韓國IT서비스學會誌
第10卷 第2號
2011年 6月

AOP를 이용한 신뢰성 있는 서비스 어플리케이션의 SOA 기반 프레임워크*

김은선** · 이재정*** · 이병정****

A SOA based Framework Using AOP for Reliable Service Applications*

Eun-Sun Kim** · Jae-Jeong Lee*** · Byung-Jeong Lee****

■ Abstract ■

Loosely coupled properties of SOA(Service Oriented Architecture) services do not guarantee that service applications always work properly. Service errors may also influence other services of SOA. These characteristics adversely affect software reliability. Therefore, it is a challenge to effectively manage system change and errors for operating services normally. In this study, we propose a SOA based framework using AOP(Aspect Oriented Programming) for reliable service applications. AOP provides a way to manipulate cross-cutting concerns such as logging, security and reliability and these concerns can be added to applications through weaving process. We define a service specification and an aspect specification for this framework. This framework also includes service provider, requester, repository, platform, manager, and aspect weaver to handle changes and exceptions of applications. Independent Exception Handler is stored to exhibited external Aspect Service Repository. When exception happened, Exception Handler is linked dynamically according to aspect rule that is defined in aspect specification and offer function that handle exception alternate suitable service in systematic error situation. By separating cross-cutting concerns independently, we expect that developer can concentrate on core service implementation and reusability, understanding, maintainability increase. Finally, we have implemented a prototype system to demonstrate the feasibility of our framework in case study.

Keyword : Framework, SOA, AOP, Reliability

논문투고일 : 2011년 04월 13일 논문수정완료일 : 2011년 05월 16일 논문게재확정일 : 2011년 06월 11일

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2010-0025477).

** 서울시립대학교 컴퓨터과학과 석사과정

*** 더-케이손해보험(주) IT팀

**** 서울시립대학교 컴퓨터과학부 교수, 교신저자

1. 서 론

최근의 IT 비즈니스 환경과 기술은 빠르게 변화하고 항상 새로운 서비스가 나타난다. 따라서 현재 IT시스템은 시장 적시성(time-to-market)을 위한 비즈니스 요구에 민첩하게 대응해야 하고, 이러한 요구를 수용하기 위해 서비스 지향 아키텍처(SOA)가 제시되었다. SOA의 주된 목적은 서로 독립된 서비스 제공자의 서비스를 연합하기 위한 수단을 제공함으로써 유연성과 재사용을 향상시키는 것이다[1]. 그러나 느슨한 결합(loose coupling) 속성은 필요한 서비스가 항상 정상적으로 작동한다는 보장이 없기 때문에 시스템의 신뢰성(reliability)에 좋지 않은 영향을 준다. 본 논문에서 신뢰성은 프로그램의 런타임 시 어떠한 오류가 발생하더라도 정상적으로 작동하는 것을 의미한다. SOA 기반 어플리케이션은 실행 시간에 예외 메시지를 발생시키고, 분산된 환경에서는 어플리케이션들이 같은 예외 메시지를 발생시킬 수 있다. 그러나 현재의 웹 서비스 기반의 SOA 기술은 발생되는 문제에 정적으로 대처하기 때문에 이러한 상황을 효율적으로 대처하지 못해 SOA 기술의 발전과 확산을 가로막고 있다. 따라서 예외 상황을 동적으로 분산 처리할 수 있으면 어플리케이션 간의 상호작용에서 일어나는 예외상황에 보다 유연하게 대처할 수 있다. 다중 에이전트 시스템의 예외처리는 독립된 비즈니스 파트너 사이에서 사용될 수 있도록 일반적이고 분산적이기 때문에 이를 SOA에 적용하여 예외처리를 독립적으로 분리해야 한다.

SOA에서 서비스 개발자는 다른 사용자의 서비스를 사용하는 도중 발생하는 예상 가능한 비즈니스 레벨의 예외를 모두 처리하여야 한다. 하지만 이러한 예외처리 기능을 서비스의 핵심 기능과 함께 두면 코드의 중복으로 관리가 어려워지고 유지보수를 어렵게 한다. 따라서 동적으로 AOP를 사용하면 횡단 관심 코드들은 관점 모듈(aspect)로 분리되고 엮는(weaving)과정에 의해 합쳐지므로, AOP는 프로그램의 유지보수성을 증가시키고 새로운 외부 기능을 동적으로 추가할 수 있다.

또한 SOA에서 서비스 개발자는 의미적 매칭을 통해 서비스를 정확하게 검색할 수 있어야 한다. 그러나 XML 기반의 웹 서비스 인터페이스 명세인 WSDL(Web Services Description Language)[2]과 UDDI(Universal Description, Discovery and Integration)[3]는 의미 정보를 표현하지 못한다. 서비스에 대한 충분한 명세를 지원하지 않는 환경에서 서비스 제공자와 요구자는 요구 사항에 맞는 정보를 탐색하는 것이 어렵다. 따라서 이를 극복하기 위해 본 연구에서는 서비스 메타모델(4C model)의 개념을 적용한다. 이 서비스 메타모델은 특정 언어 및 플랫폼 독립적인 고수준의 추상화를 이용하여 서비스를 표현하고 서비스의 재사용을 높여준다.

본 연구는 SOA 기반 프레임워크에 동적 AOP 기술을 적용하여 서비스 어플리케이션의 신뢰성, 재사용성, 유지보수성을 향상시킨다. 제 2장에서 신뢰성 향상을 위한 다중 에이전트 시스템의 예외처리와 SOA에서 AOP 기술의 활용에 관한 연구를 기술한다. 제 3장에서는 본 논문에서 제안한 SOA 기반 프레임워크를 제안하며, 제 4장에서는 사례연구, 제 5장에서는 토의를 기술한다. 마지막으로 제 6장에서 결론 및 향후 연구 방향을 서술한다.

