Journal of Korea Multimedia Society Vol. 12, No. 1, January 2009(pp. 122-129)

SOA 기반의 워크플로우를 응용한 웹 서비스 설계 및 구현

이성규[†], 김태석[††]

요 약

레거시 시스템이나 특정 벤더의 애플리케이션이 혼재 되어 있는 기존 IT 환경에서 새로운 기술을 도입한다는 것은 많은 비용과 시간이 소요되는 작업이다. 특히 특정 소프트웨어나 일부 상용 애플리케이션에서 상호 연동성이 높지 않은 경우 이러한 상황이 더 많이 발생하며 시스템 도입 중 예상하지 못하는 상황이 발생할 가능성이 높아 새로운 애플리케이션을 적용하는 시간이 더욱 길어지게 된다. 서비스 지향 아키텍처를 기본으로 인터넷 애플리케이션을 생성하면 연결된 시스템 구성에 대해 유연한 처리 방법과 느슨한 결합을 기반으로 하는 아키텍처를 구현 할 수 있다. 이러한 접근방식은 XML 기반의 Web Service를 활용하여 서비스의 통합과 재사용을 통해 복잡한 처리를 요구하는 인터넷 환경의 워크플로우 모델 개발의 가능성을 살펴보고, 하나의 복잡한 비즈니스 프로세스와 다양한 서비스들이 워크플로우를 통해 재사용 될 수 있는 방법을 제안한다.

Design and Implementation of Web Service Applying SOA Based on Workflow

Seong-Kyu Lee[†], Tai-Suk Kim[††]

ABSTRACT

Incorporating proprietary existing it solutions like legacy systems or vendor specific with new technologies is an expensive and time consuming task. Such situations take place due to the lack of ability of proprietary software to cooperate with other parties or to cooperation only with specific vendor products. Such a situation is undesirable and causes a prolonged adaptation period for new applications. This thesis is to show the new approach to creation of Internet applications in Service Oriented Architecture through loose coupling, introduces fare more flexibility into a system composed of connected applications. This approach allows one to integrate through XML based Web Service and reuse a number of arbitrary services available over the Internet in a complex processes specified as a workflow model.

Key words: Workflow(워크플로우응용), Web Service(웹 서비스), SOA(서비스 지향 아키텍처), XML (확장성 생성 언어)

1. 서 론

최근 몇 년 동안 IT 업계에서의 통합 솔루션에 대한 요구가 급격히 증가 하고, 그에 따라 유연한 소프트웨어 제품군들이 많이 나오고 있음을 알 수 있다. 이러한 이유는 서로 다른 IT 환경이나 기술 상호간 연동에 많은 비용 투자가 필요하기 때문이다. 기업이 필요로 하는 모든 기능을 단순히 하나의 애플리케이션으로 모두 담당할 수 없기 때문에 애플리케이션의 통합과 구성은 반드시 필요하게 되었으며[1], 그에 따라 재사용에 대한 문제가 중요시 되어 재개발(reengineering)을 하는 방법과 통합(integration)을

※ 교신저자(Corresponding Author) : 김태석, 주소 : 부산광역시 부산진구 엄광로 995번지(614-714), 전화 : 051) 890-1707, FAX : 051)890-2629, E-mail : tskim@deu.ac.kr
접수일 : 2008년 8월 18일, 완료일 : 2008년 10월 1일

[†] 정회원, 동의대학교 컴퓨터소프트웨어공학과 초빙교수 (E-mail : seongkl@microsoft.com)
[††] 종신회원, 동의대학교 컴퓨터소프트웨어공학과 교수

하는 방법을 이용하고 있다[2]. 통합을 위한 대표적인 개발 형식으로는 서로 다른 종류의 컴퓨터들 간에 상호작용을 하기 위한 소프트웨어 시스템인 웹 서비스가 있다. 그러나, 실제 업무 환경에서 하나의 웹 서비스에 대한 처리는 간단하게 나타낼 수 있지만 복잡한 애플리케이션에 대한 웹 서비스는 많은 요구조건들이 필요하게 되며 이런 경우 워크플로우 모델을 활용할 수 있으며 업무간의 데이터 흐름이나 조절을 구성하기 위해 전용 개발 언어로 처리되는 복잡한 로직을 캡슐화 할 수 있다[3].

