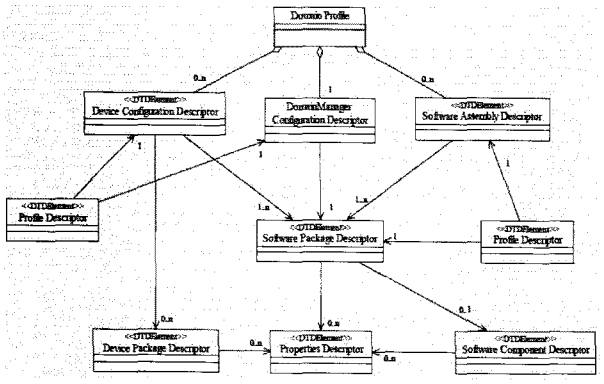


그림 2. 도메인 프로파일 구조
Fig. 2. Domain Profile Relationship.

화 개념인 Device, Application)의 구현 대상을 식별하기 위한 것이며, SCD(Software Component Descriptor)는 컴포넌트가 제공(provide)하고 사용(use)하는 모든 인터페이스에 대한 정보를 명세하며, SAD(Software Assembly Descriptor)는 Application을 구성하는 컴포넌트들에 대한 정보를 포함하며, PRF(Properties Descriptor)는 패키지에 적용할 고유 특성/속성 정보를 포함한다. DPD(Device Package Descriptor)는 설치될 디바이스를 식별하는데 사용되며, DCD(Device Configuration Descriptor)는 디바이스와 이들의 구성 정보를 포함한다. Profile Descriptor는 사용된 도메인 프로파일의 정보를 포함한다[7].

3. 도메인 프로파일 파서

도메인 프로파일 파서는 코어 프레임워크 요구에 따라 도메인 프로파일을 파싱하고 개별 정보를 제공하기 위해 필요하다. 도메인 프로파일이 XML을 사용하여 명세함에 따라 XML 파서도 필요하다.

현재, 여러 SCA 구현에서는 도메인 프로파일 파서와 XML 파서가 코어 프레임워크에 직접 연동되도록 구성되고 있으며, 파싱 결과의 재사용에 대한 다양한 고려가 반영되지 않고 있다[8]. 그 결과, 도메인 프로파일 파싱 과정의 불필요한 파싱 과정의 제외에 대한 어려움, 특정 도메인 프로파일의 중복 파싱으로 인한 비효율성이 초래되고 있다. 또한, 최근 SCA 환경이 DSP와 FPGA상에 적용되는 하드웨어 로직에 까지 확대 적용됨에 따라, 이들 컴포넌트의 프로파일 파싱을 위해 CPU상의 도메인 프로파일 파서를 공유하여 활용하려는 연구가 진행됨에 따라 도메인 프로파일 파서의 독립성과 체계화가 필요하다[9].

또 다른 관점으로, 도메인 프로파일 파서는 XML과 파서에 대한 의존성이 존재한다. 단말과 같이 제한된 환경에서 운용되는 XML 파서는 범용 XML 파서와 호환되지 않는 단순화된 API만을 제공하기도 한다. 하지만, 특정 XML 파서 벤더의 API를 직접 사용하는 도메인 프로파일 파서 구현은 COTS를 적극 활용하려는 코어 프레임워크 구성 원칙에도 벗어난다. 최근, XML의 복잡성과 XML 파서의 크기 문제로 인해 XML이 아닌 다른 대체 명세 방식의 사용을 적극 고려하고 있다[9]. 그러므로 도메인 프로파일 파서로부터 XML 파서에 대한 의존성은 도메인 프로파일 파서뿐만 아니라 코어 프레임워크의 유연성을 크게 떨어뜨리는 요인이 된다.

III. 도메인 프로파일 파서의 확장

이 장은 코어 프레임워크가 특정 XML 파서 벤더에 의존하지 않도록 하며, 단말과 같은 제한된 무선 환경에서 도메인 프로파일에 대한 중복 파싱을 지양하며 효율적으로 설치하고 실행할 수 있도록 구성한 도메인 프로파일 파서의 구현 내용을 기술한다. 또한, 도메인 프로파일 파서는 XML이 아닌 도메인 프로파일 명세 방식을 제공할 수 있는 융통성있는 구조도 지원하도록 확장하였다.

1. 코어 프레임워크와의 연동 방식

도메인 프로파일 파서는 코어 프레임워크와의 독립성을 강화하고 도메인 프로파일 파싱에 대한 효율성과 확장성을 개선하도록 체계화하였다.