2. 이론적 배경

2.1 다중 에이전트 시스템과 예외 처리

기존의 동적 변경을 지원하는 시스템은 프로그래밍 언어를 활용하여 독자적으로 기능적/비기능적 요구사항 해결 방법을 제공하고, 표준 아키텍처나 기법을 지원하지 않아 확장성, 개방성에 한계가 있다[4]. 단위 서비스의 사용불가능, 실행실패 또는 QoS 속성을 위반하였을 때 안정적으로 시스템의 실행을 유지하기 위한 연구[5]는 과거 순차적인 시스템과 동일한 중앙 집중적인 오류처리로 인해 실행 중 에이전트가 추가되거나 변경될 수

있는 에이전트의 개방적 특성에서 한계를 가진다.

다중 에이전트 시스템의 예외처리는 순차적 또는 전통적인 분산시스템의 방식과 다르다. 독립된 비즈니스 파트너 사이에서 수행되는 공개된 프로세스는 환경에 따라 적응할 수 있는 유연한 프로세스를 요구한다. 따라서 다중 에이전트 시스템의 예외처리는 서로 다른 시스템에서 사용될 수 있도록 일반적이고 에이전트의 모듈화를 위해 분산적이어야 한다[6]. 위탁 프로토콜(commitment protocols)이라는 에이전트 상호간의 프로토콜에 기반을 둔 연구[7]에서는 분산된 방식으로 순향(proactive)에이전트를 위한 예외처리를 다룬다. 예외처리기는 수행(run)의 집합으로 취급되는 프로토콜로서 표현된다. 예상하지 못한 예외는 사전에 개발자에 의해 설계되지 않는 문제를 해결하기 위해 외부의 지식 저장소(knowledge repository)를 사용하고 예외처리기는 시스템 프로토콜을 이용해서 병합한다. 이러한 연구는 에이전트의 개방적 특성과 캡슐화(encapsulation)측면에서 유용한 접근법이지만 매우 이론적이기 때문에 실제 개발될 필요가 있다. 또한 모델이 운영적 메카니즘으로 전환될 수 있는 상태에 도달해야 하며 이에 대한 검증된 결과를 보여주어야 한다.

2.2 소프트웨어 동적 변경을 위한 AOP와 SOA

대부분의 시스템에서 예외처리 코드는 종종 몇몇의 부분에 중복되고 얽혀있기 때문에 유지보수와 코드의 재사용을 저해한다. 관점 지향 프로그래밍은 시스템 전역에 걸쳐있는 횡단 관심사를 모듈화함으로써 이러한 문제를 극복한다. SOA환경에서 이러한 관점 모듈이 원래의 서비스에 새로운 기능성을 동적으로 추가하고 변경하는 문제는 새로운 과제이다. 정적 AOP는 컴파일/링크 중에 관점이 삽입되고, 동적 AOP는 관점 모듈이 실행 중에 삽입되거나 변경되기 위해 사용된다. 이를 위해 동적 AOP는 변경된 자바 가상 머신(JVM)을 사용하거나 바이트코드를 수정함으로써 이루어진다[8]. 또 포인트컷(pointcuts)과 어드바이스(advice) 명세 파일을 수정함으로써 실행 시간에 메소드를 교체하는 동적 AOP를 활용한다[9]. 이 연구들은 자바 프로그램 언어 수준에서 동적 AOP를 제공하므로 언어에 종속적이고 서비스 재사용성에 한계가 있다. 또한 이 연구들은 SOA와 같은 표준 아키텍처를 지원하지 않는다.

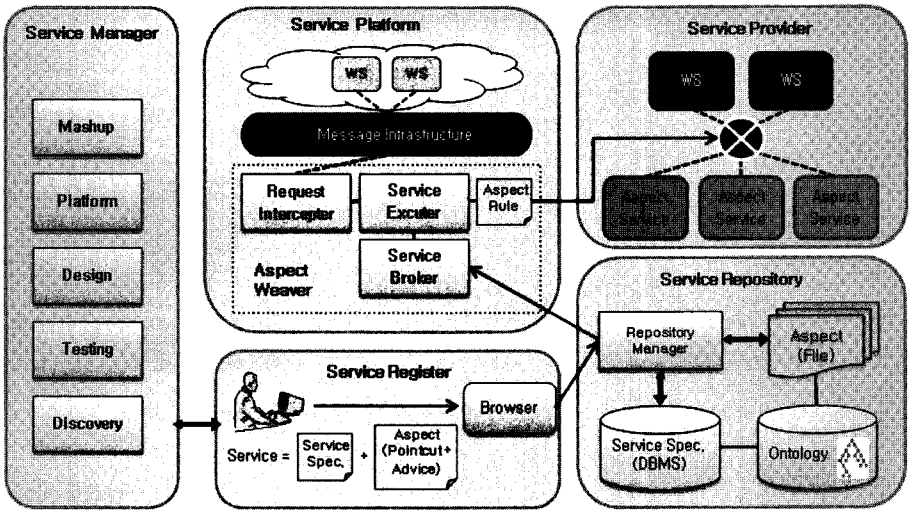
연구[10-12]는 SOA 기반 서비스 패러다임을 적용하여 시스템의 동적 변경을 다룬다. 먼저 서비스 제공자가 서비스를 중개자에게 등록하면 서비스 소비자는 필요한 서비스 명세(기능적 명세와 비기능적 명세 포함)를 작성하여 중개자에 등록된 서비스와 비교하여 매칭한다. 매칭 되는 서비스가 있으면 중개자는 그 서비스를 호출한다. 이 연구들은 시스템 동적 변경을 서비스 명세를 활용하여 다루고 있으나, 동적으로 비기능적 횡단 관심사를 다루기 위하여 동적 AOP를 적용하지 않는다.

