한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003 년 5 월 16 일-17 일 한동대학교(포항)

상태 기반 비즈니스 프로세스 트랜잭션 관리

State-driven Business Process Transaction Management

이순재, 윤장혁, 김광수

포항공과대학교, 산업공학과

Abstract

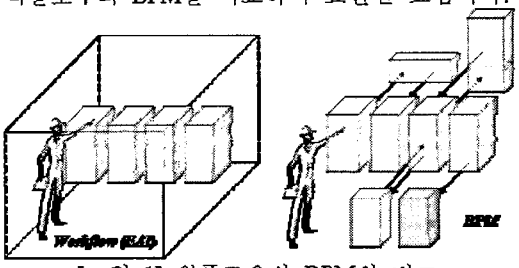
In the real world, business processes are very complex and composed of heterogeneous business activities. As the advent of the Web services enables business processes to be integrated and to be automated, it makes enterprises integrate heterogeneous business processes from different business partners as well as their internal business processes. To support recent trends in integration of business processes, BPEL, WS-C and WS-T specifications have been established since 2002. WS-C and WS-T describe the reliable business environment including compensation (undo) of completed business processes. The compensation of business processes is a basic requirement for automation of business processes among business partners. Systems implementing these specifications, however, very rarely exist. It's not only because those specifications are developed recently, but also because they are not perfect yet.

In this paper, a new business process transaction management, which complements the deficiency of WS-C and WS-T, is suggested. Furthermore, the new approach proposes the business logics for supervisory coordinators which manage serial and parallel business gates. The modification of traditional WS-T specification and the simplification of WS-C specification make business processes managed effectively.

Keywords: business process management, transaction management, compensation, Web services, BPEL, BPEL4WS, WS-Transaction, WS-Coordination.

1. Introduction

BPM(Business Process Management)을 간단하게 정의하면 차세대 웍플로우(workflow) 관리이다. 웍플로우 관리란 한정된 공간에 갇혀서 그 내부에 있는 비즈니스 프로세스들을 관리하는 것인 반면, BPM이란 통상적으로 공간에 제약을 받지 않고 내외부의 여러 관계자들을 자신의 비즈니스 프로세스로 포함시켜 관리하는 것을 말한다 [1]. 그림1은 웍플로우와 BPM을 비교하여 표현한 그림이다.



[그림 1] 웍플로우와 BPM의 비교

BPM을 통해서 새로운 비즈니스 기회를 창출하고자 할 때 BPI(Business Process Integration)를 한다. BPI란 비즈니스 파트너와 고객, 그리고 자신의 응용 프로그램(applications)을 통합하여 새로운 비즈니스 기회를 창출하는 것을 말한다. 이러한 BPI에서 웹서비스(Web services)의 역할이 증대되고 있다.

웹서비스는 기존 시스템내의 응용 프로그램들을 외부에 노출시켜 BPI를 단순화하는 역할을 한다 [1]. 개별 응용 프로그램들로 수행되던 대다수의 기존 업무가 웹서비스들로 노출되어 있다면, 구축할 새 비즈니스 프로세스에 맞는 웹서비스들을 선택, 통합하여 새로운 비즈니스 기회를 창출한다. 그림2는 웹서비스들을 통합하여 새로운 비즈니스 기회를 창출하는 BPI를 보여준다.

하지만 단순한 BPI만으로는 새로이 제공되는 비즈니스의 신뢰성을 보장 받을 수 없다. 각각의 비즈니스 프로세스들이 지정된 순서대로 올바로 수행되는지 감시(monitoring)하고 수행 중 어떤 이상이 생겼을 때에 이미

수행된 비즈니스 프로세스들을 원상복귀(compensate or undo)시키는 일이 필요하다. 이러한 역할을 맡고 있는 것이 상태기반 비즈니스 프로세스 트랜잭션 관리 시스템이다. 상태기반 비즈니스 프로세스 트랜잭션 관리를 위해 2002년 이래 여러 규격들(specifications)이 제정되고 있다.

[그림 2] 웹서비스를 통한 BPI

웹서비스를 통한 BPI를 기술하기 위해 BPEL4WS(Business Process Execution Language for Web Services, for short, BPEL)이라는 규격이 제정되었다. 또한 WS-C(Web Services Coordination)와 WS-T(Web Services Transaction)는 BPEL로 기술된 웹서비스들을 보다 신뢰도 높은 서비스로 제공하기 위해 제정되었다. WS-C와 WS-T 규격은 상태 기반 프로토콜을 수용할 수 있는 구조 (architecture)로 서술되어 있다. 하지만 두 규격은 그 세부 내용에 몇몇의 미비점들을 가지고 있어 이를 보완하는 일이 필요하다.

본 논문에서 제시한 상태기반 프로세스 트랜잭션 관리에서는 BPEL로 기술된 BPI의 신뢰도를 높이기 위해 상태기반의 WS-T 프로토콜을 수정하여 제시하고 있으며, WS-C의 구조를 보다 효율성 높은 구조로 단순화시키는 것을 그 내용으로 하고 있다. 뿐만 아니라 개선된 새로운 구조하에서 개별 웹서비스들의 상태에 따른 트랜잭션 알고리즘도 제시하였다.

