유비쿼터스 환경에서의 서비스 통합을 위한 SCG 엔진과 ESB 적용에 관한 연구

정덕원○ 민덕기
건국대학교
{dwchung○, dkmin}@konkuk.ac.kr

A Study on Service Oriented Integration Using SCG Engine and ESB for Ubiquitous Environment

요 약

분산된 서비스의 통합의 중요성이 커지고 있는 최근의 유비쿼터스 환경에서 SOA(Service Oriented Architecture)의 구현기술인 웹 서비스는 이러한 분산된 서비스를 연계하는 표준기술로서 기업내,외부뿐만 아니라 방송/통신 융합, 정보가전/홈네트워크, 임베디드 환경 등 다양한 분야에서 사용되어 지고 있다. 본 논문에서는 홈네트워크에서 사용되는 디바이스등을 상호 이질적인 플랫폼이나 프로토콜에 유연하게 제공하기 위한 서비스 변환 게이트웨이 역할을 하는 SCG 엔진(Service Convergence Gateway Engine)과 서비스 연계의 표준기술로 자리잡고 있는 웹 서비스를 기반으로 SOA를 지원하고 S/W Service와 Application Component간의 연동을 위한 경량화된 Backbone의 역할을 수행하는 ESB(Enterprise Service Bus)를 사용하여 서비스를 통합하고 관리하는 구조를 제안 한다.

1. 서 론

유비쿼터스 환경에서 분산된 서비스의 통합의 중요성이 대두되고 있는 최근, 웹 서비스는 이러한 분산된 서비스를 연계하고 통합하는 표준 기술로서 광대역 통합망 기반의 유무선 통합 응용, 방송/통신 융합, 정보 가전/홈 네트워킹, 임베디드 환경 등 다양한 분야에서 핵심 연동 기술로 그 활용 범위가 빠르게 확산되고 있다. 홈 네트워킹이 발전되면서 생활 주변의 가전기기는 기존의 기기에서 제공하는 기능뿐만 아니라 추가적인 향상된 기능을 요구하고 있다. 이런 요구 사항들을 충족하기 위하여 상호 이질적인 플랫폼이나 디바이스들에서 제공되는 서비스들 사이의 통합과 관리 모니터링이 요구되어지고 있다. 이에 표준 통신 프로토콜이 제시되어 다양한 플랫폼에서 서비스가 제공될 지라도 서로 통신을 할 수 있게 하며 표준 프로그래밍 인터페이스를 제시하여 기존에 존재하는 다양한 프로그래밍을 위한 모든 수단을 이용하여 개발을 할 수 있어야 하고 마지막으로 표준 레퍼런스 저장소와 같은 다양한 플랫폼에서 다양한 언어로 개발되어진 서비스를 상호간에 이용하기 위한 표준들을 제공함으로써 상호 운영성이나 연계 통합을 가능케 임베디드 SCG 엔진을 개발하였다. 본 논문에서는 유비쿼터스 환경에서의 분산된 서비스 통합을 위하여 SOA[1,2]를 지원하는 ESB[3] 내에 Service Engine으로 SCG 엔진을 탑재하여 자 다양한 통합 기술을 활용하면서도 시스템간 연동을 위한 대형 도로를 제공해주는 ESB 기존의 기능과 함께 SCG의 기능을 추가하여 보다 능동적이고 유연한 서비스 통합과 관리를 위한 구조를 제안한다. 또한 본 논문에서 제시하는 구조에서 ESB는 JBI(Java Business Integration)[4] 표준 기반이어서 가정 내 홈네트워크 뿐만 아니라 가정 과 가정의 서비스까지 다른 작업 없이 통합하고 관리할 수 있는 장점이 있다.

2. 관련연구

본 논문에서 제시하고 있는 SCG엔진과 ESB를 사용한 구조는 유비쿼터스 환경에서 다양한 디바이스에 존재하는 기존의 서비스들을 동일한 서비스 형태로 인식함으로서 상호운영성을 지원토록 하고 Lightweight ESB를 사용함으로서 외부 네트워크에 연결되어 서비스를 통합이용이하게 한다. SCG엔진은 상호 이질적인 플랫폼이나 프로토콜을 사용하는 기존 서비스들을 통합하며 이들 서비스들을 임베디드 디바이스와 같은 제한된 경량 컴퓨팅 환경에서 제공할 수 있도록 하고 있다. 기존에 구성되어 있는 서비스들의 컴퓨팅 환경을 통합하여 웹서비스의 환

경으로 변환하기 위해 SCG는 실제 서비스를 관리하는 Service Container와 Service Container에 저장되어 있는 서비스들에 관련된 정보들을 관리하는 Meta Service Container를 분리하여 사용한다. SCG는 두 컨테이너 사이에 커넥터를 이용하여 통신함으로써 어떠한 Service Container 일지라도 커넥터를 이용하여 이 Service Container에 있는 서비스를 사용할 수 있다.