동적 AOP와 SOA의 통합에 관한 연구는 서비스 실행 중 환경 혹은 요구사항의 변화에 따라 수행중인 서비스에 새로운 서비스의 기능을 추가 혹은 변경을 위한 기술적인 기초를 확립했다[13]. 웹 서비스의 적응성을 높이기 위해 AOP를 활용한 연구[14]에서 SOA환경의 클라이언트와 웹 서비스 사이의 횡단 관심 영역을 다루기 위해 WSDL(Web Service Management Layer)이라 불리는 추상계층을 이용한 AOP 플랫폼을 제안했다. JasCo[15]라는 AOP 언어에 기반을 둔 이 연구는 개별 서비스의 구현이 플랫폼에 종속적이 되어서 서비스의 재사용성에 한계가 있다. 또한 이 연구들은 AOP를 적용할 때 의미적 문제를 해결하는데 한계를 갖고 있다.

3. AOP를 이용한 SOA 기반 프레임워크

3.1 AOP를 이용한 SOA 기반 프레임워크

본 연구에서는 AOP를 이용하여 신뢰성있는 서



[그림 1] AOP를 이용한 SOA 기반 프레임워크

비스 어플리케이션을 위한 SOA 기반 프레임워크 [그림1]을 구축하는 것을 제안한다. SOA 기반 어플리케이션의 지속적인 실행을 보장하기 위해 예외처리기를 관점 서비스로 분리하고 외부의 서비스 저장소에 저장하였다. 서비스 저장소는 동적으로 의미적 매칭을 사용하여 서비스를 검색하고, 관점 명세서에 서술된 관점 규칙에 따라 시스템적인 오류 상황에 대처한다.

- 웹 서비스(WS)는 서비스의 핵심기능을 제공한다. 예외처리 기능과 같은 관점 기능은 관점 서비스로 독립되고 필요한 시점에 동적인 위빙 과정을 거쳐서 수행된다. 웹 서비스(WS)는 서비스 제공자로부터 제공되며, 서비스를 실행하기 위한 기능적/비기능적 특징이 정의 되어 있는 서비스 명세(service specification)의 정보를 가진다. 이 정보는 서비스 저장소에 있는 핵심 서비스들을 실행시키는데 필요하다.
- 관점 서비스(aspect service)는 관점 명세서에 정의된 관점 규칙(aspect rule)에 따라 관점 위버에 의해서 예외처리를 위해 동적으로 삽입되어 실행된다. 이러한 관점 서비스는 재사용 가능한 소프트웨어 컴포넌트로서의 역할을 가지고 있다.
- 서비스 저장소(service repository)는 공개된 서비스 저장소로써 웹 서비스와 예외 처리기와 같은 관점 서비스에 대한 서비스 명세 정보와 관점 명세서를 관리한다. 온톨로지 저장소는 시스템 내부에서 사용되는 도메인 온톨로지와 추상적인 예외 모델 온톨로지 그리고 서비스를 표현하는 서비스 온톨로지를 갖는다. 관점 위버의 요청에 의해 서비스의 동적인 검색을 위한 서비스 요청 설명과 의미정보를 사용한 서비스 비교(matching)를 수행할 때 참조된다.
- 관점 명세서(aspect specification)는 서비스 개발자가 핵심 서비스의 예외처리 기능을 담당하는 관점 서비스를 분리하기 위해 AOP 개념에 해당하는 포인트 컷과 어드바이스 정보를 포함한 관점 규칙을 명시한다.
- 관점 위버(aspect weaver)는 클라이언트의 서비스에 대한 호출 메시지를 가로채기(intercept)하여 해당 서비스 에이전트의 관점 명세서에 정의된 관점 규칙을 검사하고 적용하는 역할을 한다. 어드바이스의 종류(type)에 따라 원래의 서비스에서 사용되는 예외처리기 관점 서비스를 위빙하거나 실행 중 시스템적인 오류가 발생하였을 때 원래의 서비스와 기능 및 비기능적으로 동일한 서비스 에이전트를 동적으로 검

색하여 실행한다.

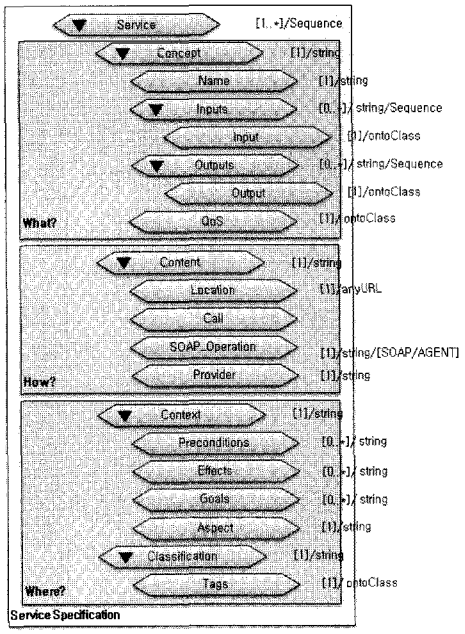
- 서비스 관리자(service manager)는 동적 AOP를 이용한 신뢰성 있는 서비스 특성을 시스템에 반영하고 전체 시스템을 구성하는 과정을 지원한다.

3.2 서비스 명세

서비스 제공자는 서비스에 대한 정보를 명확히 정의할 필요가 있다. 이를 위해 요구사항 정의 단계에서 얻은 정보를 각 서비스에 대해 <표 1>과 같이 명세한다.