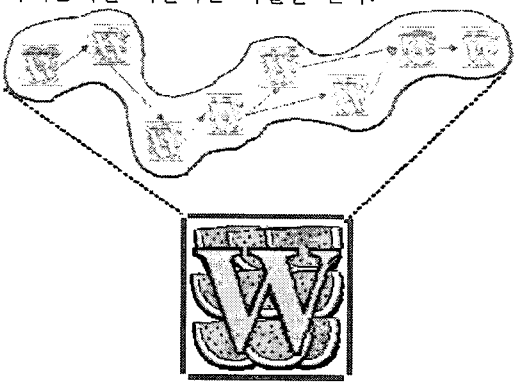
다음의 2장에서는 BPEL, WS-C, WS-T의 세 규격과 그들간의 관계에 대해 설명하고 있으며, 3장에서는 제안할 WS-T, WS-C의 수정안과 개선된 환경 하에서의 트랜잭션 알고리즘들에 대해 설명하고 있다. 4장에서는 실제 구현 예를 설명하고 있으며, 5장에서는 토의와 결론이 이어진다.

2. Related specifications

상태기반 비즈니스 프로세스 트랜잭션 관리 기술에 연관된 규격들로는 BPEL4WS, WS-C, WS-T가 있다. 2.1, 2.2, 2.3절에서 이것들에 대해 각각 간략히 설명하고, 2.4절에서는 그들이 각각 상태기반 비즈니스 프로세스 트랜잭션 관리 기술에서 어떤 역할을 하며 관련을 맺고 있는지 설명한다.

2.1. BPEL4WS

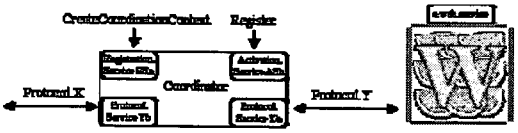
BPEL은 여러 웹서비스들을 특정한 흐름을 가진 새로운 웹서비스로 통합하여 제공하는 비즈니스 프로세스를 기술하는 규격이다 [2-3]. 즉, BPEL4WS는 그림3과 같이 다수의 웹서비스들을 어떤 순서로 어떤 흐름을 가지는지를 기술하는 역할을 한다.



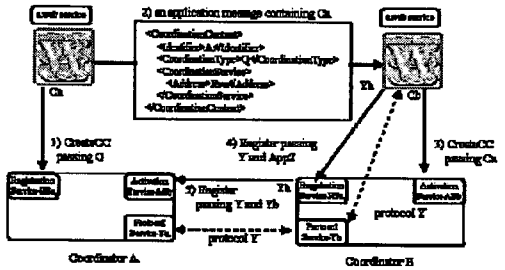
[그림3] 웹서비스 프로세스를 기술하는 BPEL

2.2. WS-Coordination

WS-C는 WS-T같은 조정 프로토콜(coordination protocols)을 전개(deploy)하기 위한 프레임웍(framework)이다. 물론 WS-T뿐만 아니라 다른 추가적이고 새로운 프로토콜에 대해서도 확장 가능하다 [2]. 그림4는 1개의 웹서비스에 대해 WS-C 규격에 따른 조정 프레임웍을 보여준다. 단일 웹서비스에 대해 등록(registration), 활성화(activation), 프로토콜(protocol) 서비스를 가진 조정자(coordinator)가 생성된다. 그림5는 다수의 웹서비스들에 대한 조정 프레임웍을 보여준다[4].

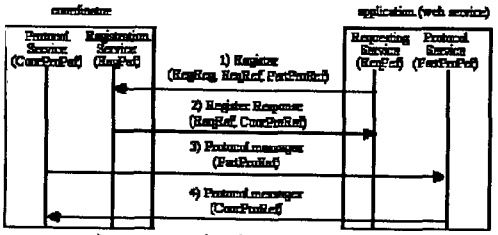


[그림4] 단일 웹서비스 조정 프레임웍 (WS-C)



[그림5] 다수 웹서비스 조정 프레임웍 (WS-C)

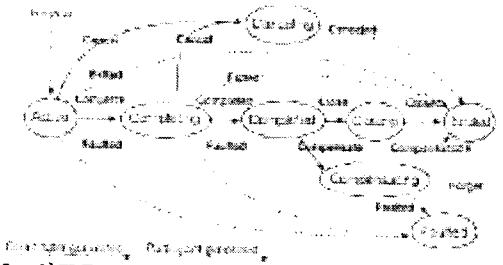
그림6은 조정 프레임웍을 구성하는 흐름을 설명한다. 조정자와 웹서비스가 등록(register), 등록응답(register response)의 메시지를 교환한 후 프로토콜 메시지를 상호 교환하는 순서로 조정 프레임웍이 구축된다 [4].



