韓國컴퓨터情報學會 論文誌
第14卷 第10號, 2009. 10.

2009-14-10-4-3

정책 기반의 합성된 웹 서비스 품질 모니터링 시스템의 설계

염 귀 덕*, 정 충 교**

A Design of Policy-Based Composite Web Services QoS Monitoring System

Gwyduk Yeom*, Choong Kyo Jeong**

요 약

웹 서비스 기술이 안정화됨에 따라 기업내 혹은 기업간 웹 서비스들을 통합한 합성된 웹 서비스의 연구가 활발히 진행되고 있다. 또한, 비기능적 속성은 다르지만 유사한 기능을 제공하는 웹 서비스들의 증가로 인하여 품질관리는 중요한 이슈로 여겨지고 있다. 품질관리를 위해서는 모니터링 기능이 필요하다. 본 논문에서는 중개자를 통한 합성된 웹 서비스의 품질 모니터링 시스템의 설계를 제안한다. 이 시스템은 OWL-S를 사용하여 합성된 서비스의 프로세스를 명세하며 WS-Policy를 사용하여 다양한 요구사항인 정책(각 서비스 입력·출력값, 품질요소 및 품질요소 값 등)을 명세하여 합성된 서비스 프로세스 모델에 첨부한다. 또한, 센서를 이용하여 명세한 정책과 실제로 모니터링된 데이터의 값을 비교하여 불일치하면 위반사항이 발생하였음을 웹 서비스 제공자 및 사용자에게 통보함으로서 품질을 관리한다. 제안한 시스템의 검증을 위하여 여행 예약 시스템 예제를 가지고 평균 응답시간과 타임아웃 정책을 적용하여 모니터링한 결과를 보여 준다.

Abstract

As the web service technology matures, research is focused on the composite web services that combine individual web services within an enterprise or between enterprises. Quality of service is the critical competitiveness factor in this mature technology stage where there are many services with similar functionalities differing only in some non-functional properties. Monitoring is the key component for the service quality management of a web service. A service quality monitoring system design using a broker is presented in this paper. OWL-S is used to specify the composite service process and a service policy(inputs and outputs of each service, quality attributes and values, etc.) built by WS-Policy is applied to the composite service process. If there is any discrepancy between the service policy and the monitored data, the service provider and the user

• 제1저자 : 염귀덕
• 투고일 : 2009. 08. 11, 심사일 : 2009. 09. 01, 게재확정일 : 2009. 09. 30.
* 애리조나주립대학교 컴퓨터공학과 박사 후 연구원 ** 강원대학교 컴퓨터학부 교수
※ 이 논문은 2008년 정부의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2008-357-D00228).

are notified of it so as to take necessary measures. We have implemented a travel reservation system as an example of the presented design and the experimental results are shown. Average response time was monitored and the timeout policy was applied in the experiment.

▸ Keyword : 합성된 웹 서비스(Composite Web Services), 품질 모니터링(QoS Monitoring), 정책 기반(Policy-Based)

I. 서 론

서비스 기반 아키텍처(Service-Oriented Architecture)는 서비스 기반의 어플리케이션 단위로 약결합(Loosely-Coupled)으로 연결하여 하나의 완성된 어플리케이션으로 개발하기 위한 새로운 소프트웨어 아키텍처로 주목받고 있다[1]. 최근 XML 기반의 웹 서비스 기술이 등장하면서 서비스 기반 아키텍처는 중요한 패러다임으로 인식되고 있다. 웹 서비스는 서비스 기반 아키텍처의 개념을 구현하는 소프트웨어 인터페이스로서 이기종 시스템들간의 연계 및 통합을 위한 개방형 분산 컴퓨팅의 표준기술로 자리잡고 있다.

웹 서비스 기술이 등장한 초기에는 주로 SOAP(Simple Object Access Protocol)[2]과 WSDL(Web Services Description Language)[3] 프로토콜과 같은 웹 서비스의 인터페이스 측면과 웹 서비스가 제공하는 기능의 연구[4]에 중점을 두었지만 웹 서비스 기술이 안정화됨에 따라 기업내 혹은 기업간의 웹 서비스들을 통합한 합성된 웹 서비스(Composite Web Services) 및 합성된 웹 서비스의 프로세스 모델, 그리고 웹 서비스들 간의 상호작용인 웹 서비스 협업(Collaboration) 연구가 활발히 진행되고 있다.

최근 비기능적 속성(Non-Functional Properties)은 다르지만 유사한 기능(Functionality)을 제공하는 웹 서비스들의 증가로 인하여 품질의 중요성이 커지고 있다. 현재 UDDI 서버는 품질 정보를 제공하지 않으므로 웹 서비스 사용자들에게 고품질의 웹 서비스를 제공하기 위해서는 신뢰할 수 있는 중개자(Broker)[5]를 통한 품질관리가 무엇보다도 중요하다. 품질관리를 위해서는 모니터링 기능이 필요하다.

