韓國컴퓨터情報學會 論文誌
第14卷 第11號, 2009. 11.

2009-14-11-4-4

OSGi 환경에서의 동적 웹서비스 조합 기법

고 성 훈*, 김 은 삼**, 이 춘 화***

Dynamic Web Service Composition Support for OSGi Environments

Sunghoon Ko *, Eunsam Kim **, Choonhwa Lee ***

요 약

OSGi는 서비스 레지스트리를 통한 런타임 서비스 검색을 지원함으로써 서비스 간의 인터랙션을 진작하는 대표적인 SOA 기술이다. OSGi와 더불어 SOA의 선두 그룹을 형성하고 있는 웹 서비스 기술은 광역 인터넷에 위치한 네트워크 서비스들이 개방 인터페이스에 따라 상호 작용하여 새로운 부가 기능을 지원할 수 있도록 하는 토대를 제공하고 있다. 본 논문에서는 OSGi 프레임워크 환경에서 웹 서비스 조합 언어인 WS-BPEL을 이용하여 OSGi 서비스와 웹 서비스를 합성하여 복합 서비스 및 응용을 구성할 수 있도록 하는 새로운 기법을 제안한다. 이 결과 랜 환경을 주 대상으로 하는 OSGi 서비스와 광역 엔터프라이즈 환경의 웹 서비스가 소속 도메인 경계를 넘어 서비스 조합에 참여할 수 있게 됨으로써 단일 도메인 내 조합에서는 가능하지 않았던 응용 기능 지원이 가능해진다.

Abstract

OSGi enables services to be dynamically discovered through its service registry for fostering interactions among themselves, positioning itself as one of the most prominent SOA technologies. Web Services also provide a mature technical base of open business services being employed over the Internet and allow more value-added applications to be built up from component services. In this paper, we propose a new architecture, built on the concept of dynamic service binding, to support interbred service compositions of OSGi and Web Services. Web Services are imported into OSGi domains, and the compositions are described in WS-BPEL language. The support for crossbred compositions of OSGi services and Web Services opens up a new opportunity of a wider range of applications beyond their respective traditional target domains of home gateways in LAN environments and business applications in global Internet environments.

▸ Keyword : 서비스 조합(Service Composition), 웹 서비스(Web Services), OSGi

• 제1저자 : 고성훈 교신저자 : 이춘화
• 투고일 : 2009. 08. 10, 심사일 : 2009. 09. 16, 게재확정일 : 2009. 11. 17.
* 롯데정보통신 연구원 ** 홍익대학교 컴퓨터공학과 조교수 *** 한양대학교 컴퓨터공학부 조교수
※ 이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음 (KRF-2007-331-D00392).

Ⅰ. 서 론

SOA(Service Oriented Architecture) 개념에 따르면 서비스는 어떤 기능을 제공하는 단위 컴포넌트로 존재하고, 이들의 조합을 통해 부가 기능의 복합 서비스나 응용 프로그램을 구성할 수 있게 된다. 이때 컴포넌트는 인터페이스 정의가 공개·발견 가능한 단위 서비스를 의미한다. 이 서비스는 분산 환경에서 표준 기반의 인터페이스를 사용하여 독립적으로 구현되어 제공되고, 서비스 사용자들은 제공된 서비스를 검색, 발견, 호출하여 사용하게 된다. 이러한 동적 서비스 사용 시나리오를 지원하기 위해 SOA는 기본적으로 서비스 제공자, 서비스 레지스트리, 서비스 사용자의 세 개체로 구성된다.

종래의 SOA 기술로는 엔터프라이즈 네트워크 환경의 EAI(Enterprise Application Integration), CBD (Component Based Development), BPM(Business Process Management) 등을 들 수 있으나, 최근에는 인터넷 웹 서비스(Web Services) 기술과 가정용 게이트웨이로 출발한 OSGi(Open Service Gateway initiative) 서비스 플랫폼 기술이 두각을 나타내고 있다. OSGi는 서비스의 설치, 활성화, 비활성화, 제거의 동작으로 이루어지는 서비스 라이프사이클 관리를 지원할 뿐 더러 프레임워크 서비스 레지스트리를 매개로 서비스 광고, 발견, 사용이 이루어질 수 있도록 한다. 또한 메모리 사용량이 적기 때문에 가정용 네트워크나 소규모 사무실 네트워크 환경에 적합하다. 반면에 웹 서비스는 현재 엔터프라이즈 환경에서 많이 사용되는 기술로서 서비스의 요청 메시지를 통해 서비스 간에 연결이 이루어지고, 메시지 처리 후 연결이 해지되는 느슨한 결합 방식의 구조를 갖는다. 서비스 연결 시 XML 기반의 SOAP(Simple Object Access Protocol) 메시지를 사용하기 때문에 플랫폼뿐만 아니라 프로그래밍 언어에 독립적이고 동기 ·비동기적인 메시지 교환도 지원한다는 특성을 갖는다. 웹 서비스는 서비스 구현에 대한 제약이 없고 HTTP 프로토콜을 이용한 서비스 간 메시지 교환이 가능하기 때문에 방화벽에 상관없이 원격지 서비스와 상호 연동이 용이하여 엔터프라이즈 환경에 많이 사용된다.

본 논문의 핵심은 지역 네트워크 지향적인 OSGi 프레임워크에 광역 네트워크를 대상으로 하는 웹 서비스 기술을 수용할 수 있는 수단을 제공하여 서비스 조합이 이루어지는 영역을 확장하고자 하는 것이다. 웹 서비스 기술 중 가장 대표적인 WS-BPEL(Web Services Business Process Execution Language)[1]을 차용하여 OSGi 프레임워크 상에서 OSGi 서비스와 웹 서비스 조합하는 방법을 제안함으로써 두 SOA 플랫폼의 서비스 통합을 기하고자 한다. 이 궁극 목표를 달성하기 위한 기술적인 접근 방법을 논하기 전에 먼저 2장에서 주요 관련 연구를 살펴본다. 3장에서는 문제 제기, 해법 제시, 프로토타입 등 논문 전반에 걸쳐 사용될 서비스 조합 가상 시나리오를 소개한다. 4장에서는 OSGi와 웹 서비스 환경에서의 서비스 조합 기술의 실태를 점검해 보고, 이어 5장에서는 본 논문에서 새롭게 제안하는 이종 도메인 서비스 통합 모델이 자세하게 다루어진다. 마지막으로 6장에서는 제안된 서비스 조합 기법의 타당성 및 가능성 검증을 위해 구현된 시스템 프로토타입이 기술된다.

Ⅱ. 관련연구

OSGi 환경에서의 서비스 조합 기법에 관한 연구는 다른 SOA 기술 특히 웹 서비스 기술의 활발한 연구에 비해 상대적으로 미진한 면을 보이고 있는데, 이는 OSGi가 기본적으로 소규모 네트워크를 목표로 설계되었기 때문인 것으로 볼 수 있다. 그러나 최근 OSGi 적용 환경이 엔터프라이즈 네트워크나 광역 인터넷으로 확장되어 감에 따라 OSGi에 기반을 둔 서비스의 조합 기법에 대한 연구의 중요성이 부각되 그러나 최 이러한 관심은 OSGi 도메인 내에서의 서비스 조합 지원에 관한 초기 연구[2][3]으로 어졌는데, 이들 연구에서는 서비스 조합을 자동화하고 조합된 서비스의 상태 모니터링과에 기 기능 제공을 우선적인 목표로 하였 서비스 다른 연구[4]에서는 서비스 조합 언어로 BPEL을 차용하고, 즉 OSGi 프레임워크 내부 서비스들의 연결과 조합을WS-BPEL로 기술하고, WS-BP동의 조정자 역할을 담당하는 BPEL 엔진이 자바 호출 서비으로 참여 OSGi 서비스들을 호출함으로써 조합 서비스가 실행될 메인 내에서의 서비그러나 이 기법은 OSGi 환경에서만의 조합을 고려하기 때문에 서비스 조합에 참여할 수 있는 서비스가 제한메인 서비스 조서비스 조합에 참여하는 모든 서비스가 디자인 타임에 정적으로 결정되어야서의 는 단점이 러나 최이와는 달기 본 논문에서는 서비스 조합에 참여하는 서비스의 영역을 OSGi 도메인호출함으로써 웹 서비스 도메인까결정 포함하도록 확장되었고, 동적인 서비스 연결을 통해 서비스가 런타임에 최종 결정되는 유연한 서비스 조합이 지원된다는 것이 기존 연구와의 주요 차별화 포인트가 된다.

