주 제

SDR 기술 (SCA를 중심으로)

ETRI 오상철, 김홍숙, 김진업

차례

I. 서 론

SDR (Software Defined Radio) 기술은 하드웨어의 교체 없이 다운로드를 통한 소프트웨어의 교체만으로 여러 가지 모드의 무선통신을 할 수 있도록 하는 기술로써 차세대 이동통신 기술의 핵심 기술로 자리 잡아가고 있다.

이러한 SDR 기술을 완성하기 위해 여러 가지 종류의 무선통신을 위한 소프트웨어와 하드웨어를 지원할 수 있는 개방형 시스템 구조가 요구되었으며 하드웨어 계층과 소프트웨어 계층의 명확한 분리를 요구하는 접근방법을 통해 소프트웨어가 임의의 하드웨어 플랫폼에서 이식가능 하도록 하였다. 따라서 이들 하드웨어 플랫폼은 휴대용 (manpack), 공수용(airborne), 해상용(naval) 또는 고정용(stationary) 무선통신과 같이 각각 다른 환경의 요구 사항을 지원함에 있어서 해당 소프트웨어의 배치 형태에 따라 자유롭게 확대 또는 축소될 수 있다. 또한, 각 배치 형태에 따라 전력 소비, 크기, 적응성 등을 최적화할 수도 있다. 이와 같이 소프트웨어 주도형 하드웨어 (software driven hardware) 플랫폼을 사용함으로써 새로운 waveform 과 기능의 구현을 쉽게 할 수 있다[1].

SDR Forum에서는 이를 위해 JTRS (Joint Tactical Radio System) JPO (Joint Program Office)에 의해서 정의된 SCA (Software Communications Architecture)를 SDR 모바일 플랫폼 구성을 위한 소프트웨어 프레임워크의 표준으로 삼고 있다[2].

SCA 기반으로 구성된 모바일 플랫폼은 사용되어지는 무선 도메인에 관계없이 공통된 개방형 프레임워크를 바탕으로 도메인간의 상호 운용성 (Interoperability), 다양한 주파수, 소프트웨어의 이식성(Portability), 범위성(Scalability) 등을 제공해 준다.

SDR 을 위한 소프트웨어 전략과 관련하여 직면한 문제는 임의의 하드웨어에 이식 가능한 소프트웨어를 만드는 것이었는데 이는 자바에서 JVM(Java Virtual Machine) 이 필요했던 이유와 맥락을 같이 한다. SCA는 현재 미군의 JTRS Radio 에 있어서 필수 요구사항이며, 군사용 SDR 의 사실상의 표준(de facto standard)이다 [3].

II. SCA 개요

SCA규격은 최근까지 발전된 통신기술을 이용하여 미래의 통신 시스템을 개발하기 위해 설립된 미국의 JTRS JPO에서 통신 시스템간 상호 연동성을 크게 개선하고 개발 및 배치 비용을 줄이고자 제안한 통신 소프트웨어 구조이다.

SCA는 이러한 JTRS의 목적을 이루기 위해 JTRS와 MSRC (Modular Software-programmable Radio Consortium)에 의해 만들어졌다. SCA는 특정 시스템에 한정되는 규격이 아니고 위의 목적을 만족시키는 통신 시스템을 만들기 위한 독립적인 시스템 디자인 프레임워크라고 말할 수 있다. SCA는 다음과 같은 몇 가지 기본 원칙을 가지고 구축되어야 한다 [4].

- SCA를 기반으로 구현된 서로 다른 시스템상의 응용 소프트웨어간에 이식성을 제공하여야 한다.
- 개발 비용의 최소화를 위해 상용 제품 표준을 최대한 사용하여야 한다.
- 모듈화 소프트웨어의 재사용을 통하여 새로운 시스템 개발의 시간을 단축시킨다.
- 상용 프레임워크와 구조를 발전시키는 것을 토대로 해야 한다.