합성된 웹 서비스의 모니터링을 위해서는 다양한 기능들이 필요하다. 첫째, 합성된 웹 서비스의 각 서비스가 제공하는 능력(서비스 이름, 동작, 호출정보 등)을 명세한 프로세스 모델이 필요하다. 둘째, 정책명세가 필요하다. 정책은 웹 서비스 제공자가 제공하는 웹 서비스의 정책들(보안, 각 서비스 매개변수의 입·출력값 및 성능관련 품질요소 및 품질요소 값 등)과 웹 서비스 사용자의 요구사항인 정책으로 나눌 수 있다. 명세된 정책은 프로세스 모델에 첨부형태로 추가된다. 셋째, 모니터링된 데이터 수집이 필요하다. 명세한 정책과 실제 웹 서비스의 데이터 값을 비교하기 위해서는 모니터링 데이터를 수집해야 한다. 마지막으로, 위반사항(Violation) 통보 기능이 필요하다. 만약, 웹 서비스 제공자의 정책인 품질요소의 품질값이 실제로 모니터링된 웹 서비스 품질값과 다른 경우 위반사항이 발생했음을 웹 서비스 제공자와 사용자 양측에 통보하는 기능이 필요하다. 본 논문에서는 중개자를 통한 합성된 웹 서비스 품질 모니터링 시스템의 설계를 제안한다. 이 시스템은 OWL-S(Web Ontology Language for WebService)[6]를 사용하여 합성된 웹 서비스(Composite Web Services)의 프로세스를 명세하며WS-Policy를 사용하여 정책(각 서비스 매개변수의 입력값 및 출력갑, 성능관련 품질 요소 및 품질요소값 등)을 명세하여 OWL-S 프로세스 모델(Process Model)에 첨부한다. 또한, 센서를 이용하여 모니터링된 데이터를 수집한다. 센서는 합성된 서비스의 각 서비스 단위의 오퍼레이션별로 생성이 되며 프로세스 모델에 기기화(Instrumentation)된다. 센서는 명세된 정책과 실제 모니터링된 데이터 값을 비교하여 불일치하면 위반사항(Violation)이 발생하였음을 웹 서비스 제공자와 사용자 양측에 통보함으로서 품질관리를 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안한 시스템을 검증하기 위해 사용된 여행 예약 서비스 예제를 소개한다. 3장에서는 관련연구를 소개한다. 4장에서는 제안한 웹 서비스 품질 모니터링 시스템의 구조와 합성된 웹 서비스 프로세스 실행을 위하여 센서 생성과 기기화 그리고 정책 명세및 실행 과정을 자세히 설명한다. 5장에서는 여행 예약 서비스 예제를 이용하여 평균 응답시간과 타임아웃 정책을 적용한 실험결과를 보여준다. 그리고 마지막 6장에서는 결론으로 마무리한다.

II. 관련 연구

웹 서비스의 초기에는 웹 서비스의 개발과 배포에 관한 연구가 많았었다. 최근에는 웹 서비스 사용자들의 다양한 요구사항을 만족시키기 위하여 웹 서비스 품질의 모니터링 측면에

연구의 초점을 두고 있다. 다양한 요구사항들을 가진 합성된 웹 서비스 프로세스의 명세 및 모니터링에 대한 다양한 연구가 진행되고 있다[7][8][9][10][11][12][13][14][15][16][17]. 몇 연구에서는 모니터링 요구사항들을 명세하기 위하여 WS-BPEL을 사용한 연구가 있다[7][13][14]. 표 1은 각 모니터링 연구 방법들의 특징들을 보여준다.

표 1. 모니터링 방법들의 비교
Table 1. The Comparison of Monitoring Methods

분류	특징
Mahbub, Spanoudakis [13]	• 모니터링 속성들을 명세하기 위하여 이벤트 사용
Baresi, Guinea [7]	• WS-Policy[8][11] 요구사항들을 검증하기 위하여 모니터링 프레임워크 제안 • 비기능적 제약사항들을 표현하기 위하여 WS-CoL(Web Service Constraint Language)를 제안 • 명세된 비기능적 제약사항들은 BPEL 명세에 추가하고 대리인(Proxy)을 사용하여 실행
Cruz [9]	• 웹 서비스 기록(Logging) 구조를 제안 • Soap 중개자를 사용하여 웹 서비스 제공자와 사용자간 주고받는 메시지 획득
Vaculin, Sycara [15]	• OWL-S로 명세된 웹 서비스의 이벤트 기반의 모니터링을 위하여 이벤트 온톨로지 모델을 제안 • 이벤트를 프로세스계산이벤트(Process CalEvent),결과평가이벤트(Result EvalEvent), 그리고 선행조건평가이벤트(Precondition EvalEvent)로 분류하고 도구(Tool) 구축 • 도구의 예로 WS-I 테스팅 도구[18]
Li [11]	• 품질 정보를 수집하고 적응적으로 서비스를 공급하기 위하여 정책 기반의 모니터링 프레임워크를 제시

위의 연구들은 정책에 대한 구체적인 분류가 없으며 위반사항 발생시 액션에 대한 고려가 없다.

