韓國컴퓨터情報學會 論文誌
第15卷 第12號, 2010. 12.

2010-15-12-4-3

BPEL 기반의 서비스 지향 비즈니스 프로세스 모델링

이 상 영*

BPEL Based Service Oriented Business Process Modeling

Sang-Young Lee*

요 약

신속히 변하는 비즈니스 환경에서 기업이 살아남기 위해서는 변화하는 환경에 자사의 비즈니스 프로세스를 적응시켜야 한다. 시스템 통합은 이러한 문제를 해결하는 중요한 문제이다. 유연한 비즈니스 프로세스와 통합을 지원하는 IT 표준으로 SOA의 채택은 피할 수 없는 방향이다. 특히 비즈니스 프로세스와 인터페이스의 통합은 점차 중요도가 증대되고 있다. 따라서 비즈니스 프로세스를 쉽고 직관적으로 구현할 수 있는 모델링이 필요하다. 이에 본 논문에서는 기업간의 비즈니스 프로세스를 분석하고 BPEL을 기반으로 서비스 지향 비즈니스 프로세스를 실현수준까지 모델링한다. 또한 비주얼 언어인 UML을 사용하여 높은 수준의 기업문제에서 낮은 수준의 프로세스까지 문제를 보다 비주얼하고 효과적으로 제시한다.

Abstract

To survive in rapidly changing business environment, the enterprise should adapt its business process to the changing environment. The system integration is an important issue to tackle this challenge. It is unavoidable direction that SOA(Service Oriented Architecture) is accepted as an upcoming IT standard to support flexible business processes and integration. In particular, the integration of business processes and interfaces are important. Therefore, intuitive and easy to implement business process modeling is required. In this paper analyzes business process between enterprises, modeling the service-oriented business process with BPEL to realization level. Also, In this paper suggests that UML is used from high-level business problems to the process of low-level problem for Visual and effectively.

▸ Keyword : 비즈니스 프로세스(Business Process), BPEL((Business Process Execution Language), 모델링(Modeling)

• 제1저자 : 이상영
• 투고일 : 2010. 09. 15, 심사일 : 2010. 09. 27, 게재확정일 : 2010. 10. 05.
* 서울대학교 보건행정학과 교수

Ⅰ. 서 론

최근 기업들은 다운사이징, 인수합병, 글로벌 경쟁과 불확실성 하에서 경쟁력 제고와 더불어 수익성 개선을 위한 방법을 모색하고 있다. 이러한 기업의 외부 환경의 변화에 따라 기업의 업무 흐름이 수시로 변화하고, 고객과 공급자 등 모두를 고려한 애플리케이션에 대한 통합 모델이 필요하다[1]. 이러한 애플리케이션 통합을 위한 가장 중요한 요소는 공동 이용 가능성이며, 이는 특정 업체나 플랫폼에 종속되지 않는 기술이어야 한다. 아울러 기존의 정보 시스템 투자를 보호할 수 있어야 하며, 개발자의 현재 보유 기술을 계속적으로 활용하고 응용해야 한다. 이러한 통합 기술로 대표되는 것이 바로 서비스 지향의 아키텍처인 SOA(Service Oriented Architecture)이다. SOA는 다양한 시스템을 통합하기 위한 일반적인 접근법으로 원격으로 호출될 수 있고 독립적인 방법으로 서비스가 가능한 잘 정의된 인터페이스를 가지며 내부로직에 관계없이 애플리케이션 기능을 사용할 수 있도록 하는 아키텍처이다[2, 3]. 특히 SOA에서는 다양한 전산시스템 위에 구현된 애플리케이션과 그 기능을 자원이라고 하는데 이들 자원들을 효과적으로 통합시켜 기능의 재활용이나 배치의 유연성을 극대화하는 것을 목적으로 한다[4]. 이러한 SOA의 기본 아이디어는 자원을 서비스화 하여 통합할 수 있다는 관점이다. 서비스는 애플리케이션의 기능 구현이 어떤 네트워크, 하드웨어, OS, 미들웨어, 프로그래밍 언어, 객체모델에 기반하고 있든지 관계없이 일관된 방법으로 그 기능을 사용할 수 있도록 포장한 것이다. 이렇게 애플리케이션 기능을 서비스화 함으로써 다양한 수준에서 애플리케이션 통합을 이루어낼 수 있다[5]. 이렇게 함으로써 시장 변화에 즉시 대응할 수 있도록 하고 비용을 절감하면 조직의 핵심역량에 집중할 수 있게 한다. 또한 IT 측면에서 보면 개발기간을 단축하고 운영을 단순화하여 비용을 절감할 수 있게 한다.