본 논문에서는 위와 같이 OSGi 도메인 내로 한정된 초기 연구 경향에서 벗어나 이종 SOA 도메인간의 서비스 조합을 지원하는 방법을 탐구하고 그 해법을 제시한다. 무엇보다도 동적 서비스 바인딩이란 독창적인 기법을 제안하여 조합되는 컴포넌트 서비스가 OSGi 서비스인지 혹은 웹 서비스인지에 상

관없이 복합 서비스 생성 시점과 주어진 상황에서 최적의 서비스 조합이 가능하도록 한다. 이는 서비스 조합 관점에서 두 서비스 도메인의 실질적인 통합을 가능하게 하고, 그 결과 기존의 단일 도메인 서비스 조합에서는 가능하지 않았던 새로운 기능을 제공하는 응용을 생성할 수 있게 한다. BPM과 SOA 기술의 융합을 통해 비즈니스 프로세스의 자동화를 시도하는 연구 노력[5][6]도 거시적인 접근 방법에 있어서 본 논문과 맥락을 같이 한다고 볼 수 있다. 하지만 비즈니스 프로세스 자동화 방법론과 프레임워크 제시, SOA 기술을 이용한 프로세스의 재사용 및 시스템 확장성보다는 본 논문에서는 OSGi 서비스와 웹 서비스를 컴포넌트로 하는 WS-BPEL 서비스의 동적 조합 지원에 주안점을 두고 있다는 점에서 다르다.

OSGi 프레임워크를 확장하려는 최근의 연구 노력도 주목할 만한데, 무엇보다 OSGi 프레임워크 상에서 OSGi 서비스를 하나의 웹 서비스로 등록하는 기법인 Knopflerfish Axis port[7]가 두각을 나타내고 있다. 하지만, 이 연구는 외부 도메인에 내부의 OSGi 서비스를 제공하는 기능을 고려하고 있을 뿐 OSGi 프레임워크 기반의 서비스 조합 기법을 다루지 있지 않다. 환언하면 이 연구에서는 개별 서비스 차원에서 OSGi와 웹 서비스의 기본적인 연결 기법만을 제공하고 있음에 반해, 본 연구에서 목표로 하는 이종 도메인간 서비스 조합은 보다 지능적이고 진보된 형태의 서비스 조합을 지원할 수 있다. 그리고 가장 대표적인 SOA 기술인 OSGi와 웹 서비스 조합 시스템을 설계하고 그 프로토타입을 구현함으로써 제안 기법의 타당성과 가능성을 증명하였다는 것에 의미를 둘 수 있을 것이다.

동적 서비스 조합의 단계인 서비스 탐색과 통합에 시맨틱 웹 기술을 활용하려는 시도 또한 주목할 만하다[8][9]. 본 연구에서 제안하는 서비스 조합 시스템에서는 현재 OSGi 바인딩과 웹 서비스 바인딩만 고려하고 있다. 새로운 바인딩을 추가하여 시맨틱 웹 온톨로지로 기술된 컴포넌트 서비스도 지원할 수 있도록 확장함으로써 위 연구 결과들을 흡수하고 통합하려는 노력은 앞으로 고려해 볼 만한 연구 과제이다.

SOA 환경에서의 서비스 조합 기법으로 플랫폼 독립적이고 구조적인 서비스 모델링 기법이 SCA(Service Component Architecture) 연구를 통해 개발되고 있다[10]. SCA는 SOA에 기반을 둔 시스템과 응용을 구축하기 위한 모델을 정의하는 기술로서 비즈니스 컴포넌트들과 그들의 연결 관계 안에 비즈니스 솔루션의 구조를 정의한다. SCA의 가장 큰 특징은 컴포넌트 구현 특정 기술에 관계없이 서비스 응용을 구축할 수 있다는 것이다. 다양한 이종 플랫폼의 서비스를 혼합 조립한다는 점에서 본 논문의 제안 기법과 맥락이 유사한 면이 있다. SCA 컴포넌트는 메시지 기반의 상호 연동을 통해 데이터를 교환하는데 현재 주요 서비스/레퍼런스 타입으로는 자바 인터페이스와 WSDL 포트 타입이 정의되고 있다.

III. 공공요금 납부 시나리오

OSGi 환경에서의 웹 서비스 지원이라는 이종 서비스의 조합을 목표로 하는 본 연구의 문제 제기, 해법 제안, 그리고 제안 기법의 타당성 검증과 시연을 위한 사용 케이스로 그림 1과 같은 공공요금 자동 납부 시나리오를 고려한다. Felix란 사용자의 집에는 OSGi 프레임워크에 기반한 홈게이트웨이가 설치·운영 중에 있어 댁내의 모든 장치를 총괄 관리하고 있다. Felix의 홈 PC나 모바일 디바이스 상에서 수행되고 있는 스케줄을 관리 프로그램(To-Do list)에는 매월 납부하여야 하는 공공요금에 대한 엔트리가 존재한다. 이 공공요금의 납부 시점은 Felix가 수동으로 결정하기도 하지만 설정에 따라 요금을 청구하는 회사의 납부기한에 맞추어 자동으로 결정될 수도 있다. 납부 스케줄 서비스는 공공요금의 납부일이 되면 홈 네트워크 상의 OSGi 서비스들인 가스 계량기(Gas Meter), 전기 계량기(Wattmeter), 수도 계량기(Water Meter) 서비스에게 요금 납부 시점이 되었음을 알린다. 스케줄 서비스의 이벤트를 받은 각 계량기 서비스들은 자신의 계량 값을 각 수도/전기/가스 회사의 웹 서비스(Utility Billing Service)에 전달하고, 이 공공요금 웹 서비스는 사용량에 따른 요금을 결정한다. 이 계산 요금은 Felix가 계좌를 가지고 있는 은행의 요금 지불 웹 서비스(Payment Service)에 의해 지불된다. 이 공공요금 납부 시나리오는 홈네트워크 상의 OSGi 서비스인 가스 계량기, 전기 계량기, 수도 계량기 서비스와 인터넷 상의 웹 서비스인 수도/전기/가스 회사의 요금 청구 서비스와 은행의 지불 서비스 간의 협력에 의해, 즉 서비스 조합을 통해, 공공요금 자동납부라는 응용 프로그램 기능이 제공되는 것을 보여주고 있다.

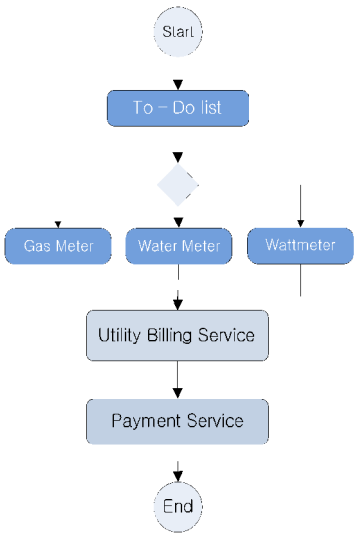

그림 1. 공공요금 납부 시나리오
Fig. 1. Utility Payment Scenario

이 시나리오는 OSGi 혹은 웹 서비스 단독으로 제공될 수 없고 두 서비스 도메인의 통합에 의해서만 제공 가능함을 나타내고자 한다. 그림 1을 보면 스케줄 서비스와 계량기 서비스는 랜 환경으로 한정되는 것이 자연스러운 OSGi 서비스이나 전기/가스/수도 회사와 은행 서비스는 인터넷 환경을 가정하는 웹 서비스이다. 계량기 서비스는 특정 계량기 하드웨어 디바이스를 나타내는 서비스로 이를 웹 서비스로 구현하는 것은 부적절하고, 반대로 공공요금 청구 서비스 및 은행 납부 서비스는 홈/사무실을 주 대상으로 하는 OSGi 서비스로 구현하는 것은 적절하지 않다. 이 공공요금 납부 시나리오는 OSGi와 웹 서비스의 통합 조합의 필요성을 제기하고, 이를 지원할 수 있는 기법의 고안, 설계, 구현 과정의 기술에 두루 사용되어 본 논문 기술의 중심이 된다.