본 논문에서는 OWL-S로 합성된 웹 서비스의 프로세스를 명세하였다. 또한, 정책을 데이터 정책(Data Policies), 기능적 정책(Functional Policies), 그리고 성능 정책(Performance Policies)으로 분류하여 WS-Policy 표준문서를 사용하여 정책을 명세하고 정책을 프로세스 모델에 첨부할 때는 WS-Policy Attachment[18]를 따랐다. 또한 센서를 사용하여 모니터링 데이터를 수집하여 명세된 정책과 품질값이 불일치한 경우 위반사항이 발생하였음을 웹 서비스 제공자와 사용자 양측에 통보함으로서 품질을 관리한다.

III. 여행 예약 서비스 예제

본 논문에서는 "여행 예약 서비스" 예제를 이용하여 본 논문에서 제안한 시스템의 검증을 보여준다. 합성된 서비스(Composite Service)는 다음과 같은 서비스들로 구성된다.

1. *Flight* Service: 비행기 검색 및 예약 서비스
2. *Hotel* Service: 호텔 검색 및 예약 서비스
3. Car Service: 차 검색 및 예약 서비스
4. *Travel* Service: 여행 계획을 위한 서비스

합성된 서비스는 네개의 오퍼레이션을 가지고 생성된다. 이 합성된 서비스는 본 장에서는 "여행 예약 서비스 프로세스" 예제를 이용하여 본 논문에서 제안한 시스템의 검증을 보여준다. 합성된 서비스는 Flightservice에 연결된 s1:reserveFlight ;

Hotelservice에 연결된 s2:reserveHotel ; Travel Service에 의해 제공되는 s4:confirmJourney ; 로 구성된다.

위의 네개의 서비스들을 합성하기 위하여 세개의 구조체가 정의된다. *c1* for Split the Process, *c2* for Split-Join the Process, 그리고, Sequence Execution. 여섯개의 세그먼트 p1, p2, p3, p4, p5, 그리고 p6가 실행된다.

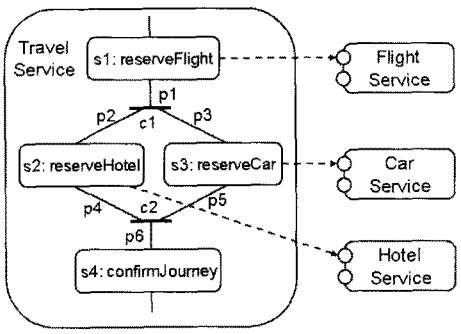

그림 1. 여행 예약 서비스의 예제
Fig. 1. The Example of Travel Reservation Service

VI. 제안한 시스템의 구조

본 장에서는 본 논문에서 제안한 웹 서비스 품질 모니터링 시스템(WS QoS Monitoring System)의 구조 및 모니터 과정을 소개한다. 또한, OWL-S를 이용한 합성된 웹 서비스의 프로세스 모델 명세 및 방법을 설명한다. 또한, 세가지 형태의 프로세스 센서들과 정책들을 소개하고 센서와 정책의 상관관계 매트리스를 설명한다. 마지막으로, 명세된 정책이 실행되는 과정을 설명한다.

그림 2는 웹 서비스 품질 모니터링 과정을 보여준다. 서비스 제공자 측의 시스템 관리자는 중개자(Broker)가 제공하는 에이전트를 다운로드해서 웹 서비스 컨테이너(웹로직)에 설치한다. 에이전트는 모니터링 대상인 관리 웹 서비스들을 중

개자측에 등록시킨다. 또한, 에이전트는 웹 서비스 사용자와 제공자간에 주고받는 Soap 메시지를 가로채서 중개자에게로 보낸다. 중개자의 로그 분석기(Log Analyzer)에 쌓인 Soap 메시지들은 모니터링 분석용으로 사용된다.

웹 서비스 품질 모니터링 시스템은 OWL-S API와 WS-Policy API를 사용하여 이클립스 플래폼[19]에서 개발되었다. 웹 서비스 품질 모니터링 시스템은 세 개의 컴포넌트인 정책 관리자(Policy Manager), 센서 관리자(Sensor Manager), 그리고 로그 분석기(Log Analyzer)와 OWL-S 프로세스 모델을 실행시키는 OWL-S Engine을 사용한다. OWL-S Engine은 기기화된(Instrumeted) 센서들과 정책을 실행(Policy Enforcement)시키는 컴포넌트로 구성된다.

합성된 웹 서비스의 프로세스를 명세한 OWL-S 프로세스 모델의 각 서비스 단위의 오퍼레이션별로 센서가 생성된다. 생성된 센서들은 OWL-S의 프로세스 모델로 기기화된다. 하나의 센서는 여러 개의 정책들을 가질 수 있으며 하나의 정책은 여러 개의 다른 센서들에 사용될 수 있다. 센서는 모니터링된 데이터 값과 정책으로 명세한 데이터 값을 비교하여 결과를 로그로 남기고 보고서를 생성하여 웹 서비스 제공자 및 사용자 양측에 보여준다. 만약, 값이 모니터링된 데이터 값이 정책검증 과정 중에 불일치하면 위반사항(Violation)이 발생하였으므로 경보(Alarm)를 생성하여 웹 서비스 제공자 및 사용자 양측에 통보한다.