다음 그림은 확장된 도메인 프로파일 파서의 구성을 보인다. 도메인 프로파일 파서는 모두 CORBA IDL로 정의하고 구현되었으며, 개별 도메인 프로파일을 파싱하기 위한 Parser들과 이들을 캡슐화하기 위한 팩토리

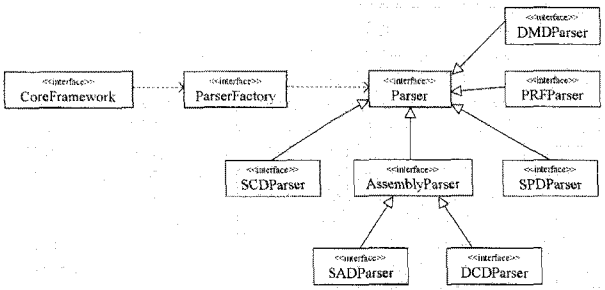


그림 3. 도메인 프로파일 파서 정의
Fig. 3. Interface Definition for Domain Profile Parser.

설계 패턴을 적용한 ParserFactory로 구성된다.

Parser는 SCA에서 정의한 7개의 파서를 추상화하기 위해 정의하였으며, 결국 개별 파서에 대응하는 해당 도메인 프로파일내 정보에 접근하는데 사용된다. 예를 들어, SADParser는 특정 어플리케이션을 구성하기 위한 요소 정보를 얻기 위해 사용된다[6]. ParserFactory는 Parser에 대한 캡슐화를 통해, 코어 프레임워크에 대해 Parser의 구현 방식이나 변경에 대한 독립성을 보장할 수 있다[10].

도메인 프로파일 파서에서 코어 프레임워크와의 독립성 보장은 매우 중요하다. 코어 프레임워크는 기본적으로 XML 파서 벤더에 대한 의존성을 갖지 않도록 해야 하며, 향후 도메인 프로파일의 명세 방식에 대한 의존성도 제거되어야 하기 때문이다. 이에 따라, 코어 프레임워크는 표준 XML 파서에 대한 직접 호출 대신에 ParserFactory에 의해 생성된 Parser를 통해 도메인 프로파일 파싱을 요청한다. 이를 위해, Parser는 일반적인 XML 데이터 추출 관점보다 코어 프레임워크 관점에 맞추어 도메인 프로파일의 파싱 결과를 반환하도록 개선된 접근 기능(Ⅲ장 4절에서 설명)을 제공하도록 구성되었다.

다음 예는 기존 방식에 의한 도메인 프로파일 파싱 과정이다. SADParser는 코어 프레임워크의 메모리 영역 내에 생성되며, XML 파서 API의 직접 사용을 통해 정보를 추출한다.

```
sad_parser * p = new SADParser();
p->getElement("name");
```

하지만, 이러한 방식은 특정 구현을 사용하게 되며, 향후 명세 방법의 변경이나 확장해야 할 경우 유연성이 떨어진다[10]. 본 논문의 방식에서는 다음과 같이 분산 환경에서 SADParser의 위치나 구현 방식, 그리고 도메인 프로파일 명세 방식에 대해서도 추상화하여, 코어 프레임워크와의 독립성을 보장한다.

```
Parser * p = ParserFactory.getParser(SAD,...);
p->getValue("dom.name");
```

또한, 도메인 프로파일에 대한 파싱 요구는 코어 프레임워크 뿐만 아니라 서비스 확장에 따라 개별 컴포넌트에서도 제기될 수 있기 때문에, 도메인 프로파일 파서에 대한 중복 적재와 도메인 프로파일에 대한 중복 파싱의 문제가 있다. 확장된 도메인 프로파일 파서는 분산 미들웨어인 CORBA를 통해 원격의 Parser와 ParserFactory를 통해, 플랫폼의 전체 생명주기내에서 대상 Parser에 대한 적재 및 해당 도메인 프로파일의 파싱은 단지 한번만 수행됨을 보장하기 때문에, 이러한 중복 문제로 인한 효율성 문제는 최소화될 수 있다.

이러한 방식은 DSP나 FPGA에 적재되는 컴포넌트에서 GPP상에 존재하는 원격의 Parser를 활용할 수 있어, FPGA에 Parser를 적재하는 오버헤드를 피할 수 있다는 장점이 있다[11].