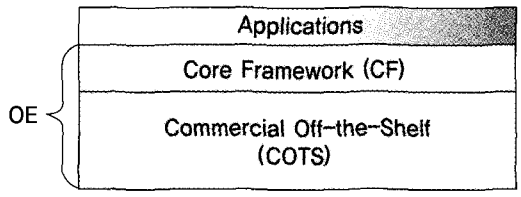
이런 원칙 하에 개발된 SCA의 소프트웨어의 프레임워크는 군사 전술용 응용분야에서 일반 상업용 응용분야에까지 이르게 되었고, 실제 이동통신 분야에서의 SDR 기술에 적용되는 단일 플랫폼 형성을 위한 내장형 제어 소프트웨어의 표준 규격으로 자리 잡게 되었다.

한편, SCA가 공식적인 상업용 소프트웨어의 표준 프레임워크로 공식화 될 전망과 더불어 많은 회사들이 SCA 프레임워크를 검증하고 개발하는 데 참여하고 있다. JTRS JPO는 1998년 처음으로 발표된 SCA Version 1.0 규격을 통하여 프레임워크 규격의 개발과 더불어 SCA 규격의 검증을 MSRC를 통하여 Raytheon을 비롯한 여러 회사에 각각 다른 프로토타입을 가지고 수행하게 되었다. 이를 통해 규격의 개발이 더욱 가속화가 되었고 현재 Version 3.0 까지 발표된 상태이다.

또한, SCA 는 다음과 같은 상용 표준을 기반으로 작성되었다.

- X.731 ITU/CCITT OSI System State Management
- POSIX (Portable Operating System Interface)
- CORBA (Common Object Request Broker Architecture)
- CCM (CORBA Component Model)
- XML (eXtensible Markup Language)

