Facade 설계 패턴을 통한 컴포넌트기반 서비스지향 아키텍처에 관한 연구

박동식, 신호준, 김행곤
대구가톨릭대학교 컴퓨터정보통신공학부

A Study on the Component based Service Oriented Architecture through Facade Design Pattern

Dong-Sik Park, Ho-Jun Shin, Haeng-Kon Kim
Dept. of Computer Science, Catholic University of Daegu

요 약

웹 서비스는 인터넷을 통해 기업 상호간에 비즈니스를 연결할 수 있도록 하는 것으로, 서비스 구축의 개발비용의 감소와 구축속도 증진이 가능하다. 또한, 다른 영역과의 통합이 쉽게 가능하며, 컴포넌트 기반으로 개발할 경우 업데이트나 수정이 재사용성(reusability)과 대체성(replace ability)을 제공함으로써 용이하다.

본 논문에서는 서비스지향 아키텍처 상에서 공급자측면에서 구현되는 웹 서비스를 개발하기 위해서 구성되는 컴포넌트를 효율적으로 통합하기 위한 아키텍처를 제안하고자 한다. 비즈니스 로직을 웹 서비스로 제공하기 위해 먼저 서비스 지향 아키텍처를 정의하고, 이를 지원하고 웹 서비스를 제공하기 위한 컴포넌트를 Facade와 Backside로 정의한다. 특히, 서비스지향 아키텍처 상에서의 웹 서비스 지원을 위한 Facade 컴포넌트와 외부 컴포넌트간의 연결을 위해 Facade 설계 패턴을 적용하여 웹 서비스를 위한 컴포넌트 통합을 제시한다. 이를 통해 비슷한 어플리케이션을 작성시에 막대한 생산비용과 개발시간의 절감을 기대할 수 있으며, 컴포넌트를 기반으로 웹 서비스를 구성하여 재사용성과 대체성에 대한 신뢰성 향상을 가져온다.

1. 서론

IT 영역에서 여러 기업들이 수익성 확보를 위해 각 기업들이 협력을 하고자 노력하고 있지만, 각자의 기업 고유의 업무 프로세스, 즉 상이한 플랫폼으로 인한 통합의 어려움, 그리고 이익 창출에 중점을 둠으로써 타 기업과의 통합이 실패하거나 중단된 전자상거래 컨소시엄이 많다. 기존 개발 환경의 한계점으로 이러한 한계를 극복하고자 웹 서비스가 등장하게 되었다. 웹 서비스는 플랫폼, 특정언어, 특정 하드웨어로부터의 종속성을 탈피한 모델이다. 즉, 공통된 프로세스인 비즈니스 로직을 일정한 공급자가 컴포넌트 모델로 구현하고 그것을 각각의 사용자가 표준 프로토콜, XML, SOAP, WSDL, UDDI를 통하여 서비스를 받음으로써 자연스럽게 기업 간에 협력 발생을 유도한다.

본 논문에서는 서비스지향 아키텍처 상에서 공급자 측면에서 구현되는 웹 서비스를 개발하기 위해서 구성되는 컴포넌트를 효율적으로 통합할 수 있는 아키텍처를 제안하고자 한다. 비즈니스 로직을 웹 서비스로 제공하기 위해 먼저 서비스 지향 아키텍처를 정의하고, 이를 지원하고 웹 서비스를 제공하기 위한 컴포넌트를 Facade와 Backside로 정의한다. 특히, 서비스 지향 아키텍처 상에서의 웹 서비스 지원을 위한 Facade 컴포넌트와 외부 컴포넌트간의 연결을 위해 Facade 설계 패턴을 적용하고자 한다. Facade 설계 패턴 적용을 통해 웹 서비스 내부 컴포넌트간의 조립을 도우며, 외부 컴포넌트간의 통합을 용이하게 할 수 있다. 따라서, Facade 설계 패턴은 웹 서비스 내부의 아키텍처 명세에 도움을 주고 외부 컴포넌트와의 상호관계성 정의를 가능하게 한다. 또한, 비슷한 어플리케이션을 작성시에 막대한 생산비용과 개발시간의 절감을 기대할 수 있다. 또한, 웹 서비스를 직접 지원하고 상호 운용되는 Facade 컴포넌트의 설계를 통한 Facade 설계 패턴의 적용된 사례를 제시한다.