2. 도메인 프로파일 접근 기능

SCA에서 도메인 프로파일은 XML로 기술하고 있다. 하지만, 도메인 프로파일은 XML 뿐만 아니라 다른 체계로 명세되는 것을 허용하여야 한다. 이는 제한된 무선환경에서 도메인 프로파일을 파싱하기 위한 XML 파서의 적재와 운용은 여러 부하를 유발하기 때문이다. 이러한 문제를 해결하기 위해, 도메인 프로파일 파서가 도메인 프로파일의 명세 방법에 의존하지 않도록 구축될 수 있도록 정의한 다음의 세 가지 인터페이스를 통해 가능하다.

구성 인터페이스는 도메인 프로파일의 분석 과정을 거쳐 기본 계층화된 구조를 생성하며, 추출 인터페이스는 이로부터 정보를 추출한다. 복합 인터페이스는 프레임워크 요구 관점의 복잡한 자료구조를 선택적으로 추가할 수 있도록 한다.

다음에서는 세 인터페이스의 기능과 역할에 대해 기술하며, 명세 방식에 대한 지원은 구성 인터페이스와

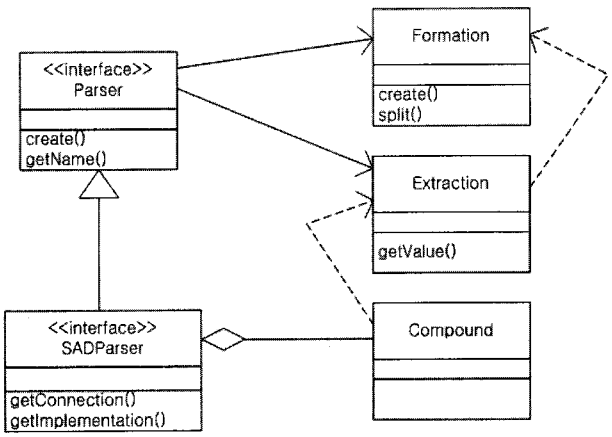


그림 4. 파싱을 위한 인터페이스
Fig. 4. Interface for parsing the domain profile.

추출 인터페이스에 대한 새로운 구현으로 달성될 수 있다.

가. 구성 인터페이스

구성 인터페이스는 명세 방식과 독립적으로 도메인 프로파일을 문법적으로 분석하고, 코어 프레임워크에게 제공하기 전 중간형태의 내부 정보를 구성한다. 명세방식이 XML인 경우 내부 구조는 전형적으로 XML 파서를 통해 얻은 DOM 트리이다. 이외, 명세 방식이 XML이 아닌 경우 다음에서 설명할 추출 인터페이스의 구현과 연계될 계층적인 혹은 자체의 고유 내부 구조를 형성한다.

먼저, XML로 명세된 다음 도메인 프로파일 명세는

<Element>
  <SubElement/>
</Element>

다음과 같은 계층 구조를 갖는 내부 구조 형태로 변환된다.

(상위요소 (하위요소))

내부 구조에서 XML의 각 Element는 요소로 구성되며, 요소간 논리적인 계층 구조를 형성한다(Element는 상위요소로 SubElement는 하위요소로). 즉, 요소는 또 다른 요소를 포함하여 계층적으로 구성될 수 있다. XML의 Attribute는 요소의 한 특성값을 명시하며, 한 요소에 다수 개의 특성값들을 정의할 수 있다. 이러한 내부 구조에 접근하기 위해서는 계층 구조를 해소(resolve)하기 위한 연산자가 필요하며, 이에 대한 정의는 추출 인터페이스에서 기술한다. 즉, 역할 측면에서, 구성 인터페이스는 내부 구조를 생성하며, 이에 대한 접근은 대응하는 고유의 추출 인터페이스를 통해 가능하다.

도메인 프로파일에 대응하여 구성되는 내부 구조의 전형적인 구성 예는 다음과 같다(○는 내부 구조의 (요소, 특성값들)을 의미함).

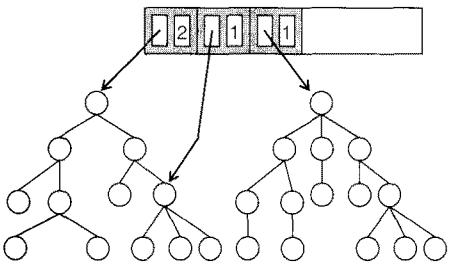


그림 5. 도메인 프로파일에 대한 내부 구조
Fig. 5. Internal Structure for Domain Profile.