지금까지 기업의 비즈니스 프로세스(business process)를 모델링하고, 명세하고, 런타임 환경에서 실행하는데 있어 다양한 기술들이 적용되어 왔다. 이중에서 워크플로우의 모델링 및 실행방법은 크게 Choreography 언어와 Orchestration 언어에 의한 방법으로 구분된다. Choreography는 특정한 주도권을 가진 엔터티가 없으면서 비즈니스 엔터티들 간에 장기적이고 계속적인 상태에서 서로 협조하면서 이벤트가 발생할 시 각자가 자기역할을 함으로써 프로세스를 처리하는 기술이다. 즉 peer-to-peer 방식으로 협업하는 모델이다. 반면에 Orchestration은 하나의 비즈니스 엔터티에 초점을 맞추는 것으로, 특정지점 한곳에서 사전에 정의된 대로 프로세스의 전체를 통제하는 것이다. 즉 비즈니스를 주도하는 하나의 참여자 관점에서 여러 프로세스들을 통합하는 모델이다[6].

BPEL(Business Process Execution Language)이 Orchestration 형식의 서비스를 지원할 뿐 아니라 표준 인터페이스를 제공함으로써 프로세스의 수평적 그리고 수직적 통합을 가능케 하는 Orchestration 언어이다. BPEL은 웹 서비스를 구성하여 비즈니스 프로세스의 행동을 명세하는 XML 기반의 언어로 프로세스를 파트너들과의 상호교류 관계로 정의한다. 파트너는 프로세스에게 서비스를 제공하거나 서비스를 요청하기도 한다. BPEL은 서비스를 의미 있고 일정한 순서로 명세함으로써 웹 서비스를 Orchestrate한다. 또한 XML 기반의 흐름 언어이므로 if 문, while 문, 병렬수행 등과 같은 구조적 프로그래밍도 가능하다.

따라서 웹 서비스 환경에서 비즈니스 프로세스를 정의하고 실행하기 위한 표준 언어인 BPEL을 기반으로 서비스 지향의 비즈니스 프로세스를 모델링한다면 비즈니스 프로세스를 생성하고, 생성된 프로세스를 런타임 환경에 적재하는 실제적인 적용이 가능하여 진다. 즉 비즈니스 프로세스인 컴포지트 서비스(composite service)를 쉽고 직관적으로 구현할 수 있다.

이에 본 연구에서는 BPEL 기반의 서비스 지향 비즈니스 프로세스 모델링을 목적으로 컴포넌트간 상호운용성을 고려하여 비즈니스 프로세스를 모델링한다. 또한 통합 비주얼 모델링 도구인 UML을 이용하여 서비스 지향 비즈니스 프로세스 모델링 모델링 방법을 제시하고 구현한다. 논문의 구성은 크게 5장으로 구성되어 있다. 서론에 이어, 2장에서는 관련연구를 제시하고, 3장에서는 모델링에 대한 설계를 하고, 4장에서는 실제적인 사례에 적용하여 구현하고 테스팅한다. 마지막 5장에서는 연구에 대한 결론 도출 및 추후 연구 방향에 대하여 제시한다.

Ⅱ. 관련 연구

점차 다양해지고 발전하는 기술들로 인해 분산 환경의 시스템에서 일부 기능들은 컴포넌트로 개발되어 재사용되고 있다. 그러나 컴포넌트는 특정한 환경에 종속적이면서 다른 컴포넌트와의 연관도가 높아 동일한 환경에서는 재사용이 탁월하지만 이질적인 환경에서는 상호운용에 문제가 발생할 수 있다[8]. 이를 해결하기 위한 방안으로 환경에 제약없이 컴포넌트를 재사용할 수 있는 SOA에 대한 많은 연구들이 수행되고 있다[9, 10, 11]. 특히 SOA를 기반으로 프로세스 통합과 실행을 위한 획기적인 계기를 마련함으로써 웹서비스의 실질적인 통합을 이루는데 근간이 되는 BPEL에 관한 연구도 이루어지고 있다[6]. IT 관점에서 SOA는 분산 객체의 형태로 존재하는 컴포넌트들 간에 메세지 통신을 통해 정보를 교환하는 느슨하게 연결된(Loosely Coupling) 형태의 소프트웨어 구

조이다[12]. 기존의 컴포넌트 방식은 시스템에 제약적인 방식으로 컴포넌트간 결합방식을 제공하였으나, SOA 서비스 방식은 사용자 인터페이스로부터 메시징 채널까지 독립된 서비스간의 약 결합(loosly-coupled)을 통해 제공된다. 따라서 SOA는 컴포넌트 방식에 비해 비즈니스 서비스를 실행하고 서비스 컴포넌트를 운용하는데 있어 보다 넓고 확장 가능한 아키텍처를 제공한다[13].