III. SCA 구조



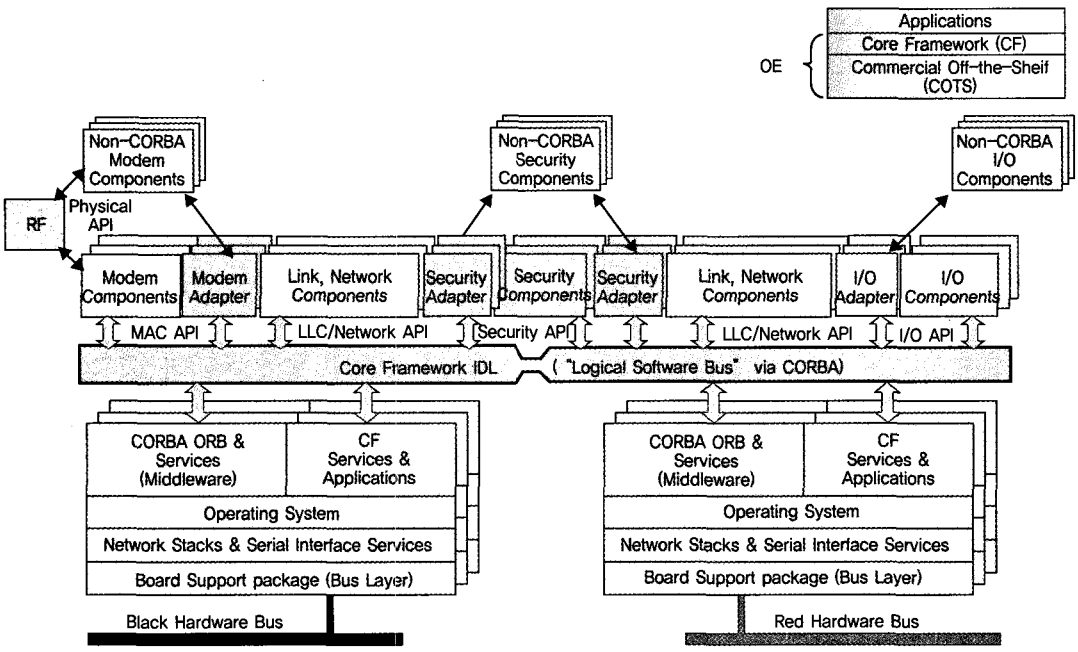
(그림 1) 개략적인 SCA 소프트웨어 구조

(그림 1)은 SCA 소프트웨어 구조의 개략적인 구조를 설명하고 있다. SCA 소프트웨어는 크게 어플리케이션과 운영환경인 OE (Operating Environment)로 나눌 수 있으며 운영환경인 OE 는 다시 코어 프레임워크(CF)와 상용 제품인 COTS (Commercial Off-the-Shelf)로 나뉘어 지는데 COTS 에는 POSIX 기반의 RTOS(Real-Time Operating System)와 CORBA가 포함된다. 이는 RTOS 와 CORBA의 경우 SCA 를 위해 새로 정의하는 것이 아니라 상용으로 이미 출시되어진 제품을 그대로 사용한다는 것을 의미한다.

SCA 구조는 이와 같이 OE 를 정의하고, OE 상에서 수행되는 어플리케이션 서비스들과 OE 간의 인터페이스를 정의하여 응용서비스들의 이식성을 제공한다. 운영환경 OE 는 SCA 규격을 따르는 무선 플랫폼에서 동작하는 어플리케이션들이 다른 플랫폼에서 쉽게 이식되어 동작되도록 설계시 제약요소를 적용한다. 이러한 설계 제약요소로는 Core Framework와 어플리케이션간의 인터페이스, Operating System의 waveform 사용에 대한 제약요소들이 포함된다. 또한, SCA는 어플리케이션 컴포넌트들간 API (Application Programming Interface)를 정의하기 위해 빌딩 블록 (BB : Building Block) 구조를 제공한다. 빌딩 블록 구조 적용은 컴포넌트 수준의 재사용을 용이하게 하고, 개발자들이 waveform에 특별한 API들을 정의하도록 유연성을 제공한다.

SCA 적용은 다음과 같은 장점을 제공한다.

- 상업용 프로토콜과 제품의 사용을 최대화
- 다 계층(multiple layers)으로 구성된 개방형, 상업용 소프트웨어 인프라 구조에서 코어 (core) 및 비코어 (non-core) 어플리케이션 (Application)들을 내장 하드웨어로부터 분리
- CORBA를 사용하는 분산 처리 환경을 통해 소



(그림 2) SCA 소프트웨어 구조

프트웨어 어플리케이션의 이식성, 재사용성, 확장성을 제공

SCA 구조는 객체 지향 방법론을 사용하여 개발되었고, OMG에서 권고하는 UML (Unified Modeling Language)을 사용하여 인터페이스를 그래픽으로 표현하고, IDL (Interface Definition Language)을 사용하여 인터페이스를 정의하였다.

(그림 2)는 SCA 소프트웨어 구조를 보여준다. (그림 2)에서 소프트웨어 구조는 CF 서비스들과 인프라 소프트웨어들(Board 지원 패키지, 운영 체제(OS) 및 서비스, CORBA 미들웨어 서비스 포함)로 구성된 OE를 정의한다. 또한, (그림 2)는 SCA 규격에서 따라 어플리케이션 소프트웨어들을 분할/적용하여 waveform을 구현하는 방법 예를 보여준다.

3.1 운영 체제

OS는 소프트웨어 아키텍처의 운영체제로서 POSIX 기반의 처리 환경을 제공한다. SCA는 애플리케이션의 이식성 및 아키텍처의 범용성 등을 지원하기 위해 애플리케이션 환경 프로파일(AEP : Application Environment Profile)을 정의하고, OS는 AEP에서 필수적으로 제공하여야 하는 서비스를 지원하도록 규정하고 있다.

POSIX는 IEEE에 의하여 정의된 현재 RTOS의 산업용 표준이다. POSIX 1003.13 표준은 애플리케이션의 모델에 따라서 PSE-51, PSE-52, PSE-53, PSE-54의 4개의 AEP를 정의하고 있다 [11]. SCA는 이중에서 SCA 요구사항을 만족시키는 최소한의 POSIX AEP인 Real-time System 프로파일을 제공하는 PSE-52를 채택하고 있다.