도메인 프로파일의 파싱 결과인 내부 구조에는 여러 Parser에 의해 반복적으로 사용되는 부분이 있으며, 한 번의 접근이후에 필요하지 않은 부분이 혼재되어 있다. 구성 인터페이스는 이러한 내부 구조를 생성할 뿐만 아니라 내부 구조에 대해 부분적인 제거, 재사용 혹은 이동을 허용한다. 그 결과, 이러한 기능은 추출 요구시점에 불필요한 내부 구조의 항해를 최소화할 수 있다. 특히, 내부 구조로서 DOM 트리가 사용되는 경우, 정보 추출은 반드시 DOM 트리내 상대적인 경로를 사용해야 한다는 제약이 존재한다. 이때 DOM 트리에 대한 변경의 허용은 기존 도메인 프로파일 파싱 방식에서 도메인 프로파일에 명시되지 않은(생략된) 정보를 추출하기 위한 요구가 발생할 경우, 전체 DOM 트리에 대한 반복적인 재파싱이 발생된다는 점에서 부분적인 해결점이 될 수 있다.

이외에도, 구성 인터페이스는 프레임워크에서 내부 구조의 해제/유지, 요구된 도메인 프로파일에 대한 내부 구조의 존재 여부 확인 등의 추가 기능들을 포함한다. 이를 통해, 단말내 코어 프레임워크에서 이미 파싱한 내용을 포함하고 있는 Parser의 공유를 통해 도메인 프로파일의 중복 파싱을 최소화할 수 있다.

구성 인터페이스를 통해, 내부 구조는 DOM 트리에 의존적이지 않으며, XML이 아닌 경우에 내부 구조는 빠른 접근을 위해 해쉬함수와 (key,value) 형태로 구성될 수도 있다. 이와 같이 내부 구조 도메인 프로파일 파서와 상호 독립성을 가지며, 오직 명세 방식의 해석 방식에 의존한다.

나. 추출 인터페이스 도출

추출 인터페이스는 도메인 프로파일의 명세 방식과 독립적으로 필요한 정보에 접근하기 위한 연산들의 모임이다. 따라서, 추출 인터페이스는 특정 구성 인터페이스의 구현(즉, 내부 구조 생성 방법에 대한 구현)과 밀접한 관계가 있으며, 반드시 내부 구조에 대응하는 고유한 추출 인터페이스의 구현이 함께 제공되어야 한다. 예를 들어, 명세 방식이 XML 방식이라면, 구성 인터페이스의 구현은 XML 파싱의 결과인 DOM 트리를 내부 구조로 설정하도록 구현되어야 하며, 이때 추출 인터페이스는 DOM 트리에 접근하기 위한 XML API로 구현되어야 한다.

추출 인터페이스는 명세 방식과 내부 구조와 독립성

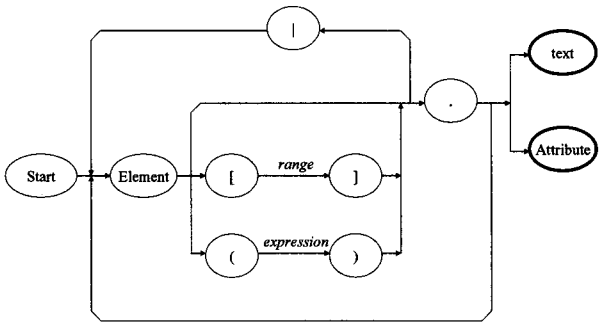


그림 6. 계층화 구조로부터 정보 추출 과정
Fig. 6. Information Extraction on Hierarchical structure.

을 갖도록 하기 위해, 다음과 같은 선언적인 형태의 추출 패턴을 사용한다. 앞의 예에서, 이러한 추출 패턴에 대한 구현은 DOM 트리에 접근하기 위한 XML API를 사용함을 의미한다.