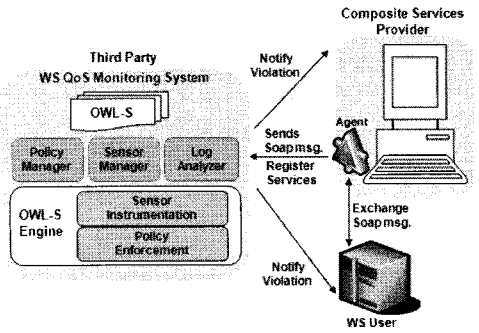

그림 2. 웹 서비스 품질 모니터링의 과정
Fig. 2. WS QoS Monitoring Process

1. 에이전트 개요

에이전트는 웹 서비스를 제공하는 서비스 컨테이너(웹로직)와 같은 소프트웨어에 탑재되어 웹 서비스에 대한 모니터링을 수행하는 어플리케이션을 말한다. 에이전트는 배치되는 형태에 따라 서비스 컨테이너의 한 모듈로 설치되어 서비스 제공자와 사용자가 알지 못하게 내부적으로 작동하는 플러그인 에이전트(Plugin Agent)와 서비스 컨테이너의 내부 또는 외부에 설치되어 서비스 제공자와 사용자 사이에서 메시지를 전달해주는 대리자 에이전트(Proxy Agent)로 구분된다. 플러그인 에이전트는 서비스 컨테이너의 내부에 설치되며 기존 서비스에 변화를 주지않고 서비스 컨테이너에서 제공하는 API를 사용하여 웹 서비스 품질 모니터링 및 관리기능을 수행한다. 이러한 방식은 에이전트의 추가 및 삭제가 전체 시스템에 영향을 주지 않으며 서비스 사용자의 입장에서도 변경이 요구되지 않기 때문에 이상적인 형태라고 할 수 있으나 서비스 컨테이너가 웹 서비스를 구성하고 제공하는 방식이 표준화된 규격을 사용해야 하며 다양한 서비스 컨테이너의 구조 특성에 맞게 에이전트를 각각 개발해야 하며 어플리케이션의 크기가 약간 커지는 단점이 있다.

본 논문에서는 웹 서비스 제공자와 사용자간 주고받는 Soap메시지를 가로채기 위하여 대리자 에이전트를 사용한다. 모든 메시지 교환이 대리자 에이전트 어플리케이션을 거치므로 발생하는 성능저하와 서비스 사용자가 메시지 요청의 목적지를 기존 서비스 제공자가 아닌 대리자 에이전트의 위치로 변경해야 한다는 즉, 웹 서비스의 Endpoint URL이 바뀐다는 단점이 있다.

표 2는 본 논문에서 개발한 에이전트의 API를 보여주는데 이는 OASIS의 WSDM(Web Services Distributed Management)[20] V1.1을 기반으로 개발되었다.

표 2. 에이전트 어플리케이션 프로그래밍 인터페이스
Table 2. Application Programming Interface of the Agent

XML 요소 이름	기능
NumberOfRequests	웹 서비스 엔드포인트가 호출받은 요청 메시지의 수를 정수값으로 제공한다.
NumberOfFailedRequests	웹 서비스 엔드포인트가 호출받은 후(요청메시지를 받은 후) 응답 메시지가 실패한(fault) 경우의 수를 정수값으로 제공한다.
NumberOfSucessfulRequests	웹 서비스 엔드포인트가 호출받은 후(요청메시지를 받은 후) 응답 메시지가 성공적인 경우의 수를 정수값으로 제공한다.
ServiceTime	웹 서비스 엔드포인트가 요청메시지를 받고 처리할 때까지 총 경과된 시간을 초 단위로 제공한다.
MaxResponseTime	모든 요청메시지를 받고 응답메시지를 받을 때까지 경과된 시간 중 최대 응답시간을 반환한다.
LastResponseTime	요청 메시지를 받고 기록된 응답시간들 중 가장 최근에 응답한 시간을 반환한다.

WSDM은 인터넷 상의 자원을 관리하기 위한 프레임워크인 Managing Using Web Services(MUWS)와 관리대상인 자원을 웹 서비스로 한정시킨 Management of Web Services(MOWS)로 나누어진다. 본 에이전트는 WSDM의

NOWS를 기반으로 개발되었다.

2. OWL-S Process Model

본 연구에서는 합성된 서비스 프로세스 모델을 명세하기 위하여 OWL-S(Web Ontology Language for WebService)를 사용한다. OWL-S는 서비스를 기술하기 위하여 세가지 구성요소인 ServiceProfile, Service Grounding, 그리고 Service Model로 구성된다. ServiceModel은 OWL 온톨로지를 이용하여 제공하는 서비스를 기술하고 서비스 호출정보 및 서비스 합성과 같은 서비스의 능력을 기술한다.