또한 외부 네트워크상의 서비스의 통합이 용이한 이유는 기존의 벤더들마다 각자 자신의 스펙대로 만들던 ESB와는 달리 자바 진영의 ESB 표준으로 자리잡고 있는 JBI를 기반으로 하고 있기 때문이다. 현재 오픈소스 ESB 동향을 살펴보면 Apache의 ServiceMix[5]와 같은 JBI 구현측과 Mule[6]과 Open ESB[7]의 JBI 미구현측으로 되어 있는데, 먼저 JBI 구현측은 기본적으로 JSR 208 JBI 스펙을 구현한다. 반면 JBI 미구현측은 JBI 스펙 구현을 직접 하지 않고 다른 JBI 구현체와 결합할 수 있는 방식으로 되어있다. JBI 미구현측이 ESB를 제대로 하지 않는 것은 아니고 다만 JBI 스펙은 2005년에 완성됐기 때문에 그 전부터 ESB를 하는 경우에는 자체적인 방식으로 구현했을 뿐이다. 또한 JBI 구현체는 ESB를 기반으로 하고 그 위의 확장으로서 ESB의 정체성을 찾는 접근도 있다. 초창기 비 JBI 구현측에 속하는 Mule과 Open ESB 역시, 현재 JBI를 따르고 있다. 이러한 JBI는 다음 그림 1과 같은 구조로 되어있으며 JBI는 SOA 환경 구축에 있어 가장 중요한 역할을 담당하게 될 ESB를 위한 자바 기반의 아키텍처이며 서비스들간의 전송 레벨의 바인딩을 담당하기 위한 BC(Bindigo Component)와 비즈니스 로직과 그 외 다른 서비스의 구현부분인 SE(Service Engine), 메시지의 표준화를 위한 NMR(Normalized Message Router)등으로 구성된다.

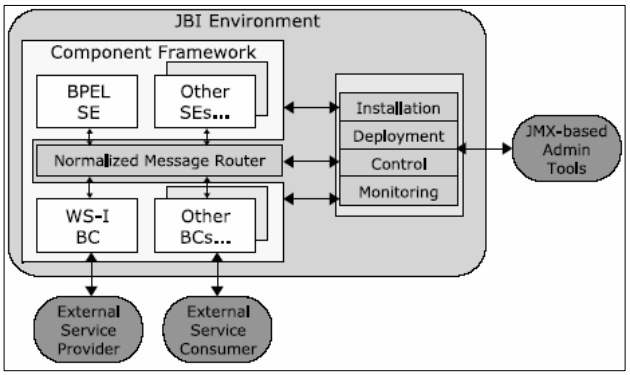

그림 1 JBI Architecture

3. SCG & ESB 시스템 구조

본 절에서는 본 논문에서 제안하고 있는 SCG & ESB 시스템의 구조를 설명한다. SCG는 다음 그림 2와 같이 표준화된 구조를 제공하기 위하여 전체 시스템을 크게 2개의 Layer로 나눈다. 첫 번째 Layer는 다양한 통신 매체를 이용하여 디바이스의 접근을 처리하기 위한 Protocol Binding Layer이다. 이 Layer에서는 유비쿼터스 환경의 디바이스를 찾거나 시스템으로 접근하는 디바이스와 통신하고 이를 웹 서비스로 변환하기 위한 프로토콜 변환 작업을 수행한다. 두 번째 Layer는 Protocol Binding Layer에서 변환 처리되어 넘겨진 각종 요청을 실제로 처리하는 Service Layer이다. 이 Service Layer는 두 가지 형태의 서비스를 관리한다. 하나는 실제로 프로그래밍되어 SOA 서비스 컨테이너에 적재된 실제 서비스이고 다른 하나는 디바이스의 서비스를 변환하여 등록한 가상의 서비스이다. Service Layer는 이 두 가지 형태의 서비스를 관리함과 동시에 웹 서비스의 요청을 위한 WSDL[8]을 생성하며 요청이 있을 때에는 호출 시퀀스에 따라 서비스를 호출하여 그 결과를 반환하는 역할을 수행하고 전체 시스템에 대한 관리를 한다.