추출 패턴은 SCA 도메인 프로파일에서 정보 추출에 유용하게 사용할 수 있는 전형적인 네 개의 연산자를 사용한다. 경로 연산자('.')는 내부 구조의 계층 구조에 따라 상위 요소(혹은 요소내 특정 특성값)로부터의 경로를 명세한다. 한정 연산자 ('[ ]')는 계층 구조내 동일 요소 구조에 대해, first, last와 같은 인덱스 혹은 first..last, first..last-1 과 같은 범위 인덱스를 사용하여 추출할 대상을 한정할 수 있다(인덱스 생략시 전체를 의미함). 조건 연산자 ('( )')는 조건을 명시하여, 조건을 만족하는 요소만을 선별적으로 지정하기 위한 목적으로 사용한다. 선택 연산자('|')는 여러 조건의 요소들을 추출 대상으로 선정할 때 사용된다. 이러한 연산자들의 최종 반환값은 문자 포인터 형(char *)이며, 반복 구조에 따라 문자 포인터의 포인터 형(char **)을 갖게 된다.

이러한 연산자의 예를 보이기 위해, XML로 작성된 다음의 도메인 프로파일 예를 사용한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<softwareassembly id="DCE:9ae8fddf-d919-4898-9ed3-0f55a2b61a7b"
name="3GE-WLAN">
  <componentfiles>
    <componentfile
id="AssemblyController_4b6e756b-6db5-4ae4-90fb-b898bfd649d0" type="SPD">
      <localfile name="/xml/AssemblyController.spd.xml"/>
    </componentfile>
    <componentfile id="3GEN_Resource_9f0482cd-a8c0-436a-b1c2-b783b5471bfe"
type="SPD">
      <localfile name="/xml/3GEN_Resource.spd.xml"/>
    </componentfile>
    <componentfile id="WLAN_Resource_7b9e2b2f-af5a-41c2-844f-923956ea4033"
type="SPD">
      <localfile name="/xml/WLAN_Resource.spd.xml"/>
    </componentfile>
  </componentfiles>
  <assemblycontroller>
    <componentinstantiationref
refid="DCE:4b6e756b-6db5-4ae4-90fb-b898bfd649d0"/>
  </assemblycontroller>
  <connections>
    <connectinterface id="Controller23GEN">
      <usesport>
        <usesidentifier>AssemblyController</usesidentifier>
        <findby>
          <namingservice name="AssemblyController"/>
        </findby>
      </usesport>
      <providesport>
        <providesidentifier>3GEN_Resource</providesidentifier>
        <findby>
          <namingservice name="3GEN_Resource"/>
        </findby>
      </providesport>
    </connectinterface>
    <connectinterface id="AssemblyControllerToWLAN_Resource">
      <usesport>
        <usesidentifier>AssemblyController</usesidentifier>
        <findby>
          <namingservice name="AssemblyController"/>
        </findby>
      </usesport>
      <providesport>
        <providesidentifier>WLAN_Resource</providesidentifier>
        <findby>
          <namingservice name="WLAN_Resource"/>