서비스 명세에서 설명란은 서비스가 수행하는 작업에 대한 서술로서 해당 서비스가 수행하는 기능의 의도를 나타낸다. 구체적인 구현 방법에 대해서는 서술하지 않는다. 그리고 서비스 정보란은 서비스의 개념에 정의된 행위를 달성하고 기능적 명세를 구현하기 위한 코드에 대한 서술로서, 해당 서비스를 사용하는데 필요한 서비스 제공자, 위치와 같은 정보를 표현한다. 또한 연관된 서비스란에는 다른 서비스에 대한 종속 관계를 표현한다. 카테고리란에는 서비스의 카테고리와 관련된 정보를 구체화하는 형식을 나타낸다. 마지막으로 연관된 관점란에 예외상황이 발생했을 경우 이를 대처하기 위한 예외처리기나 대체될 서비스에 대한 명세를 한다.

이러한 명세 내용은 실제 서비스 구축 단계에서 서비스 명세 스키마로 발전한다. 하지만 WSDL은 해당 웹 서비스의 문법적인 구조에 대한 정보만 있을 뿐, 그 웹 서비스가 갖는 의미는 표현되지 않는다. 따라서 본 연구에서는 웹 서비스의 단점을 극복하고 서비스 실현을 위하여 XML 기반 서비스 메타모델(4C model) [그림 2]를 정의한다.



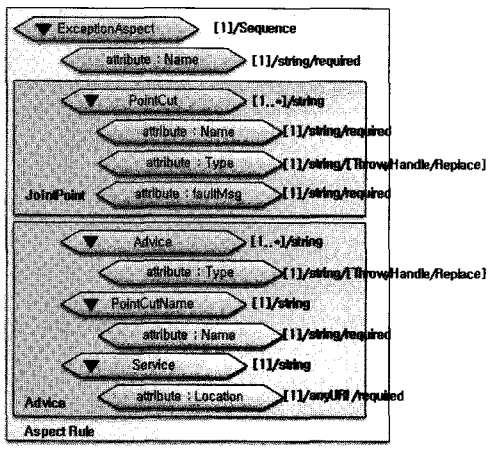
[그림 2] 서비스 명세 스키마

이 서비스 메타모델은 특정 언어 및 플랫폼 독립적인 고수준의 추상화를 이용하여 서비스를 표현하고 서비스의 재사용을 높여준다. 서비스 메타모델은 개념(concept), 내용(contents), 문맥(context), 분류(classification)를 포함하며, 앞의 요구사항 정의 단계에서 명세한 정보를 기반으로 확장한다. 개념 정보는 서비스 설명란으로부터 확장되며, 서비스의 입력, 출력과 같은 기능적 특성이 개념 요소에 표현된다. QoS에 기반을 둔 응답시간, 수행성능, 서비스 사용에 필요한 비용과 같은 비기능적 제약사항은 서비스 품질(QoS) 요소에 표현된다. 내용은 앞의 명세에서 서비스 정보란으로부터 확장되며, 해당 서비스를 사용하기 위해 필요한 자세한 정보를 표현한다. 이러한 정보는 서비스의 네트워크 위치, 호출방법, 예외 메시지 정의

〈표 1〉 서비스에 대한 명세

서비스	카테고리	서비스 설명	서비스 정보	연관된 서비스	연관된 관점
서비스 이름	카테고리 …	수행 작업 …	네트워크 위치 호출방법 …	연관된 서비스 …	예외 처리기 … 대체 서비스 …

그리고 서비스 제공자에 대한 정보를 포함한다. 문맥은 서비스 명세표의 연관된 서비스란에서 확장되고, 개념과 내용의 정의에 관련되어 서비스가 사용되는 소프트웨어적 환경에 대한 측면을 나타낸다. 또, 다른 서비스에 대한 종속 관계를 표현하고 서비스의 실행을 통해 산출되는 상태 변화를 나타낸다. 상태변화 정보는 전제조건(pre-condition)과 효과(effect) 특성을 통해 명시된다. 전제조건은 서비스를 요청하기 전에 요구되는 논리적 조건이나 상태를 정의하고 효과는 서비스가 성공적으로 실행된 후에 나타난 결과를 의미한다. 관점(Aspect)은 서비스에 위빙될 관점(예외처리 등)과 추후에 서비스 제공에 문제가 생겼을 경우 동적으로 대체될 서비스를 명세한다. 분류는 카테고리로부터 확장되며, 표준적인 영역 계층구조 및 태그와 같은 특정 분류 시스템을 포함할 수 있다.



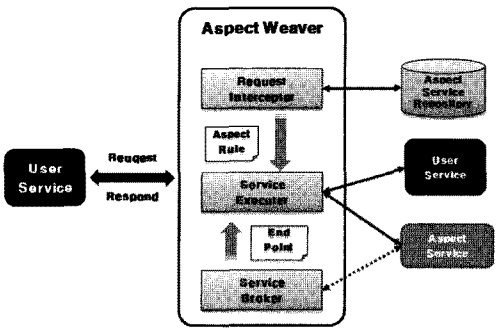
[그림 3] 관점 명세 스키마

3.3 관점 명세

일단 서비스가 개발되고 배포되면 서비스 내부에 새로운 예외처리 로직을 추가하기 어렵다. 그리고 예외처리 로직이 서비스 내부에 포함되어 있기 때문에 다른 환경의 서비스에서 재사용하기 어렵다. 이러한 문제를 해결하기 위해 본 논문에서는 AOP의 개념(포인트 컷, 어드바이스)을 활용한다. 독립된 서비스로 구현된 예외 처리기는 관점 명세에 정의된 관점 규칙에 따라 원래의 서비스에 동적으로 위빙되어 예외를 다루고 시스템적인 오류 상황에서 적합한 서비스를 대체하는 기능을 제공한다. [그림 3]은 서비스 어플리케이션의 관점 규칙을 선언하기 위한 XML 기반의 관점 명세 스키마를 보여준다.