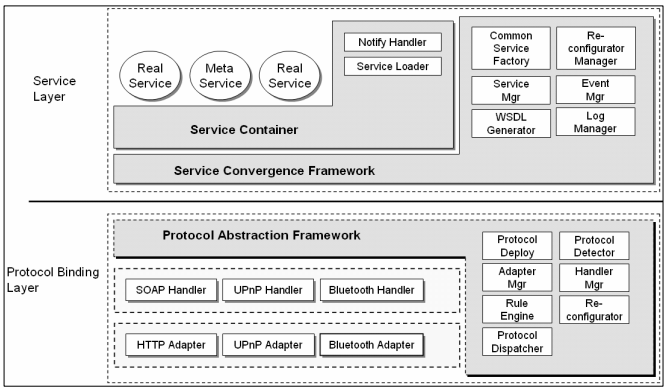

그림 2 SCG Architecture

또한 다양한 프로토콜을 지원하는 디바이스를 인식하고 통신하기 위하여 IAdapter와 IHandler의 두 가지 구조를 지원하며 언제든지 새로운 형태의 프로토콜을 지원하기 위하여 Adapter와 Handler를 개발하여 추가할 수 있는 구조를 가진다. 이렇게 하여 언제 어디서든지 원하는 종류의 프로토콜을 선택하여 해당 디바이스의 서비스를 웹 서비스화 할 수 있고 새로운 프로토콜이 변경 또는 추가되었을 때에 동적으로 새로운 Adapter와 Handler를 추가 또는 다시 배치하여 새로운 환경을 반영할 수 있도록 한다. 그리고 각각의 Adapter와 Handler는 AdapterMgr과 HandlerMgr을 통하여 동적인 추가/삭제 및 재적재 등을 실시간으로 수행 할 수 있도록 하여 실행 시에 동적인 프로토콜 바인딩을 가능하게 한다.

ESB는 Apache의 공개 프로젝트인 ServiceMix를 기반으로 하는 구조를 제시한다. ServiceMix는 2장에서 언급했던 바와 같이 JBI 기반의 ESB이다. 다른 프로젝트들에 비해 가장 JBI와의 연관성을 확실히 한 프로젝트이다. 그림 3의 ESB 구성도를 보면 JBI 구조와도 아주 비슷하게 구성되어 있다. 플러그인 구조에 위에는 Service Engine과 아래는 Binding Component가 들어가기 때문이다. ServiceMix는 실제로 Service Engine과 Binding Component로 올 수 있는 것들을 정의하고 구현하였으며 본 논문에서 제시하고자 하는 SCG엔진과의 결합이 Service Engine에 탑재되게 된다. 또한 비즈니스 프로세

스의 처리를 하는 BPEL과 XML 변환의 표준인 XSLT, 룰 엔진, 동적 스크립트 등이 포진해 있고, SOA환경의 표준 통신 프로토콜로 대표되는 SOAP[9] 등이 Binding Component로 구성되어 있다.

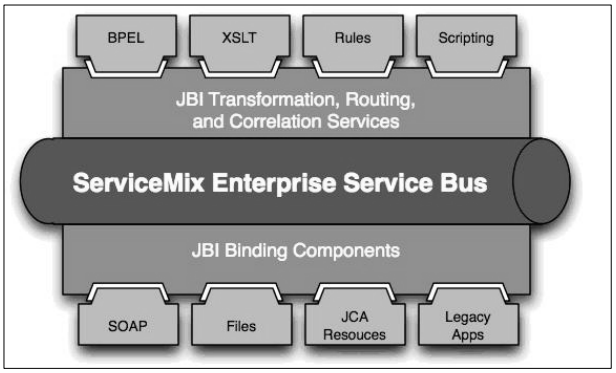

그림 3 ServiceMix Architecture

위에서 설명한 SCG엔진과 ESB를 기반으로 다음 그림 4와 같이 ESB에 플러그인 형식으로 SCG엔진을 탑재하여 분산 환경에서의 다양한 디바이스를 서비스로 인식하여 이러한 서비스를 통합함과 동시에 경량화된 구조로 제시함으로써 홈네트워크 상의 셋탑박스 형태로 구현하여 임베디드 환경에서 구동이 가능하도록 한다. 또한 기존 SCG 엔진만 사용 시 같은 네트워크상에서만 가능하던 통합 포인트를 분산된 서비스 컴포넌트를 쉽게 통합 연동할 수 있고 신뢰성 있는 메시지 통신이 가능한 ESB에 플러그인 함으로써 가정과 가정까지의 서비스 통합을 가능케 할 수 있다.

