트랜잭션 상호호환성 보장을 위한 시맨틱스 기반의 트랜잭션 프로토콜[1)]

A Semantic-based transaction protocol for guaranteeing transaction interoperability

강동우*, 이순재*, 김광수*, 이재열**

* 포항공과대학교 산업경영공학과 ({hyunil, supersun, kskim}@postech.ac.kr)
** 전남대학교 산업공학과 (jaeyeol@chonnam.ac.kr)

Abstract

웹 서비스 기반의 비즈니스 프로세스 관리 시스템은 트랜잭션에 대한 신뢰도의 확보를 위하여 트랜잭션 관리가 필요하다. 트랜잭션 관리를 위한 여러 트랜잭션 프로토콜들이 제시되었지만, 다양한 프로토콜들은 서로간의 이질성으로 인해 비즈니스 프로세스 관리 시스템들 간의 상호호환을 저해한다. 이를 해결하기 위하여 본 연구에서는 시맨틱스 기반의 트랜잭션 프로토콜을 제안한다. 시맨틱스 기반의 트랜잭션 프로토콜은 상태와 메시지에 대한 정적 시맨틱스(Static Semantics)와 상태 전이에 관한 동작 시맨틱스(Operational Semantics)로 구성된다. 정적 시맨틱스는 프로토콜의 상태와 메시지를 Web Ontology Language(OWL)을 사용하여 온톨로지 기반으로 정의한 모델로서, 트랜잭션 프로토콜에 대한 상호 이해도를 높일 수 있을 것으로 기대된다. 동작 시맨틱스는 비즈니스 트랜잭션의 상태전이를 Abstract State Machine(ASM)으로 정의한 모델로서, 트랜잭션 실행에 대한 자동화와 체계적인 모니터링을 지원할 것으로 기대된다. 트랜잭션 프로토콜들 간의 상호호환을 위하여 이러한 정적 시맨틱스와 동작 시맨틱스를 기반으로 하여 각 트랜잭션 프로토콜 및 중립 트랜잭션 프로토콜의 시맨틱스를 정의한다. 정의된 각 트랜잭션 프로토콜의 정적 시맨틱스는 중립 트랜잭션 프로토콜의 정적 시맨틱스와 온톨로지 기반의 매핑 관계를 형성한다. 중립 트랜잭션 프로토콜의 정적 시맨틱스를 매개로 한 온톨로지 매핑은 각 트랜잭션 프로토콜들 간의 상호호환을 지원할 것으로 기대된다.

1. 서론

웹 서비스는 비즈니스 프로세스를 서비스 기반으로 변환시킴으로써 비즈니스 프로세스를 보다 빠르게 재구성하고 실행할 수 있는 환경을 구축할 수 있게 한다. 그러나 서비스 기반의 비즈니스 프로세스에서 발생하는 예측하지 못한 에러들은 서비스 기반 프로세스의 신뢰도에 치명적인 손상을 입히고 있다. 이러한 신뢰도의 손상을 방지하기 위해 서비스 기반의 비즈니스 프로세스를 위한 트랜잭션 관리의 필요성이 대두되고 있다.

기존의 트랜잭션은 데이터베이스 관리 시스템(DBMS)에 적용되던 것으로, ACID(Atomicity(원자성), Consistency(일관성), Isolation(분리성), Durability(지속성)) 속성을 철저히 따르고 있다. 이러한 속성은 데이터베이스 관리 시스템의 1)단단히 결합된(Tightly-coupled) 시스템, 2)짧은 트랜잭션 처리 시간, 3)트랜잭션 참가자들 간의 강한 신뢰감, 4)트랜잭션 관리자 하에서의 트랜잭션 수행 등의 특성에 기인한다. 하지만 서비스 기반의 비즈니스 프로세스 관리 시스템(BPMS)은 데이터베이스 관리 시스템과는 달리 1)느슨하게 결합된(Loosely-coupled) 시스템, 2)긴 트랜잭션 처리 시간, 3)참가자들 간 커뮤니케이션의 불신임성, 4)참가자들 각자의 독립적인 트랜잭션 관리 등의 특성을 가진다. 이러한 특성의 차이점으로 인하여 기존의 트랜잭션 관리 방법을 서비스 기반의 비즈니스 프로세스 관리 시스템에 적용하기에는 한계점이 있다.

이러한 한계점들을 해결하기 위하여 웹 서비스 환경에서 적용될 수 있는 다양한 트랜잭션 프로토콜들이 제정되었다. WS-Transaction(WS-T), Business Transaction Protocol(BTP), Web Services Transaction Management(WS-TXM)은 대표적인 웹 서비스 트랜잭션 프로토콜들이다. 그러나 트랜잭션 프로토콜들의 다양성은 비즈니스 프로세스 트랜잭션 관리 시스템들 간의 상호호환을 저해한다. 상호호환 저해의 주요 원인은 트랜잭션을 위해 사용되는 동일 개념들이 다양한 프로토콜 내에서 이질적으로 표현되고 있기 때문이다. 이러한 표현의 이질성이 있는 반면에, 다양한 트랜잭션 프로토콜들은 모두 동일하게 2단계 커미트(Two-Phase Commit, 2PC)를 트랜잭션 메커니즘으로 사용하고 있다. 따라서 동일한 트랜잭션 메커니즘을 기반으로 하여 표현의 이질성을 해결한다면, 트랜잭션 프로토콜들 간의 상호호환성을 기대할 수 있다.