SOA에 대한 연구는 1996년의 첫 제안 이후 현재까지 국내외의 많은 학계 및 업체들에 의해 활발히 진행되어왔다[8, 9]. 먼저 SOA에 대한 접근방법들과 제안된 기술들에 대한 연구들이 있고[12, 13]. SOA의 구현에 필수적인 웹 서비스를 이용하여 기존에 존재하는 다양한 분야의 애플리케이션에 대한 응용연구들이 있다[14]. 이러한 연구들에서는 상호 이질적인 플랫폼이나 프로토콜에 디바이스 서비스를 유연하게 제공하기 위하여 서비스 변환 게이트웨이 엔진과 웹 서비스, 소프트웨어 서비스와 애플리케이션 컴포넌트들 간의 연동을 위해 ESB(Enterprise Service Bus)를 사용하여 서비스를 통합하고 관리하는 구조를 제안하고 있다.

SOA는 그동안 DCE(Dalian Commodity Exchange)에서 DCOM(Distributed Component Object Model), EJB(Enterprise JavaBeans), EAI(Enterprise Application Integration)에 이르기 까지 다양한 형태로 구현되어 왔다. SOA의 적용으로 기능의 위치와 내부구현에 영향을 받지 않고 기능을 사용하는 것은 가능해졌으나 여전히 여러 가지 면에서 상호 의존성이 존재한다. 성공적인 SOA 구현이라 할 수 있는 EJB와 DCOM의 경우를 보면 EJB 기반 애플리케이션과 DCOM 기반 애플리케이션 간에는 서로 서비스를 주고받을 수가 없다. 이것은 기업 내 혹은 기업 간 시스템 통합에 중대한 제약이 되고 있다. 이 문제를 해결하는 가장 확실한 방법은 주고받는 메시지의 포맷과 프로토콜을 통일하여 플랫폼으로부터 독립하는 것이다. 이 포맷과 프로토콜은 다양한 언어와 API에서 지원되는 형태이어야 하고 객체 모델과는 무관하여야 한다. 무엇보다도 SOA와 연관된 애플리케이션과 인프라 모두 SOA를 지원해야 한다[15]. 즉 웹 서비스를 통해 실현되는 플랫폼 독립성은 부서단위 혹은 기업 단위를 넘어서는 규모에 걸쳐 애플리케이션끼리 상호연동하고 기능을 재활용하며 손쉽게 재배치할 수 있는 여건을 제공해 줄 수 있다.

UML은 이러한 배치를 도와주는 비주얼한 모델링 방법을 제공하므로 모델링 시 효율적이다. 즉 잘못된 비즈니스 프로세스분석으로 인한 오류들이 그대로 설계로 이어져 비즈니스 요구사항을 만족시키지 못하는 IT 서비스의 생산을 초래할 수 있는 바 비주얼한 모델링 방법인 UML을 사용하여 보다 체계적으로 모델링할 필요가 있다. 이에 대한 SOA 분야의 관련연구를 보면 액티비티와 클래스 다이어그램을 활용한 비즈니스 프로세스 모델링 방법론에 대한 연구[16, 17, 18] 등이 있다. 이러한 연구에서는 UML을 기반으로 비즈니스 로직과 애플리케이션 로직간의 차이를 줄이고 서비스 인터페이스를 구현하는 연구들이 이루어졌다. 이러한 연구에서는 객체지향 개발방법론에 입각하여 점증적이고 반복적인 방법으로 비즈니스 요구사항을 분석하고 설계하여 비즈니스를 실현할 수 있는 수준까지 모델링하는 것이 부족하다. 즉 본 논문에서는 UML 모델링을 바탕으로 웹서비스의 실행할 수 있는 BPEL과 WSDL을 이용하여 SOA를 구현하고 테스팅하였다.

III. 서비스 지향의 비즈니스 프로세스 모델링 설계

서비스 지향의 비즈니스 프로세스 모델링은 비즈니스 상태 평가부터 비즈니스 프로세스 분석 및 비즈니스 프로세스 자동화까지 분석되고 설계된다. 또한 이러한 모델링에 대한 최종적인 산출물은 비즈니스 프로세스를 실체화한 비즈니스 유스케이스 다이어그램으로 작성된다.

