협력 트랜잭셔널 워크플로우에서 데이타 일관성을 고려한 철회 종속성 감지 및 관리

(Rollback Dependency Detection and Management with Data Consistency in Collaborative Transactional Workflows)

변 창 우 † 박 석 ††
(Chang-Woo Byun) (Seog Park)

요 약 데이타를 공유하며 협력적인 공동 작업을 필요로 하는 기업들간의 중요한 일을 수행하는 업무 연계와 같은 분야에서는 네트워크 인프라의 부족과 기업간의 정보 교환을 위한 표준의 부재, 서로 다른 업무들이 데이타베이스 관리 시스템에 접근하여 공유 데이타를 충돌 모드로 이용하는 경우의 데이타 일관성 유지 문제 등 많은 제약 요소에 의해 워크플로우를 적용하지 못하고 있다. 특히, 공유 데이타를 사용하고 있는 상황에서 발생될 수 있는 문제들에 대해서는 언급되어지지 않고 있다.

본 논문에서는 업무 연계 관점에서 나타나는 공유 데이타를 사용하며 병행 수행되는 협력 트랜잭셔널 워크플로우들 사이에서 임의의 워크플로우와 다른 워크플로우들의 단계들 사이에서 발생될 수 있는 철회 종속성을 하나의 워크플로우에서의 암시적 철회 종속성, 협력 워크플로우 사이에서의 암시적 철회 종속성, 협력 워크플로우 사이에서의 명시적 철회 종속성 이렇게 세 가지로 분류하고, 데이타 일관성을 고려한 세 가지 철회 종속성을 워크플로우 스키마에 명시적으로 표현하는 자동 생성 알고리즘을 제안한다. 알고리즘을 실행하여 생성된 워크플로우 스키마는 데이타 공유와 관련하여 발생되는 런타임 철회 종속성을 데이타 일관성을 유지하며 런타임에 해결할 수 있도록 함으로써 워크플로우의 정확성을 향상시키게 된다.

키워드 : 협력 트랜잭셔널 워크플로우, 데이타 일관성, 암시적 철회 종속성, 명시적 철회 종속성

Abstract Workflow is not appropriately applied to coordinated execution of applications(steps) that comprise business process such as a collaborative series of tasks because of the lacks of network infra, standard of information exchange and data consistency management with conflict mode of shared data. Particularly we have not mentioned the problem which can be occurred by shared data with conflict mode.

In this paper, to handle data consistency in the process of rollback for failure handling or recovery policy, we have classified rollback dependency into three types such as implicit rollback dependency in a transactional workflow, implicit rollback dependency in collaborative transactional workflows and explicit rollback dependency in collaborative transactional workflows. Also, we have proposed the rollback dependency compiler that determines above three types of rollback dependency. A workflow designer specifies the workflow schema and the resources accessed by the steps from a global database of resources. The rollback dependency compiler generates the enhanced workflow schema with the rollback dependency specification. The run-time system interprets this specification and executes the rollback policy with data consistency if failure of steps is occurred. After all, this paper can offer better correctness and performance than state-of-the-art WFMSs.

Key words : collaborative transactional workflow, data consistency, implicit rollback dependency, explicit rollback dependency

† 학생회원 : 서강대학교 컴퓨터학과
chang@dblab.sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
spark@dblab.sogang.ac.kr
논문접수 : 2000년 12월 29일
심사완료 : 2002년 11월 27일

1. 서 론

전통적인 워크플로우 관리 시스템은 워크플로우들의 모델링, 수행, 모니터링을 지원하는 소프트웨어 시스템으로 워크플로우를 구성하는 조직 계층 구조에 적합한 단계들의 설정 및 데이타 흐름과 제어 흐름에 대한 정확한 감지 및 관리가 주요 목적이었다[1, 2, 3, 4, 5].

그런데, 현재의 프로토타입이나 상업적인 워크플로우 관리 시스템은 개별적인 워크플로우의 단계들 사이의 실행 순서의 종속성은 지원하고 있지만, 공유 데이타를 사용하고 있는 상황에서의 장애 극복(failure handling)이나 회복 정책(recovery policy)은 수작업에 의존하고 있다. 특히, 업무 연계 관점에서 나타나는 병행 수행되는 워크플로우들 사이에서 장애 극복을 위해 수행되는 철회 정책에서의 데이타 종속성에 대한 무관심은 데이타 비일관성 문제를 초래한다[6, 7, 8, 9, 10].