이질적인 표현으로 인한 오해를 줄이고 프로토콜들 간의 상호호환성을 보장하기 위해 본 연구에서는 <그림 1>과 같은 시맨틱스 기반의 트랜잭션 프로토콜을 제안한다. <그림 1>의 서비스 기반의 비즈니스 프로세스 레벨은 비즈니스 프로세스의 흐름에 따른 실제 비즈니스 메시지의 교환을 위한 레벨이다. 트랜잭션 레벨은 서비스 기반의 비즈니스

1) 본 논문은 한국과학재단이 지원하는 특정기초연구사업에 의해서 수행되었음(R01-2003-000-10171-0)

프로세스 레벨을 지원하여 비즈니스 프로세스의 신뢰성을 확보하기 위한 레벨로서, 트랜잭션 프로토콜을 기반으로 트랜잭션 메시지를 교환하고 트랜잭션 상태 전이를 수행하는 레벨이다. 본 연구는 트랜잭션 레벨에서의 프로토콜들 간의 상호호환성 확보를 주목적으로 한다. 제시된 트랜잭션 프로토콜을 위한 시맨틱스는 정적 시맨틱스(Static Semantics)와 동작 시맨틱스(Operational Semantics)로 구성된다. 정적 시맨틱스는 트랜잭션 프로토콜 내의 상태, 메시지 등의 정적인 요소들을 Web Ontology Language(OWL)를 사용하여 온톨로지 기반으로 정의한 모델이다. 정적 시맨틱스는 트랜잭션 프로토콜의 상태와 메시지를 개념화하고, 이를 다른 프로토콜들의 상태와 매시지 개념과 연결시킨다. 이러한 개념화와 개념들 간의 연관관계 형성은 한 프로토콜이 다른 프로토콜내의 상태와 메시지를 이해할 수 있도록 지원한다. 이를 바탕으로 트랜잭션 프로토콜들 간의 표현의 이질성을 해결할 수 있을 것으로 기대된다. 동작 시맨틱스는 트랜잭션 프로토콜의 메시지 흐름에 따른 상태 전이를 Abstract State Machine(ASM)으로 정의한 모델이다. ASM은 수학적 포멀리즘을 바탕으로 상태전이를 모델링 하는 방법론으로서, 머신 가독성(machine-readability)을 갖는다. 따라서 동작 시맨틱스는 트랜잭션 운용에 관한 수학적 포멀리즘을 지원하고, 머신 가독성을 바탕으로 비즈니스 프로세스 관리 시스템이 트랜잭션을 쉽게 이해하고 수행하도록 지원할 것으로 기대된다. 정적 시맨틱스와 동작 시맨틱스는 독립적으로 존재하지만 상호 참조를 통해서, 정적 시맨틱스에서 이미 정의된 상태와 메시지를 동작 시맨틱스에서 사용할 수 있다. 이를 통해 중복적인 모델링을 줄이고 정적 시맨틱스와 동작 시맨틱스를 유기적으로 정의 할 수 있을 것으로 기대된다.

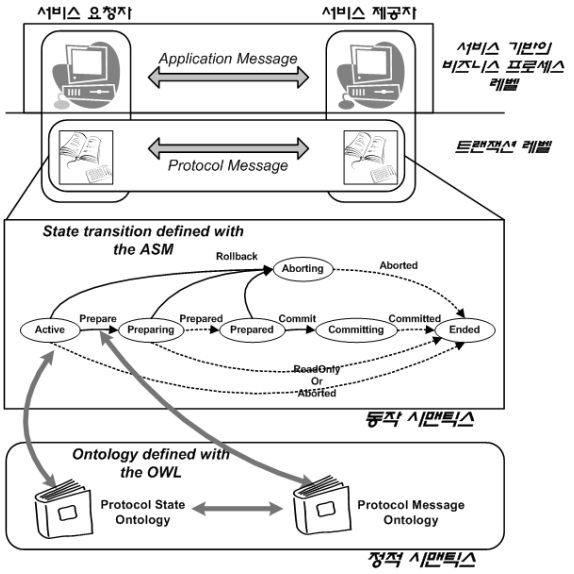


그림 1. 시맨틱스 기반의 트랜잭션 프로토콜

트랜잭션 프로토콜의 이질성으로 인한 문제를 해결하는 이상적인 방안은 모든 트랜잭션 프로토콜들을 아우르는 하나의 표준화된 프로토콜의 제정이다. 하지만 이 방안은 현실적으로 어렵기 때문에, 본 연구에서는 시맨틱스 기반의 중립 프로토콜을 매개로 한 트랜잭션 프로토콜들 간의 상호호환을 제시한다. 앞에서 제시한 정적 시맨틱스와 동작 시맨틱스를 바탕으로 각 트랜잭션 프로토콜들과 중립 프로토콜의 시맨틱스를 정의한다. 정의된 각 프로토콜들의 정적 시맨틱스는 중립 프로토콜의 정적 시맨틱스와 온톨로지 매핑(mapping) 관계를 형성한다. 이를 바탕으로 각 트랜잭션 프로토콜들의 정적 시맨틱스는 다른 프로토콜의 정적 시맨틱스와 온톨로지 매핑 관계를 형성할 수 있다. 형성된 온톨로지 매핑을 기반으로 트랜잭션 프로토콜들은 서로간의 표현의 이질성을 해결할 수 있으며, 이는 서로간의 상호호환성을 지원할 것으로 기대된다.