OWL-S 프로세스 모델을 정의하기 위하여 두 개의 컴포넌트들인 프로세스 온톨로지(Process Ontology)와 프로세스 제어 온톨로지(Process Control Ontology)가 사용되었다. 프로세스 온톨로지는 입력매개변수(Inputs), 출력매개변수(Outputs), 선행조건들(Preconditions), 그리고 결과값(Effects)(IOPE)과 같은 서비스의 속성들을 기술한다. 프로세스 제어 온톨로지는 초기 활성화, 실행, 그리고 완료와 같은 프로세스 상태를 기술한다. 각 합성된 서비스 프로세스는 순서대로 실행(Sequence), Split, Split-Join, Any-Order, 반복(Iterate), 만약 ~이라면 그렇지 않다면(If-Then-Else), 그리고 선택(Choice)과 같은 제어 구조체 중 하나를 가지게 된다.

3 센서 생성과 기기화

OWL-S 프로세스 모델에 명세된 합성된 서비스의 각 서비스 단위의 오퍼레이션별로 센서가 생성된다. 생성된 센서는 프로세스 모델로 기기화(Instrumented)된다.

기본적으로 세가지 형태의 프로세스 센서들이 정의된다.

- Struct_PS : 프로세스 모델의 구조체에 따라 생성된다. 예를 들면, 센서들은 If-Then-Else 제어 구조체의 각 분기점에서 생성된다.
- Data_PS : 서비스의 입력과 출력의 데이터 값을 검증하기 위하여 모니터링 대상인 데이터의 정의에 따라서 센서들이 생성된다. Data_PS는 실행시간에 데이터 값을 획득하기 위하여 모니터링이 될 서비스의 입력 및 출력 데이터를 연결(Bind)한다.
- Cond_PS : OWL-S의 프로세스 모델에서 정의된 선행조건과 결과(Precondition and Effect)에 따라서 센서들이 생성된다. 이 센서들은 프로세스의 상태를 검사하고 모니터링한다. Cond_PS는 실행결과를 기록하기 위하여 조건명세 이전과 이후에 추가된다.

4. 정책 명세 및 실행

본 연구에서는 정책을 데이터 정책(Data Policies), 기능적 정책(Functional Policies), 그리고 성능 정책(Performance Policies)으로 분류한다. 데이터 정책은 서비스의 입력값과 출력값을 검증(확인)하기 위한 정책이다. 기능적 정책은 예상되는 조건이나 상태에 대하여 기능적으로 정확한가를 검증하기 위한 정책이며 성능 정책은 예상되는 성능관련 품질요소 값을 확인하기 위한 정책이다. 본 연구에서는 정책을 명세하기 위하여 WS-Policy를 사용한다. WS-Policy는 W3C의 표준 프레임워크로 웹 서비스의 제공자가 제공하는 서비스의 정책들(보안, 성능관련 품질요소 및 품질요소값 등)과 웹 서비스 사용자들의 정책 요구사항들을 명세할 수 있는 표준문서이다. 명세된 정책은 OWL-S 프로세스 모델에 첨부되며 첨부하는 방법은 WS-Policy Attachment에서 정의된 첨부 방법을 따른다.

명세된 정책은 센서를 통해 실행이 되는데 본 논문에서는 정책과 센서의 상관관계 매트리스(Correlation Matrix)를 가지는 모니터링 정책을 사용한다. 모니터링 정책은 3-Tuple인 $\langle Type, Subject, Assertions\rangle$으로 정의된다.

- *Type* : 정책을 분류한다. 예를 들면, 이 정책이 데이터 정책인지(예를 들면, Flight No=3124) 응답속도와 같은 성능관련 정책인지를 표현한다.
- *Subject* : 정책이 적용될 서비스의 요소(Element)를 표현한다. (예를 들면, FlightService, CarService 등)
- *Assertions*$=\{ass_i\}$: 룰의 집합(Rule Sets)을 표현한다. 룰 집합은 "ExactlyOne"과 같은 룰 연산자들을 가진다. 각 룰은 조건(Conditions)-그리고(and)-액션(Actions)의 문장을 가진다. 조건이 "참"일 때 액션이 실행된다는 의미이다.