3.2 CORBA 미들웨어

CORBA 미들웨어는 분산 처리를 위해 Object Management Group (OMG)에서 제정한 미들웨어의 표준이다. SCA에서는 애플리케이션들 간의 통신 및 애플리케이션과 CF, OS와의 논리적인 소프트웨어 버스를 제공하여 OS로의 접근과 동시에 CF 메시지들을 전달하기 위한 미들웨어로 CORBA를 사용하고 있다. 한편, SCA에서는 OE의 요구사항 중의 하나인 Naming Service와 Log Service를 CORBA의 확장 영역에서 제공해주는 CORBA Naming Service와 Log Service를 사용한다. CORBA Naming Service는 ORB (Object Request Broker)를 통하여 원하는 객체를 찾아 그 객체의 Object Reference를 얻어주는 역할을 한다. 또한 CORBA에서는 Event Service를 제공하여 consumer object와 producer object 사이의 결합도(coupling) 를 최소화 할 수 있다 [5].

모든 CF 인터페이스는 IDL로 정의된다. CORBA 프로토콜은 메시지 전달을 위해 필요한 bit packing과 handshaking을 처리하는 위한 메시지 마샬링 (marshalling)[1]을 제공한다. SCA IDL은 컴포넌트들 간에 사용되는 오퍼레이션과 속성을 정의한다.

3.3 Core Framework

CF는 소프트웨어 어플리케이션 설계자들에게 내

1) 컴퓨터 프로그래밍에서, 마샬링은 하나 이상의 프로그램 또는 연속되어 있지 않은 저장 공간으로부터 데이터를 모은 다음, 데이터들을 메시지 버퍼에 집어넣고, 특정 수신기나 프로그래밍 인터페이스에 맞도록 그 데이터를 조직화하거나, 미리 정해진 다른 형식으로 변환하는 과정을 의미한다.

장된 소프트웨어와 하드웨어 계층에 대한 개요를 제공하기 위한 개방형 어플리케이션 계층 인터페이스와 핵심 서비스들의 집합이며, 다음과 같이 구성되어 있다.

- 기본 응용 인터페이스(Base Application Interface) : *Port, LifeCycle, TestableObject, PortSupplier, PropertySet, ResourceFactory, Resource*
- 프레임워크 제어 인터페이스(Framework Control Interface) : *Application, Application-Factory, DomainManager, Device, Loadable-Device, ExecutableDevice, AggregateDevice, Devicemanager*
- 프레임워크 서비스 인터페이스(Framework Service Interface) : *File, FileSystem, FileMan-ager, Timer*
- 도메인 프로파일(Domain Profile)

Base Application Interface는 모든 소프트웨어 애플리케이션에서 사용될 수 있는 기본이 되는 인터페이스이다. 〈표 1〉은 Base Application Interface의 구성요소와 기본 오퍼레이션을 나타낸 것이다. Base Application Interface에 대한 세부적인 UML 다이어그램과 각각의 인터페이스에서 제공되는 오퍼레이션에 대한 설명은 SCA 소프트웨어 아키텍처 규격문서를 참조하도록 한다[4].

Framework Control Interface는 크게 Domain Management, Device, Device Management Interface로 나눌 수 있다. Applicaton, ApplicationFactory, DomainManager로 구성된 Domain Management는 하나의 도메인 내에서 도메인 구성요소(Application, Devices, Device Manager, Event Channel)들의 등록/등록해제, 설치/삭제, 검색 등의 서비스를 제공한다. Device Interface는 Device, LoadableDevice, ExecutableDevice, AggregateDevice로 구성되고, 도메인 내의 Logical Device를 관리하는 기능을 제공한다. Device Management Interface는 DeviceManager 인데, 이는 Logical Device를 생성하고, 생성된 Logical Device에 서비스 애플리케이션을 시작시키는(Launch) 기능을 포함한 노드 관리를 수행한다. Framework Control Interface의 UML 다이어그램과 각각의 오퍼레이션 및 설명과 상호관계는 SCA 규격을 참조하도록 한다[4]. Framework Control Interface는 기본적으로 Base Application Interface를 상속하고, 추가적으로 위에서 설명한 각각