본 연구에서 제시하는 1) 시맨틱스 기반의 트랜잭션 프로토콜과 2)중립 프로토콜의 시맨틱스를 매개로 한 온톨로지 매핑은 프로토콜에 대한 이해를 돕고, 프로토콜들 간의 상호호환성 확보를 지원할 것으로 기대된다. 또한 트랜잭션 실행 단계에서 시맨틱스를 기반으로 자동화된 트랜잭션 수행과 트랜잭션에 대한 체계화된 모니터링을 지원할 수 있을 것으로 기대된다.

본 연구의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 기존 연구들에 대해 설명하고, 3장에서는 시맨틱스 기반의 비즈니스 프로토콜을 정적 시맨틱스와 동작 시맨틱스를 사용하여 정의한다. 4장에서는 시맨틱스 기반의 트랜잭션 프로토콜을 차용한 트랜잭션 관리 시스템을 제시한다. 마지막으로 5장에서는 본 연구에 대한 결론 및 추후 연구 과제에 대해서 설명한다.

2. 관련연구

웹 서비스 환경에서 비즈니스 프로세스의 트랜잭션을 수행하기 위한 프로토콜에 대한 연구는 활발히 이루어지고 있으며, 이미 다수의 프로토콜들이 제정된 상태이다. WS-Transaction(WS-T)은 BEA, IBM, Microsoft가 주도하고 있는 프로토콜이다. WS-T는 WS-AtomicTransactions(WS-AT)와 WS-BusinessActivity(WS-BA)의 두 하위 프로토콜로 구성된다. WS-AT는 기존의 데이터베이스 관리 시스템에서와 같이 ACID 속성을 반드시 지켜야 하는 트랜잭션에 대한 프로토콜이다. 이에 반해 WS-BA는 웹 서비스 환경에서 원활한 트랜잭션을 수행하기 위하여 원자성(Atomicity)과 분리성(Isolation)의 규제를 완화한 트랜잭션 프로토콜이다. Business Transaction Protocol(BTP)는 OASIS가 주도하는 프로토콜이다. BTP 또한 위의 WS-T와 같이 두 가지 형태의 트랜잭션을 지원한다. Atoms는 WS-AT처럼 기본적인 트랜잭션을 지원하는 프로토콜이지만, 분리성(Isolation)의 규제가 좀 더 완화되었다는 차이점이 있다. Cohesions는 WS-BA처럼 웹 서비스 환경에서의 비즈니스 프로세스를 위한 트랜잭션을 지원하는 프로토콜이다. 마지막으로 위의 두 프로토콜에 비해서 인지도가 낮은 편이지만, Sun Microsystems, Oracle, IONA Technologies 등이 주도하는 표준인 Web Services Transaction Management(WS-TXM)가 있다. WS-TXM은 Web Services Composite Application Framework(WS-CAF)의 한 부분으로서, TX-ACID, TX-LongRunningAction (TX-LRA), TX-BusinessProcess(TX-BP)로 구성되어 있다.

트랜잭션에 시맨틱스를 부여하는 연구는 아직까지 미비한 실정이다. Adams, N et al.은 온라인 서

비스의 트랜잭션에 대한 온톨로지에 관한 연구를 수행하였다. 그러나 이 연구는 단순히 트랜잭션에 사용되는 용어에 대한 정의를 한 것으로 트랜잭션 프로토콜에 시맨틱스를 적용했다고 보기는 힘들다. 또한 OWL과 같은 온톨로지 모델링 언어를 통한 정확한 모델링이 아니라 단순한 언어적 서술로 온톨로지를 정의하였다는 단점이 있다. Prinz, A. et al.은 데이터베이스 관리 시스템에서 ASM을 통하여 트랜잭션의 동작 시맨틱스를 정의하였다. 그러나 이 연구는 데이터베이스 관리 시스템에 국한된 것으로 서비스 기반의 비즈니스 프로세스 트랜잭션의 특성을 반영하지 않고 있다. DERI에서 제시한 Web Service Modeling Ontology(WSMO)는 온톨로지와 ASM을 활용하여 웹 서비스에 대한 모델링을 지원하지만 <그림 1>에서의 트랜잭션 레벨이 아닌 서비스 기반의 비즈니스 프로세스 레벨에서의 모델링이라는 차이점이 있다.