[그림6] WS-C에 따른 조정 흐름(flow)

2.3. WS-Transaction

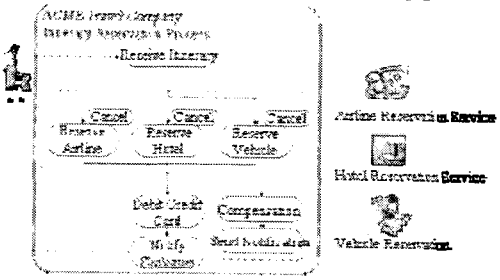
WS-T는 웹서비스의 상태(states)를 기술하는 조정 프로토콜이다. WS-T를 통해 2단계실행(2 phase commitment)를 요구하는 단기 트랜잭션(short-lived transaction) 뿐만 아니라 신뢰기반관계(trust based relationship)인 장기 트랜잭션(long-lived transaction)을 관리한다 [2]. 다음의 그림7은 WS-T에 따른 조정 프로토콜들 중 하나인 BusinessAgreementWithComplete 프로토콜을 보여주고 있다 [5].



[그림7] BusinessAgreementWithComplete 프로토콜

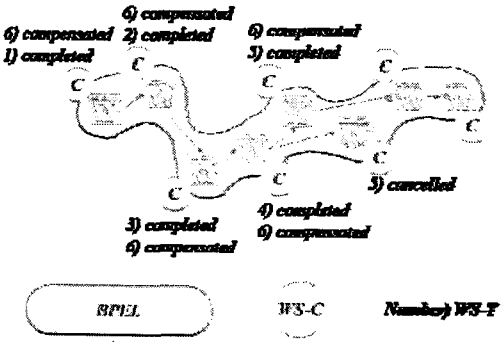
2.4 The relationship among 3 specifications

BPEL과 WS-C, WS-T를 이용하여 포괄적인 비즈니스 프로세스 자동화 프레임웍을 구축할 수 있다. 그림8은 ACME Travel 여행사의 예를 보여주고 있다. ACME Travel사는 여행패키지에 따른 고객의 주문을 받아 각각의 스케줄에 맞게 차량, 호텔, 항공편을 예약한 후 신용카드로 결제하고 이를 고객에게 알리는 비즈니스 프로세스를 구축하여 새로운 비즈니스 기회를 얻으려고 한다. 물론 차량, 호텔, 항공편 예약은 각각 다른 비즈니스 파트너들로부터 웹서비스로 노출되었다고 가정하자. 하지만 이런 비즈니스 프로세스를 BPEL로 기술하는 것만으로는 완벽하지 않다. 만약 항공사는 예약을 하였지만 호텔을 예약하지 못했다면 이 여행패키지는 판매할 수 없다. 이 경우 이미 예약한 항공사의 예약은 원상복귀시키고 고객에게 주문을 수행하지 못했다고 알려주는 과정이 필요하다 [6].



[그림 8] ACME Travel의 사례

BPEL과 WS-C, WS-T는 이러한 원상복귀 프로세스를 지원하여 상태기반 비즈니스 프로세스 트랜잭션 관리와 자동화를 지원하는 규격이다. 그림9는 BPEL과 WS-C, WS-T의 관계를 간단명료하게 보여준다. BPEL은 웹서비스들의 복잡한 흐름을 기술하고 WS-C는 각각의 웹서비스들에 대한 조정자를 달아 조정 프레임웍을 완성한다. 그리고 WS-T에 따라 번호순서대로 임시완료 (completed)되다가 중간에 외부 요인에 의해 중단되어 더 이상 전체 비즈니스를 수행할 수 없을 때 수행되었던 단위 비즈니스들을 원상복귀 시키는 과정을 보여준다.



[그림9] BPEL, WS-C, WS-T의 관계

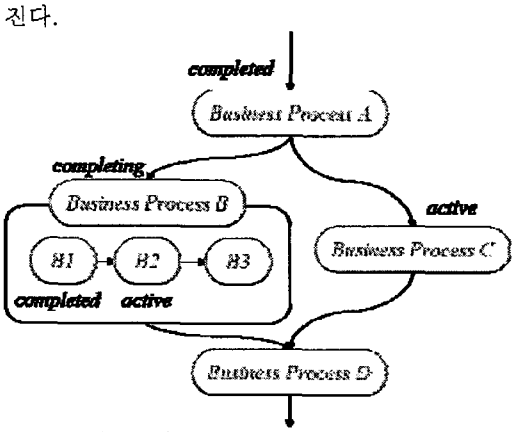
3. State-driven business process transaction management

제안된 상태기반 비즈니스 프로세스 트랜잭션 관리에서는 WS-T 규격내의 프로토콜에 대한 수정안과 WS-C 구조의 단순화 방안을 포함하고 있다. 3.1절과 3.2절에서는 이들을 설명하고, 3.3절에서는 새로 제안된 조정 프레임웍 내에서 병렬 또는 직렬로 서술된 비즈니스 프로세스들의 상태에 따른 트랜잭션 관리 알고리즘에 대해 설명한다. 그리고 3.4절에서는 병렬, 직렬 프로세스들을

통합하여 트랜잭션을 관리하는 방법론을 기술한다.