본 논문에서는 서비스 지향 아키텍처를 기본으로 인터넷 애플리케이션을 생성하는 새로운 접근 방법을 보여주고자 하며, 연결된 시스템 구성에 대해 유연한 처리 방법과 느슨한 결합을 기반으로 하는 웹 서비스를 통하여 통합과 재사용을 이용한 복잡한 처리를 요구하는 인터넷 환경의 워크플로우 모델 개발을 연구 하였다.

2. 관련 기술 및 연구 배경

웹 서비스는 엔터프라이즈 환경에서 분산 처리에 대한 문제 해결을 하기 위해 최근에 많이 시도되고 있으며 이전의 다른 방식과는 확연한 차이를 가지고 있다. 2장에서는 웹 서비스의 기본이 되는 XML 기술과 서비스 지향 아키텍처에 대한 기반 기술을 살펴본다.

2.1 XML 처리 - 네임스페이스, 스키마, XSLT

XML(eXternsible Markup Language) 문서는 상당히 복잡하고 구조화 된 데이터를 포함하고 있기 때문에 문서 내에 존재하는 XML 문법들을 재사용하는 것은 XML 작성 시 많은 노력을 절약할 수 있고 XML 처리기가 엘리먼트의 분석을 보다 빨리 처리할 수 있게 한다. XML 1.1의 네임스페이스 규약에 의하면, 네임스페이스는 XML 요소, 속성 이름을 URI(Uniform Resource Identifier)로 식별되는 이름과 연관시켜 구분하고 전 세계적으로 고유한 요소나 속성 이름을 통해서 상호간 이름 생성에 대한 충돌을 방지하고 의미를 구분하게 한다[4]. 그리고 DTD (Document Type Definition)를 포함한 스키마의 재사용에도 활용될 수 있다. 최근에는 이미 존재하는 XML 문서를 HTML 컨텐츠로 변형시키는 것이 주요 관심사가 되면서 인터넷 브라우저가 XML 데이터를 원할히 표현하기 위해 연구 개발된 것이 바로 XSL(eXtensible Stylesheet Language) 이다[5]. XSL은 XML을 사용하고 있는 웹을 통해 보내어지는 데이터가 사용자에게 어떻게 보여질 것인지를 설명한 스타일 시트를 만드는데 사용되는 언어로서 XSLT(XSL Transformation)와 XSL 형식화(Formatting)라는 두 가지 단계로 구성된다.

2.2 표준 웹 서비스 아키텍처

웹 서비스는 네트워크상에 자신이 제공할 수 있는 서비스가 무슨 내용인지, 어디에 있는지 또 어떻게 서비스를 요청할 수 있는지에 대한 내용을 만들어 모듈화되어 발표된 애플리케이션이다[6]. 웹 서비스는 웹으로 대변되는 인터넷, IT 그리고 객체지향 기술의 통합으로 프로그램들 사이에 원활한 통신과 서비스를 위해 다양한 표준을 사용한다. 웹 서비스 아키텍처는 스택(Stack)을 구성하는 다양한 구성 부분과 기술의 관계를 설정하고 있다. 이 스택을 살펴보면 아래의 그림 1과 같다.

SOAP(Simple Object Access Protocol)는 HTTP와 같은 프로토콜 위에 있는 XML-based Messaging Facility이며. WSDL(Web Service Description Language)은 웹 서비스의 위치와 그 내용을 기술한다. UDDI(Universal Description Discovery & Integration)는 자신의 서비스를 등록하고 또 다른 서비스를 발견할 수 있는 Facility를 제공하며 WSFL(Web Service Flow Language)은 일련의 웹 서비스를 통한 비즈니스 애플리케이션의 플로우를 만들 수 있도록 한다. 현재 웹 서비스 아키텍처에서의 보안 및 관리 기능, 그리고 서비스의 질에 대한 표준화 작업이 계속 연구 중에 있다[7]