〈표 1〉 Base Application Interface

인터페이스	오퍼레이션 및 설명
Port	포트들간의 관리를 위한 오퍼레이션을 제공한다. connectPort(), disconnectPort()가 있다.
PortSupplier	특정 객체에 대한 포트의 참조를 얻을 수 있는 메커니즘을 제공한다. getPort() 오퍼레이션이 있다.
LifeCycle	특정 객체의 초기화 및 해제를 관리한다. CF의 ApplicationFactory를 사용하며 Constructor 이후 초기화 및 종료시 메모리 해제, File Close를 담당한다. initialize(), releaseObject()가 있다.
Testable-Object	특정한 객체에 대한 Built-In Tests(BIT)를 제공한다. 테스트 Id 및 Value는 도메인 프로파일의 Properties Descriptor에 따르고 runTest()가 있다.
PropertySet	특정 객체에 대한 Property XML 파일에 정의된 Configuration 파라미터의 값을 요구 내지 변경하는 메커니즘을 제공한다. query(), configuration()이 있다.
ResourceFactory	응용 Resource를 생성하고 삭제하는 데 사용. createResource(), releaseResource(), shutdown()이 있다.
Resource	컴포넌트의 제어와 구성을 위한 공통된 API를 제공한다. Application이나 Device를 설정하기 위해 HCI(Human Computer Interface)에 의해 사용된다. start(), stop()이 있다.

의 기능에 따른 오퍼레이션이 추가된다.

Framework Service Interface는 Core Application과 Non-core Application 모두에 서비스를 제공하는 인터페이스로, File, FileSystem, File Manager가 있다. File은 파일시스템 안의 파일에 대한 읽고, 쓰기(read, write, sizeOf, close, setFilePoint) 등의 기능을 제공한다. FileSystem은 CORBA 오퍼레이션 제공을 통하여 물리적으로 떨어진 파일시스템에 원격접속을 허용하고 기본적인 파일 액세스에 관련된 기능(remove, copy, exist, list, create, open, mkdir, rmdir, query)을 제공한다. FileManager는 한 개 이상의 분산된 파일 시스템을 관리하는 기능을 갖는데, 이 FileManager는 실제 파일 시스템이 두 개 이상으로 떨어져 있어도 하나의 파일시스템에만 존재한다. 이것을 "federated file system"이라고 한다. FileManager는 이러한 파일 시스템을 관리하는 기능(mount, unmount, getmounts)을 제공한다.

마지막으로 Domain Profile은 하나의 SCA 시스템 도메인을 구성하는 하드웨어 디바이스와 소프트웨어 컴포넌트들을 기술하는 파일들의 집합이라고 할 수 있다. 도메인 프로파일은 하드웨어 디바이스와 소프트웨어 컴포넌트의 위치, 상호 의존관계, 용량, 식별(identity), 능력(capability) 등을 기술하고 있으며, 여러 개의 XML 디스크립터(Descriptor) 파일들로 구성되고, 이들 파일들은 Document Type Definitions(DTD) 포맷을 따른다.

3.4 어플리케이션 계층(Application Layer)

SCA는 어플리케이션 소프트웨어의 portability와 dynamic instantiation이 매우 중요하기 때문에,CORBA 기반의 객체지향 오픈 아키텍처를 규정하고 있다. 즉 하드웨어에 독립적인 응용 소프트웨어를 위한 인터페이스를 지향하고 있다.

어플리케이션은 (그림 2)의 SCA 소프트웨어 구조에서 CF IDL 상위의 user-oriented 한 "non-core" 어플리케이션을 말한다. 이는 "core"의 상대적인 개념으로서의 의미일 뿐이고 이 역시 중요한 부분이다. 이는 모뎀 디지털 신호처리 (modem-level digital signal processing), 네트워크 프로토콜 처리 (network-level protocol processing), 라우팅 (routing), 외부 액세스(external access), 보안 (security), 내부의 유틸리티 (embedded utility)를 포함하는 광범위한 통신 기능을 수행한다. 모든 non-core 어플리케이션은 CF의 Resource와 Device 클래스의 하부 클래스(subclass)들로 구성된다. 또한 Resource 는 CF의 기본 응용 인터페이스를 상속 받는다.