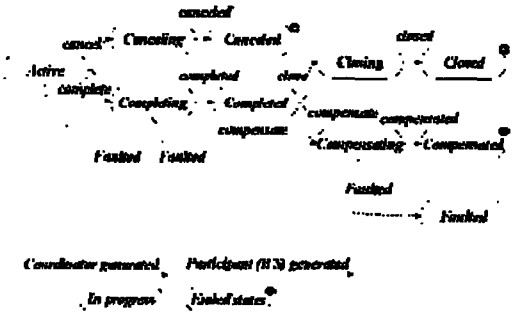
3.1 Modification of WS-T

그림 7에 나타난 WS-T 규격에 따른 BusinessAgreementWithComplete 프로토콜은 상태의 서술에 문제점들이 있다. 첫 번째 문제는 작업 수행 중(completing)인 상태에서 취소 중(canceling) 상태로 바뀐다는 것이다. BPEL로 기술된 비즈니스 프로세스는 최종적으로 다시 웹서비스로 노출된다. 만약 completing 상태에 있는 참여 웹서비스(participant)가 BPEL로 기술되어 여러 하위 웹서비스들로 구성된 경우, 하위 웹서비스의 어느 정도까지 수행되었는지를 알 수 없다. 이 때문에 completing 상태의 웹서비스는 canceling 상태가 아니라 원상복귀 명령을 받아 원상복귀 중(compensating) 상태로 바뀌어야 한다. 다음의 그림10은 하부 웹서비스의 통합으로 이루어진 웹서비스가 completing에서 compensating으로 가야 되는 이유를 보여준다. 상태가 completing인 비즈니스 프로세스 B의 내부에는 하위 비즈니스 프로세스 B1이 이미 수행되었으므로 이를 원상복귀하기 위해, 상위 비즈니스 프로세스 B에서 compensating의 상태로 바뀌어야 한다. 그렇지 않고 기존의 WS-T 규격에 따라 canceling 상태로 바뀌게 되면 취소(cancel)을 위한 조치에 원상복귀 알고리즘을 다시 한 번 구현해야 하므로 효율성이 떨어지고, 시스템이 매우 복잡해 진다.



[그림10] 하부구조를 가진 웹서비스의 completing에서 compensating로 상태 변화 이유

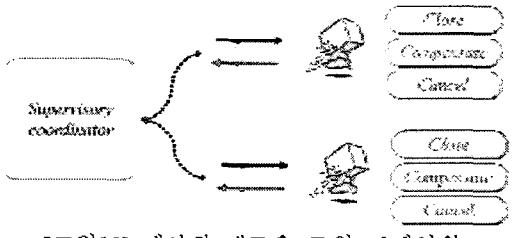
두 번째 문제는 항상 상태도(state diagram)가 ended로 끝난다는데 있다. 이 문제는 BusinessAgreement 프로토콜에도 동일하게 존재한다. 이러한 종료상태(ended state)로의 전이는 상태도를 복잡하게 하여 이해하기 힘들게 만든다. 따라서 이들을 분리하여 관리할 필요가 있다. canceled와 compensated, closed 3개의 상태들을 종료상태들로 간주하면 상태도가 매우 간단해 진다. 뿐만 아니라 최종 웹서비스의 상태들을 저장함으로써 CRM (Customer Relationship Management)분석 등에 활용할 수 있는 기초자료로 쓸 수 있다는 부가적인 장점까지 생긴다. 그림 11은 최종적으로 개선된 WS-T 프로토콜을 보여준다.



[그림11] 수정된 BusinessAgreementWithComplete 프로토콜

3.2 Simplification of WS-C

WS-C는 웹서비스마다 조정자를 부여하는 것을 그 내용으로 하고 있다. 하지만 이 경우 BPM을 위해서 복잡한 원상복귀 알고리즘을 조정자들마다 가지고 있어야 한다. 뿐만 아니라 이러한 알고리즘은 BPEL에서 정의된 비즈니스 프로세스 흐름에 따라 달라진다. 따라서 조정자의 재사용성이 떨어지고 BPEL에서 정의할 수 있는 모든 경우의 수에 따라 알고리즘을 부여해야 하므로 비현실적인 규격이라 하겠다. 이에 각각의 웹서비스에 대해 조정자를 만들어 부여하는 대신, 각각의 웹서비스에 close, compensate, cancel의 3C 메소드(method)를 추가함으로써 조정 프레임웍을 단순화 할 수 있다. 이러한 프레임웍을 갖추고 난 뒤에 상태에 따라 3C 메소드를 호출할 상위 조정자(supervisory coordinator)를 구축한다. 다음의 그림12에는 제안된 새로운 조정 프레임웍을 보여주고 있다.



[그림12] 제안된 새로운 조정 프레임웍

이 후의 3.2.1, 3.2.2, 3.2.3 절에서는 새로 제안된 조정 프레임웍에서 구현될 close, compensate, cancel 메소드에서 어떠한 알고리즘이 구현되어야 하는지 설명하고 있다.