        </findby>
      </providesport>
    </connectinterface>
  </connections>
...
</softwareassembly>
```

내부 구조는 XML과 동등하게 요소간의 계층 구조를 그대로 수용한다. 전형적인 추출 패턴의 예제는 다음과 같다.

(1) dom.assemblycontroller.refid

Assemblycontroller의 refid 속성값을 얻기 위한 요구는 dom.softwareassembly.assemblycontroller.refid와 같이 명세함으로서 refid 속성값으로 단일 문자열 데이터인 String:'DCE:4b6e756b-6db5-4ae4-90fb-b898bfd649d0'를 얻을 수 있다. 이때, 도메인 프로파일내에서 'assemblycontroller' 요소는 유일하게 정의되므로 dom.assemblycontroller.refid와 같은 축약된 표현이 가능하다.

(2) dom.componentfile[ ].localfile.name

도메인 프로파일에서 하위 컴포넌트의 모든 도메인 프로파일의 명칭을 추출하는 요구에 대한 명세는 "dom.componentfile[].localfile.name"로 할 수 있다. 이 명세에서 dom은 내부 구조의 최상위 요소를 지칭하는 논리적 지시어이며, 계층 구조상 그 하위에 'componentfile' 노드로 부터 'localfile' 노드까지 경로가 있다면, 해당 'localfile' 요소의 'name' 속성을 추출하고 한정연산자 '[]'에 의해 1차 집합 형태로 값을 반환한다. Parser는 이 요구의 결과로 문자열 데이터 집합인 String[]:{'/xml/AssemblyController.spd.xml', '/xml/3GEN_Resource.spd.xml','/xml/WLAN_Resource.spd.xml' }을 얻을 수 있다. 한정 연산자는 최대 2개까지 가능하며, [1-2]와 같이 순서번호를 지정하여 앞선 두 개 정보만을 추출할 수도 있다.

(3) dom.connectioninterface(id='Controller23GEN').providesport.namingservice.name

'Controller23GEN'로 식별되는 CORBA 연결에 대해 포트를 제공하는 컴포넌트의 CORBA 네이밍 컨텍스트를 추출하기 위한 요구는 dom.connectinterface(id='Controller 23GEN').providesport.namingservice.name을 통해 명세될 수 있으며, 결과적으로 String: '3GEN_Resource'를 얻을 수 있다. 이 요구는 두개의 connectioninterface 요소 중에서 선택 조건을 명시하기 위해, connectioninterface의 id 특성값에 제약을 명시하였다.

비슷하게, 대상 리소스의 연결이름을 얻기 위해서는 'name' 대신에 dom.connectinterface(id='Controller23GEN').providesidentifier.value를 통해 'provides identifier' 요소의 값인 String: '3GEN_Resource'를 참조할 수 있다.

추출 인터페이스는 도메인 프로파일 파싱후 결과 추출시, 특정 명세 방식에 독립적이므로 유연하게 구성이 가능하며, 또한 정형화되지 않은 내부 구조 탐색으로 인한 오버헤드 등의 문제를 해결할 수 있다.

다. 복합 인터페이스

도메인 프로파일 파서는 GPP, DSP, FPGA 등으로 이루어지는 분산 환경에 대해 위치 독립성을 갖는다. 즉, 도메인 프로파일을 파서와 파싱을 요구하는 컴포넌트는 동일 프로세서에 존재하지 않을 수 있다. 분산환경에서 도메인 프로파일 파서와 컴포넌트간 전송 부하를 줄이기 위해, 상호 응집력이 강한 정보들은 하나의 복합 구조로 구성된다. 대표적인 구조인 Implementation, Connection, ProvidePort, FindBy, UsesPort, ComponentPlacement, Component Instantiation 등과 같은 구조이다. 이들에 대한 범위는 SCA의 코어 프레임워크와 도메인 프로파일간 인터페이스에 따라 정의되었다.

SPD의 Implementation 복합 구조에 대한 정보 추출 과정은 아래와 같다.