3. 시맨틱스 기반의 트랜잭션 프로토콜

서론에서 언급하였듯이 현재 제시된 트랜잭션 프로토콜들은 동일한 개념에 대해 서로 이질적인 표현을 사용하고 있기 때문에, 2단계 커미트를 기반으로 한 동일한 트랜잭션 메커니즘에도 불구하고 서로간의 호환이 어렵다. 표현의 이질성으로 인해 하나의 프로토콜을 통해 나타내어진 트랜잭션을 다른 프로토콜에서 해석하는데 어려움을 겪을 수 있으며, 프로토콜들 간의 오해의 소지가 될 수도 있다. 특히 트랜잭션 프로토콜 명세서가 자연어(Natural Language)를 통해 표현되어 있기 때문에 트랜잭션 용어의 개념이 모호하며, 트랜잭션 동작에 대한 정형성이 부족하다. 이러한 문제점들을 해결하기 위하여 시맨틱스 기반의 트랜잭션 프로토콜을 제안한다. 서론에서 언급하였듯이 시맨틱스 기반의 트랜잭션 프로토콜은 정적 시맨틱스와 동작 시맨틱스로 구성된다. 본 장에서는 BTP를 예로 들어 시맨틱스 기반의 트랜잭션 프로토콜을 구성해 보고자 한다.

3.1 OWL을 이용한 정적 시맨틱스의 정형화

정적 시맨틱스는 프로토콜에서 정의하는 프로토콜 상태와 프로토콜 메시지의 의미들을 온톨로지를 기반으로 기술함으로써 확보될 수 있다. 프로토콜 상태와 메시지에 대한 온톨로지를 정의하기 위해서 우선 그들이 가지는 의미의 유사도에 따라 위의 <표 1>, <표 2>와 같이 상태와 메시지를 분류할 수 있다. 상태와 메시지에 대한 의미적인 분류 후에 상태와 메시지들에 대한 각각의 개념(Concept) 및 개념들 간의 관계(Relation)를 온톨로지를 기반으로 정형화(Formalize)한다. 예를 들어 "ENROL"이라는 메시지는 트랜잭션을 수행하기 위해 참가자가 조정자에게 "트랜잭션에 등록을 요청한다."라는 의미로 보내는 메시지이다. 이 메시지는 갱신 준비 단계 중 등록 과정에서 전달되는 메시지로 분류될 수 있다. "ENROL"이 가지는 이러한 정의와 제약 조건들은 온톨로지를 통해 개념화된다. "ENROL"이 가지는 참가자 정보, 조정자 정보, 전송된 시간, 파기 일자 등은 개념에 대한 속성이다. 속성 중에서 조정자 정보 등은 다른 온톨로지에서 미리 정의해 놓은 개념들을 참조하여 사용한다.

정적 시맨틱스의 구체적인 모델링을 위해 본 연구에서는 온톨로지 모델링 언어인 OWL을 사용하였다. 트랜잭션 프로토콜들의 상태, 메시지를 OWL을 이용하여 정형화하는 과정은 다음과 같다. 먼저, 프로토콜에서 사용되는 각각의 상태와 메시지는 OWL의 class로 개념화된다. 위에서 예를 든 "ENROL" 메시지는 <그림 2>에서와 같이 ***<owl:Class rdf:ID="ENROL"/>***로 정의될 수 있다. 정의된 개념들 간의 계층 구조는 ***rdfs:subClassOf*** 태그로 표현될 수 있다. 예를 들어 "ENROL"은 메시지라는 것을 표현하기 위해 <그림 2>에서와 같이 ***rdfs:subClassOf***를 사용하여 "ENROL"와 "Message"의

표 1. 트랜잭션 프로토콜별 트랜잭션 상태 용어 비교

Phase		BTP	WS-AT	WS-BA(PC)	WS-BA(CC)	WS-TXM (ACID)
갱신준비 단계 (Pre-commit phase)	등록 (Register)	Enrolling	-	-	-	-
		Enrolled	Active	Active	Active	Active
	준비 (Prepare)	Preparing	Preparing	-	Completing	Preparing
		Prepared	Prepared	Completed	Completed	Prepared
	종료 (Resign)	Resigning	-	Exiting	Exiting	-
		Resigned	-	Ended	Ended	-
갱신단계 (Commit phase)	수행 (Commit)	One-phase-confirming	-	-	-	One-phase-commit
		Confirming	Committing	Closing	Closing	Committing
		Confirmed	Ended	Ended	Ended	Committed
	취소 (Cancel)	Canceling	Aborting	Canceling Compensating	Canceling Compensating	RollingBack
		Cancelled	Ended	Ended	Ended	RolledBack
	에러 (Error)	Contradicting	-	Faulting	Faulting	-
		Contradiction	-	Ended	Ended	-