3.2.1 Close method

Close 메소드는 BPEL에서 서술된 모든 웹서비스의 상태가 정상적 완료(completed)일 때 최상위조정자가 모든 웹서비스들의 상태를 closing로 바꿈과 동시에 호출하는 메소드이다. 따라서 해야 할 비즈니스 프로세스를 완료하는 알고리즘이 구현되면 된다. 예를 들어 예약을 하는 웹서비스라면 예약 대기상태에 있다가 예약을 확정하는 알고리즘이 구현되면 된다.

3.2.2 Compensate method

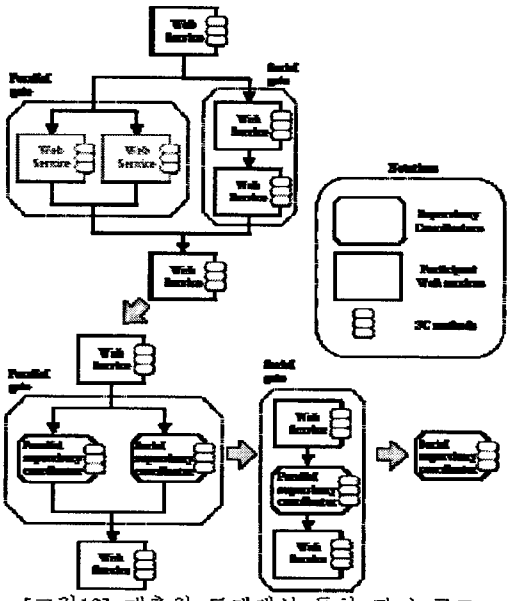
개별 웹서비스는 정상적으로 업무처리를 할 수 있어 completed나 completing상태에 있는 비즈니스 프로세스이지만 외부요인에 의해 BPEL에 기술된 모든 웹서비스들을 원상복귀해야 하는 경우 compensate 메소드가 호출된다. 여기에는 이미 취해진 비즈니스 프로세스를 원상복귀하는 알고리즘을 구현해 넣어야 한다. 예를 들어 예약 대기상태에 있던 웹서비스의 경우 예약을 취소하는 알고리즘이 들어가야 할 것이다. 만약 앞의 그림10에서와 같이 하위 프로세스를 내포하고 있는 상위 조정자인 경우에는 하위 프로세스들이 complete나 completing 상태라면 compensate 메소드를 호출하고, active라면 cancel 메소드를 호출하는 알고리즘이 구현되어야 한다.

3.2.3 Cancel method

Cancel 메소드는 active화 했지만 아직 참여 웹서비스로의 비즈니스 프로세스를 수행하지 않았을 경우에 호출하는 메소드이다. 따라서 여기에는 특별한 알고리즘이 구현되지 않고 일을 시작하지 않았으니 잘 취소되었다는 확인 메시지를 콜백(callback)해 주는 알고리즘만을 작성하면 된다. 기존의 WS-T 규격에 따르게 되면 이 cancel 메소드 내에 때에 따라 compensate 메소드와 동일한 알고리즘을 중복해 구현해야 하는 문제점이 있었으나 제안된 새로운 구조(architecture)에서는 앞서 언급한 간단한 알고리즘 만을 구현하면 된다.

3.3 Business process transaction algorithms

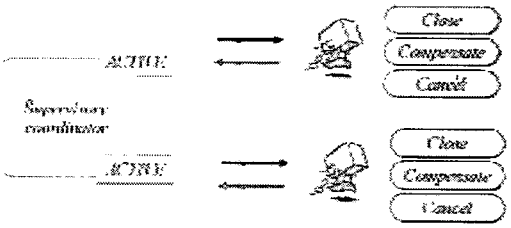
비즈니스 프로세스들의 통합은 병렬, 직렬 프로세스 등 여러 가지 형태의 혼합으로 이루어 진다. 제안된 방법론에서는 이러한 병렬, 직렬의 비즈니스 프로세스들을 parallel gate(AND), serial gate등으로 통합하여 새로운 상위조정자에 의해 트랜잭션 관리를 받는 시스템을 제안한다. 그림13에서는 이러한 계층적 트랜잭션 통합구조를 보여주고 있다. 다음의 3.3.1, 3.3.2절에서는 parallel gate(AND)와 serial gate에 대한 트랜잭션 알고리즘을 제시하고, 이어지는 3.4절에서는 parallel gate(AND)와 serial gate를 계층적으로 통합하여 관리하는 알고리즘에 대해 언급한다.



[그림13] 계층적 트랜잭션 통합 관리 구조

3.3.1 Parallel gate (AND)

Parallel gate는 그림 8의 ACME Travel 사의 차량, 호텔, 항공편 예약 과정과 같이 동시에 그리고 모두 반드시 일어나야 하는 비즈니스 프로세스들을 말한다. 만약 그 중 하나라도 수행이 불가능하면 이미 수행된 예약을 다시 원상복귀 시키는 과정이 필요하다. 이러한 과정은 그림 14와 같은 병렬 상위 조정자(parallel supervisory coordinator)를 두어 트랜잭션을 처리한다.