다수의 Resource들로 구성된 "non-core" 어플리케이션은 하나의 무선 도메인에서의 Programmable Modular Communication System (PMCS) 참조 모델에 근거하여 기능별로 구분한다면 크게 *Modem Device, I/ODevice, SecurityDevice, Network Resource, LinkResource, UtilityResource* 등으로 나눌 수 있다.

3.5 Adapters

Adapter는 CORBA 도메인에서 CORBA 기반으로 작성되지 않은 요소들을 지원하는 *Resources* 혹은 *Devices*로서, CORBA 기반으로 작성되지 않은 컴포넌트 혹은 디바이스들과 CORBA 기반 요소들간의 변환을 위해 사용한다. Adapter들은 특히 CORBA 기반으로 설계되지 않은 Modem, Security, Host 처리 요소의 지원에 유용하다.

IV. 주요 기관의 활동 상황

4.1 JTRS(Joint Tactical Radio System)

JTRS 프로그램은 1997년 처음으로 시작되었으며 waveforms에 대한 범위성(scalability), 이식성(portability) 및 확장 가능성(extensibility)에 대한 요구사항 만족을 위한 공통적인 소프트웨어 통신 구조의 연구 개발을 목적으로 하고 있다. 현재와 미래의 군사적인 요구사항에 적합한 공통적이고 개방적인 구조를 개발하기 위하여, JTRS JPO는 프로그램은 개발단계를 여러 단계로 나누어 시작하였다. 즉, 1단계에서는 구조적인 프레임워크 개발, 2단계에서는 아키텍쳐 개발, 3단계에서는 공통 서비스 요구사항을 만족하는 waveform 소프트웨어 개발로 나누어 진행되었다.

공개 경쟁을 통하여 JTRS JPO는 세 개의 업계 콘소시엄들이 초기 구조를 만들도록 하였다. 세개의 콘소시엄은 대학교를 포함하여 총 30개의 회사가 포함되었으며 Boeing, Motorola와 Raytheon이 초기 1단계 개발을 각각 주도하였다.

현재까지 Speak Easy, Have Quick, DMR(Digital Modular Radio), MBITR(Multi-Band Intra Team Radio) 와 같은 시스템을 개발해 온 경험을 바탕으로 200여가지 형태의 Radio 를 고려하고 있다 [3].

4.2 SDR Forum

SDR Forum 은 1996년 3월 MMITS (Modular Multifunction Information Transfer System) Forum 을 시초로 1998년 12월 SDR Forum으로 개명하여 SDR 기반 이동 통신 시스템 구조의 국제표준화를 위하여 활동하고 있으며 사실상 SDR 기술의 표준화를 주도하고 있는 단체이다.

현재 SDR Forum의 Technical Committee는 상기와 같은 6개의 워킹그룹과 1개의 Ad Hoc 그룹으로 분류되어 있고 2003년 미국 NASA의 요구에 의해 SDR Space Requirements Study Group이 포함되었다. 이 그룹은 2003년 11월 미국 올랜도에서 개최된 36차 SDR Forum General Meeting 에서 첫번째 미팅을 가졌고 NASA, US JTRS JPO, US Air Force가 참여하여 우주선 등의 시스템 적용에 대하여 논의하고 있다 [2].

4.3 OMG

OMG는 1989년 4월에 3COM Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems 및 Unisys Corporation 등 11개의 회사로 시작하였으면, 현재 800개 이상의 회사들을 회원으로 거느리고 있는 세계에서 가장 큰 소프트웨어 개발 컨소시엄이다 [6].

OMG내의 SDR관련 DTF(Domain Task Force)인 SBC (Software-Based Communication)는 software-defined communication device의 개발, 전개, 운영 및 관리를 지원하는 표준 명세의 개발을 위하여 구성되었으며 다음과 같은 목표를 가지고 다른 OMG그룹들과 작업을 하고 있다.