1. 비즈니스 프로세스 모델링

본 논문에서 제안하는 비즈니스 프로세스 모델(M)은 아래 그림 1과 같이 5개의 구성요소로 이루어진다.

```
M = {BSE, BPM, SBP, BPELC, ST}
        BSE : Business State Evaluation;
        BPM :  Business Process Modelling;
        SBP : Service based Business Process;
        BC : BPEL Creation;
        ST : Service Testing;

 BSE = {BG, BRS, ATO, SAG}
        BG : Business Glossary;
        BRS :  Business Rules and Service;
        ATO : Assess Target Organization;
        SAG : Set and Adjust Goals;
 BSE = {BPI, BPD, BPRD, BPEA}
        BPI : Business Process Identifictaion;
        BPD :  Business Process Definition;
        BPRD : Business Process Realization Design;
        BPEA : Business Process Explore Automation;
 SBP = {Primary Activity, Structure Activity}
         PA : Primary Activity
         SA : Structure Activity
 BSE = {PT, Var, ProcessDef}
        PT : Partner Type;
        Var :  Variable;
        ProcessDef : Process Definition;
 BSE = (Tname}
        Tname :  Tname;
```

그림 1. 모델 구성요소
Fig. 1 Model Elements

이러한 구성요소를 기반으로 아래 그림2와 같은 비즈니스 프로세스 모델링 순서에 따라 전체적으로 모델링된다[19].

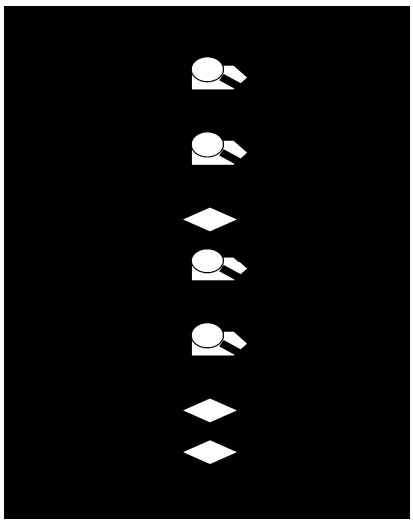

그림 2. 비즈니스 프로세스 모델링
Fig. 2 Business Process Modeling

그림에서 보는 바와 같이 비즈니스 프로세스 찾기 및 정제에서는 실제 비즈니스 프로세스를 분석하고 설계할 수 있는 핵심적인 설계 및 모델링이 이루어진다. 실제적인 비즈니스 액터와 비즈니스 유스케이스가 도출되며 서비스 관련 비즈니스 규칙이 정의되고 정제되는 단계이다. 이러한 정제를 거쳐 비즈니스 활동들을 실현할 수 있도록 변경되고 실체화되며 최종 시스템에 사용될 수 있도록 비즈니스 자동화단계를 거친다. 각각의 실체화된 비즈니스 유스케이스들에 대한 자동화 단계를 거쳐 최종 산출물인 비즈니스 유스케이스 다이어그램이 설계된다.

2. 서비스 지향 비즈니스 프로세스 모델링

서비스 지향 관점에서 보면 유스케이스 관점이 서비스를 조립하고 조정하는 중심 역할을 수행하며 비즈니스 프로세스를 실현하는 BPEL의 설계요소가 된다는 점에서 매우 중요하다. 본 논문에서는 서비스 지향 비즈니스 프로세스 구축 시 비즈니스 프로세스를 여러 개의 스텝으로 구성하고 각각의 스텝을 액티비티로 구분한다. 각 액티비티는 다른 웹 서비스를 호출하거나 메시지 전달, 동기화 작업의 응답, 할당, 대기, 종료와 같은 primary 액티비티와 프로세스의 단계를 정의할 수 있는 알고리즘을 수행할 수 있는 structure 액티비티로 구분한다. 다음 표 1은 primary 액티비티와 structure 액티비티를 정의하고 나타낸다.

표 1. primary 액티비티, structure 액티비티
Table 1. primary activity, structure activity

primary 액티비티	structure 액티비티
<Invoke>: 다른 웹 서비스의 호출 <receive>: 클라이언트가 메시지를 전달함으로써 비즈니스 프로세스를 호출할 때까지 대기 <reply>: 동기화 작업의 응답 생성 <assign>: 데이타 변수의 값 할당 <throw>: 폴트 및 예외 발생 <wait>: 일정 시간 대기 <terminate>: 전체 프로세스의 종료	Sequence (<sequence>): 일정한 순서로 호출되는 액티비티 집합을 정의 Flow (<flow>): 병렬적으로 호출되는 액티비티의 집합을 정의 Case-switch construct (<switch>): 분기 조건의 구현 While (<while>): 루프 정의 <pick>: 다양한 실행 경로 중 하나를 선택 <partnerLink>를 통해 파트너 링크를 정의 <variable>을 통해 변수를 선언

다음 그림 3은 이러한 본 논문에서 적용한 BPEL을 설계하는 과정을 보여준다.

