클라우드 컴퓨팅에서 BPEL 분석 및 검증을 위한 Onion 언어로의 변환

최재홍*, 온진호*, 이문근*
*전북대학교 컴퓨터공학과
e-mail : {jaehong.choe, jjinghott}@gmail.com, moonkun@jbnu.ac.kr

Transformation of BPEL to Onion Language for Analysis and Verification of BPEL in Cloud Computing

Jae-Hong Choe*, Jin-Ho On*, Moon-Kun Lee*
Dept. of Computer Engineering, Chon-Buk University

요 약

클라우드 컴퓨팅에서 사용되는 웹 서비스들은 워크플로우에 따라 서비스가 설계되어 조합된다. 대표적인 웹서비스 명세언어인 BPEL 의 검증방법에는 *Petri nets*, *Abstract State Machine*(ASM), *BPE-Calculus* 등이 존재한다. 하지만 기존의 방법은 설계와 검증이 분리되어 있어 일관성이 부족하고, 시각화 문제, 동일성, 시간에 대한 제약조건의 문제점이 존재한다. 이에 대한해결방안으로 이동성, 재구성성, 동일성, 시간속성등의 새로운 분석 방법을 제시하는 Onion 언어가 제안되었다. 본 논문은 BPEL 로 명세된 서비스를 Onion 시스템에 적용시키기 위한, 변환 과정에 대해서 다룬다. 이에 대한 과정으로 BPEL 의 액티비티를 Onion 으로 변환하고, 워크플로우 패턴을 적용하여, 3 가지 패턴을 Onion OVL 로 변환을 적용하였다. 이를 통하여 BPEL 을 Onion OVL 로 변환하는데 문제가 없음을 보였으며, 효율적인 표현이 가능함을 보였다. 추후 Onion 시스템의 컴포넌트로 적용하여, BPEL 로 작성된 서비스를 Onion 시스템을 통해 분석/검증할 수 있다.

1. 서론

클라우드 컴퓨팅에서 사용되는 웹 서비스들은 *Business Process Execution Language*(BPEL)[1]에 의해 여러 서비스들이 조합되어 설계된다. BPEL 은 실행 레벨에서 서비스의 결합을 지원하여, 사용하기 전 분석/검증이 필요하다.

BPEL 로 조합된 웹 서비스들의 검증 방법으로는 Petri nets[2], *Abstract State Machine*(ASM)[3], BPE-Calculus[4] 등의 검증방법이 존재한다.

기존의 BPEL 의 검증 방법은 설계와 검증이 따로 분리되어있고, 각각의 검증방법과 검증하고자 하는 데이터에 따라 BPEL 을 각 방식에 맞게 변환하여 사용하므로 통일성이 부족하여, 기존의 검증방법으로는 시각화 문제, 동일성, 시간에 대한 제약조건 등 검증하기 어렵다. 이러한 문제점을 극복하기 위해, 다양한 검증 방법을 제공하는 새로운 명세언어를 도입할 필요성이 있다.

Onion[5]에서는 BPEL 의 다른 검증방법에서 제공하지 않은, 이동성, 재구성성, 동일성, 시간속성 등의 새로운 분석 방법을 제시하고, 시각화 및 통합화를 통하여 기존의 분석 및 검증방법의 한계를 극복하였다. 따라서, 새로운 명세 언어로 Onion 을 도입하기 위해서 기존의 BPEL 로 명세된 서비스를 Onion 에서 사용하기 위한 BPEL 의 문법을 Onion 의 문법으로 변환하는 과정이 필요하다.

본 논문은 2 장에서 관련연구로 BPEL 과 워크플로우 패턴 그리고 Onion 에 대해서 다루고, BPEL 의 핵심 문법 및 패턴을 예로 들어 설명한다. 3 장에서 BPEL 을 Onion 으로 변환하는 방법에 대해서 살펴본다. 마지막으로 4 장에서 결론 및 향후연구를 통하여 결과와 앞으로의 진행 방향에 대하여 논한다.

2. 관련연구

2.1 BPEL

BPEL 은 비즈니스 협업을 지정하고, 이를 구현하는데에 사용되어지며, 실행레벨에서 서비스 결합을

이 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No.2010-0023787)

지원한다.