<ExceptionAspect>요소는 <PointCut>과 <Advice>요소로 이루어진다. <PointCut>요소는 세 개의 속성을 가지는데, 이름(name)속성은 포인트컷의 유일한 식별자 이름이다. 종류(Type)속성은 해당 예외를 어떻게 다룰 것인지 지정한다. 예외(faultMsg)속성은 해당 서비스에서 발생 가능한 예외의 이름을 명시한다.

<Advice>요소는 예외를 다루는 관점 서비스를 사용하기 위해 필요하다. <Advice>요소의 종류(Type)속성은 "Throw/Handle/Replace"의 값을 가질 수 있다. 이름(name)속성은 미리 정의된 포인트 컷들 중 특정한 포인트 컷의 식별자 이름을 나타낸다. <Service>요소는 예외를 처리할 수 있는 관점 서비스의 URI를 나타내는 위치(location)속성을 가진다.



[그림 4] 관점 위버

3.4 관점 위버

[그림 4]는 클라이언트가 어떤 서비스를 실행하는 도중 예외가 발생하였을 때 예외를 다루는 관점 위버의 구조를 보여준다. 클라이언트는 원래의

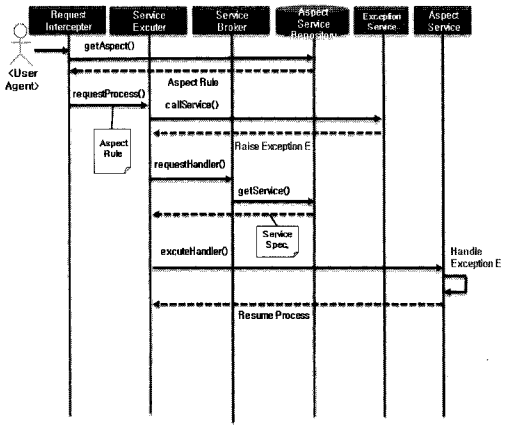
서비스 어플리케이션에게 요청 메시지 인터페이스를 이용하여 서비스를 호출한다. 요청 인터셉터(request intercepter)는 클라이언트의 요청을 가로채고 관점 서비스 저장소에서 해당 서비스에 대한 관점 명세를 가져온다. 그 다음에 요청 인터셉터는 관점 명세에 정의된 관점 규칙 정보를 서비스 실행자(service executer)로 전달한다. 서비스 실행자는 서비스 수행 중 예외가 발생하였을 때 어드바이스를 적용할 수 있는 포인트 컷이 있는지 검사한다. 그리고 서비스 실행자는 클라이언트의 요청 메시지를 이용해서 원래의 서비스를 수행한다. 만약 서비스 실행 중 포인트 컷에 명시된 예외가 발생한다면 서비스 실행자는 어드바이스의 타입 속성을 검사한다.

```
...
<ExceptionAspect name =
"UpdateMyStatusException">
<PointCut name = "twitter" type = "Handle"
faultMsg = "UpdateMyStatusException"/>
<Advice name = "twitter" type = "Handle">
<Service location =
"TwitterServiceAgent@se.uos.ac.kr : 1099/JADE"/>
</Advice>
</ExceptionAspect>
...
```

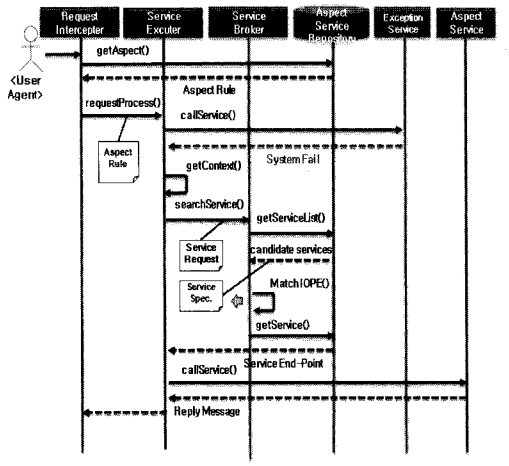
[그림 5] 예외 관점 명세의 예

[그림 5]와 같이 타입이 "Handle"이라면 서비스 실행자는 <Service>요소의 위치(location)에 명시된 관점 서비스를 서비스 중개자(service broker)를 통해 요청하여 실행한다.

[그림 6]은 관점 서비스를 요청하여 예외 처리기를 수행하는 프로세스를 보여준다. 먼저 서비스 중계자는 서비스 저장소에서 예외처리를 위한 관점 서비스를 가져온 후, 해당 관점 서비스의 실행을 위한 정보 (end point)를 서비스 수행엔진에게 되돌려준다. 서비스 수행엔진은 예외처리를 위해 엔드포인트 정보를 이용해서 관점 서비스를 수행한다. 만약 어드바이스의 타입의 "Replace"라면 서



[그림 6] 예외 처리기 프로세스

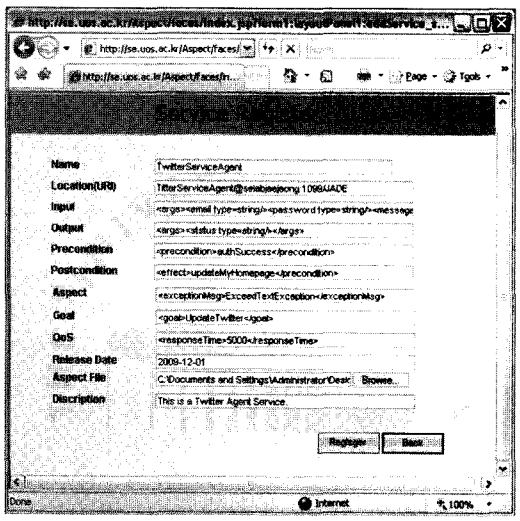


[그림 7] 관점 서비스의 검색 프로세스

비스 수행엔진은 서비스 중계자를 통해서 원래의 서비스와 동일한 관점 서비스의 검색을 진행한다. [그림 7]은 관점 서비스 검색을 위한 프로세스를 보여준다. 먼저 서비스 수행엔진은 현재의 문맥을 분석하여 서비스 요청메시지를 생성한 후 서비스 중계자에게 전달한다. 서비스 중계자는 관점 서비스 저장소에서 서비스 요청 메시지에 적합한 서비스를 검색하는 질의를 보낸다. 서비스 요청 메시지는 특정 서비스 개체를 요청하지 않는다. 요청 메시지는 필요한 서비스에 대한 의미적인 서비스 타입에 대한 정의를 포함한다.