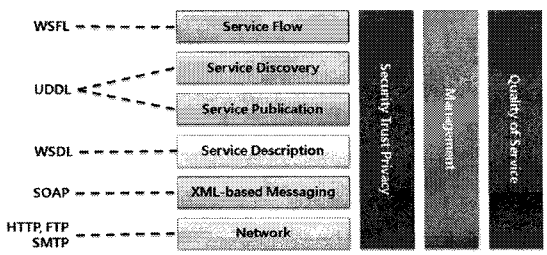


그림 1. 개념적인 웹 서비스 스택

2.3 서비스 지향 아키텍처(Service-Oriented Architecture)

서비스 지향 아키텍처는 로직의 개별 단위들이 서로 고립되지 않고 상호 작용하면서도 자율적으로 존재할 수 있도록 한다. 로직 단위는 원칙에 따라 공통성과 표준성을 유지하며 독립적으로 구성되며, SOA에서는 이러한 로직 단위를 서비스라고 한다. SOA는 이러한 서비스들을 기반으로 하는 소프트웨어 아키텍처로 애플리케이션의 기능들을 사용자(Consumer)에게 적합한 크기로 공개한 서비스들의 집합을 제공하고 정책(Policy) 또는 프레임워크(Framework)를 통해 바인딩 가능하도록 구현한다. 이 때 서비스는 단일한 표준 기반의 인터페이스 형태를 통하여 구현하며 독립적으로 추상화되고, 호출(Invoke), 공개(Publish), 발견(Discovery)과 같은 오퍼레이션을 수행 한다. 즉, SOA란 서비스로 정의되는 분할(Decomposition)된 애플리케이션 조각들을 단위로 느슨하게 연결해 하나의 완성된 애플리케이션으로 만드는 아키텍처이다[8].

3. 웹 서비스를 위한 워크플로우 모델링

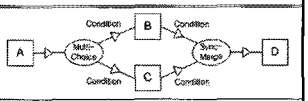
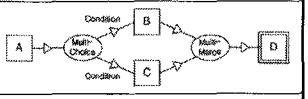
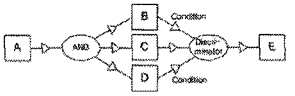
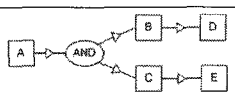
분산 애플리케이션의 구조를 체계화하기 위해서는 3-Tier 아키텍처로 불려지는 비즈니스 계층, 프리젠테이션 계층 등이 필요하다[9]. 본 논문에서 가장 중심적으로 다루고자 하는 내용은 비즈니스 로직 계층에 대한 부분이다. 분리된 애플리케이션의 비즈니스 로직들은 워크플로우 모델로 열거될 수 있으며 이러한 워크플로우 모델(비즈니스 프로세스)은 복잡한 프로세스로 지정된 업무들을 실행하기 위한 결합으로 구성된다. 이번 장에서는 본 연구에서 구현하고자 하는 모의 주식 거래 시뮬레이션 웹 서비스에 필요한 워크플로우 패턴을 기반으로 애플리케이션 활동을 모델링 하는데 필요한 다양한 패턴을 설계 하고, XPDL(XML Process Definition Language)에서 지원하지 못하는 부분에 대해서는 필요한 속성을 추가하여 이번 연구에서 제시하는 프로세스 모델링에 반영한다.

3.1 워크플로우 패턴 설계

페트리 넷은 플레이스와 트랜지션의 집합으로 구성된 그래프 기반 정형화 도구로서 특히 병렬 시스템의 동적 특성을 효과적으로 표현하고 분설할 수 있는 방법을 제공한다[10]. 페트리 넷을 이용하여 워크플로우를 표현하면 토큰의 이동에 따른 동적 변화를 표현할 수 있기 때문에 워크플로우의 구조적 특성은 물론 동적인 특성까지 함께 모델링이 가능하다. 페트리 넷을 이용하여 표1 같이 기본 패턴을 응용하고 다양한 조건에 만족할 수 있는 워크플로운 패턴을 설계하고 전체 ERD(Entity-Relationship Diagram) 다이어그램에서 활용하였다.