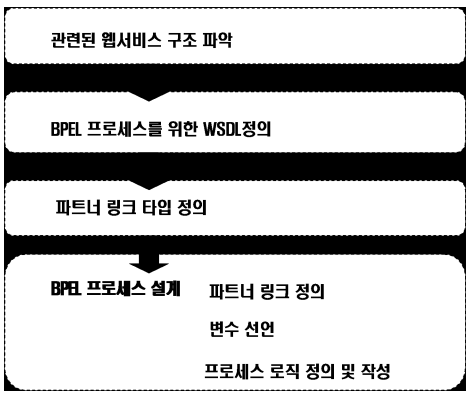

그림 3. BPEL 설계
Fig. 3 Design of BPEL

그림에서 보는 바와 같이 먼저 BPEL 프로세스 정의 작업을 시작하기 전에 프로세스에서 호출하는 웹서비스 구조를 파악한다. 그리고 BPEL 프로세스를 웹서비스의 형태로 공개하는데 사용할 WSDL을 정의한다. 이 프로세스에서는 클라이언트로부터 메시지를 전달받고 그 결과를 반환하는 과정이 수반된다. 다음으로, 파트너 링크 타입(partner link type)을 정의한다. 파트너 링크 타입은 BPEL 프로세스와 관련 개체들 간의 상호작용을 정의하는데 사용된다. 여기에는 BPEL 프로세스가 호출하는 웹서비스와 BPEL 프로세스를 호출하는 클라이언트에 대한 정의가 함께 포함된다. BPEL 프로세스를 구현하는 과정에서 파트너 링크 타입을 이해하는 것은

매우 중요하고 파트너 링크 타입의 정의가 완료됨으로써 비즈니스 프로세스 작성단계를 처리하게 된다.

그리고 BPEL 프로세스는 클라이언트로부터 메시지를 받아 비즈니스 프로세스를 실행하는 형태로 동작한다. BPEL 액티비티 네임스페이스는 다음과 같이 정의된다. 설계되는 프로세스 정의는 이러한 액티비티 네임스페이스에 따라 작성되며 다음과 같은 세 가지 요소로 구성된다.

- 파트너 링크 정의
- 변수 설정
- 프로세스 로직 정의(Process Logic Definition) 작성

먼저 파트너 링크 정의는 아래 그림 4와 같은 정의 구문 기술 방식으로 정의하고 변수를 설정한다.

```
<process name="BusinessTravelProcess" ... >
   <partnerLinks>
     <!-- The declaration of partner links -->
   </partnerLinks>
   <variables>
     <!-- The declaration of variables -->
   </variables>
   <sequence>
    <!-- The definition of the BPEL business process
main body -->
   </sequence>
</process>
```

그림 4. 프로세스 정의 구문
Fig. 4 Process definitions statement

아울러 BPEL 프로세스 로직에 대한 정의를 보면 BPEL 프로세스 메인 바디에서는 먼저 파트너 웹서비스를 호출하는 순서를 정의한다. 먼저 <sequence>를 사용하여 순차적으로 실행되는 액티비티를 정의 한다. Sequence에서 비즈니스 프로세스를 실행할 때 사용할 입력 메시지를 정의하기 위해 <receive>를 활용하며 <receive>는 매치되는 메시지가 입력될 때까지 대기한다. 이때 입력 메시지를 <receive> 구문 안에 직접 정의하는 대신, 파트너 링크와 포트 타입, 작업 이름(operation name)을 정의하고 필요한 경우 후속 작업의 메시지 입력에 사용할 변수를 지정한다. 또한 두 개의 <copy>를 포함하는 <asign> 구문을 활용하여 변수를 복사하거나 값을 할당한다.

병렬 수행을 위한 액티비티는 <flow> 액티비티를 이용하며 <flow> 액티비티 안에 하나의 <sequence> 액티비티를 사용하여 두 가지 액티비티를 포함할 수 있다. 병렬적 의미와 다르게 분기의 표현은 <switch> 액티비티와 <case> 액티비티를 사용하여 정의할 수 있다.