2. 관련 연구

2.1 서비스 지향 아키텍처

SOA는 서비스 지향 활동을 위한 기술적 기반이며, 원격으로 객체나 서비스에 접근하는 능력, 서비스 검색, 동적 바인딩 그리고 Loose coupled를 지원하는 특징을 가진다[1].

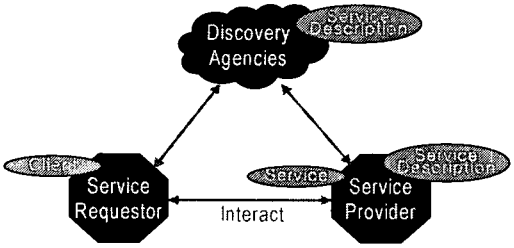

그림 1. Service Oriented Architecture

(그림 1)과 같이 아키텍처내부의 역할들은 Services Requestor, Discovery Agencies, Service Provider로 나누어져 있다. 서비스의 명세는 XML표준을 사용하며, 통신을 하기 위한 SOAP(Simple Object Access Protocol), 어떤 서비스를 제공하는지 기술하는 WSDL(Web Services Description Language), 서비스 브로커역할을 제공하는 UDDI(Universal Description Discovery and Integration)과 같은 표준을 사용한다. 서비스의 제공은 XML 기반의 메시지 흐름으로 이루어진다.

2.2 Facade 설계 패턴

패턴은 어떤 실제 상황에서 유용했던 아이디어로, 그것이 향후 다른 상황에서도 유용할 가능성이 있는 것이다. 패턴의 정의는 상당히 포괄적인데, 객체지향 개발에서는 설계 단계에 활용할 수 있는 객체 모형을 중심으로 디자인 패턴이라 하여, E.Gamma, Kent Beck, Ward Cunningham, Jim Coplein, D.Hay 등에 의하여 활발히 논의되어 왔다. 이러한 설계 패턴을 이용하여 설계한다면 시간과 노력을 줄일 수 있을 뿐만 아니라 더욱 효과적으로 설계할 수 있다[2, 3].

설계 패턴의 구성은 패턴 이름, 해결방법, 해결할 수 있는 문제, 예제 등으로 이루어지며, 해결하고자 하는 문제와 관련된 패턴을 찾고자 할 때는 각 패턴이 해결할 수 있는 문제를 검토해야한다. 또한, 각 패턴이 제시하고 있는 해결방법 및 예를 검토하여 해결하고자 하는 문제에 적용 가능한지 검토한다.

Facade 설계 패턴의 의도는 서브시스템을 합성하는 다수의 객체들의 인터페이스 집합에 대해 일관된 하나의 인터페이스를 제공할 수 있게 한다. 또한, 서브시스템을 사용하기 쉽게 하기 위한 포괄적 개념의 인터페이스를 정의한다. 그 목적은 그림 1과 같이 시스템을 서브시스템으로 구조화하는 것은 복잡성을 줄이기 위해서이다. 그러므로 일반적으로 설계 단계에서의 목표는 서브시스템간의 연결성과 종속성을 최소화하려는 것이다. 이런 목표를 달성하도록 도와주는 패턴이 Facade 객체로서 서브시스템의 복잡한 인터페이스를 단순화된 하나의 인터페이스로 제공하려는 것이다.

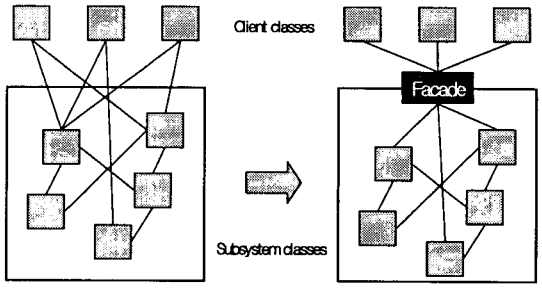

그림 2. Facade 설계 패턴의 구조도

3. Facade 설계 패턴과 컴포넌트 기반 SOA

웹 서비스의 개발은 크게 UDDI를 기준으로 Provision과 Consumer 두 부분으로 나눈다. Provision은 UDDI에 웹 서비스 등록을 위한 개발 프로세스이며, Consumer는 등록되어 있는 웹 서비스를 이용하여 개발하는 부분이다. 컴포넌트 기반의 웹 서비스를 만들기 위해서 고려한 부분은 전체 개발 프로세스에서 Provision에 대한 부분이다. 그 중에서도 테스트와 게시 부분은 생성된 웹 서비스에 대한 테스트와 UDDI 등록에 관한 부분이다.