```
SPDImplementation::SPDImplementation(DOMElement* _elem)
{
  implementationID = ParserAPI::getValue( _elem, "implementation.id");
  codeFile = ParserAPI::getValue( _elem, "implementation.code.localfile.name");
  entryPoint = ParserAPI::getValue( _elem, "implementation.code.entrypoint.text");

  char *_ct = ParserAPI::getValue( _elem, "implementation.code.type");

  if (strcmp (_ct, "KernelModule") == 0)
    codeType = CF::LoadableDevice::KERNEL_MODULE;
  else if (strcmp (_ct, "SharedLibrary") == 0)
    codeType = CF::LoadableDevice::SHARED_LIBRARY;
  else if (strcmp (_ct, "Executable") == 0)
    codeType = CF::LoadableDevice::EXECUTABLE;
  else if (strcmp (_ct, "Driver") == 0)
    codeType = CF::LoadableDevice::DRIVER;
  else {
    String msg = "[SPDImplementation:setCodeType] wrong code type passed. ";
    msg += " Type received is '" + ct + "'";
    throw new SCA.CF.InvalidProfile(msg);
  }
}
```

복합 구조내 세부 정보의 추출은 앞 절에서 정의한 추출 인터페이스들을 사용한다. 특히, 내부 구조에서 관련 요소들은 계층적으로 근접하여 위치해 있으므로, 항상 최상위 요소(즉, dom.)로 부터의 항해 대신에 가능하다면 마지막 검색된 요소의 위치로부터 상대적으로 항해하면서 검색한다. 즉, 마지막 요소의 위치를 저장하고 다음 검색은 이 위치로부터 정보 추출과정을 재시작할 수 있다.

Ⅳ. 구현 결과

1. 구현 환경

본 논문에서 사용하는 SCA 구현은 공개 프로젝트인 OSSIE[8]이다. OSSIE는 범용 XML 파서인 Xerces[12]를 사용하고 DOM을 기반으로 구현된 도메인 프로파일 파서를 포함하고 있다[8]. 도메인 프로파일 파서는 라이브러리 형태로 코어 프레임워크에 제공된다.

도메인 프로파일 파서의 구현에 대한 기존 방식과의 비교를 위해, 첫째 단계로, 본 논문에서 설명한 도메인 프로파일 파서 확장의 적용, 두 번째 단계로, XML 명세 방식이지만 XML 파서 벤더의 변경시 필요한 추출 인터페이스 재구현, 그리고, 마지막으로 플랫파일(flat-file) 명세 방식으로의 변경시 필요한 구성 및 추출 인터페이스 재구현 결과에 대해서 기술한다.

2. 구현 비교

도메인 프로파일 파서 확장을 기존 방식과 비교하기 위해, 세 단계의 구현 결과를 보인다.

첫째, 코어 프레임워크와 명세 방식간의 독립성을 보장하기 위해, Ⅲ장에서 기술한 방식에 따라 OSSIE의

표 1. 메모리 용량 비교
Table 1. Comparison of memory capacity.

(단위, byte)

라이브러리크기	기존방식	개선방식
코어 프레임워크	1,871,872(140,706)	1,178,840(167,982)
도메인 프로파일 파서	1,372,160(173,658)	774,144(79,570)

도메인 프로파일 파서를 재구현하였고 다음과 같은 결과를 보인다.

확장된 방식을 적용한 결과 코어 프레임워크는 적용 전 대비 67%로 감소하였고, 도메인 프로파일 파서도 적용 전 대비 55%로 경량화되었다. 이러한 경량화 결과는 기존의 코어 프레임워크와 도메인 프로파일 파서가 직접 XML 파서를 통해 도메인 프로파일을 파싱하고 있는 것이 주원인이며, 파싱 과정의 반복 및 중복이 다수 포함되어 있음을 알 수 있다.

두 번째로, OSSIE는 Xerces[12]를 기본 XML 파서로 적재하고 있지만, 이는 매우 큰 부담이다. 추출 인터페이스의 구체화 부분을 Xerces가 아닌 다른 XML 파서 API로의 교체를 통해 코어 XML 파서의 변경이 가능하다. Xerces 대비 10%정도 크기이며 제한된 XML API만을 제공하는 소형 XML 파서인 scew[13]로 직접 사상이 가능하였다. 이때, 코어 프레임워크에 대한 어떤 변경도 없이 연동될 수 있었다.

세 번째로, 비 XML의 명세 방식인 경우, 구성 인터페이스와 추출 인터페이스에 대한 구체화가 필요하다. 명세 방식이 플랫파일인 경우, 구성 인터페이스의 내부 구조는 해쉬 함수와 데이터 쌍(요소 명칭과 속성값)으로 구체화하였다. 요소에 대한 변경은 해쉬 엔트리의 변경으로 대응되며, 계층 구조는 존재하지 않는다. 추출 인터페이스는 해쉬 함수에 대한 호출로 재구현되었으며, 수 시간의 노력으로 교체가능하다.

이와 같이, 기존 도메인 프로파일의 확장을 통해, 보다 효율적인 메모리 효율성을 제공하며, 코어 프레임워크와 도메인 프로파일 파서간의 독립성을 보장할 수 있으며, XML 파서 벤더와 명세 방식에 대한 투명성을 보장할 수 있다.

V. 결 론

SCA 기반 플랫폼에서 XML로 기술된 도메인 프로파일과 이의 파싱에 대한 여러 고려가 필요하다. 코어 프레임워크와 도메인 프로파일 파서가 XML 파서에 대한 의존성이 크다는 점, 단말과 같은 제한된 환경에서 XML 파서의 적재는 플랫폼에 큰 부하요인이라는 점 등이 있다. 이에 따라, 단말에 적재 가능하지만, 표준 XML API지원이 아닌 소형 XML 파서의 활용뿐만 아니라 XML이 아닌 도메인 프로파일 명세 방식의 지원도 고려되어야 한다.