- Software radio 기반의 응용 프로그램의 개발, 전개, 관리에 사용하는 표준 도구의 제공
- Radio network, 서비스, 접근 노드및 터미널에 대한 재구성(reconfiguration) 능력의 제공
- Software radio 기반의 소프트웨어 지원을 위한 플랫폼 독립적인 표준 구조의 제공
- 응용 프로그램에서 사용할 수 있는 radio 기반의 표준 서비스의 개발의 촉진

- 응용 프로그램 개발에 사용할 수 있는 radio기반의 표준 인터페이스 정의 개발의 촉진
- 다음과 같은 해결책에 필요한 인프라구조 및 인터페이스 정의의 촉진
  - 투명한 보안 해결책 (Transparent security solutions).
  - 안전이 중요한 해결책 (Safety critical solutions).
  - 장애 감내 해결책 (Fault tolerant solutions).
  - 실시간 및 내장형 해결책 (Real-Time and embedded solutions).
  - Software radio와 관련되어 만들어진 표준 명세의 호환성 및 일관성의 보장

2004년 9월 미국에서 제1회 Software-Based Communication Workshop을 "From Mobile to Agile Communications"라는 주제로 개최하였다.

Waveform개발에 활용과 SDR간 waveform들의 이식성 촉진을 위한 Radio 인프라구조 기능을 위하여 SWRADIO Components용 플랫폼 독립적인 모델(PIM: Platform Independent Model)과 CORBA 플랫폼 종속적인 모델(PSM: Platform Specific Model)에 대한 명세 초안이 2004년 5월에 채택되었다. 이 명세는 크게 SWRadio용 UML Profile, PIM과 PSM에 대한 내용으로 구성되어 있다.

4.4 ITU

ITU(International Telecommunication Union)에서는 SDR 표준화를 위해 ITU-R(Radio communication)을 중심으로 활동을 하고 있다.

ITU-R에서는 SDR 에 대한 권고사항을 개발하기 위해 Study Group 8(Mobile, radio determination, amateur and related satellite services)에서 연구 중이며 내부 Working Party 8F 에서는 이미 IMT-2000 과 관련된 SDR 의 PNDR(Preliminary New Draft Recommendation)의 작성을 시작하였다[7].

V. 주요 벤더의 제품

5.1 CRC(Communication Research Center)

캐나다 CRC(Communication Research Center)의 Advanced Radio System Research(RARS) 그룹에서는 SDR Forum의 후원으로 SCA 규격의 참조 구현(SCARI : SCA Reference Implementation) 프로젝트를 2000년에 처음으로 실시하였고, 이를 통해 실제 구현 입장에서 여러 가지 문제를 고려하면서 SCA 코어 프레임워크의 수정 보완을 추진하였다[8]. SCARI를 통해 SCA 규격의 기술적인 부분의 구체적이고, 체계적인 분석과 인터페이스로 정의된 많은 부분들의 실제 behavior들이 정의되었다. SCARI 프로젝트는 SCA2.1 규격을 기반으로 만들어졌으며, 약 60,000라인의 코드와 30개의 예제 프로그램과 UML 시퀀스 다이어그램을 포함하는 300 페이지 이상의 기술문서로 이루어져 있다. 또한 CRC는 SCARI를 통하여 SCA 규격의 CF의 문제점을 설명하고 변경 제안을 JTRS에 제출하였다. SCARI 프로젝트의 모든 결과(소스, 기술문서) 등은 CRC 홈페이지에서 무료로 이용할 수 있다. 개발 환경 및 개발 툴의 변화로 SCA의 CF 성능을 발전시켜 가기 위해 SCARI 프로젝트는 현재 SCARI2, SCARI-Hybrid, SCARI++ 로 확장되어 계속해서 진행되고 있다.

5.2 Harris 와 Spectrum Signal Processing

미국의 주요 방위업체인 Harris 에서는 이미 자사의 dmTK(tm)(Domain Manager Tool Kit) 라는 SCA 관련 제품을 상용으로 판매하고 있으며 Spectrum Signal Processing 사에서는 이를 사용하여 SDR-3000(tm) 시리즈를 출시하였다 [9].