표 1의 다양한 패턴을 표현하는데 필요한 정보를 이용하여 XPDL에서 지원하지 못하는 패턴을 표현할 수 있도록 필요한 정보를 추가하고 본 연구에서

표 1. 워크플로우 패턴

이름	설명	패턴
Synchronizing Merge	프로세스 안의 한 액티비티는 같은 프로세스 안의 수행할 수 있는 이전 액티비티중에 워크플로우 관련 데이터를 기반으로 트랜지션 조건을 만족하는 모든 액티비티 완료 후에 수행된다.	
Multi Merge	프로세스 안의 한 액티비티는 같은 프로세스 안의 이전 액티비티들이 비동시적으로 결합된다.	
Discriminator	프로세스 안의 한 액티비티는 같은 프로세스 안의 이전 여러 액티비티들 중 조건을 만족하는 수의 액티비티가 완료되면 수행된다.	
Cancel Activity	프로세스 안의 한 액티비티는 프로세스 실행중에 액티비티를 생략할 수 있다.	
Implicit Termination	프로세스 안의 액티비티 중 더 이상 실행할 수 있는 액티비티가 없을때 프로세스를 종료시킨다.	

제안하는 웹 서비스 애플리케이션에 대한 ERD작성에서 복잡한 비즈니스 로직 처리에 적용하였다.

3.2 W-BPEL 적용

BPEL(Business Process Execution Language)은 개별적인 여러 서비스를 앤드-투-앤드 프로세스 흐름으로 구성하는 방법을 제시하는 업계 표준이다. 즉 웹 서비스에서 비즈니스 프로세스를 사용 가능하게 하고 정의하는 데 쓰이는 언어다. BPEL4WS(BPEL for Web Services)는 WSDL, XML Schema, XPath에 의존하며 이들 중, WSDL이 BPEL4WS에 가장 큰 역할을 한다. BPEL4WS는 XML을 기반으로 만들어진 언어이기 때문에, <process>라는 루트 엘리먼트 아래에 여러 정보를 구조적으로 담고 있으며, Primitive 액티비티와 Structured 액티비티로 크게 구분할 수 있다.

3.2.1 Primitive 액티비티

BPEL에서 정의 할 수 있는 액티비티는 크게 기본 액티비티와 흐름을 나타내는 구조적 액티비티로 구분 할 수 있다. 아래 표 2에 BPEL 기본 액티비티에서 제공하는 필요한 내용을 설명하며 각 기능을 요약하였다.

표 2. BPEL Primitive 액티비티

이 름	역 할
Receive	파트너에 의한 지정된 작동의 호출을 통해 매시지를 수락한다.
Reply	Receive 액티비티를 통해 이전에 수락된 요청에 대한 응답으로서 메시지를 보낼 때 사용한다.
Invoke	파트너가 제공한 웹 서비스 작동을 호출한다.
Assign	컨테이너에 있는 데이터를 생성, 수정하는 데 사용된다.
Empty	실행 효과가 없다.
Throw	프로세스가 오류 신호를 보낼 때 사용함
Wait	정해진 시간 동안 또는 데드라인에 도달했을 때 프로세스 지연 시작함
Compensate	실행된 지정 범위에서 보상 핸들러 호출이 이루어진다.
Terminate	비즈니스 프로세스 인스턴스가 그 실행을 포기하도록 한다.

표 3. BPEL Structured 액티비티

이 름	역 할
Flow	병행성과 동기화를 제공한다.
Scope	액티비티를 위한 실행 범위를 정한다.
While	지정된 반복 액티비티의 지원, 지정된 부울 조건이 true를 포함하지 않을 때 가지 실행이 지속된다.
Sequence	순차적으로 실행되는 한 개 이상의 액티비티를 포함한다.
Pick	한 개 이상의 이벤트 발생을 기다리고, 발생한 이벤트와 관련된 액티비티를 실행한다.
Switch	지정된 조건에 의존하는 한 개 이상의 케이스 분기 실행을 지정할 수 있도록 하여 조건 작동을 지원한다.

3.2.2 Structured 액티비티

BPEL4WS의 흐름 정의 액티비티는 BPEL4WS 문서의 구조화 작동을 위한 것으로 표 3에서 각 내용에 대해 설명 한다.