표 2. 트랜잭션 프로토콜별 트랜잭션 메시지 용어 비교

Phase		BTP	WS-AT	WS-BA (PC)	WS-BA (CC)	WS-TXM (ACID)
갱신준비 단계 (Pre-commit phase)	등록 (Register)	ENROL	-	-	-	-
		ENROLLED	-	-	-	-
	준비 (Prepare)	PREPARE	Prepare	-	Complete	PrepareMessage
		PREPARED	Prepared	Completed	Completed	VoteMessage
	종료 (Resign)	RESIGN	-	Exit	Exit	-
		RESIGNED	-	Exited	Exited	-
갱신단계 (Commit phase)	수행 (Commit)	CONFIRM	Commit	Close	Close	CommitMessage
		ONE_PHASE_CONFIRM	Committed	-	-	OnePhaseCommitMessage
		CONFIRMED	-	Closed	Closed	CommittedMessage
	취소 (Cancel)	CANCEL	Rollback	Cancel Compensate	Cancel Compensate	RollbackMessage
		CANCELLED	Aborted	Canceled Compensated	Canceled Compensated	RolledbackMessage
	에러 (Error)	CONTRADICTION	-	Fault	Fault	heuristicFaultMessage
			-	Faulted	Faulted	-
읽기 모드 (ReadOnly)		-	ReadOnly	-	-	-

```
<?xml version="1.0"?>
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://www.TPO.com/ProtocolRole"/>
  <owl:imports rdf:resource="http://www.TPO.com/ProtocolState"/>
</owl:Ontology>
<owl:Class rdf:ID="ENROL">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A request to a
    Coordinator to enrol a Participant</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Message"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Restriction>
  <owl:ObjectProperty rdf:ID="participant">
    <rdfs:range rdf:resource="http://www.TPO.com/ProtocolRole#Participant"/>
    <rdfs:domain rdf:resource="#ENROL"/>
  </owl:ObjectProperty>
  <owl:cardinality>1</owl:cardinality>
</owl:Restriction>
<owl:DatatypeProperty rdf:ID="expireDate">
  <rdfs:domain rdf:resource="#ENROL"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#time"/>
</owl:DatatypeProperty>
```

그림 2. "ENROL" 메시지가 가지는 의미의 온톨로지를 통한 정형화

상속 관계를 정의하였다. 정의된 개념에 대한 속성은 rdf:Property 등의 OWL 태그를 통해 표현될 수 있다. 예를 들어 "ENROL" 메시지에 참가자 정보, 파기일자 등의 속성을 표현하기 위해 ***<owl:ObjectProperty rdf:ID="participant">***, ***<owl:DatatypeProperty rdf:ID="expireDate">*** 등의 태그를 추가할 수 있다. 또한 ***owl:minCardinality***, ***owl:maxCardinality***, ***owl:cardinality*** 등의 태그를 사용

하여 메시지나 상태의 속성에 대한 제약을 줄 수 있다. <그림 2>에서는 ***<owl:cardinality> 1 </owl:cardinality>*** 태그를 통하여 하나의 생성된 "ENROL" 메시지는 반드시 하나의 참가자를 속성으로 가져야 한다는 것을 표현하였다. 이외에도 OWL의 다양한 태그들을 사용하여 프로토콜 메시지와 상태에 대한 속성들을 표현할 수 있다. 또한 ***owl:imports*** 태그를 통하여 기존에 정의되어 있던 다른 온톨로지들을 참조하여 사용할 수 있다. 예를 들어 <그림 2>에서 볼 수 있듯이 "participant"라는 메시지 속성은 “http://www.TPO.com/ProtocolRole”에서 미리 정의된 개념을 참조하여 사용한 것이다. 이때 ***<rdfs:range>*** 태그는 “participant" 속성이 값으로 가질 수 있는 개체의 집합을 제한한다. 즉, "participant" 속성은 값은 http://www.TPO.com/ProtocolRole#Participant로 제한된다. ***<rdfs:domain>*** 태그는 ”participant" 속성을 적용할 수 있는 개체의 집합을 제한한다. 즉, “participant" 속성은 적용대상이 "ENROL"로 제한된다.

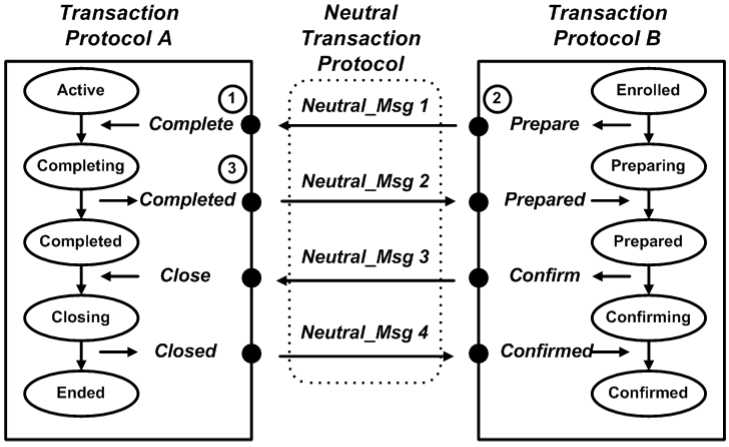
이렇게 정의된 프로토콜들의 정적 시맨틱스는 서로간의 관계 형성을 통하여 상호호환을 지원할 수 있다. 서로간의 관계는 OWL의 ***owl:imports***, ***owl:sameAs***, ***owl:differentFrom*** 등의 태그들을 사용하여 표현할 수 있다. 예를 들어 BTP 이외의 트랜잭션 프로토콜에서 BTP의 메시지의 하나인 "ENROL"을 아무런 수정 없이 사용하고자 한다면, OWL의 ***owl:imports*** 태그를 활용할 수 있다. 다른 트랜잭션 프로토콜에서 BTP의 "ENROL"과 의미는 같지만 "REGIST"라는 다른 메시지 명칭을 사용하는 경우에는 OWL의 ***owl:sameAs*** 태그를 통하여 서로 같은 메시지임을 표현 할 수 있다. 다른 트랜잭션 프로토콜에서 "ENROL"과 명칭은 같지만 다른 의미로 메시지가 정의된 경우에는 OWL의 ***owl:differentFrom*** 태그를 통하여 서로 다른 메시지임을 표현할 수 있다. 이를 기반으로 트랜잭션 프로토콜들의 개념들 간의 관계가 정의되며, 이러한 관계를 바탕으로 트랜잭션 프로토콜들과 중립 트랜잭션 프로토콜의 매핑을 구축할 수 있다. 트랜잭션 프로토콜들과 중립 트랜잭션 프로토콜 간의 매핑을 기반으로 한, 트랜잭션 프로토콜들 간의 상호호환성 보장에 대한 개념도는 <그림 3>과 같다.