여행 예약 서비스 예제에서 그림 3은 성능 정책을 서비스의 응답시간 타임아웃을 300 밀리초(Milliseconds)로 정의하는 예제이다.

```
<wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xMlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="PerformancePolicy">
  <wsp:ExactlyOne>
    <wsp:All>
        <ResponseTime TimeOut = "300"/>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

그림 3. 정책명세의 예제
Fig. 3. Example of Policy Specification

하나의 센서(Sensor)는 여러 개의 정책들을 가질 수 있으며 하나의 정책은 여러 개의 다른 센서들에 사용될 수 있다. 예를 들어, 여행 예약 서비스 예제에서 *reserve*Flight 서비스에 내장된 센서는 최대 평균응답시간과 최대 평균처리량과 같은 여러 개의 성능정책을 가질 수 있다. 또 다른 예로, *ResponseTimeOut* 정책은 합성된 서비스에서 여러 개의 각 서비스 요소에 정의될 수 있다. 매트리스(Matrix)는 센서들과 정책들간의 m-n 상관관계를 기록하기 위하여 정의된다.

Definiton 1: $C : S \times P$ 는 상관관계 매트리스이다. 여기서 S는 센서들의 집합, P는 정책들의 집합, 그리고 c_{ij} 는 다음과 같이 정의된다 :

$$\begin{cases} c_{ij} = 1, & if\ p_j \in P\ is\ assigned\ to\ s_i \in S \\ c_{ij} = 0, & otherwise \end{cases}$$

명세된 정책들 간에 Complete and Consistent(C&C)는 중요한다. 카노우 맵(K-Map: Karnaugh Map)을 사용하여 정책 룰에서 조건들의 조합을 검사함으로서 정책들의 적응성 분석을 할 수 있다. 또한 새로운 정책들을 추가하거나 삭제가 용이하므로 정책 정확성 유지를 위해서는 C&C 분석 실행은 필요하다. 예를 들면, 그림 4는 K-Map을 이용하여 정책들에서 조건들의 조합들을 보여준다.

---

if (Not A and $\overline{B}$ and (C or D)) Assert {Constraint 1};

if (A $\overline{B}$ and (C or D)) Assert {Constraint 2};

if (Not ABCD) Assert {Constraint 3};

if (A $\overline{B}$ CD)Assert{Constraint4};

if (Not AB Not CD) Assert {Constraint 5};

그림 4. 정책 룰의 조합
Fig. 4. Combination of Policy Rules

정책들은 정책 엔진(Policy Engine)에서 실행된다. 그림 5는 프로세스 엔진, 센서들, 그리고 정책 엔진들 간의 협업관계를 통하여 정책이 실행되는 과정을 보여준다.

① 프로세스 엔진(Process Engine)은 여러 개의 서비스가 합쳐진 합성된 서비스의 프로세스 모델을 각 서비스 단위로 분해하기 위하여 프로세스 모델을 해석(Interpreter)한다.

② 일단, 하나의 서비스가 식별되고 선행조건(Pre-Conditions)이 만족되면 프로세스 엔진은 서비스를 연결(Bind)하고 호출한다. 프로세스 모델의 끝까지 이 과정은 반복된다.

③ 다양한 센서들이 프로세스 엔진에 생성되고 기기화(Instrumented)된다. 이러한 센서들은 이벤트 리스너(Event Listener)로서의 역할을 한다. 센서들은 프로세스 엔진이 실행되는 동안 모니터링된 이벤트들과 데이터를 획득(Capture)한다.

④ 센서 형태에 기반하여 메시지(Messages), 기능(Functions), 그리고 성능 매개변수들(Performance Parameters)과 같은 다양한 종류의 데이터들이 수집되고 필터링되며 계산된다.

⑤ 수집된 데이터는 미리 정의된 정책들과 비교하여 검증하는 과정을 거친다.

⑥ 정책 엔진(Policy Engine)은 센서와 관계되고 모니터링된 데이터에 적용된 정책들을 확인한다.

⑦ 정책 엔진은 정책들을 정의한 정책명세서를 해석(읽어 들인다)한다.

⑧ 정책 엔진은 정책과 모니터링이 된 데이터와 비교를 한 후 결과값을 센서로 반환한다.

⑨ 센서는 모니터링된 데이터와 정책 검증 결과를 로그로 남기고 보고서를 생성하여 웹 서비스 제공자 및 사용자 양측에 보여준다. 만약, 값이 모니터링된 데이터 값이 정책 검증 과정 중에 불일치하면 위반사항(Violation)이 발생하였으므로 경보(Alarm)를 생성하여 웹 서비스 제공자 및 사용자 양측에 통보한다.

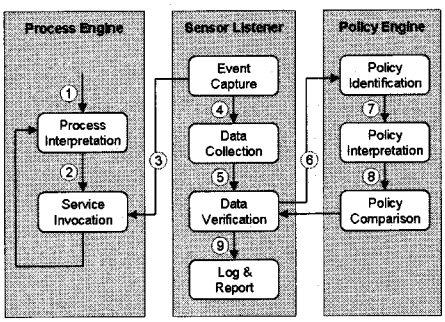

그림 5. 정책 실행 과정
Fig. 5. Policy Enforcement Process

V. 시스템 구현 및 실험