[그림14] Parallel supervisory coordinator (AND)

(i) 정상적으로 모두 예약 되고 최종 사용자에게 확인을 보내는 시나리오

1) 제일 처음 주문이 들어오면, 상위 조정자에 각각의 웹서비스들의 상태를 "ACTIVE"로 저장한다.
2) 상위 조정자는 각각의 웹서비스에 대해 예약 관련 메소드를 호출하고 상태를 "COMPLETING"으로 둔다.
3) 하나의 예약 웹서비스가 예약과정을 모두 마치면 상위 조정자에 콜백을 돌려준다.
4) 콜백을 받음과 동시에 해당 웹서비스의 상태를 "COMPLETED"로 바꾼다.
5) 전체 프로세스에 참가한 모든 웹서비스의 상태가 "COMPLETED"가 되면 전체 웹서비스에 close 메소드를 호출하고 상태를 동시에 "CLOSING"로 바꾼다.
6) 각각의 웹서비스로부터 close에 대한 콜백이 돌아오면 해당 웹서비스의 상태를 "CLOSED"로 바꾼다.
7) 전체 웹서비스의 상태가 모두 "CLOSED"가 되면 주문을 넣은 최종 사용자에게 확인을 보낸다.

(ii) 일을 처리하는 도중 최종 사용자의 주문을 취소하는 시나리오

1) 제일 처음 주문이 들어오면, 상위 조정자에 각각의 웹서비스의 상태를 "ACTIVE"로 저장한다.
2) 상위 조정자는 준비된 웹서비스에 대해 예약 관련 메소드를 호출하고 상태를 "COMPLETING"으로 둔다. 이때 각각의 웹서비스는 서로 다른 비즈니스 파트너들에 의해 제공되므로 메소드 호출에 걸리는 시간이 다를 수 있다.
3) 상위 조정자에 주문에 대한 원상복귀 명령이 들어오다.

4-1) "COMPLETING"이나 "COMPLETED" 상태에 있는 웹서비스에 대해서 "COMPENSATING"상태로 바꾸고, 해당 웹서비스의 compensate 메소드를 호출한다.

4-2) "ACTIVE" 상태에 있는 웹서비스에 대해서는 "CANCELING" 상태로 바꾸고, 해당 웹서비스의 cancel 메소드를 호출한다.

5) 4-1과 4-2에서 호출한 메소드에 따라 콜백을 받아 "COMPENSATED"나 "CANCELED"상태로 바꾼다.
6) 전체 웹서비스의 상태가 모두 "COMPENSATED"나 "CLOSED" 중 하나가 되면 주문을 넣은 최종 사용자에게 확인을 보낸다.

Parallel gate(AND)를 관리하는 상위 조정자는 재귀적으로 다시 serial gate등의 다른 종류의 상위 조정자에 통합되어 트랜잭션 관리를 위한 컴포넌트의 역할도 다해야 한다. 따라서 다음과 같이 자신의 상태를 바꾸는 일이 필요하고 자신도 또한 cancel, compensate, close의 메소드를 가지는 일도 필요하다.

1) 제일 처음 주문이 들어오면, 상위 조정자에 하위 웹서비스들의 상태를 "ACTIVE"로 저장한 후 상위 조정자 자신의 상태도 "ACTIVE"로 둔다.
2) 1의 상태에서 하위 웹서비스 중 하나라도 "COMPLETING"으로 전환되면 상위 조정자의 상태를 "COMPLETING"으로 전환한다.
3) 모든 하위 웹서비스들의 상태가 "COMPLETED"로 전환되면 상위 조정자의 상태를 "COMPLETED"으로 전환한다.
4) 1의 상태에서 하위 웹서비스 중 하나라도 "CANCELING"으로 전환되면 상위 조정자의 상태를 "CANCELING"으로 전환한다.
5) 모든 하위 웹서비스들의 상태가 "CANCELED"로 전환되면 상위 조정자의 상태를 "CANCELED"으로 전환한다.
6) "CLOSING", "CLOSED"로 전환도 4), 5)와 같은 과정으로 한다.
7) 하위 웹서비스 중 하나라도 "COMPLETING"이나 "COMPLETED"에서 "COMPENSATING"으로 전환되면 상위 조정자의 상태를 "COMPENSATING"으로 전환한다.
8) 모든 하위 웹서비스들의 상태가 "COMPENSATED"나 "CANCELED" 중 하나로 전환되면 상위 조정자의 상태를 "COMPENSATED"로 전환한다.

3.3.2 Serial gate

그림 8의 ACME Travel 사의 3가지 예약 웹서비스가 parallel gate(AND) 처리에 의해 1개의 상위 조정자로 잘 통합되었다고 가정하자. 그러면 ACME Travel사의 비즈니스 프로세스는 여행스케줄접수, 예약, 신용카드 결제, 알림의 4가지 웹서비스를 순차적으로 해야 하는 serial gate로 정의된다. 이러한 serial gate의 경우도 parallel gate와 마찬가지로 1개의 직렬 상위 조정자(serial supervisory coordinator)를 두어 트랜잭션을 처리한다. 하지만 비즈니스 트랜잭션 관리는 매우 단순하다.