상태와 메시지에 대한 개념들을 온톨로지를 기반으로 체계화하고 명시화함으로써 이질적인 프로토콜들이 서로를 이해할 수 있게 지원할 수 있다. 또한 OWL을 사용하여 정적 시맨틱스를 모델링함으로써, 정형성(formalism)과 기계 가독성(machine-readability)을 확보할 수 있으며, 이는 프로토콜들의 용어에 대한 정확한 이해를 도울 수 있을 것으로 기대된다. 마지막으로 트랜잭션 프로토콜들의 상태 시맨틱스의 온톨로지 매핑을 기반으로 프로토콜들 간의 상호호환성을 보장할 수 있을 것으로 기대된다.

3.2 ASM을 이용한 동작 시맨틱스의 정형화

동작 시맨틱스는 트랜잭션에서 프로토콜 메시지에 의한 상태의 전이를 ASM을 기반으로 정의한 모델이다. ASM은 수학적인 포멀리즘을 바탕으로 상태 머신(State machine)을 표현하는 방법론이다. 본 연구에서는 Microsoft 사의 ASM 모델링 언어인 Abstract state machine Language(AsmL)를 사용하여 동작 시맨틱스를 정형화 하였다.

ASM은 if..then..else...를 통해 상태 전이를 표현한다. 예를 들어 BTP는 초기 상태에서 "ENROL"이라는 메시지가 발생할 경우 "Enrolling"이라는 프로토콜 상태로 전이한다. 이는 <그림 4>와 같이 ASM으로 모델링 될 수 있다. 이와 같은 방식으로 트랜잭션 프로토콜의 동작 시맨틱스를 정의할 수 있다.



● *Ontology mapping*

①
```
<owl:Class rdf:ID="Complete">
   <owl:sameAs rdf:resource="whttp://ww.TPO.com/Message#Neutral_Msg1"/>
</owl:Class>
```

②
```
<owl:Class rdf:ID="Prepare">
   <owl:sameAs rdf:resource="http://www.TPO.com/Message#Neutral_Msg1"/>
</owl:Class>
```

③
```
<owl:Class rdf:ID="Completed">
   <owl:sameAs rdf:resource="http://www.TPO.com/Message#Neutral_Msg2"/>
</owl:Class>
```

그림 3. 온톨로지 매핑을 통한 상호호환성 보장

```
if currentTransactionState = null then
    if inputMessage = ENROL then
        step currentTransactionState := Enrolling
```

그림 4. ASM 모델링 예제

ASM은 수학적 포멀리즘을 바탕으로 기계 가독성을 갖는 상태전이 모델을 정의하는 방법이다. 따라서 ASM 기반의 동작 시맨틱스는 트랜잭션 운용에 대한 포멀리즘을 보장하고, 비즈니스 프로세스 관리 시스템이 트랜잭션 운용을 정확하고 쉽게 이해하도록 지원할 수 있을 것으로 기대된다. 또한 ASM 모델의 기계 가독성은 자동화된 트랜잭션 운용과 체계적인 비즈니스 프로세스 트랜잭션 모니터링을 지원할 것으로 기대된다.

위에서 살펴보았듯이 동작 시맨틱스를 표현하기 위해서는 프로토콜의 상태와 메시지가 필수적이다. 하지만 <그림 4>의 동작 시맨틱스는 상태와 메시지를 정적 시맨틱스와 별도로 정의한다. 이는 중복된 모델링이며, 서로간의 이질적인 모델링으로 혼란을 줄 수 있다. 따라서 동작 시맨틱스는 정적 시맨틱스를 참조하여, 이미 정의된 상태와 메시지를 사용할 필요가 있다. 온톨로지 기반으로 정확하고 구체적으로 모델링된 정적 시맨틱스의 상태와 메시지를 동작 시맨틱스에서 사용함으로써, 동작 시맨틱스를 효율적이고 구체적으로 정의할 수 있으며, 상태와 메시지에 대한 정의의 일관성을 유지할 수 있을 것으로 기대된다.

본 연구에서는 동작 시맨틱스에서 정적 시맨틱스를 참조하기 위해 Universal Resource Identifier(URI)를 사용하고자 한다. AsmL은 현재 URI를 지원하고 있지 않기 때문에 본 연구에서는 이를 표현하기 위하여 "***uri***"라는 명령어를 제안한다. "***uri***" 타입으로 정의된 변수는 정적 시맨틱스의 상태나 메시지 온톨로지를 지칭하는 URI를 값으로 갖는다. 예를 들어 <그림 2>에서 정의된 정적 시맨틱스의 "ENROL" 메시지는 <그림 4>의 ENROL이라는 변수가 참조할 수 있다. 이는 <그림 5>와 같이 "***uri***" 명령어를 사용함으로써 표현할 수 있다.