4. 사례 연구

본 연구에서는 JADE를 사용하여 서비스 어플리케이션의 프로토타입을 구현하였다. 최근 기업들은 디지털 콘텐츠의 빠른 확산을 위해 트위터(Twitter)를 활발히 활용한다. 트위터는 홈페이지에 자신의 현재 상황을 텍스트로 업데이트 해주는 서비스이다. 본 연구에서 개발한 트위터 서비스는 트위터의 Open API를 이용하여 사용자의 트위터 홈페이지에 원하는 메시지를 업데이트할 수 있고 게시된 메시지를 확인할 수 있는 기능을 가지고 있다. 이 트위터 서비스를 사용해서 자신의 블로그 혹은 웹 사이트에 사진이 업데이트되면 팔로워에게 원하는 메시지를 보내주는 서비스를 개발하는데 활용할 수 있다.

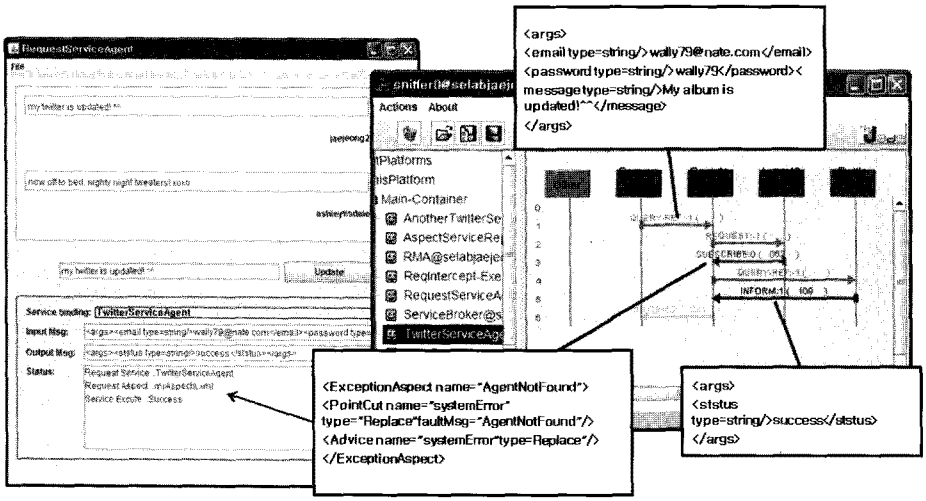


[그림 8] 서비스 등록기를 위한 입력 정보

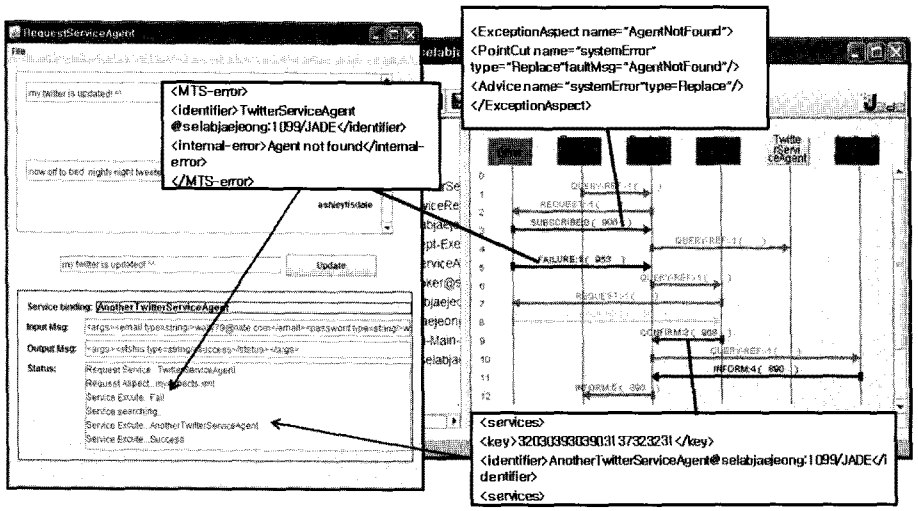
[그림 8]은 서비스 제공자가 서비스 등록기를 통해 트위터 서비스의 명세정보를 관점 명세서와 함께 서비스 저장소에 저장되는 화면이다. [그림 8]에서는 해당 서비스의 기능 및 접근을 위한 정보(URI)와 서비스의 품질에 기반을 둔 응답시간을 포함한다. 트위터 서비스를 사용하기 위해 자신의 계정정보(이메일, 패스워드)와 텍스트 메시지를 입력받고 등록 결과를 출력한다.