BPEL 의 프로세스 구조는 프로세스에 의해 사용되는 데이터 변수를 정의하는 *Variable,* 비즈니스 프로세스와 상호작용하는 서로 다른 상대편들을 정의하는 *PartnerLinks*, 할당과 승인 서비스의 호출로부터 발생하는 결함의 결과로 수행되어야하는 단위업무를 정의하는 *faultHandler* 와 기본 액티비티와 구조적인 액티비티로 구분이 가능한 *Activity* 로 되어 있다.

BPEL 을 Onion OVL 로 변환하기 위해, BPEL 액티비티를 살펴본다. 표 1 은 BPEL 에서 사용되는 기본 액티비티와 구조적인 액티비티의 종류와 역할이다. 크게 나누어 통신, 에러처리, 선택, 반복, 병렬처리 등으로 구분할 수 있다.

<표 1> BPEL 액티비티 종류

이름	내용
Invoke	invoking operations offered by partner Web services
Receive	waiting for messages from partner Web services
reply	for capturing interactions
Assign	updating variables
Wait	delaying the process execution
Throw	signaling faults
compensate	triggering a compensation handler
empty	Doing nothing
exit	Ending a process immediately
Flow	activities being executed in parallel
Sequence	activities being executed sequentially
if	capturing conditional routing
While	structured looping
repeatUntil	structured looping
forEach	executing multiple instances
Scope	grouping activities into blocks
Pick	capturing race conditions

2.2 워크플로우 패턴

비즈니스 프로세스 관리에서 그 형태가 반복적으로 나타나는 워크플로우 패턴이 존재한다[6]. 워크플로우 패턴은 소프트웨어 엔지니어링 또는 비즈니스 프로세스 엔지니어링 분야에 정의된 디자인 패턴의 특수한 형태로 *Loops, decisions and sequence flows* 와 같은 가장 중요한 20 가지의 기본 패턴으로 구분할 수 있다. BPEL 2.0 에서는 20 가지 패턴중에 16 가지의 패턴을 지원하고 있다.

표 2 는 BPEL 이 지원하는 워크플로우의 분류와 그 패턴이다.

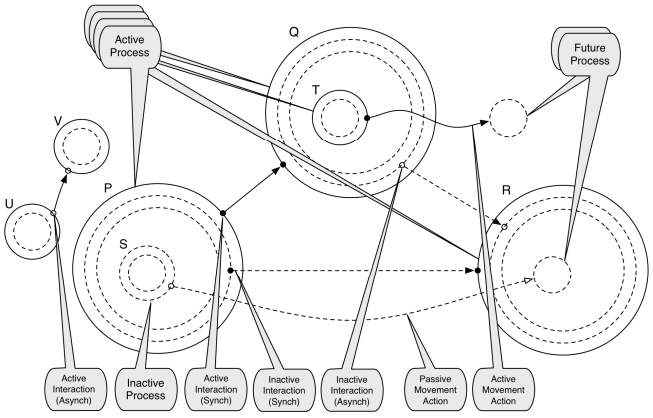
<표 2> BPEL 이 지원하는 워크플로우 패턴 종류

분류	패턴
Basic Control Flow pattern	Sequence Parallel Split Synchronization Exclusive Choice Simple Merge
Advanced Branching & Synchronization Patterns	Multi choice Synchronization Discriminator
Multiple Instance (MI) Patterns	MI without Synchronization MI with a Priori Design-Time Knowledge
	MI with aPriori Run-Time Knowledge
State-based Patterns	Deferred Choice
Iteration & Termination Patterns	Implicit Termination
Cancellation Patterns	Cancel Activity Cancel Case

2.3 Onion

Onion 은 기존의 프로세스 대수의 제약과 복잡성을 극복하기위하여 시각적 언어와 시각화된 형태로 프로세스의 분산, 이동 그리고 상호작용 액션에 대해서 시각적으로 표현할 수 있는 프로세스 대수를 제안하며, 가능한 모든 액션에 대한 시각화된 표현, 액션의 시간 속성에 대한 시각화, 결과적으로 Onion 은 통합된 환경에서 *in-the-large*(ITL)와 *in-the-small*(ITS) 관점으로 시스템을 시각적이고 정량화된 형태로 이해할 수 있게 한다.