정적 시맨틱스에서 자세하고 구체적으로 모델링된 상태와 메시지를 동작 시맨틱스에서 사용함으로써, 비즈니스 프로세스 관리 시스템 및 다른 트랜잭션 프로토콜들이 동작 시맨틱스를 더욱 정확하고 자세하게 이해할 수 있을 것으로 기대된다. 또한 동작 시맨틱스에서 참조한 정적 시맨틱스를 참고함으로써, 비즈니스 프로세스 관리 시스템은 트랜잭션에 대한 체계적인 모니터링을 수행할 수 있을 것으로 기대된다. 예를 들어, 동작 시맨틱스의 트랜잭션 상태가 "Enrolling"일 때, 비즈니스 프로세스 관리 시스템은 정적 시맨틱스의 “Enrolling" 상태 온톨로지를 살펴봄으로써 트랜잭션 상태를 더욱 자세하고 구체적으로 이해할 수 있다.

4. 트랜잭션 관리 시스템

<그림 6>은 시맨틱스 기반의 트랜잭션 프로토콜을 차용한 트랜잭션 관리 시스템을 나타내고 있다. 관리 시스템은 크게 시맨틱스 저장소(Semantics

```
enum EnumState
  Enrolling:uri="http://www.TPO.com/ProtocolState#Enrolling"
  Enrolled:uri="http://www.TPO.com/ProtocolState#Enrolled"
                                (중략)
enum EnumMessages
    ENROL:uri="http://www.TPO.com/ProtocolMessage#ENROL"
    ENROLLED:uri="http://www.TPO.com/ProtocolMessage#ENROLLED"
                                (중략)
//Transaction when Enrolling state
      if currentTransactionState = Enrolling then
        if inputMessage = ENROLLED then
          step currentTransactionState := Enrolled
                                (생략)
```

그림 5. 동작 시맨틱스에서 정적 시맨틱스 참조

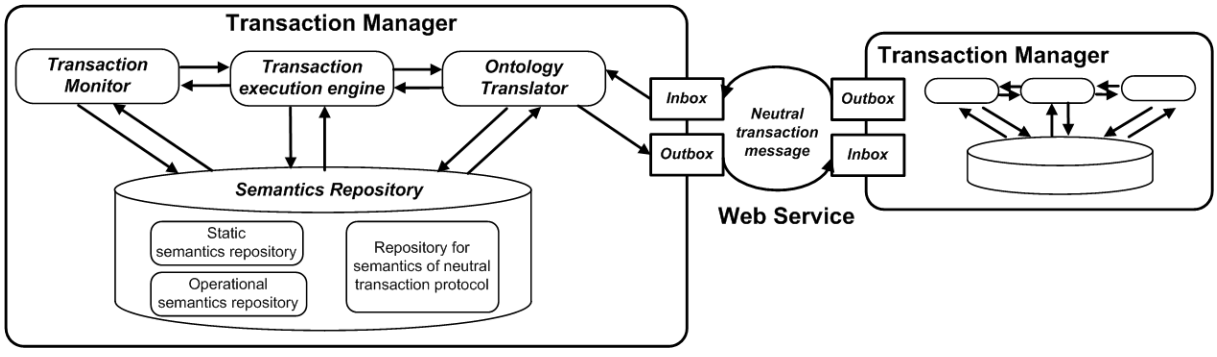


그림 6. 시맨틱스 기반의 트랜잭션 프로토콜을 차용한 트랜잭션 관리 시스템

repository), 트랜잭션 실행 엔진(Transaction execution engine), 온톨로지 변환기(Ontology translator), 트랜잭션 모니터(Transaction monitor)로 구성된다. 시맨틱스 저장소는 트랜잭션 관리 시스템이 적용하고 있는 프로토콜의 시맨틱스와 중립 프로토콜의 시맨틱스를 저장하고 있다. 트랜잭션 실행 엔진, 온톨로지 변환기, 트랜잭션 모니터는 시맨틱스 저장소에 저장된 시맨틱스를 이용하여 자신들의 역할을 수행한다.

트랜잭션 실행 엔진은 시맨틱스 저장소로부터 프로토콜의 동작 시맨틱스와 정적 시맨틱스를 읽어들여 트랜잭션을 수행한다. 트랜잭션 수행은 동적 시맨틱스의 상태 전이 ASM 모델을 기반으로 수행되며, 이때 발생하거나 받아들이는 메시지는 정적 시맨틱스의 메시지 온톨로지를 사용한다.

온톨로지 변환기는 시맨틱스 저장소에 저장된 시맨틱스를 기반으로 트랜잭션에서 발생하는 메시지 온톨로지를 변환시켜주는 역할을 한다. 즉, 트랜잭션 중에 상대방에게 보내는 메시지 온톨로지는 중립 트랜잭션 프로토콜의 시맨틱스를 기반으로 온톨로지 매핑을 통하여 중립 트랜잭션 프로토콜의 메시지 온톨로지로 변환된다. 또한 상대방에게서 받는 메시지는 중립 트랜잭션 프로토콜의 시맨틱스를 기반으로 온톨로지 매핑을 통하여 자신의 프로토콜 메시지 온톨로지로 변환된다.