4. 워크플로우를 응용한 웹 서비스 설계 및 구현

본 논문에서 구현한 가상의 시스템은 주식 거래에 대한 비즈니스 로직을 구현해 보았다. 주식 거래 시스템은 외부로는 은행이나 다른 금융 시스템과 연계가 필요하며 내부로는 매매, 주문요청 등에 대한 서비스 통합을 필요로 하므로 워크플로우를 적용하여 비즈니스 로직을 구성하기에 가장 적합한 사례로 판단하였다. 구현하고자 하는 시스템의 목표는 서비스 지향 아키텍처를 기반으로 분산 애플리케이션이 다른 애플리케이션과 완벽히 조합이 가능한지 살펴보고 BPEL4WS를 이용한 서비스 캡슐화를 통하여 워크플로우 모델 기반의 비즈니스 프로세스를 웹 서비스에 적용하고자 함이다.

4.1 시스템 구조 설계

사용자의 복잡한 비즈니스 로직을 처리하기 위해 유연하며 관리와 확장성이 가능하도록 3-Tier (Presentation, Database, 그리고 Business Logic layer)기반의 시스템 아키텍처로 구성 하였다.

이러한 구성에서는 각 레이어에서의 문제가 다른 계층에 영향을 미치지 않는 다는 장점이 있다. 그림

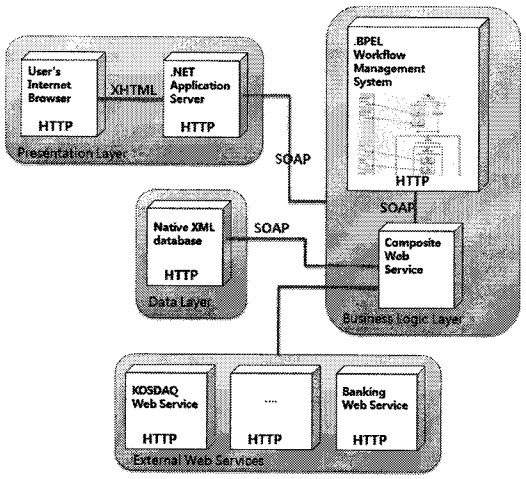


그림 2. 시스템의 분산 컴포넌트 구조

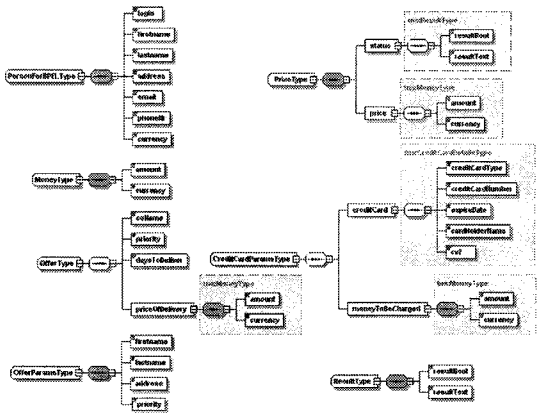


그림 3. XML 스키마의 확장된 구조

2는 구현 시스템의 분산 컴포넌트 구조를 나타낸다. 비즈니스 로직 계층에서 SOA의 기본 개념을 도입함으로서 시스템의 유연성을 높이게 되며 다수의 웹 서비스 집합을 워크플로우를 통하여 조합하게 된다. 본 시스템의 비즈니스 로직 레이어에 사용한 워크플로우 모델은 BPEL 기준 모델에 따라 구성 하여 특정 플랫폼이나 벤더의 종속성을 가지지 않는다. 데이터베이스 계층은 순수 XML 데이터베이스를 받아 저장하고 웹 서비스 커뮤니케이션에서 발생하는 요청을 XQuery 언어를 이용하여 데이터베이스로부터 직접 필요한 결과를 얻을 수 있다.