Ⅳ. 서비스 조합 기술

본 논문의 중심 주제인 OSGi와 웹 서비스들을 OSGi 환경에서 통합하기 위한 새로운 서비스 조합 방식을 본격적으로 서술하기에 앞서 OSGi 프레임워크와 웹 서비스 환경 각각에 대해 현 서비스 조합 기술을 살펴보기로 한다.

4.1 OSGi 환경에서의 서비스 조합

OSGi 스펙은 두 서비스를 상호 연결하는 매개체로 Wire 객체를 이용한 서비스 조합 방법을 정의하고 있다. 즉, 생산자 서비스(producer service)는 Wire를 통해 소비자 서비스(consumer service)에게 데이터를 전달할 수 있다. 이 Wire 객체는 OSGi 프레임워크 시스템 서비스인 Wire Admin 서비스를 통해 체계적으로 관리된다. 그러나 Wire Admin 서비스는 Wire 객체를 통한 서비스의 연결이라는 기초적인 기능 제공에 국한된다는 제약이 있다. 이러한 문제는 서비스의 자동 조립과 유지 보수 등의 고차원의 기능을 제공하는 Plumber 시스템 서비스라는 저자들의 이전 연구에서 해결되었다고 볼 수 있다[2]. Plumber 서비스는 OSGi 환경에서 동적이고 자동화된 서비스 조합을 지원하도록 설계된 OSGi 프레임워크상의 시스템 서비스로서 시스템 관리자에 의해 주어진 XML 형식의 기술서(description)인 서비스 조합 그래프(service composition graph)를 토대로 서비스를 연결하고 연결 상태의 모니터링과 필요 시 자동 복구 기능을 수행한다.

그림 2는 Plumber 서비스에 의해 파이프로 연결되는 OSGi 서비스의 데이터 교환을 보여준다. 생산자 서비스는 소비자 서비스로 데이터 항목을 전달하기 위해 파이프 상의 update() 메소드를 이용하는데, 이 파이프는 Plumber 서비스의 용도에 맞도록 어탭터로 감싼 OSGi Wire 객체이다.

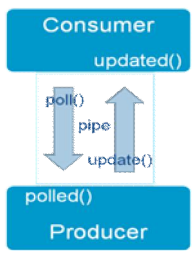

그림 2. OSGi 프레임워크에서의 서비스 연결
Fig. 2. Service Interconnection on OSGi Framework

서비스들 간의 상호 연동은 생산자 서비스가 파이프의 update() 메소드를 호출하면 파이프는 소비자 서비스의 updated() 메소드를 호출함으로써 이루어진다. 반대로 소비자 서비스가 생산자 서비스를 호출하게 될 때에는 파이프의 poll() 메소드를 호출하고, 파이프는 생산자 서비스의 polled() 메소드를 호출하여 상호 연동하게 된다. Plumber 서비스에 의해 조합된 서비스들은 이러한 방식을 통해 메시지를 교환하게 된다. Plumber 서비스는 Service Tracker 서비스, Event Admin 서비스 등의 OSGi 시스템 서비스들을 추가로 이용하여 서비스 연결을 모니터링하고 상태변화에 대응할 수 있는 기능을 구현한다.

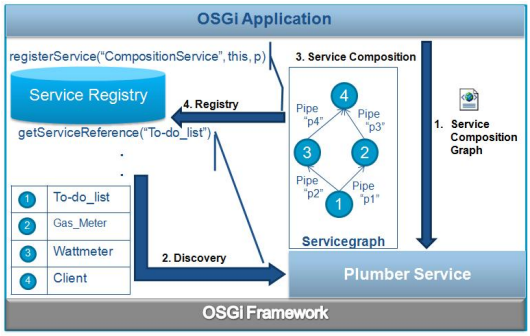

그림 3. Plumber 시스템 서비스에 의한 서비스 조합
Fig. 3. Service Composition Support by Plumber System Service

서비스 조합 그래프를 이용하여 서비스가 조합되는 과정이 그림 3에 도시되어 있다. 먼저, Plumber 서비스는 시스템 관리자 또는 상위의 응용으로부터 넘겨받은 서비스 조합 그래프를 해석하여 OSGi 서비스 레지스트리로부터 서비스 조합에 필요한 Gas_Meter, Wattmeter, To-Do_list 등의 요소 서비스들을 검색해 온다. 이 서비스들은 그림 3의 서비스 그래프와 같이 연결되어 복합 서비스(composite service)를 형성하는데, 이 복합 서비스는 OSGi 서비스 레지스트리에 등록되어 타 서비스나 응용에 의해 사용될 수 있다.

본 논문에서는 OSGi 프레임워크 상에서 서비스의 동적인 조합 및 관리 기능을 제공하던 기존 Plumber 서비스를 확장하여 서비스 조합에 OSGi 서비스뿐만 아니라 웹 서비스도 참여할 수 있도록 문호를 개방하는 이종 도메인 서비스 조합 기술을 연구한다.

4.2 웹 서비스 조합 기술

웹 서비스의 구성 요소는 서비스 제공자, 서비스 사용자, 서비스 정보의 저장소인 UDDI(Universal Description, Discovery, and Integration)의 세 개체이다. 웹 서비스의 기본적인 동작을 보면 먼저 서비스 제공자는 서비스의 기능을 요약한 XML 형태의 문서인 WSDL(Web Service Definition Language)의 주소 정보를 UDDI 등록한다. 서비스 사용자는 UDDI에 등록된 서비스의 리스트를 검색하여 필요로 하는 서비스의 주소 정보를 찾은 다음 해당 웹 서비스의 WSDL 파일을 가져온다. 클라이언트는 이 WSDL 파일을 해석하여 서비스를 자세히 이해할 수 있게 되고, 이 정보를 토대로 서비스를 호출하게 된다.

이러한 단위 웹 서비스들을 연결 통합하여 부가적인 기능을 제공하는 복합 서비스나 응용 프로그램을 구성할 수 있다. 대표적인 웹 서비스 조합 기술 언어는 앞서 간단하게 언급된 바와 같이 어떤 비즈니스 로직에 따라 순차적으로 서비스를 호출하여 서비스를 조합하여 응용을 구성하는 WS-BPEL이다. WS-BPEL은 XML 형태의 BPEL 기술서를 사용하여 단위 서비스들의 조합을 기술한다. WS-BPEL 기술서에는 서비스들의 호출 순서인 비즈니스 프로세스, 데이터를 처리하는 방식을 서술한 비즈니스 로직, 그리고 서비스에 대한 인터페이스 등을 정의하는 용도로 사용되는 파트너 링크가 중심을 이룬다. 특히 파트너 링크는 웹 서비스가 정의를 통해 다른 웹 서비스와 연결되기 때문에 중요성이 강조된다.

```
〈process name="BankTransfer" ..... "〉
<variables>
  <variable name="clientRequest"
            messageType="
            BankTransferRequestMessage"/>
   ...
<variables>
<partnerLinks>
   ...
  <partnerLink
  name="MyBankTransferService"
    partnerLinkType="BankTransferService"
    partnerRole="BankTransferServiceProvider"
    />
</partnerLinks>
  ......
<sequence>
  <receive name="receiveInput"
  partnerLink="client"
    portType=" BankTransfer"
      operation="startprocess"
      variable="clientRequest"
      createInstance="yes"/>
    ......
  <invoke
  partnerLink="MyBankTransferService"
      portType=" BankTransferService"
      operation="checkBalance"
      inputVariable="CBRequest"
      outputVariable="CBResponse"/>
      ......
  <invoke
  partnerLink="MyBankTransferService"
    portType="BankTransferService"
    operation="transfer"
    inputVariable="transferRequest"
    outputVariable="transferResponse"/>
      ......
</sequence>
  ...
</process>
```