5.3 PrismTech

실시간 내장형 CORBA 제품인 e*ORB 로 우리에게 더 친숙한 미국의 PrismTech 사에서도 SPECTRA(tm) 라는 자사의 SCA 관련 제품을 상용으로 판매하고 있다 [10].

VI. 맺음말

본 고에서는 SDR 기술의 핵심 요소 기술로 자리잡고 있는 SCA 에 대해 살펴 보았고 SDR Forum, OMG, ITU-R 과 같은 표준화 기관 및 업체의 동향에 대해 알아 보았다. 앞서 언급한 것과 같이 다양한 하드웨어 플랫폼을 하나의 소프트웨어 플랫폼을 이용하여 제어할 수 있다면 그 위에서 개발되는 소프트웨어는 이식성(Portability)을 비롯한 많은 장점을 가질 수 있다. 이는 또한 SDR 을 실제 구현함에 있어서 필수 요구사항으로 이야기 되고 있고 SCA 가 그 자리를 메우고 있다. 세계 각국의 이동통신분야에서는 크게 기지국 시스템과 이동 단말 시스템 영역으로 나뉘어 SCA 를 적용하고 있으며 현재 ETRI 에서는 이러한 기술동향에 따라 이미 다중모드를 지원하는 기지국 시스템을 SCA 구조를 기반으로 개발하고 있다. SCA 구조 기반의 개발전략은 이동 단말의 영역으로까지 확장될 것이다.

[참 고 문 헌]

[1] Walter Tuttlebee, Software Defined Radio: Origins, Drivers and International Perspectives, pp. 62-66, 279, 298, John Wiley & Sons, Ltd., 2002.

[2] SDR Forum, http://www.sdrforum.org

[3] Joint Tactical Radio System(JTRS), http://www.jtrs. saalt.army.mil/

[4] Modular Software-programmable Radio Consortium Software Communication Architecture Specification MSRC-5000SCA V2.2, Joint Tactical Radio System, Nov. 17, 2001.

[5] P.A. Eyermann and M.A. Powell, Maturing the Soft-ware Communications Architecture for JTRS," Proc. of MILCOM, pp. 28-31, Vol. 1, Oct. 2001.

[6] Object Management Group (OMG), http://www.omg.org

[7] ITU-R SG 8 - Mobile, radio determination, amateur and related satellite services, http://www.itu.int/ITU-R/study-groups/rsg8

[8] CRC, http://www.crc.ca/

[9] Spectrum Signal Processing, http://www.spectrumsignal.com/

[10] Prism Tech, http://www.prismtechnologies.com/

[11] POSIX 1003.13: Standardized Application Environment Profile-POSIX Realtime Application Support (AEP), IEEE Std 1003.13-1998.

오상철

1995년 광운대학교 전자통신공학과(학사)
1997년 광운대학교 전자통신공학과(석사)
1997년 ~ 2000년 대우통신(주)
2000년 ~ 현재 한국전자통신연구원 이동통신연구단 SDR연구팀 선임연구원
관심분야 : 3G/4G 이동통신망, SDR, 핸드오버, 유무선 프로토콜

김홍숙

1994년 서강대학교 컴퓨터학과(학사)
1996년 서강대학교 컴퓨터학과(석사)
1996년 ~ 1998년 현대정보기술(주) 정보기술연구소 선임연구원
2003년 한국정보통신대학교 공학부 Computer System and Theory Track (박사)
2001년 ~ 2004년 엔솔테크(주) 기술연구소 BIT S/W개발실장
2005년 ~ 현재 한국전자통신연구원 이동통신연구단 SDR연구팀 선임연구원
관심분야 : SDR, 병렬처리, 컴파일러

김진업

1978년 ~ 1985년 고려대학교 전자공학과 (학사)
1985년 ~ 1987년 KAIST 전기및전자공학과 (석사)
1990년 ~ 1996년 KAIST 전기및전자공학과 (박사)
1987년 ~ 현재 한국전자통신연구원 이동통신연구단 팀장/책임연구원
관심분야 : SDR, 무선접속기술, Data Compression, Channel coding