상대방이 보낸 메시지는 인박스(Inbox)로 일단 받아서, 누구로부터 온 메시지인지 확인한 후 온톨로지 변환기를 거쳐 트랜잭션 실행 엔진으로 이동하여 자신의 트랜잭션 인스턴스의 상태를 전이시킨다. 자신이 보내는 메시지는 온톨로지 변환기를 거쳐 아웃박스(Outbox)로 이동한 후, 아웃박스에서 누구에게 보내는 메시지인지를 판별하여 상대방에게 메시지를 보내게 된다.

트랜잭션 모니터는 트랜잭션 실행 엔진의 트랜잭션 수행을 모니터링하는 기능을 수행한다. 트랜잭션 모니터는 트랜잭션 인스턴스의 상태 전이를 살펴봄으로써, 현재 트랜잭션 상태를 점검할 수 있으며, 시맨틱스 저장소의 시맨틱스를 기반으로 현 상태와 관련된 구체적인 정보, 관련 메시지 등을 얻을 수 있다.

5. 결론 및 추후 연구 과제

비즈니스 프로세스 관리 시스템은 비즈니스 프로세스의 신뢰성 확보를 위하여 트랜잭션 관리가 필요하다. 트랜잭션 관리를 위한 다양한 트랜잭션 프로토콜들이 제시되고 있지만, 이러한 다양한 프로토콜들의 표현의 이질성은 서로간의 상호호환의 장애가 되고 있다. 이를 해결하기 위해 본 연구에서는 시맨틱스 기반의 트랜잭션 프로토콜을 제안하였다. 제안된 시맨틱스 기반의 트랜잭션 프로토콜은 상태와 메시지에 대한 정적 시맨틱스와 상태 전이에 관한 동작 시맨틱스로 구성된다. 정적 시맨틱스와 동작 시맨틱스는 각각 OWL과 ASM을 통하여 모델링되었으며, 동작 시맨틱스에서 정적 시맨틱스를 참조하여 사용하기 위한 방법을 제시하였다. 정의된 각 트랜잭션 프로토콜의 정적 시맨틱스는 중립 트랜잭션 프로토콜의 정적 시맨틱스와 온톨로지 기반의 매핑 관계를 형성한다. 중립 트랜잭션 프로토콜의 정적 시맨틱스를 매개로 한 온톨로지 매핑은 각 트랜잭션 프로토콜들 간의 상호호환을 지원할 것으로 기대된다. 또한 온톨로지 기반의 정적 시맨틱스는 트랜잭션 프로토콜에 대한 상호 이해도를 높일 수 있을 것으로 기대되며, 마지막으로 ASM 기반의 동작 시맨틱스는 트랜잭션 실행에 대한 자동화와 체계적인 모니터링을 지원할 수 있을 것으로 기대된다.

본 연구의 한계점 및 추후 보완되어야 할 사항은 다음과 같다. 우선 정적 시맨틱스로부터 정확하고 다양한 정보를 얻기 위한 온톨로지 추론(Reasoning) 방법에 대한 연구가 필요하다. 또한 에러 처리와 보상 정책을 포함한 시맨틱스 기반의 트랜잭션 프로토콜에 대한 연구도 필요하다

참고문헌

Adams, N., Fraser, J., Macintosh, A., and McKay-Hubbard, A. (2002), Towards an Ontology for Electronic Transaction Services, *Int. J. Intell. Sys. Acc. Fin.Mgmt.*, **11**, 173−181.

Arjuna, BEA, Hitachi, IBM, IONA, Microsoft (2005), Web Services Atomic Transaction (WS-AtomicTransaction), available at ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf.

Arjuna, BEA, Hitachi, IBM, IONA, Microsoft (2005), Web Services Business Activity Framework (WS-BusinessActivity), available at ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf.

Arjuna, Fujitsu, IONA, Oracle, Sun Microsystems (2003), Web Services Transaction Management (WS-TXM), available at http://developers.sun.com/techtopics/webservices/wscaf/wstxm.pdf.

Börger, E. and Stärk, R. (2003), Abstract State Machines: A Method for High-Level System Design and Analysis, Springer-Verlag, USA.

Dalal, S. Little, M. Potts, M. Temel, S., and Webber, J. (2003), Coordinating Business Transactions on the Web, *IEEE Internet Computing Special Edition on Web Services*, 30-39.

ESSI WSMO Working group (2005), Web Service Modeling Ontology (WSMO), available at http://www.wsmo.org/TR/d2/v1.2/.

OASIS (2004), Business Transaction Protocol, available at http://xml.coverpages.org/BTPv11-200411.pdf.

Prinz, A. and Thalheim, B. (2003), Operational Semantics of Transactions, *Proceedings of the Fourteenth Australasian database conference on Database technologies*, **17**, 169-179.

W3C (2004), OWL Web Ontology Language Overview, available at http://www.w3.org/TR/owl-features/.