그림 4. BankTransfer BPEL 기술서
Fig. 4. BankTransfer BPEL Description

그림 4는 BankTransfer BPEL 프로세스의 BPEL 기술서를 보여준다. 그림의 기술서를 보면 BankTansfer BPEL 프로세스의 기능은 사용자로부터 계좌 정보와 송금액을 포함

하는 BPEL 프로세스 상에 정의된 clientRequest 타입의 변수 값을 입력 받아 해당 금액을 송금하는 서비스이다. 그림의 WS-BPEL 프로세스는 컨테이너 액티비티인 〈sequence〉 엘리먼트부터 실제 프로세스가 동작하게 된다. 이 〈sequence〉 엘리먼트는 내부의 액티비티를 순서대로 실행하는 것을 의미하므로 최상단의 〈receive〉 액티비티부터 실행된다. 이러한 액티비티를 실행하는 BPEL 프로세스와 외부 서비스와의 관계는 파트너링크에 정의된다. 〈invoke〉 액티비티를 보면 MyBankTransferService 파트너링크는 기술서의 파트너링크 정의에 의해 BankTransferService 파트너타입이라는 것을 알 수 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyBankTransferService" .....
>
  <message name="debitRequest">…</message>
  .....
  <portType name="BankTransferService">
    <operation name="checkBalance"…>
      <inputmessage="CBRequest"…/>
      <outputmessage="CBResponse"…/>
    </operation>
    <operation name="transfer"…>
  .....
  </portType>
  <binding
  name="BankTransferServiceSoapBinding"
      type="BankTransferService">
  .....
  </binding>
  <service name="FooBankTransferService">
    <addresslocation="http://localhost:8080/axis/
    services/FooBankTransferService"/>
  </service>
  <partnerLinkType
  name="BankTransferService">
    <role
        name="BankTransferServiceProvider">
        <portType
            name="BankTransferService"/>
    </role>
〈role
    name="BankTransferServiceRequester"〉
    〈portType
        name="BankTransferCallback"/〉
 〈/role〉
  </partnerLinkType>
</definitions>
```

그림 5. BankTransfer 프로세스 서비스의 WSDL
Fig. 5. WSDL Descriptioin for BankTransfer Service

이 파트너링크 타입은 서비스를 외부에 공개하기 위해 작성되는 그림 5와 같은 WSDL 파일에 의해 정의된다. 그림 5를 보면 서비스 자신은 BankTransferCallback 포트 타입이고 호출 대상 서비스는 BankTransferService 포트 타입의 서비스임을 알 수 있다. 따라서 결과적으로 BPEL 프로세스는 그림 6과 같이 웹 서비스와 연결될 수 있다.

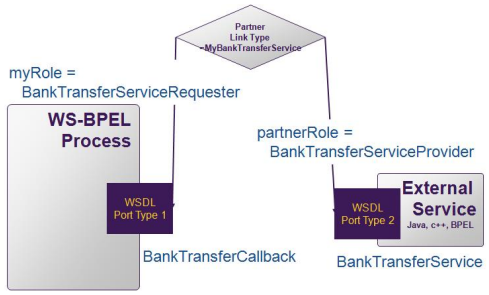

그림 6. 파트너 링크를 이용한 상호 연동
Fig. 6. Service Interaction via Partner Link

서비스의 상호연동 과정을 살펴보면 그림 5에 정의된 기술서의 파트너링크에는 서비스의 이름을 의미하는 파트너링크 이름, 서비스의 형태를 의미하는 파트너링크 타입, 그리고 해당하는 파트너링크의 역할이 정의된다. 이러한 파트너링크의 정의는 BPEL 프로세스를 위한 descriptor 파일의 파트너링크 타입정의를 통해 연결될 서비스의 WSDL의 주소를 확인하는데, 그림 7에서 이 부분을 볼 수 있다. BPEL descriptor로부터 알게 되는 MyBankTransferService 서비스의 WSDL 주소는 http://localhost:8080/axis/services/MyBankTransferService?WSDL인데, 이는 그림 5의 WSDL 파일을 나타낸다. 즉, BPEL 프로세스는 BPEL 기술서에서 파트너링크를 참조하고, 이를 이용하여 BPEL descriptor로부터 WSDL 파일의 정보를 얻고, 다시 이로부터 서비스 주소를 획득하여 서비스와 연결할 수 있게 된다.