본 논문에서는 웹 서비스를 컴포넌트로 구현할 수 있는 아키텍처와 이 아키텍처 내에서의 컴포넌트 조립 및 기본 구성을 Facade 설계 패턴에 기반하고자 한다. 그림 4는 기존의 주요 비즈니스 로직을 담당하는 핵심 시스템을 웹 서비스로 대응하며, 이를 컴포넌트 기반의 웹 서비스로 정의하기 위해 Facade 설계 패턴을 적용한 것이다. 방화벽을 중심으로 외부시스템 혹은 외부사용자 시스템과 웹 서비스영역으로 나뉜다. 프레젠테이션 계층은 User Web Session과 Workflow가 해당된다.

외부 사용자는 3가지의 방법으로 웹 서비스를 제공받을 수 있다. 첫 번째, 웹 브라우저를 사용하여 User Web Session에서 제공하는 웹 페이지의 형식으로 서비스를 제공받는다. 두 번째, 외부 시스템을 사용하여 Workflow 서비스를 통하거나 마지막으로, 직접 웹 서비스를 제공받을 수 있다. Workflow는 내부의 웹 서비스들에 대한 작업의 흐름을 제어하는 웹 서비스이다. 내부 사용자도 유사한 방법으로 서비스를 제공받을 수 있다. 단지 내부 사용자의 경우 방화벽 내부에 존재하기 때문에 방화벽을 통과하지 않아도 되고, 외부 시스템을 사용할 필요가 없다.

웹 서비스의 내부는 Backside 컴포넌트와 Facade 컴포넌트로 분류하였다. Backside 컴포넌트는 일반적

인 비즈니스 로직 컴포넌트와 데이터 컨트롤 컴포넌트와 같이 주로 내부 로직을 담당하는 컴포넌트들로 구성이 된다.

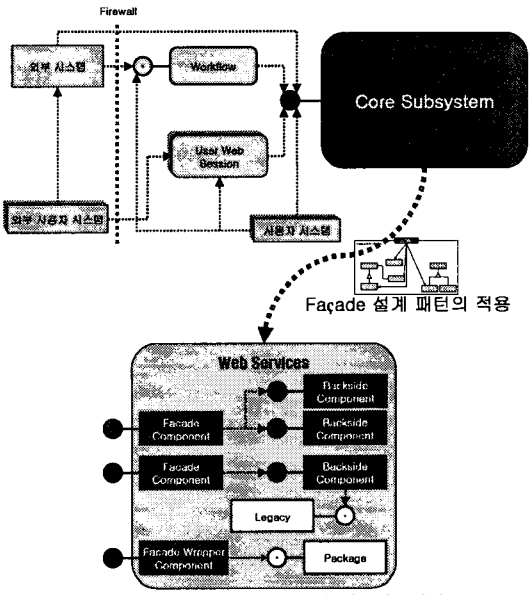

그림 3. Facade 설계 패턴의 적용된 컴포넌트기반 SOA

Facade 컴포넌트는 SOAP과 WSDL로 명세 되어 웹 서비스가 가능하도록 해준다. Backside 컴포넌트는 기존의 개발된 컴포넌트의 대체가 가능한 부분이라면, Facade는 웹 서비스의 주요 기능을 개발해야할 목표 컴포넌트이다. 이는 SOAP의 구현을 통해 실제적인 웹 서비스를 제공하고, Presentation Layer에 위치한 웹 서비스와 웹 기능 모듈과 상호 운용되며, 외부 시스템이나 외부사용자와도 통신이 가능하다.

웹 서비스의 내부 컴포넌트는 기존의 Facade 설계 패턴의 서브시스템이나 서브 클래스에 해당되며, 이를 통합하기 위한 Facade는 Facade 컴포넌트에 해당된다. 따라서, Facade 컴포넌트는 외부 시스템에 대한 통합된 인터페이스를 제공하고, 내부의 다른 컴포넌트와 상호연관서을 가진다.