Ⅳ. BPEL 기반의 서비스 지향 비즈니스 프로세스 모델링 적용

비즈니스 모델링을 기반으로 한 서비스 지향 BPEL 모델링은 일반적인 웹사이트의 회원가입 서비스인 memberService에 대한 내용을 중심으로 기술한다. 먼저 유스케이스 다이어그램을 실현하기 위해 클래스 뷰에 memberService를 실체화한 memberService 클래스에 대한 구현을 위해 클래스를 역할에 따라 분리하였고 가운데의 컨트롤을 위한 memberControl 클래스와 엔터티를 위한 memberInfo 클래스로 구분한다. 역할에 따른 클래스 다이어 그램은 아래 그림 5와 같이 모델링 된다.

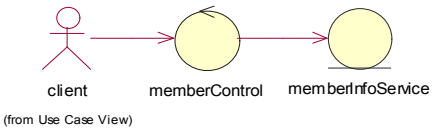

그림 5. memberService의 클래스 다이어그램(클래스 뷰)
Fig. 5. memberService Class Diagrams (Class View)

그리고 그림 6과 같이 실체화 된 클래스에 대해 활동을 표현한 액티비티 다이어그램을 이용하여 모델링한다.

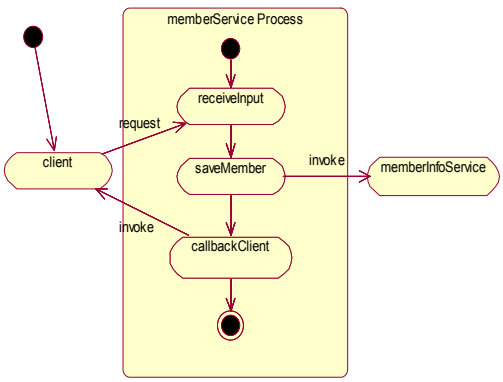

그림 6. memberService 액티비티 다이어그램
Fig. 6. memberService Activity Diagrams

마지막으로 memberService의 액티비티 다이어그램을 이용하여 SOA의 비즈니스 프로세스 실행언어인 BPEL을 이용하여 액티비티를 모델링한다. 웹서비스를 근간으로 하는 SOA의 BPEL 프로세스 모델링 순서는 다음과 같다.

다음은 본 논문에서 적용한 BPEL를 이용하여 비즈니스 프로세스를 모델링하는 과정이다.

- 관련된 웹 서비스의 구조 파악
- BPEL 프로세스를 위한 WSDL을 정의
- 파트너 링크 타입(partner link type)을 정의
- BPEL 프로세스 설계

파트너 링크 정의
변수 선언
프로세스 로직 정의(process logic definition) 작성

이러한 BPEL 프로세스 설계를 기반으로 만들어진 회원가입 서비스인 memberService는 그림 7과 같이 BPEL로 구현된다.

```
<process name="MemberService"
targetNamespace="http://xmlns.oracle.com/MemberService"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpath20" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://xmlns.oracle.com/BPELProcess1"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/MemberService"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.ExtFunc">
... 생략
<partnerLinks><!-- The 'client' role represents the requester of this service. It is used for callback. The location and correlation information associated with the client role are automatically set using WS-Addressing. -->
<partnerLink name="client" partnerLinkType="client:MemberService"
myRole="MemberServiceProvider" partnerRole="MemberServiceRequester"/>
    <partnerLink myRole="BPELProcess1Provider" name="MemberInfoService"
partnerRole="BPELProcess1Requester" partnerLinkType="ns1:BPELProcess1"/>
  </partnerLinks><!--
== -->
<!-- VARIABLES -->
<!-- List of messages and XML documents used within this BPEL process -->
  <variables><!-- Reference to the message passed as input during initiation -->
... 생략
<!-- services integrated portType="client:MemberServiceCallback"
operation="onResult" inputVariable="outputVariable"/>
  </sequence>
</process>
```

```
within this business process -->
  <sequence name="main">
    <receive name="receiveInput" partnerLink="client"
portType="client:MemberService" operation="initiate"
variable="receiveInput_initiate_InputVariable" createInstance="yes"/>
<!-- Asynchronous callback to the requester.
      Note: the callback location and correlation id is transparently handled
      using WS-addressing.-->
    <invoke name="saveMember" partnerLink="MemberSevice"
portType="ns1:BPELProcess1Callback" operation="onResult"
inputVariable="saveMember_onResult_InputVariable"/>
    <invoke name="callbackClient" partnerLink="client">
```

그림 7. memberService.bpel
Fig. 7. memberService.bpel

이러한 비즈니스 프로세스 모델링에 대한 검증이 필요한데 즉 실제 서비스의 수행여부 파악, 에러여부 등에 대한 서비스 테스팅이 필요하다. 본 논문에서는 일반적인 소프트웨어 테스팅 방법 중 블랙박스 테스팅 방법인 스트레스 테스팅을 수행하였다. 스트레스 테스팅에서는 엔터프라이즈 환경에서 비즈니스 서비스를 구현함에 있어 수행 서비스가 동시 처리영역, 한계영역 등 성능 및 품질을 보증할 수 있는지를 제시하기 위한 목적으로 수행하였다. 다음 그림 8은 구현된 웹서비스를 테스트하기 위한 환경이다.