4.2 시스템 디자인 및 구현

본 연구에서 설계한 모의 주식 거래 시스템에 대한 비즈니스 로직은 크게 두 부분으로 나뉘어져 있다. 웹 서비스를 통해 제공되는 업무 처리와 BPEL 워크플로우 모델로 구성된 서비스의 느슨한 결합 이다. 주식 거래 시스템을 위해 생성되고 재사용되는 웹 서비스는 하나의 웹 서비스 집합이며 다중 오퍼레이션 기능을 가지고 있으며 XML 데이터를 요구하는 모든 웹 서비스 집합은 XML 스키마를 참조 하게 된다. 공통 데이터 모델이 여러 곳에서 사용되는 경우 XML 스키마를 사용하여 변경된 항목에 대해 각 모듈의 소스코드 수정 없이 XML 스키마의 수정을 통해 모두 변경 내용이 적용될 수 있다. 구성된 웹 서비스로부터 WSDL 파일을 생성하여 전체 시스템의 XML 스키마를 보다 확장할 수 있으며, 아래 그림 3는 확장 스키마 구조 이다.

BPEL 적용 워크플로우 모델을 작성하기 위해 먼저 공용 프로세스(Public Process) 모델을 설계하였다. BPEL 워크플로우의 공용 프로세스는 웹 서비스 인터페이스를 통해 표현되며 사용자는 인터페이스 접근을 통해 비즈니스 로직에 접근할 수 있다. 사용자나 클라이언트에서 많은 인스턴스들이 동시에 요청 처리를 받더라도 비동기 커뮤니케이션 방식을 통해 시스템 블록킹현상을 방지할 수 있다. 비즈니스 로직의 응답은 대기중인 사용자를 위해 끝점(endpoint) 위치로 다시 콜백되어 처리 한다. 공용 프로세스의 일부분으로 개인 프로세스는 외부에 연결될 필요가 있을때 캡슐화 작업을 통해 연결되기도 한다. BPEL 개인 프로세스는 외부로 노출되지 않지만 WSDL 파일로 표현 되는 공용 프로세스는 BPEL 개인 프로세스와 상호작용을 허용한다. 그러나 대부분의 개인 프로세스는 내부 업무에서 핵심 적인 부분의 처리를 담당하는 비즈니스 로직을 담당하는데, 그림 4는 웹 서비스에서 비즈니스 조회업무를 이용하여 조건에 맞는(예, 가장 낮은 가격) 로직을 처리하는 부분을 나타내고 있다. 웹 서비스가 여러 개의 다중 액티비티를 수용하고자 할 때 비동기처리 방법으로 진행되는 패턴으로 구현하였으며 스위치 조건에 따라 다음 액티비티가 처리되게 된다.

XML 스키마와 BPEL 워크플로우 모델을 이용하여 비즈니스 로직을 설계하고 실제 비즈니스 로직 구성을 위한 코딩을 위해 각 내용에 따라 업무 단위로 스키마 정보를 가져왔다. 스키마에서 들어온 해당 정보를 클래스 매핑 작업과 속성 정보를 입력하는

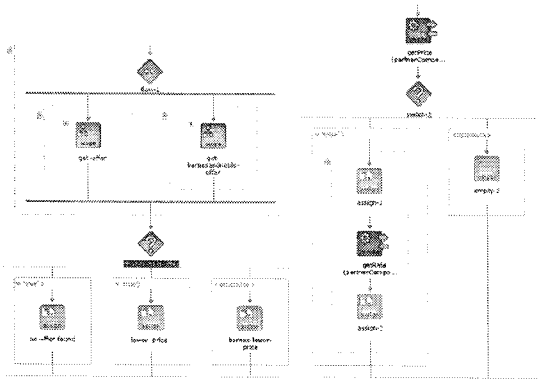


그림 4. 조건에 따라 진행되는 Private Process

방식으로 시스템을 구현하였다. 각 클래스 정보는 태스크 단위에 따라 구분된다.

5. 성능 분석 및 평가

5.1 시뮬레이션 환경 구성

본 논문에서 제시한 BPEL 워크플로우 기반의 비즈니스 로직을 처리하는 시스템 성능을 상대적으로 비교하기 위하여 워크플로우를 사용하지 않고 일반적인 애플리케이션과 같이 순수한 코딩만으로 구성한 동일 로직의 프로그램을 작성하여 테스트 결과를 비교하였으며 비즈니스 로직의 성능평가에 사용된 시스템 환경은 다음과 같다.