[그림 9]는 트위터 서비스를 사용하는 클라이언트의 요청에 대한 서비스 위빙 과정을 보여주는 화면이다. 트위터 서비스 사용 중 "AgentNotFound" 오류 발생 시 서비스 수행엔진은 "Replace" 어드바이스의 타입에 따라 또 다른 트위터 서비스를 이용해서 클라이언트의 요청을 처리하는 시나리오를 보여준다. 화면의 왼쪽 상단은 서비스를 호출하기 위한 클라이언트의 사용자 인터페이스를 나타낸다. 왼쪽 하단은 서비스 수행엔진이 사용 중인 서비스에 대한 입력 및 출력 정보와 서비스 위빙 과정을 보여주기 위한 로그 메시지를 보여준다. 그리고 오른쪽은 관점 위버를 구성하는 컴포넌트들 간의 상호작용을 보여주는 창이다. 서비스 클라이언트는 자신의 트위터 친구들에게 상태 변경을 알리기 위해 메시지를 입력하고 "Update" 버튼을 누른다. 이 때 클라이언트 프로그램은 트위터 서비스를 이용해서 상태변경에 대한 메시지를 등록한다. 관점 위버의 요청 인터셉터는 클라이언트의 요청을 가로채서 원래의 서비스에 대한 관점 규칙을 서비스 중계자를 통해 관점 서비스 저장소에서 가져온 후 서비스 수행엔진에 전달한다.

서비스 수행엔진은 원래의 서비스를 실행하고 관점 규칙에 정의된 "AgentNotFound" 오류가 발생하는지를 검사한다. 만약 서비스가 오류 없이 정상적으로 수행된다면(a), 서비스 수행엔진은 원래의 서비스에 대한 응답결과를 클라이언트에게 전달한다. 만약 서비스 실행 중 오류가 발생하면(b), 서비스 수행 엔진은 문맥정보를 생성하여 서비스 검색을 위한 요청 메시지를 생성한다. 그리고 서비스 요청 메시지를 이용하여 서비스 중계자를 통해서 관점 서비스 저장소에서 원래의 서비스와 동일한 기능의 새로운 트위터 서비스를 검색한다. 서비스 수행엔진은 검색된 서비스의 접근 정보를 이용해서 클라이언트의 요청을 수행하고 정상적인 수행 결과를 요청 인터셉터를 통해 클라이언트에게 되돌려준다. 클라이언트는 서버에서 발생한 서비스 위빙 프로세스를 알지 못하고 정상적으로 트



(a) 동적 서비스 위빙 이전



(b) 동적 서비스 위빙 이후

[그림 9] 동적 서비스 위빙 결과 화면

위터 서비스를 이용해서 메시지를 등록한다.

5. 토의

SOA는 분산된 네트워크 환경의 컴포넌트를 통합할 수 있고 최근 기업의 IT환경은 이를 바탕으로 급속히 변화하고 있다. 서비스 지향 어플리케이션은 서비스 사용자에 상관없이 모든 조건과 환경에서 일관성 있게 작동하는 것이 특징이다. 따라서 이러한 특성에 맞게 예외처리 역시 서로 다른 시스템에서 사용될 수 있도록 일반적이고 분산적이어야 한다. 본 논문에서 제안한 분산 예외처리 모델에서 하나의 서비스는 또 다른 서비스에게 비동기 메시지 기반의 예외발생 메시지를 전달한다. 이러한 예외 메시지는 서비스들 간의 통신 동안 발생할 수 있다.

〈표 2〉 관련연구와의 비교

비교항목 \ 관련연구	[5]	[14]	본 연구
개발 용이성	높음	높음	높음
재사용성	낮음	낮음	높음
유지보수성	낮음	낮음	높음

<표 2>는 관련연구와 본 연구의 비교항목에 대한 평가를 나타낸 것으로, 비교항목에 대한 고려가 충분하다고 판단되면 '높음', 그렇지 않으면 '낮음'으로 표현하였다. 가디언 예외처리 모델[5]은 글로벌 예외 처리기를 사용하며, 과거의 순차적 방식의 예외처리와 비슷한 중앙 집중적인 방식에 기반을 둠으로써 재사용성에 매우 취약한 문제가 있다. 연구[14]에서는 동적 AOP와 SOA의 통합으로 클라이언트와 웹 서비스 사이의 횡단 관심 영역을 다루기 위한 AOP 플랫폼을 제안했다. 이 연구에서는 횡단관심을 분리했다는 점에서 개발 용이성은 나아졌지만, 개별 서비스의 구현이 플랫폼에 종속적이어서 재사용성과 유지보수성이 취약하다. 반면, 본 논문의 아키텍처에서 예외 처리기는 공개된 외부의 관점 서비스 저장소에 저장되고 일반적인 상황에서 재사용 가능하도록 구현된다. 따라서 글로벌 예외처리기를 사용하는 관련연구에 비해 분산된 방식으로 유연하게 예외처리를 할 수 있기 때문에 유지보수에 상당한 장점이 있다. 그리고 본 연구에서는 요구 조건이 다양한 상황에서 서비스 지향 어플리케이션을 보다 빠르고 쉽게 개발하기 위한 해결책으로 프레임워크를 제공하기 때문에 서비스 개발자는 핵심 업무 기능을 구현하는데 집중할 수 있다. 앞에서 언급한 바와 같이 인증, 예외처리, 로깅기능과 같은 횡단관심 기능이 어플리케이션의 여러 모듈에 중복적으로 나타나 있기 때문에 일관성이 떨어지고 해당 기능의 수정을 어렵게 한다는 점에 착안하여, 횡단 관심 계층에서 담당해야할 오류처리를 재사용 가능하도록 독립적으로 분리하여 어플리케이션의 횡단 관심 계층을 보다 빠르고 견고하게 개발할 수 있도록 한다. 이는 결과적으로 어플리케이션에 부가적인 횡단 관심 기능보다도 핵심 서비스 개발에 집중하도록 함으로써 효율적으로 어플리케이션을 개발할 수 있음을 의미한다.

6. 결 론

본 논문에서는 SOA 기반 서비스 어플리케이션의 신뢰성 향상을 위한 AOP 개념을 활용한 아키텍처를 제안하였고, 이를 위해 분산 예외처리 모델을 소개하였다.