```
<?xml version="1.0"?>
<BPELProcess id="BankTransfer"
src="BankTransfer.bpel">
  <partnerLinkBindings>
    .....
    <partnerLinkBinding
      name="BankTransferService">
      <property name="wsdlLocation">
      http://localhost:8080/axis/services/MyBan
      kTransferService?WSDL
      </property>
      .....
    </partnerLinkBinding>
  </partnerLinkBindings>
</BPELProcess>
```

그림 7. BPEL Descriptor의 WSDL 위치 정보
Fig. 7. WSDL Location Information in BPEL Descriptor

이러한 파트너링크의 사용은 BPEL 프로세스를 정적인 서비스 조합으로 한정시킨다는 제약이EL 프로즉, BPEL 프로세스의 디자인 타임에 모든 구성 서비스에 대한 파악은 물론

관계된 서비스가 기술서에 명시적으로 정의되어야 하고 런타임에 변경될 수 없 프로이 정적인 서비스 조합의 제약은 기술서 작성 후 런타임에 동적으로 실제 조합 서비스가 결정될 수 있도록 본 논문록 본새롭게 제안하는 동적인 서비스 바인딩 기법에 의해 극복될 수 있다.

V. OSGi 서비스와 웹 서비스의 통합 조합

본 연구의 핵심은 웹 서비스가 OSGi 환경에서 동적으로 서비스 조합에 참여하는 이종 도메인간 서비스 조합을 지원함으로써 보다 풍부한 서비스 및 응용 기능을 지원할 수 있도록 하는 것이다. 그림 8은 이종 도메인 서비스 조합 기법에 따른 OSGi 도메인 서비스와 웹 도메인 서비스의 연동을 도시하고 있다. 기본 아이디어는 웹 서비스 혹은 원격지 OSGi 프레임워크에서 웹 서비스 형태로 수출된 서비스를 수입하여 OSGi 프레임워크에서의 서비스 조합에 참여할 수 있도록 하여 두 도메인간의 서비스 연동을 가능하게 한다는 것이다.

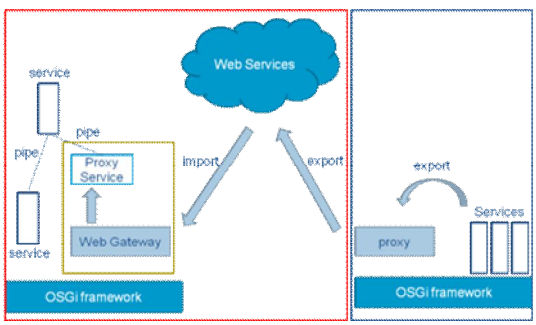

그림 8. 이종 도메인 서비스 연동
Fig. 8. Service Interactions across Heterogeneous Domains

이종 도메인간의 서비스 연동을 지원하기 위해 본 논문에서는 WS-BPEL을 차용하여 서비스 연결도를 기술하고, OSGi 프레임워크 상에서 Plumber 시스템 서비스가 가용한 실제 서비스 개체에 바인딩하는 서비스 조합 기법을 제안한다. 기존과는 달리 웹 서비스 조합용인 WS-BPEL을 서비스 조합 기술 언어로 채택하여 범용성을 확보하였다. 또한, 동적 서비스 바인딩을 제안함으로써 주어진 시점에서 가용한 서비스들 사용하여 최적의 서비스 조합이 가능하도록 할 뿐더러 서비스 조합의 유연성을 극대화한다는 점을 주요 장점으로 볼 수 있다.

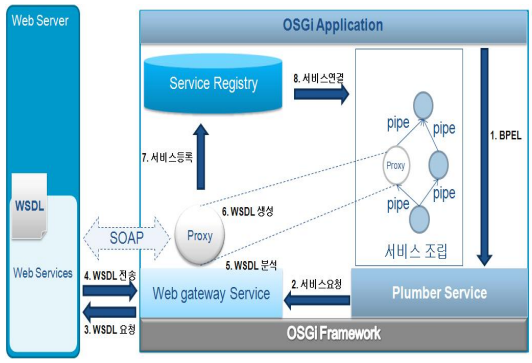

그림 9. 서비스 조합 과정
Fig. 9. Service Composition Process

그림 9는 이러한 서비스 조합 기법의 동작 과정을 전체적으로 보여주고 있다. 먼저 BPEL 기술서가 시스템 관리 응용 프로그램으로부터 전달되면 Plumber 서비스는 기술된 조합도에 따라 서비스들을 연결하게 된다. 이때 Plumber 서비스는 두 서비스 타입을 구별하여 서비스 조합 과정을 진행시키는데, 필요한 단위 서비스가 OSGi 서비스이면 OSGi 서비스 레지스트리로부터 서비스를 검색하여 조합하고, 웹 서비스일 경우에는 서비스의 WSDL 파일의 URL을 Web Gateway 서비스에게 넘겨 준다. Web Gateway 서비스는 URL이 가리키는 곳에서 다운로드한 서비스의 WSDL 기술을 해석하여 해당 웹 서비스를 나타내는 프록시를 생성하여 OSGi 서비스로 서비스 레지스트리에 등록한다. 이후 Plumber 서비스는 자신이 찾는 서비스가 등록되어 사용 준비가 완료된 것을 Web Gateway 서비스의 이벤트와 OSGi 프레임워크의 이벤트를 통해 인지하고 해당 서비스를 서비스 조합에 사용한다.

5.1 Web Gateway Service

Web Gateway 서비스는 웹 서비스의 WSDL 파일을 분석함으로써 프록시 생성에 필요한 정보를 획득하여 OSGi 서비스 형태의 프록시를 생성할 수 있게 된다. 이 생성 프록시를 OSGi 서비스 레지스트리에 등록하고 추적 관리하는 기능을 수행한다.

서비스의 동작 과정을 보면 먼저 Web Gateway 서비스가 Plumber 서비스로부터 호출되면 파라미터 값으로 전달된 URL이 지시하는 곳으로부터 웹 서비스의 WSDL 파일을 다운로드한다. 이렇게 WSDL 파일이 웹 서비스를 제공하는 웹 서버로부터 확보되면 Web Gateway 서비스의 WSDL Analyst 모듈은 웹 서비스의 WSDL을 분석하여 프록시 생성에 필요한 정보를 확보한다. 다음으로 Web Gateway 서비스의 Proxy Generator 모듈은 WSDL Analyst에 X값으획

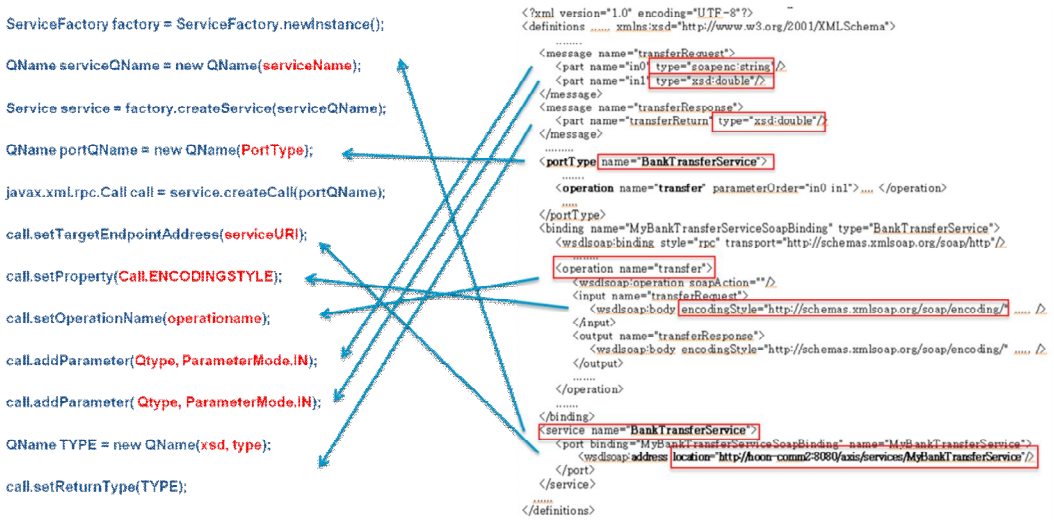

그림 10. BankTransfer 서비스를 위한 Call 객체 생성
Fig. 10. Call Object Creation for BankTransfer Service

득된 정보를 토대로 프록시를 생성한다. 이 프록시는 OSGi 서비스와X값으획득된 정에 사용되는 서비스로 웹 서비스의 프로토콜 컴포넌트와 OSGi 으획득된컴포넌트를 포함한다. 즉, 프록시 서비스는 OSGi 서비스와 웹 서비스의 연결 및 중계 기능을 제공한다. 이때 타 OSGi 서비스와 프록시의 연결은 파이프를 통해 이루어지고, 프록시와 웹 서비스의 통신은 SOAP을 이lyst이루어진다. 프록시는 웹 서비스에 대해 UR클라이언트의 역할을 수행 파일을 다운로드er 된 UR대행 서비스로의 역할을 담당한다. 마지막으로 Web Gateway 서비스의 Proxy Manager 모듈은 프록시 서비스를 관리한다. 먼저 Proxy Manager는 프록시를 해당하는 웹 서비스의 이름으로 OSGi 서비스 레지스트리에 가상 OSGi 서비스로 등록하고일을자체적으로도 프록시 서비스의 리스트를 유지한다. 그리고 Proxy Manager는 웹 서비스의 상태를 SOAP 헤T WS이lyst주기적으로 확 W받고 서비스의 업데이트b 는 유실 등의 이벤트에 대응한다. 즉, Proxy Manager는 웹 서비스의 이벤트 리스너의 역할도 수행한다.

5.2 Proxy 서비스

Proxy 서비스는 웹 서비스의 대리자로서 OSGi 프레임워크에 한 서비스로 등록되어 OSGi 서비스와 직접 연결될 수 있어야 하므로, 웹 서비스의 대리 기능을 수행하기 위해 SOAP 바인딩과 OSGi 서비스 연결을 위한 파이프 바인딩을 동시에 지원해야 한다.

Proxy 서비스는 웹 서비스 모듈, OSGi 모듈, 메시지 변환 모듈의 세 부분으로 구성된다. 먼저 웹 서비스 호출/처리 모듈은 Axis[11]에서 웹 서비스 호출을 위해 제공하는 JAX/RPC 타입의 API인 Call 객체의 invoke() 메소드를 통해 웹 서비스와 바인딩 한다. 이는 별도의 스텁 클래스나 웹 서비스의 인터페이스 없이 Call 객체를 이용한 동적 호출 인터페이스(dynamic invocation interface) 클라이언트의 호출 방식을 Proxy 서비스가 사용한다는 것을 의미한다. Call 객체의 주요 장점은 먼저 객체를 생성하고 차후에 Call 객체의 속성 설정을 통해 연동되는 웹 서비스 타입을 동적으로 변경하는 것을 지원한다는 것이다. 따라서 Web Gateway 서비스가 웹 서비스에 대한 Proxy 서비스를 생성할 때 WSDL의 URL을 제외한 어떠한 정보도 필요로 하지 않게 된다. 요약하면, Proxy 서비스는 서비스의 호출시 동적으로 Call 객체의 생성을 통하여 웹 서비스를 호출하는데 이는 내부의 웹 서비스 호출/처리 모듈에 의해 처리된다.

그림 10은 이러한 Call 객체의 생성 과정을 코드를 통해 보여준다. 그림에서 보듯이 Call 객체를 통해 서비스를 호출하기 위해서는 서비스 이름, 인터페이스, 서비스 URL, enconding 타입, 메소드 이름 및 변수(파라미터) 타입 등의 필수정보가 설정되어야 한다. 이러한 정보는 Call 객체의 인터페이스를 통해 등록되고 추후에 사용된다. Call 객체는 invoke 메소드에 파라미터 값을 설정하여 서비스를 호출하는데, 이는 Call 객체가 특정 웹 서비스에 종속되지 않고 자유롭게 속성의 설정에 따라 변경될 수 있어 본 논문의 핵심 아이디어인 동적 서비스 바인딩에 적합한 방식이다.

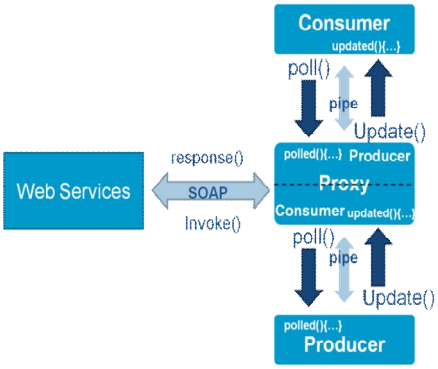

그림 11. Proxy 서비스의 메시지 처리
Fig. 11. Proxy Service's Message Processing

다음으로 OSGi 호출/처리 모듈은 Plumber 서비스의 파이프를 이용하여 타 OSGi 서비스와의 연결을 지원한다. Plumber 서비스의 파이프는 앞서 설명된 바와 같이 동적이고 자동적인 서비스 조합을 가능하게 한다. Plumber 서비스를 사용함으로써 자바 서비스 인터페이스를 직접 호출하는 OSGi 프로그래밍 모델과는 달리 서비스 연결이 자동으로 이루어지고, 연결 상태의 모니터링, 상태 변화에 따라 타 대등 서비스로의 대체 복구가 지원될 수 있다.

마지막으로 메시지 변환 모듈은 이종 서비스간의 연동을 위해 OSGi 도메인과 웹 도메인 간 데이터 메시지 타입을 변환하는 모듈이다. 메시지 변환은 웹 서비스가 Call 객체 호출을 통한 SOAP을 바인딩을 수행하고, OSGi 프레임워크는 파이프를 이용해 바인딩이 이루어지기 때문에 필요하다. 즉, 메시지 변환은 바인딩 타입 변환을 의미한다.

그림 11은 프록시의 전체적인 동작을 나타내고 있다. 그림의 메시지 처리를 보면 먼저 하위의 생산자 서비스는 상위의 프록시 서비스에게 자신이 생산한 데이터를, 즉 메시지를, 전달하기 위해 파이프 상의 update() 메소드를 호출한다. 이 파이프의 update() 메소드는 상위 프록시 서비스의 updated() 메소드를 호출하게 되고, updated 메소드는 내부에 정의된 웹 서비스 처리 모듈에 의해 웹 서비스와 상호 연동한다. 이때 하위의 서비스로부터 전달된 값이 사용되고 웹 서비스 호출의 결과 값은 상위의 소비자 서비스에게 전달된다. 이와는 반대로 poll() 메소드를 사용하여 상위의 서비스가 하위의 서비스를 호출할 수도 있는데, 프록시 서비스는 polled() 메소드에 정의된 웹 서비스 호출 모듈을 통해 서비스 요청을 처리한다.

5.3 Plumber Service

Plumber 서비스는 BPEL 기술서를 분석하여 단위 서비스들을 OSGi 프레임워크 환경에서 서비스 조합을 실행 및 주관하는 시스템 서비스이다. 이 단위 서비스가 웹 서비스인 경우 전술한 Web Gateway 서비스를 통해 생성된 프록시 서비스가 OSGi 도메인에서 그 서비스를 대리하게 된다. Plumber 서비스는 서비스 조합의 자동화라는 궁극적인 목표에 있어서는 이전 연구[2]와 유사하나, BPEL을 서비스 조합 기술 언어로 채택하고 웹 서비스와의 연동을 담당하는 Web Gateway 서비스를 도입하는 등 처음부터 완전히 재설계되었다.

서비스 조합 시나리오를 정의한 WS-BPEL 기술서가 주어지면 Plumber 서비스는 이를 분석하여 OSGi 서비스와 웹 서비스를 구별한다. OSGi 서비스는 별 문제없이 파이프를 통해 합성될 수 있지만,만약 은행 웹 서비스라면 Web Gateway 서비스에 앞의 그림 7과 같이 BPEL descriptor를 통해 확인된 WSDL 파일의 주소인 "http://localhost:8080/axis/services/MyBankTransferService?WSDL"를 파라미터 값으로 하여 프록시의 생성을 요청해야 한다. 이어서 프록시 생성 완료가 확인된 후 파이프를 이용하여 서비스를 합성한다.

5.4 동적 서비스 바인딩

WS-BPEL 스타일의 기술서를 이용한 서비스 조합의 문제점은 연결되는 서비스들과 그 주소가 사전에 결정되어야 하는 정적인 서비스 바인딩에 있다. 이러한 제약을 극복하기 위한 방안으로 WS-BPEL은 런타임에 새 서비스를 정의하는 것을 가능하게 하도록 하는 WS-Addressing 표준[12]을 도입하였다. 본 논문에서는 이러한 WS-Addressing 표준의 엔드포인트 레퍼런스(endpoint reference)를 사용한 동적 파트너 링크(dynamic PartnerLink)[1]를 OSGi 환경에서 발전시켜 OSGi 서비스 레지스트리에서 검색된 서비스가 서비스 조합에 사용될 수 있도록 하는 동적 서비스 바인딩 개념을 새로이 주창한다. 동적 서비스 바인딩은 EndpointReference 변수 타입으로 WS-BPEL 기술서에 BPEL 프로세스에 포함시킬 대상 서비스를 임시로 정의한 후, 즉 런타임에 적절한 서비스가 대체될 수 있도록 한 후, OSGi 프레임워크 서비스 레지스트리의 검색을 통해 서비스가 최종적으로 결정될 수 있도록 하는 기법이다.

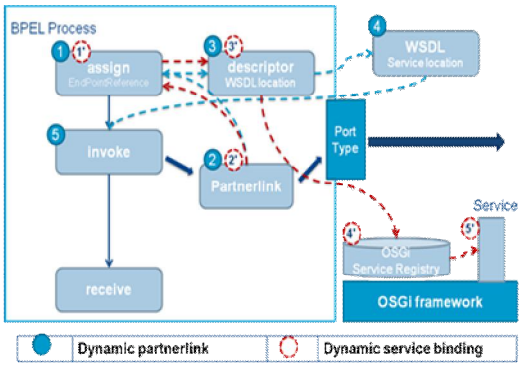

그림 12. 엔드포인트 레퍼런스를 이용한 동적 서비스 바인딩
Fig. 12. Dynamic Service Binding via Endpoint Reference

그림 12는 WS-BPEL에서 서비스를 임시로 EndpointReference 형의 변수를 사용하여 정의하고, 실제 서비스는 프로세스 동작 시점에서 할당하는 동적 파트너링크에 의해 결정되는 BPEL 프로세스의 동작 과정을 보여준다. 그림을 보면 먼저 (1) EndpointReference를 확인하면, (2) 프로세스는 해당하는 파트너 링크를 참조하고, (3) descriptor 파일을 통해 WSDL의 주소를 확인하여, (4) 그림 13과 같은 WSDL 파일을 통해 실제 서비스의 이름과 주소 정보를 획득하는 과정으로 동작한다. 원래의 BPEL과 비교하면 참여 서비스의 세부 내용의 최종 결정을 수행 시점까지 미룰 수 있도록 하여 서비스 조합의 유연성을 지원하는 것으로 볼 수 있다.

본 연구에서는 이에서 한발 더 나아가 동적인 서비스 바인딩이라는 새로운 기법을 제안하여 서비스 조합의 유연성을 획기적으로 개선하였다. 이는 기존의 BPEL descriptor 방식에 국한되지 않고 OSGi 서비스 레지스트리를 검색하여 사용할 서비스를 동적으로 선정할 수 있도록 한다. 즉, 동적 서비스 바인딩은 서비스에 대한 엔드포인트를 참조하기 위해 그림 12의 단계 (4')와 같이 OSGi 서비스 레지스트리를 이용한다.