웹 서비스 품질 모니터링 시스템은 OWL-S API와 WS-Policy

API 이클립스 플랫폼에서 개발되었다. 그림 6은 센서 생성, 기기화, 정책 정의, 그리고 모니터링 결과 기록을 위한 툴의 인터페이스를 보여준다. 여행 예약 서비스 예제를 가지고 각 웹 서비스의 평균 응답시간을 산출하기 위해 필요한 데이터가 수집되었다고 가정한다. 웹 서비스 호출 전과 호출 후에 센서들이 기기화되고 시간 차이를 계산한다. Table 1은 인터페이스 센서 (*IS_P*)와 프로세스 센서 (*PS_P*)에 의한 모니터링 결과값들의 목록을 보여준다. 그림 6에서 세 개의 서비스들의 평균 응답시간은 큰 차이가 없으며 이 세 개의 서비스들 모두 타임아웃 정책 요구사항들을 만족시키고 있다.

그러나, HotelService의 네트워크 지연은 (182.79ms)로 높은데 이것은 합성 프로세스 (327.63ms)와 정책 요구사항(300ms)의 위반사항이 원인이다.

표 3. 여러가지 서비스의 평균응답시간과 타임아웃 정책
Table 3. Average Response Times of Various Services and the Timeout Policy

서비스명	오퍼레이션	평균 응답시간		타임아웃 정책	
		PS_P	IS_P	PS_P	IS_P
Flight Service	reserve _Flight	191. 62ms	134. 86ms	300 ms	150 ms
Hotel Service	reserve _Hotel	327. 63ms	144. 84ms	300ms	150 ms
Car Service	reserve Car	240. 50ms	107. 35ms	300 ms	150 ms
Travel Service	confirm _Journey	99. 60ms	102. 33ms	150 ms	150 ms
Travel Service	confirm _Journey	821. 08ms	111. 88ms	900 ms	150 ms

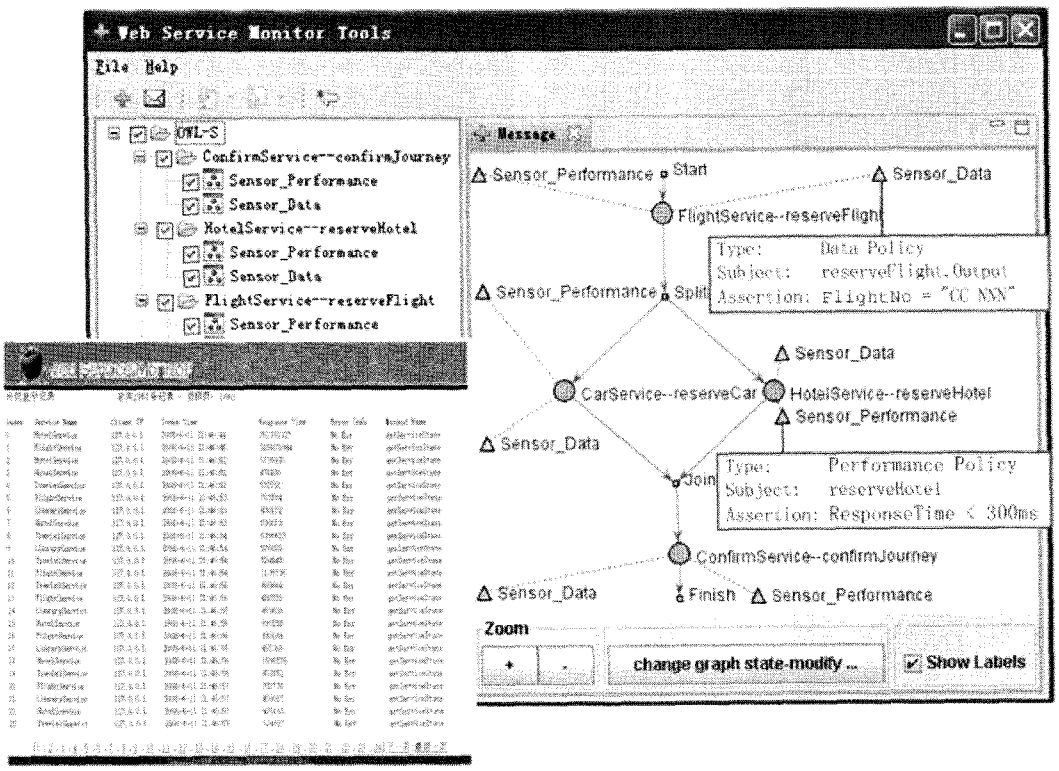

그림 6. 웹 서비스 모니터링을 위한 도구 인터페이스
Fig. 6. Tool Interface for Web Service Monitoring

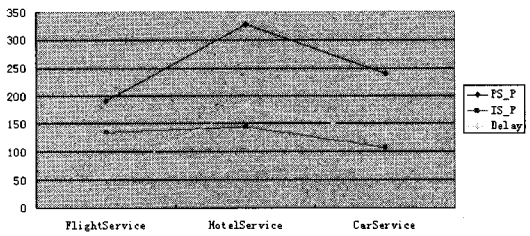

그림 7. 여러가지 서비스의 지연시간
Fig. 7. Delays of Various Services

Ⅵ. 결론

본 논문에서는 정책기반의 합성된 웹 서비스의 품질 모니터링 시스템의 설계를 제안하였다. OWL-S를 이용하여 합성된 웹 서비스 프로세스를 명세하였다. OWL-S는 모니터링 명세를 지원하지 않으므로 WS-Policy를 이용하여 모니터링 정책을 명세하였으며 명세된 정책은 WS-Policy Attachment에 따라 OWL-S프로세스에 첨부하였다. 센서를 이용하여 수집된 모니터링된 데이터와 명세된 정책을 비교하여 품질값이 불일치하는 경우 웹 서비스 제공자와 사용자에게 위반사항이 발생하였음을 통보함으로서 품질을 관리하였다.

기존의 연구들은 정책에 대한 구체적인 분류가 없으며 위반사항 발생시 액션에 대한 고려가 없었다. 그러나 본 논문에서는 정책을 데이터 정책, 기능적 정책, 그리고 성능 정책으로 분류함으로서 효과적으로 품질을 관리할 수 있었으며 센서를 생성하고 실행하는 메카니즘을 통하여 모니터링 시스템의 지능화와 자동화를 향상시킬 수 있었다. 그러나, 부적합한 센서의 기기화는 서비스의 성능에 영향을 미칠 수 있다.

본 논문에서 제안한 모니터링 시스템은 다음과 같은 기능들이 추가적으로 고려되어야 한다. 메시지의 흐름을 자동으로 추적하여 서비스 장애 발생시 해당 서비스 또는 해당 메시지를 실시간으로 모니터링 할 수 있는 기능이 필요하다. 또한, 시스템 에러 또는 메시지 내 오류 데이터에 대한 에러를 체크하여 에러 발생 시에는 에이전트에게 통보해주는 기능이 추가되어야 한다.

참고문헌

[1] W3C Working Draft, *http://www.w3.org/TR/ws-arch/* Nov. 2002.
[2] W3C Simple Object Access Protocol (SOAP), *http://www.w3c.org/TR/soap*, April, 2007.
[3] W3C Web Services Description Language(WSDL), *http://www.w3c.org/TR/wsdl*, March, 2001.
[4] M.Adel Serha, Rachida Dssouli, Abdelhakim Hafid and Houari Sahraoui "A QoS broker based architecture for efficient web services selection," IEEE, 2005.
[5] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, "A Concept for QoS Integration in Web Services," 4th International Conference on Web Information Systems Engineering, Dec. 2003.