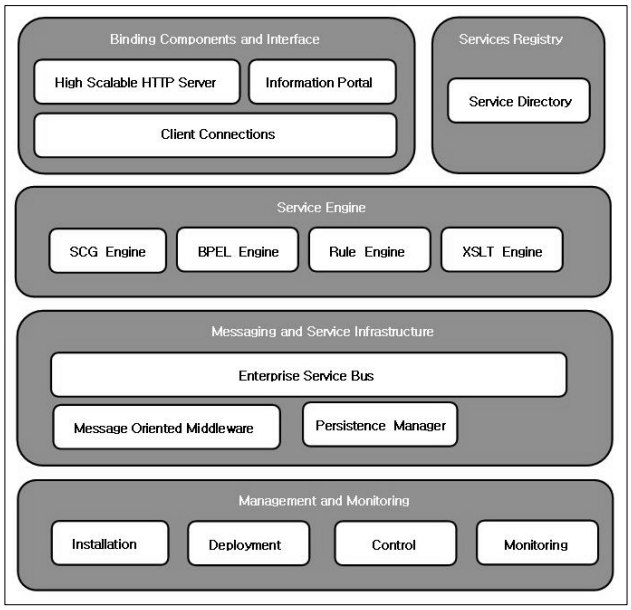

그림 4 SCG & ESB Architecture

Binding Component와 Interface

설치되어 있는 서비스들 간의 전송 레벨의 바인딩을 담당한다. 따라서 외부 시스템과 표준 전송 프로토콜로 원격 통신하기 위해 혹은 ESB 엔진 내부에서 실행 중인 두 서비스들 간의 호출을 위해 주로 사용되며, 그 외에도 WS-I 프로파일에서 사용하는 서비스들 간의 통신에도 이 바인딩 컴포넌트는 전송 수준의 서비스를 제공한다.

Service Engine

비즈니스 로직과 그외 다른 서비스들의 구현 부분이다. 이들 SE 컴포넌트들은 내부적으로 매우 다양한 기술과 설계에 의해 구현될 수 있다. 따라서 이것은 아주 간단한 변환 로직이 담긴 컴포넌트일 수도 있으며 BPEL 같은 매우 복잡한 비즈니스 프로세스가 처리되는 로직이 구현될 수도 있다. 이 서비스 엔진에 프로토콜 서비스 변환 로직이 들어있는 SCG엔진이 플러그인 된다. SE와 BC의 분리를 통해 비즈니스 로직은 그 로직의 호출에 필요한 상세한 정보와 분리됨으로써 비즈니스 로직에 전념할 수 있는 효과를 가져온다. ESB 내의 모든 서비스 제공자와 소비자는 완전히 분리되어 있으며 이들은 서로 WSDL을 통해 연결된다. WSDL을 사용하여 SE와 BC 컴포넌트 사이를 연결하며 이러한 연결의 밑바탕에는 NMR이라고 하는 표준 메시지 라우터가 존재하여 이들 간의 느슨한 연결에 의한 통신을 중재한다.

Messaging and Service Infrastructure

신뢰성 있는 메시지 전송을 위하여 MOM의 Queue를 사용한 JMS 통신을 한다. 이때 퍼시스턴스를 위하여 DB나 파일을 사용하여 in-memory 상태의 퍼시스턴스 유지가 아닌 네트워크가 끊어지는 상태나 시스템의 오류로 인한 상황에서도 완벽하게 신뢰성있는 메시징을 제공한다. 이를 위한 JMS 미들웨어와 퍼시스턴스 관리자가 존재한다.

Management and Monitoring

컴포넌트들을 설치, 적재하고 관리하며 감시하는 기능으로서 JMX 기반으로 구성되어 있다. 기존 SCG엔진에서 미구현되어있는 관리 및 모니터링 기능을 ESB의 관리를 이용하여 각 서비스의 운영 및 상태체크등을 가능케 한다.