```
<?xml version="1.0" encoding="UTF-8"?>
 ...
 <service name="FooBnakTransferService">
  <port name="BankTransferService" binding =
  "BankTransferServiceSoapBinding" >
   <addresslocation="...
   /FooBankTranferService"/>
  </port>
 </service>
 <service name="BarBankTransferService">
  <port name="BankTransferService" binding =
  "BankTransferServiceSoapBinding">
   <addresslocation="...
   /BarBankTranferService"/>
  </port>
 </service>
 ....
</wsdl:definitions>
```

그림 13. BankTransfer 서비스의 WSDL
Fig. 13. BankTransfer Service WSDL Description

그림 14는 OSGi 서비스 레지스트리를 이용할 수 있도록 그림 7의 BPEL desciptor 파일에 OSGi 서비스 참조가 추가된 것을 보여준다. 그림의 osgiLocation 속성에 주어진 서비스 정보는 Plumber 서비스에 의한 동적 서비스 검색과 바인딩에 이용된다. 예를 들어, Plumber 서비스는 OSGi 프레임워크의 getServiceReference() 메소드를 통해 OSGi 서비스 레지스트리를 검색하고, 서비스들의 property 정보를 확인하여 org.osgi.service.PartnerlinkType property가 BankTransferService인 서비스를 찾는다. OSGi 서비스 레지스트리가 서비스의 상태 정보를 동적으로 추적하여 반영하고 있기 때문에 검색된 서비스는 현 상황에서 가장 적합한 서비스임을 보장받을 수 있게 된다.