[문제점] 같은 데이타를 접근하며 협력(병행 수행하는) 트랜잭셔널 워크플로우들 사이에서 발생되는 장애를 극복하기 위해 수행되는 철회 정책에 있어서 공유 데이타를 고려하지 않은 철회 제어성은 데이타 비일관성을 초래한다.

그림 1은 공유 데이타를 사용하는 워크플로우들의 협력작업에 대한 예이다. 고객으로부터 물품 주문을 받아 배송업체와의 협업을 통해 물품을 생산하고 고객에게 전달하는 물품주문 워크플로우와 생산관리 업무를 위한 생산관리 워크플로우 그리고 재고관리 업무를 위한 재고관리 워크플로우가 서로 협력적인 관계를 갖고 있다. 또한, 이들의 업무는 물품 예약 DB와 생산 DB, 재고 DB와 같은 공유 데이타를 이용하여 진행되고 있으며, 워크플로우 설계자에 의해 이들 공유 데이타의 충돌 모드는 통제된다.

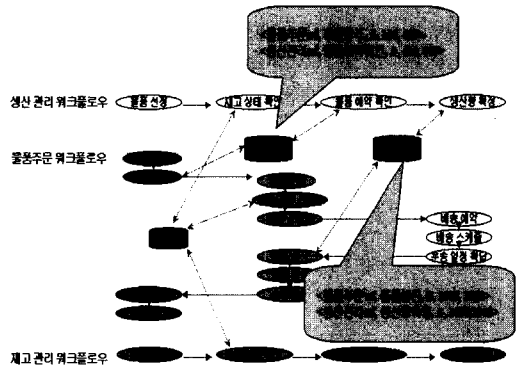


그림 1 데이타 일관성을 고려하지 않은 철회 정책의 문제

만약 생산관리 워크플로우의 물품 예약 확인 단계와 물품 주문 워크플로우의 주문 발주 단계는 서로 물품 예약 DB를 공유하고 있어 서로에게 영향을 주고 있는데, 주문 발주 단계에서 물품에 대한 예약 정보를 추가하고, 이를 물품 예약 확인 단계에서 추가사항을 확인한 상황에서 물품 주문이 취소가 되었을 때, 물품 예약 DB는 원래 상태로 돌아와야 할 것인데, 기존 연구에 따르면 주문 발주 단계부터 철회가 일어나 보상정책을 통해 데이타를 수정할 것이고, 그 후 물품 예약 확인 단계 역시 보상 정책으로 보상되겠지만, 데이타의 변경은 주문 발주가 발생한 후의 데이타로 남아 있어, 초기의 데이타와는 다른 데이타 비일관성을 초래하게 된다. 생산량 확정 단계와 물품 생산 단계에서도 같은 상황이 발생된다.

이와 같이 데이타 일관성을 고려한 철회 종속성에 대한 문제에 대해 기존 워크플로우에서는 고려하지 않았고, 고려한다 하더라도 런타임 때 수작업에 의존하기 때문에 오류 가능성이 있고, 워크플로우의 진행을 지연시키게 된다. 특히, 기존 워크플로우 관리 시스템에 대한 연구에서는 런타임 시에 어느 워크플로우 인스턴스와 다른 워크플로우 인스턴스가 상호 제어 흐름의 연결을 갖는지에 대한 통제를 하지 않는다. 이를 보완하고자 워크플로우 프로세스 모델링 구축단계에서 데이타 흐름 종속과 제어 흐름 종속을 기록한 워크플로우 스키마 정보를 워크플로우 데이타베이스에 저장하여 이들 정보를 이용하여 발생될 수 있는 문제점들을 미리 예방하여 워크플로우의 정확성을 향상시키게 된다.

따라서, 본 논문은 협력 트랜잭셔널 워크플로우를 정의하고 기존 워크플로우 시스템에서 요구하는 종속성과 관련한 세 가지 요구사항에 데이타 공유와 관련하여 데이타 일관성을 고려한 철회 종속성 요구사항을 추가한다. 이를 기반으로 발생될 수 있는 런타임 철회 종속성에 대해 데이타 일관성을 유지하며 런타임에 해결할 수 있도록 워크플로우 프로세스 모델링 구현시 워크플로우 스키마에 데이타 일관성을 고려한 철회 종속성에 대한 명세를 자동으로 생성하여 이들 정보를 워크플로우 데이타베이스에 저장, 관리함으로써 데이타 일관성을 고려한 워크플로우 관리시스템의 능력을 향상시키는데 초점을 둔다.