[6] W3C, *http://www.w3.org/submission/OWL-S*, Nov. 2004.
[7] Baresi, L and Guinea, S., "Towards Dynamic Monitoring of WS-BPEL Processes," in Proceedings of ICSOC, pp.269-282, 2005.
[8] Bhargavan, K., Fournet, C., and Gordon, A.D.., "Verifying Policy-Based Web Services Security," ACM Transactions on Programming Languages and Systems, Vol. 30, No. 6, pp. 30-59, 2008.
[9] Cruz, S.M.S.d., Campos, M. L. M., Pires, P. F., and Campos, L. M., "Monitoring E-Business Web Services Usage through a Log Based Architecture," In Pro. of the IEEE International Conference on Web Services(ICWS), pp. 61-69, 2004,
[10] Lazovik, A., Aiello, M., and Papazoglou, M., "Associating Assertions with Business Processes and Monitoring their Execution," In Proc. of the 2nd international conference on Service oriented computing(ICSOC), pp.94-104, 2004,
[11] Li, F., Yang, F., Kai, S., Su S., "A Policy-driven Distributed Framework for Monitoring Quality of Web Services," In Proc. of the 2008 IEEE International Conference on Web Services, pp. 708-715, September 2008.
[12] Li Z., Jin. Y., Han. J., "A Runtime Monitoring and Validation Framework for Web Service Interactions," In Proc. of the 17th Australian Software Engineering Conference, pp. 70-79,

2006.

[13] Mahbub, K. and Spanoudakis, G. A, "Framework for Requirements Monitoring of Service Based Systems," In Proceeding of ICSOC, pp. 84-93, 2004.

[14] Pistore. M., Traverso P., "Test and Analysis of Web Services," Springer Berlin Heidelberg, pp. 307-335, 2007.

[15] William N. Robinson, "Monitoring Web Service Requirements," In Proc. of the 11th IEEE International Conference on Requirements Engineering, pp. 65-74, 2003.

[16] Roman Vaculin and Katia Sycara, "Semantic Web Services Monitoring: An OWL-S based Approach," In 41st. Hawaii International Conference on System Sciences(HICSS), Jan. 2008.

[17] Sahai, A., Machiraju,V., Sayal,M., Moorsel, A., and Casati, F., "Automated SLA Monitoring for Web Services," Lecture Notes in Computer Science, vol. 2506, pp. 28-41, 2002.

[18] Understanding the WS-I Test Tools, *http://www.ibm.com/developerworks/webservices/library/ws-wsitest/*, Nov. 2003.

[19] *Eclipse Platform*, http://www.eclipse.org

[20] *OASIS WSDM 1.1*, *http://www.oasis-open.org/*

저 자 소 개

염 귀 덕

2006년 : 건국대학교 컴퓨터공학공학 박사

06~07년 : 서울시립대학교 컴퓨터학부 BK연구교수

08~현재 : 애리조나주립대학교 컴퓨터공학 Post Doc.

관심분야 : 웹 서비스 품질관리 SOA, Robotics

정 충 교

1989년 : 한국과학기술원 전기전자공학 공학박사

89~94년 : LG정보통신(주) 책임연구원

95~현재 : 강원대학교 컴퓨터학부 교수

관심분야 : 웹 서비스 품질관리, 컴퓨터네트워크 보안