SOA 환경에서의 동적 시맨틱스의 응용

김웅섭
동국대학교 IT 학부 정보통신공학전공
e-mail : woongsup@dongguk.edu

On Using Dynamic Semantics for SOA Services

Woongsup Kim*
*Dept. of Information Communication Engineering
e-mail : woongsup@dongguk.edu

요 약

서비스 기반 아키텍처(SOA)는 최근에 많은 관심을 끌고 있다. SOA 는 독립된 컴포넌트 기반 소프트웨어 아키텍처로서 소프트웨어의 기능이 독립된 서비스로서 제공되는데 독립된 서비스를 다양하게 구성하여 복잡한 또는 새로운 기능의 서비스를 제공할 수 있다는 장점을 가진다. 하지만 QoS 또는 Safety 를 보장할 수 없다는 한계점을 가지고 있으며 우리는 동적 시맨틱스를 사용하여 이를 극복할 수 있는 방법을 제안하고자 한다. 동적 시맨틱스는 Finite State Machine 과 시맨틱스를 결합한 형태로 구성되며 서비스의 기능적 요소를 시맨틱스로 표현하고 서비스의 비기능적 요소를 천이가 가능한 상태들로 표현하자는 것이다. 이러한 동적 시맨틱스를 상용하여 서비스의 사용자 또는 복합 서비스의 설계자는 서비스의 성능을 예상할 수 있으며 이를 통하여 보다 신뢰도 높은 서비스를 제공할 수 있을 것으로 예상한다.

1. 서론

서비스 기반 아키텍처(SOA)는 최근에 많은 관심을 끌어왔다. SOA 는 loosely coupled 된 독립된 소프트웨어 컴퍼넌트 기반의 아키텍처로서 개방된 네트워크 상에서 인터넷 또는 인트라넷을 통하여 제공될 수 있는 독립된 서비스들의 집합으로 구성되어 있다 [1,2,3]. SOA 에서의 각 서비스는 독립된 형태로 stateless 하게 운영되기 때문에 다양한 기능의 서비스들을 구성함에 의하여 복잡한 기능의 서비스를 제공하기 쉽다는 특징을 가지고 있으며 따라서 다양한 시장의 요구사항의 맞춤형 서비스를 쉽고 빠르게 제공할 수 있다는 장점을 가지고 있다.

하지만 서비스 기반 아키텍처는 그 장점에도 불구하고 내재적인 특성상 가지는 문제점 – 서비스의 Safety 또는 QoS 를 보장할 수 없는- 으로 미션 크리티칼 시스템에 적용할 수 없다는 한계점을 가지고 있다[4,5, 6].

이를 극복하기 위하여 서비스 기반 아키텍처에 시맨틱스를 적용하려는 움직임들이 있어왔다. 시맨틱스를 사용한 방법 중 하나는 서비스를 활성화하기 위한 제한조건을 명시한 계약서(contract)를 구성하는 것이며 서비스 실행 직전이나 또는 서비스 실행직후에 사전/사후 상태(pre/post condition)를 검사하고 서비스 진행 상황을 추론, 서비스 실행여부를 결정하는데 활용된다. 또 다른 방법은 복합 서비스의 프로세스 중간 중간에 체크포인트(checkpoint)를 두어 프로세스의 진행 내용을 모니터하는 것인데 이때 시맨틱스는 서비스의 기능을 정의하고 진행상황을 이해할 수 있도록 하는데 사용되어 왔다 [7,8,9].

하지만 이러한 방법 역시 제한점을 가지고 있는데 위의 방법들은 모두 서비스 실행 중에 진행상황을 모니터링하는 것으로 서비스 구성단계에서 서비스의 성능을 예상하고 그에 따른 서비스를 구성할 수가 없다는 단점을 가진다. 즉 공개된 시맨틱스에 따라 복합 서비스(aggregate service)를 서비스를 구성하고 그 진행되중에 예외상황을 파악할 수는 있으나 구성된 서비스의 성능이나 또는 위험도를 파악할 수가 없다는 단점을 가지고 있으며, 추가적으로 공개된 서비스의 기능에 거짓이 있거나 또는 예외상황이 발생하는 경우 서비스를 동적으로 재구성할 필요가 있는데 기존의 시맨틱스를 통한 접근 방법으로는 동적으로 재구성하는데 한계점 역시 지니고 있다.