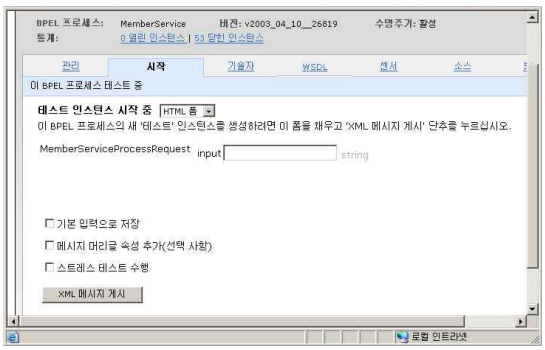

그림 8. 웹서비스 테스트 환경
Fig. 8. Web Services Test Environment

서비스 인터페이스에 대한 성능을 알아볼 수 있도록 동시 발생 스레드와 루프 및 지연간격을 지정하여 수행하였다. 아래 그림 9에서는 이러한 스트레스 테스트 화면 및 결과화면을 보여준다.

그림 9. 스트레스 테스트 결과
Fig. 9. Stress Test Result

테스트 시 MemberService 에 대한 테스트는 16바이트의 입력문서를 사용하였고 10개의 스레드와 5개의 루프를 수행하며 지연시간을 1초 간격으로 주어 테스트하였다. 실험 결과 스트레스 테스트의 상세 결과에서 delivery-request나 그 하위항목인 receive-invoke가 각각 1.05472 초가 걸려 가장 스트레스가 많았고 그 하위항목 중에서 호출 메시지 저장(invoke-store) 부분이 1.028초의 스트레스를 받고 압축하지 않은 페이로드(uncompress- payload) 항목이 0.0062초로 거의 스트레스를 받지 않았다. 또한 호출 저장(invoke-store) 중 호출 메시지 삽입(invoke-insert-msg)이 0.38506초로 많은 스트레스를 받고 있으며 호출 압축(invoke-compress)이 0.00032초로 가장 스트레스가 적은 것으로 나타났다. 그림 10은 이러한 스트레스 테스트 결과를 보여준다.

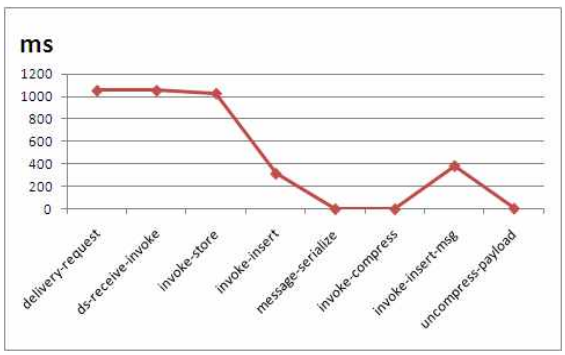

그림 10. 스트레스 테스트 취합
Fig. 10. Stress Test Result

본 논문에서 제시하는 방법을 통하여 다양한 시스템을 통합하기 위한 접근이 가능하고 독립적인 방법으로 서비스가 가능한 것을 알 수 있으며, 서비스 인터페이스를 통하여 내부로직에 관계없이 애플리케이션 기능을 사용할 수 있도록 해줌을 알 수 있다.

V. 결 론

본 논문에서는 비주얼 통합 모델링 언어인 UML을 이용하여 저 수준의 비즈니스 요구사항 분석으로부터 서비스 기반 아키텍처인 SOA의 모델링 방법을 제안하였다. 즉 객체지향 개발방법론에 입각하여 점증적이고 반복적인 방법으로 비즈니스 요구사항을 분석하고 설계하여 비즈니스를 실현할 수 있는 수준까지 모델링하였으며, 이러한 모델링을 바탕으로 웹서비스의 실행과 책임을 기술하는 BPEL을 이용하여 SOA를 구현하고 테스팅하였다.