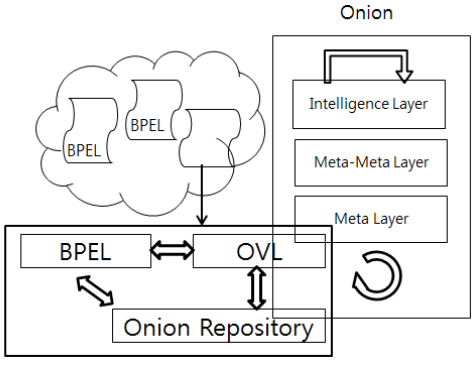
Onion 은 크게 프로세스대수로 표현되는 *Onion Text Language*(OTL), 이를 기반으로 한 시각화 언어인 *Onion Visual Language*(OVL), 공간상에 표현되는 *Geo-Temporal Block*(GTB), 시각적 제약 조건을 나타내는 *Visual Logic*(VL)의 4 가지로 구분된다.



(그림 1) Onion OVL 에서의 액션의 종류

그림 1 은 Onion 에서 OVL 로 프로세스와 액션을 보여주는 예이다.

3. BPEL 을 Onion 으로 변환



(그림 2) 개념 모델

그림 2 에서 논문에서 제안하는 클라우드의 BPEL 로 표현된 서비스를 Onion 문법으로 상호 변환하는 것을 보여준다. 본 논문에서는 BPEL 을 OVL 로 표현하는 것까지 제안을 하였고, 추후에 Onion Repository 로 변환하는 것을 추가할 예정이다.

먼저 BPEL 의 액티비티를 OVL 로 표현해보면, 표 3 과 같이 나타낼 수 있다. 표 3 은 BPEL 문법 샘플과 그에 따른 Onion OVL 을 보여준다.

<표 3> BPEL 액티비티를 Onion OVL 로 변환 예

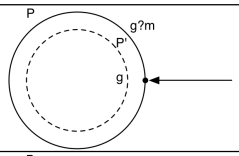
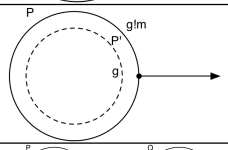
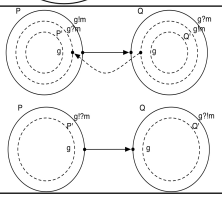
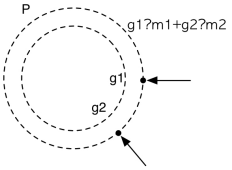
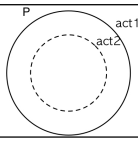
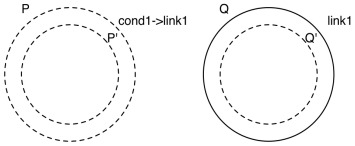
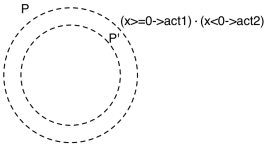
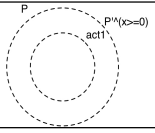
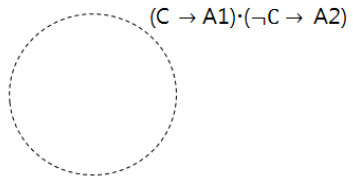
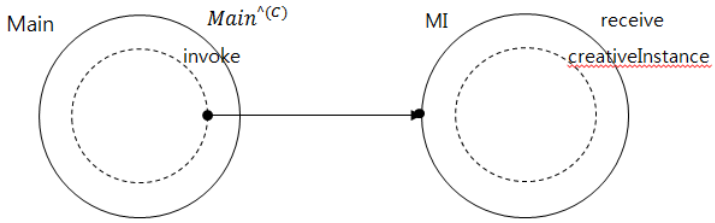
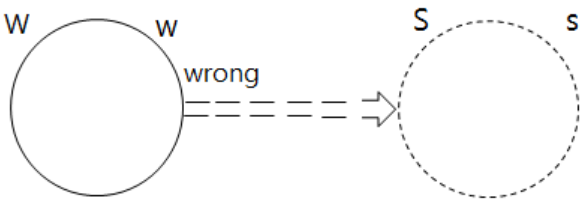
Sample BPEL	OVL
<receive ... variable="m"> </receive>	
<reply ... variable="m"> </reply>	
<invoke … invar="mS" outvar="mR"> </invoke>	
<pick …> <onMessage … variable="m1"> <… act1 …> </onMessage> <onMessage … variable="m2"> <… act2 …> </onMessage> </pick>	
<sequence …> <… act1 …> <… act2 …> </sequence>	
<flow …> <…act1…> <source linkname="link1" condition="cond1"/> </act1> <…act2…> <target linkname="link1"/> </act2> </flow>	
<switch> <case condition="bpws:getVariableData(x)>=0"> <..act1..> </..act1..> </case> <otherwise> <..act2..> </..act2..> </otherwise> </switch>	
<while condition="bpws:getVariableData(x)>=0"> <..act1..> </..act1..> </while>	

표 3 에서는 표 1 에서의 액티비티 중 지면상의 이유로 일부분을 선정하여 표현하였다.