이러한 SOA 의 한계점을 극복하기 위한 한가지 방법으로 본 연구에서는 동적 시멘틱스를 활용하고자 한다. 본 연구에서 동적 시멘틱스는 서비스의 제공환경과 프로세스 중간 진행 상황, 서비스 히스토리에 따라 다양한 적용이 가능한 시멘틱스로 정의된다. 동적 시맨틱스에서는 FSM 과 시맨틱스를 결합한 형태로 구성되며 기능적(functional) 요소는 시맨틱스로 비기능적(non-functional) 요소는 상태(state)로 표현되는 특징을 가진다. 각 상태는 서비스의 히스토리 또는 사용자의 평판도(reputation)에 의하여 결정되며 상태에 따라 특정 서비스의 공개된 정보의 신뢰 여부 또는 사용여부를 결정할 수 있다.

본 논문에서 이러한 동적 시맨틱스를 어떻게 정의하였고 SOA 기반 아키택처에서 어떤 방식으로 적용

할 수 있는지를 기술한다.

2. 동적 시멘틱스

우리가 제안하는 동적 시맨틱스는 서비스의 기능적 요소를 표현하는 부분과 비 기능적 요소를 표현하는 부분으로 구성되어 있다. 서비스의 기능적 요소를 표현하는 부분은 OWL[10]이나 WSML[11]과 같은 온톨로지 기반 언어로 표현할 수 있으며 다음과 같은 튜플로 정의된다.

$V =< p, o, q >,$

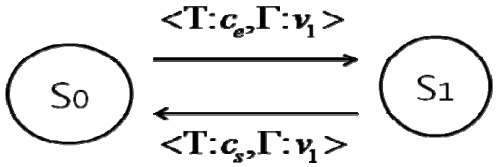
- V 는 SOA 환경에서의 서비스의 집합을 나타낸다.
- o 는 서비스가 제공하는 연산을 의미한다. o 는 실제 서비스가 호출되어 작업을 하는 부분으로 호출인자와 리턴인자로 구성된 함수로 표현된다.
- p 는 pre condition 을 나타내며 pre condition 은 연산 o 가 시행되기 위한 조건을 의미하고 p 가 만족될 경우에만 p 를 사용자가 호출할 수 있다는 것을 의미한다. 서비스 고객이 만족시켜야 하는 호출 조건 (예를 들면 가격, 회원가입 여부)와 서비스 제공자가 만족시켜야 하는 서비스 조건(예를 들면 사용 가능시간)을 모두 의미할 수 있다.
- q 는 post condition 으로 서비스 실행후의 effect 로 표현할 수도 있으며 서비스 호출이 끝난 후에 만족시킬만한 고객의 상태 또는 시스템의 상태를 의미하며 미리 값을 지정하여 예외상황 여부를 판단할 수 있다.

동적 시맨틱스는 위의 시맨틱스의 기반위에서 다음의 구조와 같이 정의된다.

$DS =< E, S, s_0, \Delta >$

- V 는 SOA 환경에서 웹 서비스로서 제공하는 서비스들을 나타낸다.
- $S \subseteq \varphi(V)$ 는 각 서비스의 시맨틱 상태 (semantic state)의 집합을 의미하며 각 서비스들의 현재 활성 정도 또는 신뢰도를 의미한다.
- $s_0 \in S$ 은 S 의 초기 상태를 의미한다.
- Δ 는 FSM 기반의 상태 천이 관계로 다음과 같은 형식으로 표현된다. $\Delta \subseteq S \times (\tau(C) \times \varphi(V)) \times S$. Δ 는 상태 $s \in S$, $s' \in S$ 간의 천이 관계를 표현하며 상태천이는 C 로 표현되는 특정 조건이 V 로 표현되는 특정 서비스에 발생하는 경우 천이가 발생하는 것을 나타낸다.

상태의 천이는 다음의 예를 통하여 설명한다. $V=\{v_1\}$, $S=\{s_0, s_1\}$, 로 정의하며 s_0 으로 초기화된다. 하지만 특정 서비스가 호출되고 실행되고 그 post condition 이 c_e 로 표현되는 제약 조건을 만족시키게 되고 v_1 이 Γ 로 표현되는 서비스 타입에 속하게 된다면 Δ 관계에 따라 상태가 천이된다. 이러한 상태에 따라 사용자는 호출 여부를 판단할 수 있다. 이러한 상태의 천이는 FSM 으로 표현되는데 <그림 1>과 같다..



<그림 1> Δ관계를 표현한 상태천이도

Γ 는 각 서비스가 제공하는 연산의 종류를 매핑하는 함수로서 연산의 종류에 따라 특정 post condition 이 상태천이를 유발하기도 하고 유발하지 않을 수도 있으며 또한 같은 post condition 이 매핑된 서비스의 연산 종류에 따라 다른 상태로 천이될 수 있음을 보여준다.