4. Facade 패턴 적용 사례

서비스 지향 설계 단계에서는 Facade와 Backside 컴포넌트 명세와 아키텍처를 생성한다. 아키텍처 생성 시에는 Fade 설계 패턴을 적용하여, 구성할 컴포넌트의 통합을 위한 상호운영성을 고려하여야 한다. 또한, 서비스 지향 컴포넌트 구현 단계에서는 서비스 지향 설계에서 식별한 웹 서비스의 기능을 수행할 수 있는 서비스 지향 컴포넌트(Facade 컴포넌트)를 서비스 지향 컴포넌트 아키텍처를 기반으로 구현한다.

웹 서비스 조립 단계에서는 Facade 설계 패턴으로 구성된 서비스 지향 컴포넌트 아키텍처를 기반으로 서비스 지향 컴포넌트와 서비스 지향 설계에서 식별된 기존 자산을 조립하여 웹 서비스를 생성한다. UDDI 등록을 위해 조립된 웹 서비스에서 WSDL 문서와 SOAP 배포 명세를 정련화한다.

본 논문에서는 웹상에서 구축된 상용 컴포넌트 리파지토리에서는 비정기적으로 다양한 사용자가 검색을 통해 요구에 맞는 컴포넌트 식별 작업을 하는 것을 사례로 들고 있다. 적절한 컴포넌트가 있을 경우 이를 선택하고 구매한다. 이러한 시스템에서는 사용자 로긴을 통해 정보에 접근가능하고 운영 조직에 따라 분류 기준을 정의하고 이를 기반으로 컴포넌트 카테고리별 관리한다.

새로운 시스템에서는 사용자가 로긴했을 경우에 상용 컴포넌트 리파지토리에 새롭게 등록된 컴포넌트 목록을 자동적으로 제시하는 서비스를 하고자 한다. 이는 사용자가 최후 로그아웃된 시점에서 다음 로그인하는 시점사이에 갱신된 컴포넌트 목록을 보여주는 것으로, 각각의 사용자를 대상으로 개발 및 제공 하고자 한다.

그림 4는 컴포넌트의 리파지토리에 컴포넌트를 등록하고 검색하는 간단한 사례를 제시한 것이다. 이는 요구사항 분석 단계에서 작업의 흐름을 통해 어떠한 작업이 있는가를 파악하기 위하여 생성된 산출물이다. 이 모델을 통하여 어떠한 기능이 컴포넌트화 될 수 있는지의 요구사항을 파악할 수 있다.

그림 5는 초기 아키텍처 모델과 서비스 Workflow 모델을 기반으로 서비스 지향 컴포넌트 상호 작용 분석 단계에서 생성된 산출물이다.