표 2 의 워크플로우 패턴 중 지면상의 이유로 패턴 3 가지를 선택하여 Onion 으로 변환하면 표 4 와 같다.

Choice 패턴은 Condition C 에 따라 액티비티 A1 이 실행되거나 A2 가 실행될 수 있다는 것을 표현하고, Multiple Instance without Synchronization 패턴은 condition C 인 동안 메인은 반복해서 MI 를 호출하고, MI 는 새로운 인스턴스를 생성하여 메시지를 받는 것을 보이고 있다. 마지막 Cancel Activity 는 에러발생시 FaultHandler 에 의해 액티비티 S 가 실행되고, 그렇지 않으면 액티비티 W 가 실행되는 것을 표현하고 있다. 이 3 가지 예제를 통하여, 대표적인 BPEL 워크플로우 패턴이 Onion OVL 로 변환될 수 있음을 확인 하였다.

<표 4> BPEL 워크플로우 패턴 변환 예

Pattern	BPEL	Onion
Choice	<if> <condition>C</Condition> activity A1 <else> activity A2 </else> </if>	
MI without Synchronization	<process name="Main"> <while> <condition>C</condition> <invoke process-MI …. /> </while> </process> <process name="MI"> <receive process-Main … creativeInstance="yes"> activity A </process>	
Cancel Activity	<scope> <FaultHandlers> <catch faultName="wrong"> activity S </catch> </FaultHandlers> <sequence> <throw faultName="wrong" /> activity W </sequence> </scope>	

4. 결론 및 향후연구

클라우드 컴퓨팅에서 웹서비스 명세언어로 사용되는 대표적인 언어인 BPEL 의 다양한 검증방법들의 통일성 문제와 시각화 문제, 동일성, 시간에 대한 제약조건 등의 문제점이 존재한다. 분석/검증에 있어서 다양한 장점이 존재하는 Onion 언어를 통하여 BPEL 의 분석 및 검증의 제약을 극복하기 위한 방법을 제안하였다. 기존의 BPEL 로 명세된 서비스를 적용하기 위해서, BPEL 의 액티비티와 워크플로우를 정리하였고, 이것을 Onion OVL 으로 변환하는 과정과, BPEL 의 워크플로우 패턴을 Onion OVL 으로 변환한 과정을 다뤘다.

BPEL 문법에서 나타나는 패턴들을 이론적으로 Onion OVL 로 표현이 가능하였으며, 변환 결과를 통하여, 효율적인 표현이 가능함을 볼 수 있다.

향후연구로는 Onion Repository 문법으로 변환하는 과정을 거치고, Onion 시스템의 컴포넌트의 일부분으로 구현하여 활용할 수 있도록 할 것이다.

참고문헌

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, Business process execution language for web services, version 1.1, Feb 2007.

[2] X. Yi and K.J. Kochut, "Process composition of web services with complex conversation protocols: a colored Petri nets based approach,"In Proceedings of the Design, Analysis, andSimulation of Distributed Systems Symposium,pages 141–148, Arlington, VA, USA, April 2004.

[3] D. Fahland and W. Reisig, "ASM-based semantics for BPEL: The negative control flow," In D. Beauquier, E. Borger, and A. Slissenko, editors, Proceedings of the 12th International Workshop onAbstract State Machines, pages 131–151, Paris, France, March 2005.

[4] M. Koshkina and F. van Breugel, "Modelling and verifying web service orchestration by means of the concurrency workbench," ACM SIGSOFT Software Engineering Notes, 29(5), September 2004.

[5] J. On, "Onion: A Graphical Language for Process Algebra," in Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual, 2011, pp. 708–711.

[6] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22 , BPMcenter.org, 2006.