이러한 각 상태는 상태만의 시맨틱스를 가지고 있는데 예를 들면 s_0 를 모든 종류의 서비스 소비자에게 빠르고 정확한 서비스를 제공할 수 있는 상태를 의미한다면 s_1 의 경우 예외상황이 발생한 적이 있는 서비스 또는 응답시간이 늦은 서비스로 정의할 수 있다. 이때 s_1 으로 천이된 서비스의 경우 신뢰도를 회복할 수 있는 조건 c_s 를 만족시켜야 s_0 으로 재 천이될 수 있다.

3. SOA 환경에서의 동적 시맨틱스 지원

동적 시맨틱스를 지원하기 위해서는 중앙 시스템을 두어 서비스의 비기능적 특성을 저장할 수 있어야 한다. 서비스의 비기능적 특성의 경우 상태로 표현되기 때문에 중앙 시스템에서는 상태를 먼저 정의하여야 하고 상태가 의미하는 바를 명확히 표현하여 각 상태별로 접근권한 또는 사용권한을 분류할 수 있어야 한다.

또한 각 상태로 천이되는 조건을 명확하고 정확하게 기술할 수 있어야 하는데 중앙 시스템에서는 서비스의 히스토리 또는 평판도를 기록할 수 있는 시스템을 갖출 수 있어야 한다. 히스토리 또는 평판도는 상태 천이를 유발하는 조건이 되고 특정 서비스의 사용 가능 정도를 판단하고 예측할 수 있는 근거가 된다.

4. 결론

우리는 이번 논문에서 동적 시맨틱스를 사용하여 SOA 의 한계점을 극복하려는 시도를 하였다. 우리가 제안하는 동적 시맨틱스는 시맨틱스를 서비스의 실행

내용과 서비스의 비기능적 요소를 표현하는 것으로 확장하였다는 것에 가장 큰 의미를 가진다고 볼 수 있다. 서비스의 비기능적 요소들은 기존에도 시맨틱스로 표현하는 시도가 있었지만 동적으로 변화하는 비기능적 데이터의 특성상 동적 시맨틱스의 적용이 가장 적합하다고 보며, 이러한 시맨틱스의 표현의 확장은 온톨로지를 사물의 기술에서 벗어나서 프로세스의 실행내용 또는 프로세스의 예측에 사용할 수 있다는 가능성을 열어두고 있다고 본다.

현재 시맨틱스의 구현은 WSDL 로 시맨틱스를 구현하였으며 OWL[10]이나 기타 온톨로지 언어로도 구현가능하다고 생각된다. 시스템에서의 구현은 실험실 규모의 서비스 호출과 응답 선에서 이루어져 있으며 서비스의 실행은 JAVA 기반으로 구현되었다.

앞으로는 해야 할 일은 구체적인 비기능적 서비스 요소를 분류하고 이를 동적 시맨틱으로 구현하는 과제가 남아있다고 본다.

참고문헌

[1] T. Erl, *Service Oriented Architecture: concepts, technology and design*, Prentice Hall, Upper Saddle River, NJ, 2005.

[2] G. Lewis, E. Morris, L. O'Brien, D. Smith, and L. Wrage, SMART: *The service-oriented migration and reuse technique,* Technical Report CMU/SEI-2005-TN-029, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 2005.

[3] M. Amar and N. Muhammad, *Service Oriented Registry.* http:idenet,bth.se/servlet/download/element/36064/SOA-Registry.pdf

[4] L. O'Brien, P. Brebner, and J. Gray, Business *Transformation to SOA: Aspects of the Migration and Performance and QoS Issues*. SDSOA 08, Germany

[5] B. Saran-Starosta, R.E.K. Stirewalt and Laura K. Dillon, *Contract and Middleware for Safe SOA Applications*, SDSOA07, 2007

[6] T.G.J. Scepers, M.E. lacob, P.A.T. Van Eck, *A lifecycle approach to SOA governance,* SAC08, Brazil, 2008

[7] N. Wilde, S. Simmons, M. Pressel, and J. Vandeville, *Understanding Features in SOA: Some Expriences from Distributed Systems*, SDSOA08, Germany

[8] P. Brebner, L. O'Brien, J. Gray, *Performance Modeling for Service Oriented Architectures*, research demonstration track, ICSE 2008.

[9] H. Ludwig, *Web services QoS: external SLAs and internal policies or: how do we deliver what we promise?* WISEW 2003.

[10] OWL Web Ontology Language Overview, *http://www.w3.org/TR/owl-features*

[11] WSML-Web Service Modeling Language, *http://www.wsmo.org/wsml*.