- OS : Windows Server 2003, IIS 6.0
- .NET Framework 3.0 기반의 ASP.NET웹 서비스
- CPU : 3.1Ghz x 2(Dual)
- RAM : 2GB
- 부하테스트 : Application Center Test (ACT)
- 성능 수집도구 : Windows Performance Monitor

5.2 RPS 성능 비교

ACT를 이용하여 사용자의 요청(Request)에 대한 시스템 처리량의 차이를 살펴보았다. RPS는 시스템이 매초당 수행한요청 처리회수를 나타내며 RPS (Request per Second) 값으로 표현한다.

그림 5는 4장에서 제시한 워크플로우를 적용한 웹 서비스 환경의 테스트(1)와 워크플로우를 적용 하지

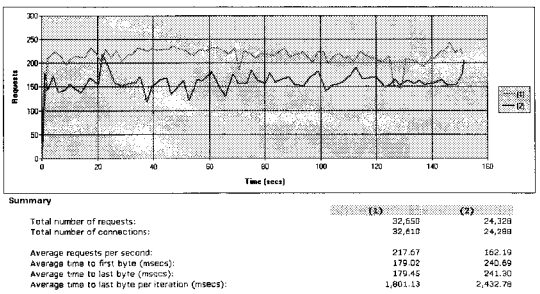


그림 5. RPS 성능평가 결과

않은 환경(2)의 테스트 비교 차이를 확인 할 수 있다. 평균 RPS가 (1)번 테스트에서는 RPS가 217.67을 기록하였으며, (2)번 테스트에서는 162.19를 나타내어 워크플로우를 적용한 웹 서비스 시스템이 초당 평균 55회의 Requests를 더 많이 처리하는 것을 확인 할 수 있었다. 또한 한 요청(Request)에 대한 평균 처리 시간도 (1)번 테스트가 평균 64.67ms 빨리 처리하고 있음을 알 수 있다. 동일한 시스템 환경에서 보다 높은 RPS가 나타날 수 있는 이유는 워크플로우에 의한 비즈니스 로직 처리 수행 시간이 그만큼 짧게 걸려 상대적으로 수행할 수 있는 요청건수가 많아진 것으로 해석할 수 있다.

시스템 자원을 얼마나 소모 하는지 비교 분석하기 위해 Request Queue와 CPU의 사용률에 대한 데이터를 확인 하였다. 시스템 성능 테스트는 동일한 시스템 상에서 동일한 부하 조건을 가했을 때의 차이를 보기 위한 것으로 부하의 가중치는 테스트 환경에서 구현된 시스템이 최적으로 처리할 수 있는 동시 접속자 40명을 기준으로 두 테스트 모두에 대해 공통으로 적용하여 진행하였으며 그 결과 워크플로우를 적용하지 않은 비즈니스 로직에서 평균 5건 정도의 요청(Requests)이 큐에 더 적체 되는 것을 알 수 있으며, 이는 실제 업무 환경에 적용되어 부하에 대한 가중치가 증가할수록 비례적으로 늘어날 것으로 예상된다. 표 4의 결과와 같이 Queue Length 의 최소 또는 최대 값 또한 상대적으로 워크플로우를 적용한 비즈니스 로직에서 보다 작은 값을 보여 주고 있다.

표 4. Queue Length 값 비교

비즈니스 로직	평균	최소	최대
Workflow 적용	14	4	31
Workflow 비적용	19	7	45

표 5. CPU 사용률 비교

비즈니스 로직	평균	최소	최대
Workflow 적용	67.622	39.5	100
Workflow 비적용	30.829	16.0	60.0

그러나 시스템 리소스 중에서 CPU에 대한 사용률은 오히려 워크플로우를 적용한 경우가 상대적으로 좀 더 높은 CPU 값을 나타내는 것을 확인 하였다. 이는 Queue Length 값을 측정할 때와 동일한 부하 조건에서 테스트를 진행하였으며 Workflow를 적용한 경우가 표 5에서 알 수 있듯이 약 두 배 정도의 CPU 리소스를 더 사용하고 있다.