Service Registry

서비스의 위치정보를 저장한 디렉토리이며 사용자 원하는 서비스를 알고 있다면 사용하지 않지만 원하는 서비스의 위치정보를 모른다면 서비스를 찾기 위하여 디렉토리를 검색한다.

다음 그림 5는 본 논문에서 제시하는 구조를 기반으로 사용하는 상황을 예상한 도식화 한 것이다.

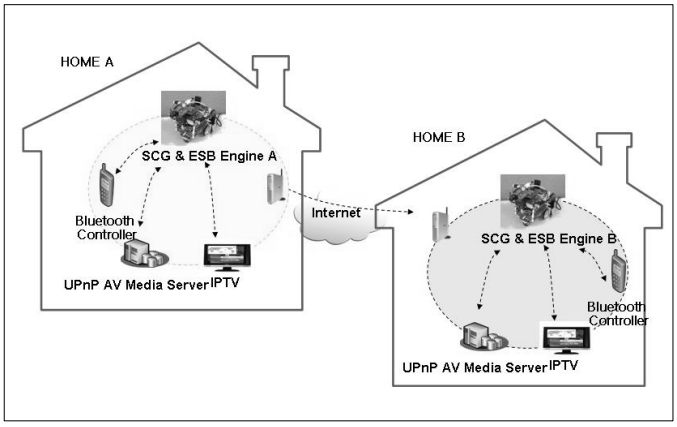

그림 5 외부 서비스 연계

기존의 SCG엔진으로는 불가능하였던 외부 디바이스 서비스와의 연계도 가능하게 해준다. 또한 서비스의 관리 및 모니터링의 부족 부분을 ESB의 관리 파트가 커버한다. 따라서 사용자는 원하는 서비스를 가정내 뿐만 아니라 외부 어디에서도 연계할수 있으며 서비스의 모니터링을 통하여 상태 체크 및 운영이 가능하게 된다.

4. 결론

본 논문에서 제시하고 있는 SCG엔진과 ESB를 사용한 서비스 통합은 최근 빠르게 발전하고 있는 유비쿼터스 경에서 상호운용성의 문제를 해결하고 유비쿼터스 환경에 존재하는 다양한 프로토콜을 지원하는 디바이스의 서비스를 SOA의 기술인 웹 서비스로 변환하여 일반 디바이스를 서비스의 레벨로 추상화시킬 수 있도록 할 뿐만 아니라 내부를 뛰어 넘어 외부 어디에 있는 서비스들과 연계도 가능하게 해준다. 또한 실행 환경을 임베디드의 환경에 맞추어 우리 주변의 어떤한 디바이스이던지 포팅하여 사용가능하게 한다.. 이를 위하여 기존에 존재하는 서비스 컨테이너의 기능을 갖추고 이에 다양한 기능을 가진 경량화된 구조를 제시하였다. 이로써 기존에 존재하는 다양한 프로토콜의 통합을 위한 추가적인 기술의 개발 없이도 서로간의 연결이 자유롭게 이루어질 수 있다. 그리고 기존의 각종 프로토콜에게 제공하기 어려웠던 서비스를 간단하게 제공한다. 그러나 SCG엔진에서 아직 구현전인 HAVi 나 Jini 등과 같은 프로토콜이나 그 외 다른 프로토콜들이 제공될 수 있도록 하는 것과 보안 문제가 향후과제가 될 것이다. 향후 본 논문에서 제시한 구조를 기반으로 시스템을 구현하여 테스트 및 성능 측정을 통하여 문제점을 파악, 보완할 것이다.

5. 레퍼런스

[1] Software-Oriented Architecture at IBM Web Site, http://www-306. ibm.com/software/solutions/soa/
[2] James Pasley, How BPEL and SOA are changing Web services development, Cape Clear Software, 2003, IEEE CNF
[3] Min Luo, Goldshlager, B., Liang-Jie Zhang, "Designing and implementing Enterprise Service Bus (ESB) and SOA solutions", Services Computing, 2005 IEEE International Conference on, Vol2, July 2005
[4] JBI, http://www.jcp.org/en/jsr/detail?id=208
[5] ServiceMix, http://www.servicemix.org
[6] Mule, http://mule.codehaus.org/
[7] Open ESB, https://open-esb.dev.java.net
[8] WSDL , http://www.w3.org/TR/wsdl
[9] SOAP MTOM spec, http://www.w3.org/TR/soap12-mtom