```
<?xml version="1.0"?>
<BPELProcess id="BankTransfer"
src="BankTransfer.bpel">
 <partnerLinkBindings>
  .....
  <partnerLinkBinding name=
        "BankTransferService">
   <property name="osgiLocation">
     BankTransferService.osgi </property>
   <property name="wsdlLocation">
http://localhost:8080/axis/services/BankTransferService
?WSDL</property>
  </partnerLinkBinding>
 </ partnerLinkBindings>
</BPELprocess>
```

그림 14. 동적인 서비스 바인딩을 위한 BPEL descriptor
Fig. 14. BPEL Descriptor for Dynamic Service Binding

다시 그림 12를 보면 Plumber 서비스는 먼저 (1') EndpointReference를 확인하면, (2') 파트너 링크를 참조하여, (3') BPEL descriptor로부터 파트너링크와 관련된 서비스의 정보를 획득한다. 이때 그림 14와 같이 descriptor 파일에 OSGi 참조가 정의되면, (4') Plumber 서비스는 OSGi 서비스 레지스트리 검색을 통해 서비스를 찾는다. 이렇게 동적으로 검색된 서비스를 런타임에 조합할 수 있도록 하고 있다. 동적 서비스 바인딩은 descriptor 파일과 WSDL 파일만을 기반으로 하여 고정된 파일의 서비스 정보에 의존하는 동적 파트너링크와는 서비스 조합 시 유연도 측면에 있어 본질적인 차이가 있는 전혀 새로운 기법임을 재차 강조한다.

동적 서비스 바인딩은 서비스 연결 시 파트너링크와 descriptor 파일을 참조하여 OSGi 서비스 레지스트리로부터 서비스의 이름과 주소를 검색하고, 그 결과로 엔드포인트 레퍼런스 변수로 임시 정의된 서비스를 대체하여 실행 시간에 완전히 동적인 서비스의 바인딩을 지원한다. 이 동적 서비스 바인딩 개념에 기초하여 설계된 OSGi와 웹 이종 도메인의 서비스 조합 시스템은 기존의 순수 OSGi 조합 방식과 비교하면 Plumber 서비스의 파이프를 이용해 서비스를 바인딩함으로써 자동 서비스 조합과 관리가 가능하다는 점이 두드러진다. 또한, 서비스 조합 기술에 WS-BPEL을 사용하기 때문에 기술적인 토대와 안정성을 확보할 수 있고, 개발 툴 등의 측면에서도 기존 연구 성과를 그대로 이용할 수 있다는 장점이 있다.

Ⅵ. 시스템 프로토타입 구현

앞서 3장에서 소개한 바 있는 공공요금 납부 시나리오를 예로 하여 OSGi 프레임워크 환경에서 OSGi 서비스와 웹 서비스를 통합하여 새로운 응용을 구성하는 서비스 조합 시스템 프로토타입의 구현을 본 장에서 설명한다. 구현 환경으로 웹 서비스 및 WS-BPEL 측을 보면 JVM J2SE 6.0을 기반으로 하여 Axis 서버 환경을 위해 Tomcat 5.5가 사용되었고 웹 서비스 메시지 처리를 위해 Apache Axis 2.0[11]가 사용되었다. 또한 WS-BPEL을 위한 BPEL 엔진으로 Active-BPEL 5.0[13]이 Tomcat 5.5 환경에서 사용되었다. OSGi 프레임워크 측에서는 JVM J2SE 6.0을 기반으로 하고 OSGi 스펙을 구현한 Knopflerfish 2.05 상에서 서비스 조합 시스템 프로토타입을 개발하였다.

```
<?xml version="1.0" encoding="UTF-8"?>
<process ..... >
 .....
 <sequence>
  <receive..... variable="AutoPaymentVar">
  <invoke partnerLink="To-Do_list"
  operation="schedule"
  outVariable="scheduleVar">

  <switch name = "MeterReadings">
   <case
   condition="getVariableDate('scheduleVar.t
   ype')='gas'">
    <invoke partnerLink="gas_meter"
    operation="read"/>
   </case>
   <case
   condition="getVariableDate('scheduleVar.t
   ype')='water'">
    <invoke partnerLink="water_meter"
    operation="read"/>
   </case>
   <case
   condition="getVariableDate('scheduleVar.t
   ype')='electicity'">
    <invoke partnerLink="wattmeter"
    operation="read">
   </case>
  </switch>