이에 따라, 본 논문에서는 코어 프레임워크와의 연동 방식과 도메인 프로파일 접근 기능을 포함하여 확장된 도메인 프로파일 파서에 대해 기술하였고, 이의 실제 적용을 통해 코어 프레임워크와 도메인 프로파일 파서의 경량화 결과와 개선 사항을 제시하였다.

이를 통해, XML을 포함한 다양한 명세 방식을 수용하며, 코어 프레임워크에서 도메인 프로파일의 반복적인 파싱을 통한 DOM 트리 생성에 대한 오버헤드, 특정 XML 파서 벤더에 의존한 호환성 저하 문제 등을 해결할 수 있다.

현재, 도메인 프로파일의 명세 방법의 단순화와 다양화가 진행 중에 있으며, 적응형 도메인 프로파일 파서는 기존 XML 뿐만 아니라 다양한 도메인 프로파일 명세를 지원하기 위한 방법이므로, 이에 대한 선행 연구로 활용될 수 있을 것이다.

참 고 문 헌

[1] JTRS Technical Document, "Software Communications Architecture Specification V2.2." Nov. 2002.
[2] Object Management Group, "The Common Object Request Broker Architecture: Architecture and Specification," Ver. 3.0, June 2002.
[3] WWW Consortium Recommendations, "Extensible Markup Language(XML): Part 1. Syntax," 1998.
[4] ANSI/IEEE Std. 1003.1, POSIX(Portable Operating System Interface) Part 1: System Application Program Interface, July 1996.
[5] Eric R Christensen, Annamarie Miller, and Byron Traver, "Object Management and Base Class Interfaces in the Distributed-Object Computing S", SDRF Contribution Report.
[6] JTRS Technical Document, "Software Comm unications Architecture Specification V2.2. API Supplements" Nov. 2002.
[7] JTRS Technical Document, "Support and Rational Document for the Software Communications Architecture Specification V2.2." Dec. 2002.
[8] OSSIE Project home page, http://ossie.mprg.org/
[9] 홍성수, 김세화, 유종훈, "다중모드 SDR을 위한 SCA 기반 소프트웨어 구조", SK Telecommunication Review, Vol. 17, pp. 420-431, June 2007.
[10] Elisabeth Freeman and Eric Freeman, "Head First Design Patterns," O'Reilly, 2004.

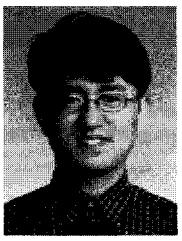
[11] S. Aslam-Mir, "Using CORBA on DSPs, FPGAs and Microcontrollers: Synthesizing a Ubiquiyous SCA Machine in Soft-Radio Devices," PrismTech White Paper, 2004.
[12] Xerces project home page, http://xerces.spache.org/
[13] scew Project home page, http://savannah.nongnu.org/projects/scew/

저 자 소 개

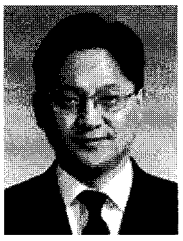
배 명 남(정회원)
1991년 전북대학교 전산통계학과 학사 졸업
1993년 전북대학교 전산통계학과 석사 졸업
1998년 전북대학교 전산통계학과 박사 졸업
1998년～현재 한국전자통신연구원 USN전송기술연구팀 선임연구원
<주관심분야 : SDR, 통신 미들웨어, 개방형 플랫폼, 임베디드 하드웨어, 무선통신>

박 애 순(정회원)
1987년 충남대학교 전자계산학과 학사 졸업
1997년 충남대학교 전자공학과 석사 졸업
2001년 충남대학교 컴퓨터과학과 박사 졸업
1988년～현재 한국전자통신연구원 차세대이동단말연구팀장(책임연구원)
<주관심분야 : 4세대 이동통신, 이동통신망, 이동성관리, 이동단말기술>

김 내 수(정회원)
2000년 한남대학교 컴퓨터공학과 박사 졸업
1986년～1990년 국방과학연구소
1990년～현재 한국전자통신연구원 RFID/USN 연구본부 USN전송기술 연구팀장(책임연구원)
<주관심분야 : RFID/USN, 위성통신, 컴퓨터 네트워크>



이 병 복(정회원)
1991년 호원대학교 전자계산학과 학사 졸업
1993년 전북대학교 전산통계학과 석사 졸업
2005년～현재 고려대학교 전산학과 박사과정
1993년～현재 한국전자통신연구원 USN전송기술연구팀 책임연구원
<주관심분야 : 이동통신단말기 시스템, 임베디드 시스템 개발 및 실행환경, 무선 센서네트워크 전송기술>



이 인 환(정회원)
1988년 한양대학교 전기공학과 학사 졸업
1990년 한양대학교 전기공학과 석사 졸업
2006년～현재 중앙대학교 컴퓨터공학과 박사과정
1990년～1993년 (주)동아전기 연구원
1993년～현재 한국전자통신연구원 USN전송기술연구팀 책임연구원
<주관심분야 : 디지털 이동통신, 신호처리, ASIC 설계>