본 논문에서 제안한 서비스 지향의 비즈니스 모델링 방법은 일반적인 비즈니스 프로세스 모델링과는 다르게 서비스 단위인 웹서비스라는 단위를 이용하여 각각 독립적으로 구현되고 참조된다. 이를 위해 비즈니스 프로세스 모델링 과정을 사례연구를 통해 실험하였다. 실험결과 비주얼 모델링 접근법을 사용함으로써 객관적이고 명확한 분석과 설계가 가능하다는 점과 서비스와 관련된 정의 및 실행에 있어서 보다 체계적인 접근이 가능하였다. 본 논문의 활용적 측면과 의의는 소프트웨어 개발에 있어 서비스를 기반으로 한 재사용성과 아키텍처를 통해 제공되는 안정성에 있어 빠른 개발 틀을 제시하고 비즈니스에 있어 시장적응과 활용에 대해 낮은 비용과 빠른 적용수단 그리고 거시적 레벨의 서비스와 미시적 레벨의 서비스를 쉽게 상호 운용할 수 있는 방법을 제시함으로써 소프트웨어와 비즈니스간의 안정적인 접목을 제공한다는 점이다. 향후에는 웹서비스 환경에서 정보노출의 위험성을 다루는 정보 관리 및 보안에 관한 연구들이 이루어져야 할 것이다.

참고문헌

[1] 장양자, 김상수, 조수형, “서비스 기반 통합, 마스터 데이터 통합 및 사례”, 정보과학회지, 제 26권, 제 9호, 23-29쪽, 한국정보과학회, 2008년.

[2] Carey, M.J., "SOA What?", Computer, Vol. 41, Issue 3, pp. 92- 94, 2008.

[3] Huhns, M. N., M. P., "Service-Oriented Computing: Key Concepts and Principles", Internet Computing, Vol. 9, Issue 1, pp. 75-81, IEEE, 2005.

[4] Stone, A., "Demanding Internet enterprise", Internet

Computing, Vol. 8, Issue 3, pp. 13 -14, IEEE ,2004.
[5] 서경기, “유연한 비즈니스를 위한 유연한 IT 환경-SOA”, 정보과학회지,제 25권, 제 1호, 55-58쪽, 한국정보과학회, 2007년.
[6] 박동진, 장병훈, “제조실행시스템에서의 BPEL 기반 워크플로우 관리시스템의 적용“, 한국IT서비스학회지, Vol. 8, No. 4, pp. 165-172, 한국IT서비스학회, 2009.
[7] Stoilov, T. and K. Stoilova, “E-business workflow modeling and execution tools”, International Conference Automatica and Informatics, 2006.
[8] T. Erl, "Service-oriented architecture: concepts, technology, and design", Prentice Hall PTR Upper Saddle River, 2005.
[9] T. Erl, "Service-oriented architecture: a field guide to integrating XML and web services", Prentice Hall PTR Upper Saddle River, 2004.
[10] 이충헌, 이종학, 서정만, 조완섭, “BPM과 SOA기반의 비즈니스 프로세스 자동화와 분석기법”, 한국컴퓨터정보학회 논문집, Vol. 14(4), pp. 171-178, 한국컴퓨터정보학회, 2009.
[11] 김태호, 이홍철, 천현재, 김준룡, “개인사용자를 포함하는 JXTA 기반의 Service Oriented Architecture 구현”, 한국컴퓨터정보학회 논문집, 제 12권, 제 4 호, 21-31쪽, 2007년.
[12] C. Baroudi, J. Hurwitz, and R. Bloor, "Service -Oriented Architecture for Dummies", Wiley, 2006.
[13] Mike P. Papazoglou and Willem-Jan van den Heuvel, "Service Oriented Architectures: Approaches, Technologies and Research Issues", 2005.
[14] Takaaki Moriya, "A Support System for Designing Ubiquitous Service Composition Scenarios", 2007.
[15] 김유경, 윤홍란, “SOA를 위한 서비스 지향 개발 프로세스”, 한국전자거래학회지, 제 12권, 제 2호, 75-93쪽, 2007년.
[16] T. Erl,, “Service-Oriented Architecture Concepts, Technology and Design”, Prentice“, Hall, 2005.
[17] K. Mantell, “From UML to BPEL, 2005, *http://www.ibm.com/developerworks/library/ws-uml2bpel/?ca=dnt-436*
[18] M. P. Singh, M. N. Huhns, “Service-Oriented Computing semantics Processs Agents”, John Wiley & Sons, Ltd, 2005.
[19] H. Eriksson, M. Penker, “"Business modeling with UML: business patterns at work"", John Wiley & Sons, 2000.

저 자 소 개

이 상 영

1994: 숭실대학교 공학사.
1998: 전북대학교 공학석사.
2004: 전북대학교 이학박사
2005 - 현재: 남서울대학교 보건행정학과 교수
관심분야 : 유비쿼터스기술, 의료정보, HCI, 의료보안 등