SOA 기반 동적인 관점 지향 프로그래밍 지원 기술은 프로그램의 구성요소인 서비스를 선언적으로 기술하는데 사용된다. 이와 같은 방법은 절차적으로 프로그램의 종합을 기술하는 기존 방법에 비해 프로그램을 구성하는 각 부분이 독립적이고 명시적으로 드러난다는 장점이 있다. 이는 소프트웨어의 실행 중에 다른 기능의 서비스가 필요하거나 또는 예외 상황이 발생했을 때 특정한 부분을 재구성하는 것을 표현하기 쉽게 만들어주기 때문에, 적응형 소프트웨어의 개발에 활용될 수 있다. SOA 기반 서비스는 신 성장 동력 산업인 로봇과 같은 지능형 소프트웨어에 활용될 수 있다.

그러나 제안된 프레임워크는 네트워크를 통해서 동적으로 관점 서비스를 위빙하므로 수행성능에 한계가 있어 이러한 제한점을 개선시키는 연구가 필요하며, SOA 기반의 엔터프라이즈 시스템 개발 시 분산된 환경에서 인증, 로깅 등의 횡단관심을 원자적(atomic)트랜잭션으로 보장하기 위한 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] Malloy, B. A., N. A. Kraft, J. O. Hallstrom, and J. M. Voas, "Improving the Predictable Assembly of Service-Oriented Architectures," *IEEE Software*, Vol.23, No.2(2006), pp.12-15.

[2] W3C, "Web Service Description Language

(WSDL) v1.2", Whitepaper, W3C Technical Publications, http://www.w3.org/TR/wsdl12/.

[3] Clement, L., A. Hately, C. Riegen, and T. Rogers, *Universal Description, Discovery and Integration*, OASIS UDDI spec. 2004.

[4] Asadollahi, R., M. Salehie, and L. Tahvildari, "StarMX : A Framework for Developing Self-Managing Java-based Systems", *Proc. of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, (2009), pp.58-67.

[5] Miller, R. and A. Tripathi, "The Guardian Model and Primitives for Exception Handling in Distributed Systems", *IEEE Trans. on Software Engineering*, (2004), pp.1008-1022.

[6] Platon, E., S. Honiden, and N. Sabouret, "Challenges in Exception Handling in Multi-Agent Systems", *Proc. of International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, 2006.

[7] Mallya, A. U. and M. P. Singh, "Modeling exceptions via commitment protocols", *Autonomous Agents and Multi-Agent Systems*, ACM Press, (2005), pp.122-129.

[8] Vasseur, A., "Dynamic AOP and Runtime Weaving for Java-How does AspectWerkz Address It?", *Proc. of International Conference on Aspect-Oriented Software Development*, 2004.

[9] Salehie, M., S. Li, and Tahvildari, L., "Employing Aspect Composition in Adaptive Software Systems : A Case Study", *Proc. of ACM Practices of Linking Aspect Technology and Evolution Workshop*, (2009), pp.17-21.

[10] Mukhija, A., A. Dingwall-Smith, D. S. Rosenblum, "QoS-Aware Service Composition in Dino", *Proc. of IEEE European Conference on Web Services*, 2007.

[11] Foster, H., A. Mukhija, D. S. Rosenblum, and S. Uchitel, "A Model-Driven Approach to Dynamic and Adaptive Service Brokering Using Modes", *Lecture Notes in Computer Science*, Vol.5364(2008), pp.558-564.

[12] Kim, J. and B. Lee, "A SOA-based Dynamic Service Composition Framework using Web Services and OpenAPIs", *Journal of KIISE : Software and Applications*, Vol.36, No.3(2009), pp.187-199.

[13] Irmert, F., M. Meyerhofer, and M. Weiten, "Towards Runtime Adaptation in a SOA Environment", *Proc. of ECOOP Workshop on Reflection, AOP, and Meta-Data for Software Evolution*, (2007), pp.17-26.

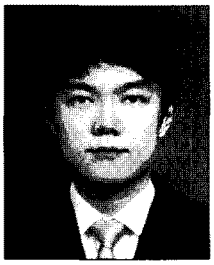
[14] Verheecke, B., M. A. Cibran, and V. Jonckers, "Aspect-Oriented Programming for Dynamic Web Service Monitoring and Selection", *Proc. of the European Conference on Web Services*, Erfurt, Germany, 2004.

[15] Vanderperren, W., D. Suvee, B. Verheecke, Cibran, M. A., and Jonckers, V., "Adaptive Programming in JAsCo", *Proc. of the 4th International Conference on Aspect-Oriented Software Development*, Chicago, Illinois, (2005), pp.75-86.

◈ 저 자 소 개 ◈

김 은 선 (eskim1208@gmail.com)

경원대학교 전자거래학과를 졸업하고, 현재 서울시립대학교 컴퓨터과학과 석사 과정에 재학 중이다. 주요 관심분야는 소프트웨어 방법론, SOA, AOP, 소프트웨어 진화 등이다.



이 재 정 (jaejeong2@gmail.com)

동국대학교 정보통신공학과를 졸업하고, 서울시립대학교 컴퓨터통계학과에서 석사학위를 취득하였다. 현재 더-케이손해보험㈜에 재직 중이다. 주요 관심분야는 소프트웨어 진화, 시맨틱 웹 서비스 등이다.

이 병 정 (bjlee@uos.ac.kr)

서울대학교 계산통계학과를 졸업하고, 서울대학교 전산과학과에서 석사 학위를, 서울대학교 전기컴퓨터공학부에서 박사학위를 취득하였다. 현대전자㈜ 소프트웨어 연구소에서 근무하였으며, 현재 서울시립대학교 공과대학 컴퓨터과학부 부교수로 재직 중이다. 주요 관심분야는 소프트웨어공학, 웹 공학 등이다.