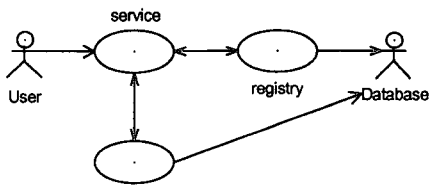

그림 4. Use Case Diagram

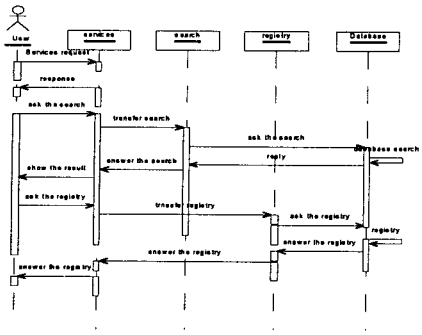

그림 5. Sequence Diagram

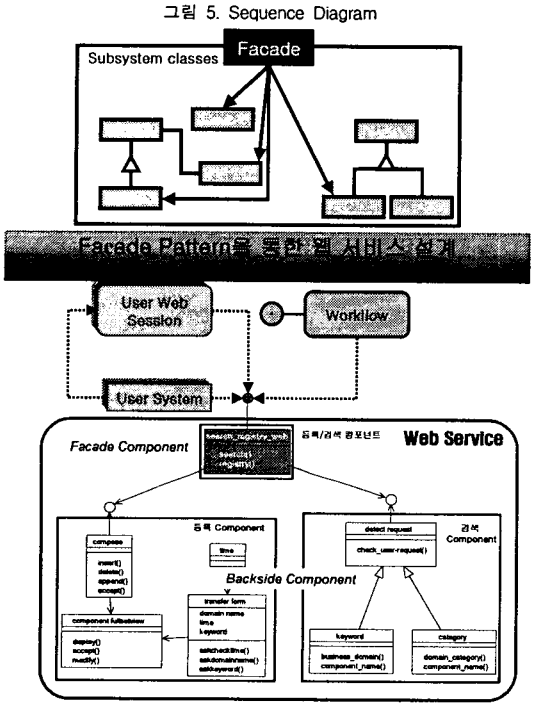

그림 6. Facade 설계 패턴을 적용한 등록 검색 웹 서비스

그림 6은 요구사항 분석 단계에서 생성된 유즈 케이스 모델을 사용하여 서비스 지향 컴포넌트 식별 단계에서 생성된 산출물이다. 그림에서는 서비스 개념 모델과 초기 아키텍처 모델을 함께 표현하였다. 웹 서비스를 제공하는 기능을 Facade와 Backside 컴포넌트로 구성된 것을 나타낸다. 특히, 웹 서비스의 등록과 검색을 위한 웹 서비스 내부의 컴포넌트를 Facade 컴포넌트를 통해 통합된 서비스로 제공하며, 외부에 사용자 시스템과 작업흐름 서비스와 연관시키고 있다. 이는 Facade 설계 패턴을 적용한 웹 서비스 아키텍처를 나타내며, 컴포넌트를 구성하는 클래스를 제시하고 있다.

5. 결론 및 향후 연구

다양한 벤더의 플랫폼(다양한 프로그래밍 언어와 OS)을 위해 작성된 애플리케이션들을 하나로 연결하는 상황이 올 때 웹 서비스가 큰 도움이 될 것이다. 일반적으로 웹 서비스는 네트워크를 통해 XML 포맷의 메시지를 보내서 액세스될 수 있는 소프트웨어이다. XML은 특정 프로그래밍 언어나 OS에 제약을 받지 않는다. 따라서 이종 시스템간 정보 교환용 프로토콜로 쓰일 수 있다. Web Services Definition Language (WSDL)은 웹 서비스가 제공하는 인터페이스를 기술하는데 사용된다.

이러한 웹 서비스 기술은 소프트웨어의 폭넓은 사용으로 인해 다른 영역과의 통합을 위한 부분에 효과적으로 사용될 수 있는 기술이다.

본 논문에서는 신뢰성과 민첩성을 보장하는 웹 서비스 개발을 위해 서비스 지향 아키텍처를 구성하였다. 이는 빠른 개발과 신뢰성과 재사용성을 보장하는 컴포넌트기반으로 아키텍처를 제공하였으며, 웹 서비스 내부에서의 컴포넌트를 Facade, Backside 컴포넌트로 정의하였다. 제시된 SOA(Service Oriented Architecture)에서 컴포넌트의 조립과 다른 외부 시스템과의 상호 운용성을 위해 Facade 설계 패턴을 통해 웹 서비스 내부의 아키텍처를 정의하고, Facade, Backside 컴포넌트의 통합을 쉽도록 한다. 또한, 웹 서비스를 직접 지원하고 상호 운용되는 Facade 컴포넌트 설계 및 사례를 제시함으로써, Facade 설계 패턴의 적용을 보였다. 향후 연구로써 Facade 및 Backside 컴포넌트의 구현 및 조립을 통한 웹 서비스 구성과 서비스 제공을 해야할 것이며, 다른 웹 서비스와의 성능에 대한 평가가 요구된다.

참고문헌

1. Michael Champion, Chris Ferris, Eric Newcomer, Iona and David Orchard, "Web Services Architecture : W3C Working Draft," http://www.w3.org/TR/ws-arch/, 2002.
2. Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns : Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.
3. Craig Larman, Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design, Prentice Hall, 1998.
4. John Cheesman, John Daniels, UML components A Simple Process for Specifying Component -Based Software, Addison-Wesley, 2001.

5. Richard Veryard, "Modeling for SOA," CBDi Journal, pp. 11-18, February, 2003.
6. Ivica Crnkovic, Component-based Software Engineering - New Challenges in Software Development, Software Focus, 2001.
7. 김민수, "웹 서비스 표준화," 정보처리학회지, Vol. 9, No. 4, pp. 31-35, 2002.
8. 최하정, 김행곤, e-비즈니스 컴포넌트 개발에 관한 설계 및 구현, 정보처리학회논문지D, Vol. 10-D, No. 1, pp. 85-100, 2002.