  <invoke… partnerLink="utilityCharge"…
  operation="calculate">
  </invoke>
  <invoke operation="startprocess"
  partnerLink="BankTransferRequest"  >
   〈assign name="DynamicServiceBinding"〉
    〈copy〉
     <from>
      <EndpointReference...>
       <Address />
      </EndpointReference> </from>
     <to variable="partnerReference" />
    〈/copy〉
    ……
   〈/assign〉
  </invoke>
  <reply ....>
 </ sequence>
</process>
```

그림 15. AutomaticPayment BPEL 기술서
Fig. 15. AutomaticPayment BPEL Description

그림 15는 공공요금 납부 시나리오에 따른 AutomaticPayment BPEL 프로세스의 기술서를 보여준다. 기술서를 보면 호출한 서비스로부터 전달된 값이 AutoPaymentVar에 할당되고, 이 값을 이용해 비즈니스 로직이 순차적으로 이루어짐을 알 수 있다. 기술서의 서비스 동작을 보면 먼저 To-Do_list 파트너 링크를 갖는 서비스를, 즉 공공요금 자동 납부 시나리오의 To-Do List 서비스를, 동기식으로 호출하는 invoke를 이용해 호출하는데, 이는 To-Do List 서비스에서 스케줄 이벤트가 발생하여 조합 서비스로 전달되기 전까지 조합 서비스는 대기 상태로 유지된다는 것을 의미한다. 다음으로 조합 서비스는 To-Do List 서비스로부터 전달된 ist 서비스에서 스 변수의 값을 대조하여 해당 계량기 서비스를 호출하여 전기/가스/수도 사용량을 얻는다. 사용 요금 계산 후 그림 4와 같은 은행 서비스인 BankTransfer BPEL 프로세스 서비스를 호출되는데, 이때 그림 15에 나타난 바와 같이 동적 서비스 바인딩이 이용된다. 즉, 서비스가 엔드포인트 레퍼런스 타입의 변수로 정의되고, 이후 수행 시 동적으로 서비스를 검색하여 실제 서비스로 연결된다.

기술서에 따라 조합되는 서비스들은 Plumber에 의해 파이프로 상호 연결된다. 이 합성 서비스에서 메시지는 서비스 간의 연결인 파이프를 통하여 목적지 서비스까지 전달된다. Plumber 서비스는 파이프를 관리하고, 파이프에 연결된 서

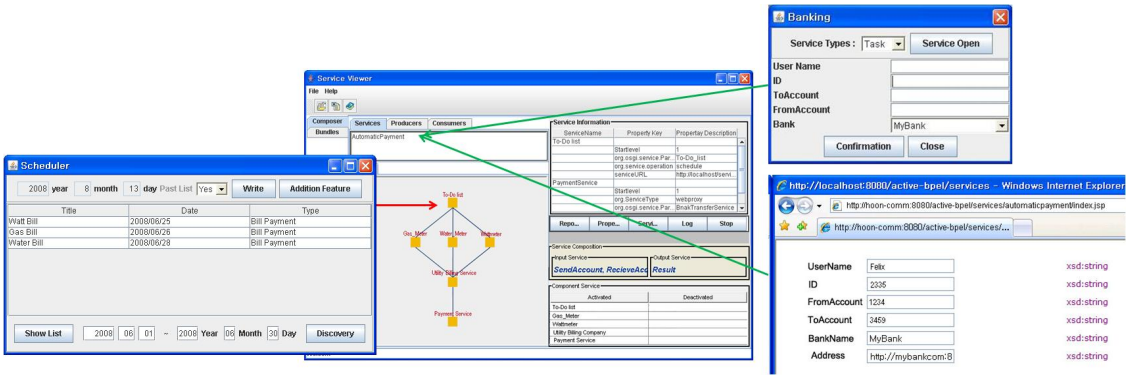

그림 16. 시스템 프로토타입 구현
Fig. 16. System Prototype Implementation

비스의 상태를 모니터링하는 임무를 맡고 있지만 직접적인 메시지 처리에는 개입하지 않는다.

그림 16은 공공요금 납부 시나리오의 프로토타입으로 구현된 시스템 관리 툴과 응용 서비스들을 보여주고 있다. 그림 중앙은 단순 기능의 서비스를 조합하여 복합 서비스나 응용 프로그램을 구성하기 위해 시스템 관리자가 사용하는 서비스 조합기 화면이다. 조합된 서비스 응용의 동작을 위해 그림 우측과 같이 은행 정보가 입력되면, 좌측의 스케줄 서비스에서 지정된 스케줄에 따라 공공요금이 자동적으로 납부될 수 있게 된다.

Ⅶ. 결론

최근의 OSGi 기술 동향을 살펴보면 가정용 게이트웨이를 넘어 자동차 및 모바일 디바이스를 위한 프레임워크 혹은 엔터프라이즈 응용 프레임워크로서의 가능성 모색 등 새로운 적용 분야와 응용에 대한 연구가 활발히 진행되고 있다. 하지만 이에 비해 서비스 환경의 확장이나 이종 도메인 서비스를 수용하고 통합하는 서비스 조합에 관한 연구는 상대적으로 미진하다. 본 논문은 OSGi 프레임워크 상으로 웹 서비스를 수용하여, OSGi와 웹 서비스라는 두 이종 도메인 서비스를 동적으로 연결할 수 있는 동적 서비스 바인딩에 근거한 새로운 서비스 조합 기법을 제안한다. 특히 동적 서비스 바인딩은 기존에 시도되지 않은 완전히 독창적인 기법으로, OSGi와 웹 서비스가 SOA 기술에서 차지하는 위치나 비중으로 볼 때 이들의 유연한 통합을 가능하게 하는 본 논문의 연구 결과는 상당한 가치와 파급 효과를 가질 것으로 기대된다.

참고문헌

[1] OASIS, "*Web Services Business Proces Execution Language Version 2.0* *http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html*.

[2] C. Lee, S. Ko, S. Lee, W. Lee, and S. Helal, "Context-Aware Service Composition for Mobile Network Environments," In Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing, pp. 941-952, Jul. 2007.

[3] H. Cervantes and R. Hall, "Autonomous Adapation to Dynamic Availability Using a Service-Oriented Component Model," In Proc. of the 26th International Conference on Software Engineering (ICSE'04), pp. 97-104, May 2004.

[4] R. P. D. Redondo, A. F. Vilas, M. R. Cabrer, J. J. P. Arias, and M. R. Lopez, "Enhancing Residential Gateways : OSGi Service Composition," IEEE Transactions on Consumer Electronics, Vol.53, No.1, pp. 87-95, Feb. 2007.

[5] 이충헌, 이종학, 서정만, 조완섭, "BPM과 SOA 기반의 비즈니스 프로세스 자동화와 분석기법," 한국컴퓨터정보학회논문지, 제 14권, 제 4호, 171-178쪽, 2009년 4월.

[6] I. Bajwa, A. Samad, S. Mumtaz, R. Kazmi, and A. Choudhary, "BPM Meeting with SOA: A Customized Solution for Small Business Enterprise," In Proc.

of International Conference on Information Management and Engineering, pp. 677-682, Apr. 2009.
〔7〕 Knopflerfish, "Knopflerfish Axis Port," *https://www.knopflerfish.org/svn/knopflerfish.org/trunk/osgi/bundles_opt/soap/axis.html.*
〔8〕 이용주, "시멘틱 e-워크플로우 프로세스를 이용한 동적 웹 서비스 조합," 한국컴퓨터정보학회 논문지, 제 10권, 제 1호, 101-112쪽, 2005년 3월.
〔9〕 R. Redondo, A. Vilas, M. Cabrer, J. Arias, J. Duque, and A. Solla, "Enhancing Residential Gateways: A Semantic OSGi Platform," IEEE Intelligent Systems, vol.23, no.1, pp. 32-40, Jan.-Feb. 2008.
〔10〕 Open SOA, "*Service Component Architecture Specifications*" *http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications.*
〔11〕 Apache Software Foundation, "*Apache Axis2 Architecutre Guide*" *http://ws.apache.org/axis2/1_3/Axis2ArchitectureGuide.html.*
〔12〕 W3C, "Web Services Addressing (WS-Addressing)," *http://www.w3.org/Submission/ws-addressing.*
〔13〕 ActiveVOS Forums, "*The ActiveBPEL Community Edition Engine*" *http://www.activevos.com/community-open-source.php.*

저 자 소 개

고 성 훈

2008년: 한양대학교 전자컴퓨터통신공학과 석사
2008년 ~ 현재: 롯데정보통신 연구원
관심분야 : 웹서비스, BPM, SOA

김 은 삼

2006년: University of Florida 컴퓨터공학과 박사
2007년 ~ 현재: 홍익대학교 컴퓨터공학과 조교수
관심분야 : 분산 멀티미디어 시스템, 컴퓨터저장시스템, IPTV 시스템

이 춘 화

2003년: University of Florida 컴퓨터공학과 박사
2004년 ~ 현재: 한양대학교 컴퓨터공학부 조교수
관심분야 : P2P 네트워킹 및 시스템, 인터넷 프로토콜, SOA, 클라우드 컴퓨팅