(i) 정상적으로 모두 예약 되고 최종

사용자에게 확인을 보내는 시나리오

1) 제일 처음 주문이 들어오면 첫 번째 웹서비스의 상태만 "ACTIVE"로 저장한다.
2) 상위 조정자는 첫 번째 웹서비스에 대해 예약 관련 메소드를 호출하고 상태를 "COMPLETING"으로 둔다.
3) 첫 번째 웹서비스가 업무 과정을 모두 마치면 상위 조정자에 콜백을 돌려준다.
4) 콜백을 받음과 동시에 그 웹서비스의 상태를 "COMPLETED"로 바꾼다.
5) 다음 웹서비스의 상태를 "ACTIVE"로 저장하고 2)~4)의 일을 마지막 웹서비스까지 반복한다.
6) 전체 프로세스에 참가한 모든 웹서비스의 상태가 "COMPLETED"가 되면 전체 웹서비스의 상태를 동시에 "CLOSED"로 바꾼다.

(ii) 일을 처리하는 도중 최종 사용자의 주문을 취소하는 시나리오

1) (i)번 시나리오의 1)~5)의 과정을 수행한다.
2) 업무 수행 중에 상위 조정자에 주문에 대한 원상복귀 명령이 들어오다.
3-1) "COMPLETING"이나 "COMPLETED" 상태에 있는 웹서비스에 대해서 "COMPLETING"상태로 바꾸고, 해당 웹서비스의 compensate 메소드를 호출한다.
3-2) "ACTIVE" 상태에 있는 웹서비스에 대해서는 "CANCELED" 상태로 바꾸고, 해당 웹서비스의 cancel 메소드를 호출한다.
4) "COMPENSATING" 상태에 있는 웹서비스가 compensate 메소드로 작업을 모두 끝내면 상위 조정자에 콜백을 준다. 이를 받은 후 해당 웹서비스의 상태를 "COMPENSATED"로 바꾼다.

3.4 Integration of parallel and serial gates

실제의 업무는 그림8의 ACME Travel 사와 같이 parallel gate와 serial gate들이 복잡하게 혼재 되어 있는 형태가 일반적이다. 따라서 이러한 경우의 처리는 그림 13에서 보여진 바와 같이 최하위 단계의 parallel, serial gate를 1개의 상위 조정자로 통합하고 통합된 비즈니스 프로세스를 1개 단위로 보고 다시 serial, parallel gate로 처리하는 방법으로 해결이 가능하다. 그리고 이 때 유의할 점은 상위 조정자가 관리하는 해당 gate내의 웹서비스가 모두 "COMPLETED" 상태라고 해서 바로 gate내의 웹서비스들의 close 메소드를 호출하고 상태를 "CLOSED"로 바꾸면 안 된다는 것이다. 즉, BPEL이 정의하는 모든 웹서비스의 상태가 "COMPLETED" 상태가 되었음을 확인한 후에, 전체의 웹서비스의 상태를 "CLOSED"로 바꾸면서 close 메소드를 호출해야 한다는 것이다. 또 한가지의 주목할만한 사실은 본 논문에서 제안한 WS-C, WS-T의 수정안에 따르면, BPEL로 정의한 비즈니스 프로세스가 정상적으로 끝난다면 전체 웹서비스의 상태가 모두 "CLOSED"가 되어야 하며, 중간에 원상복귀가 일어난 경우에는 "COMPENSATED"나 "CANCELED" 상태로 되어야 한다는 것이다. 이 "CLOSED", "COMPENSATED", "CANCELED" 상태는 이전의 WS-T에 따르면 "ENDED" 상태여서 비즈니스 프로세스의 처리가 어떻게 되었는지를 이후에 확인하기 어렵지만, 새로운 WS-T 수정안에 따르면 "CLOSED", "COMPENSATED", "CANCELED"의 상태로 마무리된다. 이러한 처리 상황은 CRM의 분석에 활용할 수 있어 이윤창출에 도움이 되며, 보다 나은 비즈니스 프로세스 트랜잭션 관리 시스템 구축 등에도 도움이 될 것이다.

4. Implementation

4.1 Environments

* Microsoft Windows 2000
* BEA WebLogic Platform 7.0
* Pentium IV 1.4GHz, Memory 256 MB

4.2 Example scenario

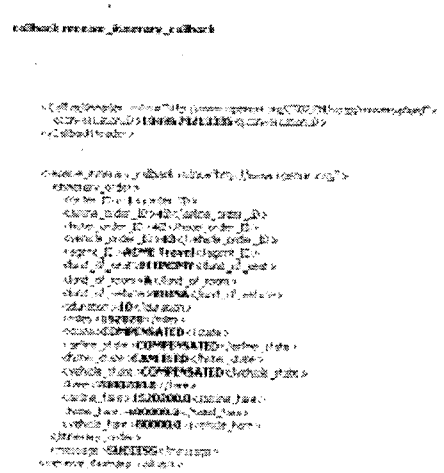
앞서 가정한 ACME Travel 사의 예에서 3개의 예약을 처리하는 부분에 대한 시나리오를 구현해 보았다. 시스템의 input으로는 고객 ID, 좌석의 종류 (항공기), 차량종류, 방의 종류(호텔), 체류기간, 인천공항으로부터의 항공이동 거리 등 이다.