테스트 과정을 통해 워크플로우를 적용한 웹 서비스의 경우 요청(Request)에 대한 응답 속도와 처리시간 등이 보다 단축되는 것을 알 수 있으며 엔터프라이즈 환경과 같이 전사환경에서 적용할 경우 효과는 더욱 클 것으로 보인다. 그러나 웹 서비스를 처리하는 프로세스의 핸들수의 증가로 인한 CPU 사용량에 따라 워크플로우를 적용한 SOA 기반의 웹 서비스를 구현 시에는 서버의 사이징이나 스펙 결정시 보다 높은 사양의 시스템을 필요로 할 것으로 판단된다.

6. 결 론

본 연구는 서비스 지향 아키텍처를 표준으로 하는 웹 서비스 애플리케이션을 구성할 때, 분산된 애플리케이션의 통합을 쉽게 적용할 수 있도록 워크플로우의 적용 방안을 모델링 하고 구현하였다. 웹 서비스는 BPEL 워크플로우 모델을 이용하여 복잡한 하나의 프로세스로 만들어지고, 이 프로세스를 통해 웹 서비스 통합이 가능함을 보여주었다. 제안된 시스템의 성능 평가는 비즈니스 로직을 담당하는 구현 모듈 중 일부에 대해 워크플로우를 적용하지 않는 방식과 비교하여 부하 테스트에 대한 시뮬레이션을 진행하였다. 사용자의 요청에 대한 응답 속도와 큐(Queue)에 적체 되는 비율 등을 확인한 결과 기존 환경에 비해 비즈니스 로직 처리의 신속성이 보다 높은 것을 알 수 있었으며 이는 대용량 사례로의 접근을 위해 엔터프라이즈 환경에 적용될 경우 일반적인 웹 서비스 환경 보다 높은 개선 효과가 기대된다.

마지막으로 향후 연구과제로서 웹이 사용자 요구를 만족시키는 방법에 대하여 보다 지능적으로 또한 환경에 따라 보다 직관적으로 만들어질 수 있다는 개념으로 도입된 시맨틱 웹의 연구를 본 논문에서 구현한 애플리케이션에 적용하고, WSMX와 같은 시스템을 통해 데이터와 프로세스의 불일치를 쉽게 완화할 수 있는 방법에 대해 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] C. Bussler. B2B Integration. Berlin, Heidelberg: Springer, 2003.

[2] M. Chowdhury and M. Iqbal, Integration of Legacy Systems in Software Architecture.

[3] A. Sharp and P. McDemott, Workflow Modeling: Tools for Process Improvement and Application Development. 2001.

[4] T. Bray, D. Hollander, and A. Layman, Namespaces in XML 1.1. *http://www.w3.org/TR/xml-names11/*. 2004.

[5] Assaf Arkin, Business Process Modeling Language, *http://www.bpmi.org,*. November 13, 2002.

[6] D. Booth. Web Service Architectural Roles. *http://www.w3.org/2002/ws/arch/2/10/r-oles_clean.htm, W3C Recommendation*, 2002.

[7] D. Booth. and H. Haas. Web Services Architecture. http://www.w3.org/TR/ws-arch/, *W3C Recommendation,* 2004.

[8] T. Erl. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. 2004.

[9] 조성진, ASP.NET 2.0 웹 프로젝트와 실전 프로그래밍, 2006.

[10] J. Peterson. Petri Net Theory and the Modeling of Systems. Prentice-Hall, 1981.

이 성 규

1900년 경남대학교 컴퓨터공학과 졸업(공학사)
1900년 광운대학교 정보통신공학전공 (공학석사)
2008년 동의대하교 컴퓨터응용공학전공 (공학박사)

2005년~현재 동의대학교 컴퓨터소프트웨어공학과 초빙교수
2000년~현재 한국마이크로포스트 이사 영남지사장
관심분야 : 웹사비스시스템, 정보시스템

김 태 석

1981년 경북대학교 전자공학과 졸업(공학사)
1989년 일본게이오대학 이공학부 계산기과학전공 (공학석사)
1993년 일본게이오대학 이공학부 계산기과학전공 (공학박사)

1993년 일본 국제전신전화연구소(KDD) 기술고문
1993년 일본게이오대학 이공학부 객원연구원
1994년~현재 동의대학교 컴퓨터소프트웨어공학과 교수
관심분야 : 인터넷비즈니스, 자연언어처리, 정보시스템