4.3 Screenshots

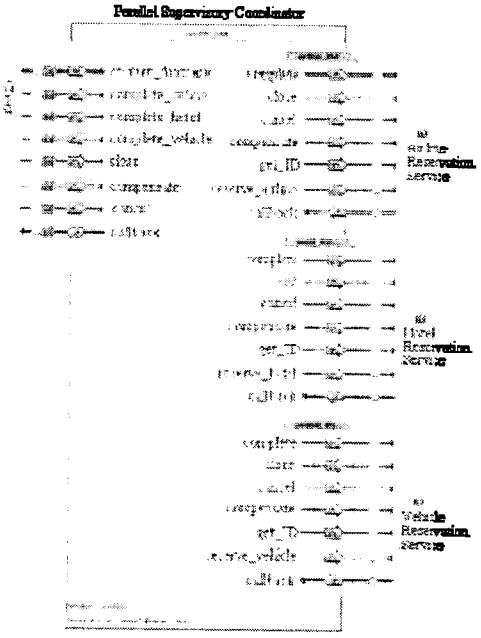
그림 15에는 compensate 메소드에 대한 콜백을 보여준다. 그리고 그림 16에는 병렬 상위 조정자의 디자인을 보여준다.

5. Discussion and Conclusion

현실에서 일어나는 비즈니스 프로세스들은 이종적인(heterogeneous) 업무들이 통합된 매우 복잡한 프로세스이다. 웹서비스 기술의 발달에 따라 이러한 비즈니스 프로세스를 효과적으로 통합하고 자동화 할 수 있는 기반환경이 마련되었다. 웹서비스를 통해 기업내의 이종적인 업무 통합뿐만 아니라 기업간, 사업자과 고객간의 이종적인 업무의 통합도 가능해 졌다.



[그림15] hotel은 canceled되고 airline, vehicle은 compensated된 후의 콜백 화면



[그림16] 병렬 상위 조정자 디자인

이를 뒷받침하기 위해 BPEL, WS-C, WS-T등의 효과적인 규격이 2002년 말부터 제정되고 있다. 이들 중 WS-C와 WS-T는 업무 수행 중 주문의 취소에 대해 이미 수행된 업무를 원상복귀 시킬 수 있는 환경을 정의하고 있다. 업무의 원상복귀는 자동화된 비즈니스 프로세스를 위해서 비즈니스 파트너 서로 간의 믿을 만한 업무 처리를 위해 반드시 필요한 항목이다. 하지만 현재 이러한 규격을 실제 구현한 시스템은 거의 없다. 이는 아직 규격들이 마련된 지가 얼마 되지 않아 WS-C와 WS-T의 규격이 아직 완벽하지 않기 때문이다.

본 논문에서는 구현 과정에서 드러난 여러 가지 단점들을 보완한 새로운 상태기반 비즈니스 프로세스 트랜잭션 관리 구조를 제시하였다. 그리고 제안된 상위 조정자들이 비즈니스 프로세스의 parallel, serial gate에서 처리해야 할 비즈니스 알고리즘을 언급하였다. 기존의 WS-T 규격의 수정과 WS-C 규격의 단순화를 통해 보다 쉽게 비즈니스 프로세스 트랜잭션 관리 시스템을 구현할 수 있도록 하였으며, WS-T 규격에 정의된 프로토콜의 종료 상태를 기존의 1개 상태에서 상황에 따른 "CLOSED", "COMPENSATED", "CANCELED"의 3가지 상태로 분화함으로써 이후 이 상태 자료들을 CRM 분석 등에 활용할 수 있도록 하였다.

이후 진행될 할 연구로는 parallel gate(OR), parallel gate(XOR) 등에 대한 비즈니스 알고리즘을 마련할 계획이며, 비즈니스 파트너 간의 협상(negotiation)을 가능하게 하는 IBM사의 conversation support 등의 연구를 바탕으로 자동화된 협상을 한 후, 비즈니스 프로세스를 진행시키는 방향으로 발전될 계획이다.

6. Reference

[1] Eric B. Parizo, "Experts decipher real impact of BPEL4WS spec", http://searchwebservices.techtarget.com/qna/0,289202,sid26_gci850320,00.html, 11 Sep 2002.

[2] Simon Yates, "BPEL4WS: Another big step for Web service standards", http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci849630,00.html, 09 Sep 2002.

[3] "Business Process Execution Language for Web Services, Version 1.0", http://www-106.ibm.com/developerworks/library/ws-bpel/, 31 July 2002.

[4] "Web Services Coordination (WS-Coordination)", http://www-106.ibm.com/developerworks/library/ws-coor/, 9 August 2002.

[5] "Web Services Transaction (WS-Transaction)", http://www-106.ibm.com/developerworks/library/ws-transpec/, 9 August 2002.

[6] James Snell, "Automating business processes and transactions in Web services", http://www-106.ibm.com/developerworks/webservices/library/ws-autobp/?loc=dwmain, August 2002.