논문의 나머지는 다음과 같이 구성되어 있다.

2장에서는 워크플로우 관리 시스템에 대한 개요와 본 논문과 관련된 기존 연구에 대해 언급하고, 협력 트랜잭셔널 워크플로우 환경에 대해서 정의하며, 3장에서는 협력 트랜잭셔널 워크플로우에서 고려되는 정확성 요구사항들을 알아본다. 4장은 협력 트랜잭셔널 워크플로우 사이에서 단계들간의 데이타 일관성을 고려한 철회 종속성

관리에 대해서 설명하며 5장에서 결론을 맺는다.

2. 협력 트랜잭셔널 워크플로우 환경의 정의

이번 장에서는 워크플로우 관리 시스템에 대한 간단한 개요 설명과 본 논문의 초점인 협력 트랜잭셔널 워크플로우에 대한 정의 및 특성을 설명하며, 협력 트랜잭셔널 워크플로우 환경에서 발생되는 트랜잭션에 대한 요구사항을 전통적인 트랜잭션의 ACID 기준과 비교하여 기술한다. 또한, 본 논문에서 이용되는 워크플로우 스키마를 정의하는 언어인 LAWS(LAnguage for Workflow Specification)에 대해서 설명한다.

2.1 워크플로우 관리 시스템 개요 및 관련 연구

협력 트랜잭셔널 워크플로우의 정의를 내리기 전에 먼저 일반적인 워크플로우 시스템의 개요에 대해 설명한다. 워크플로우 시스템(WFS)은 과업을 수행하는 단계(step, 활동이라고도 함)들과 그들을 연결한 비즈니스 프로세스를 말한다. 한편, 워크플로우 관리 시스템은 다수 조직의 복잡한 비즈니스 프로세스들간에 협력(cooperation)과 대등관계(coordination)를 지원하고 제어하고 관리하는 소프트웨어 시스템으로 프로세스 모델링, 개시된 프로세스의 원활한 진행과 요구되는 활동들간의 긴밀한 관계에 대한 제어, 수행에 대한 모니터링을 지원한다[1, 2, 3, 7].

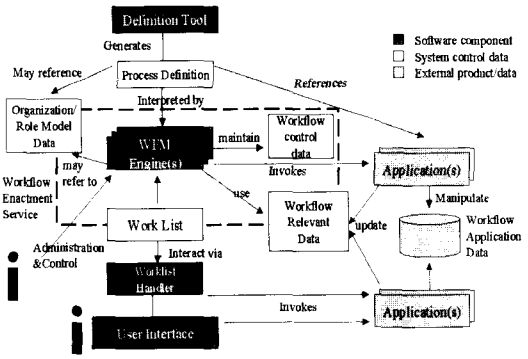


그림 2 WfMC의 워크플로우 관리 시스템 표준 모델

그림 2는 WfMC(Workflow Management Coalition)에서 제시한 워크플로우 관리 시스템으로 소프트웨어 컴포넌트(프로세스 정의 툴, 워크플로우 관리 엔진, 워크리스트 핸들러, 참여자 인터페이스)와 시스템 통제 데이타(하나 이상의 소프트웨어 컴포넌트에 의해 사용되며 프로세스 정의, 조직/규칙 모델 데이타, 워크플로우 통제 데이타, 워크리스트, 워크플로우 관련 데이타), 그리고 외부적인 요소 (응용과 응용 데이타베이스)로 구성되어 있다.

구축 단계(Build-time)에 비즈니스 프로세스를 분석하여 모델링하고 각 단계에 대한 정의가 이루어지고, 런타임때 인스턴스화되어 워크플로우 엔진에서는 프로세스 및 프로세스를 이루는 각 단계들의 상호 작용과 종속성을 유지시키며 흐름을 제어하게 된다. 따라서, 구축 단계에서 워크플로우 수행에서 발생될 수 있는 문제점들을 최대한으로 찾고 이에 대한 보상 혹은 회복 정책을 워크플로우 스키마에 반영하여 런타임 시에 발생하는 문제를 런타임에 수정하여 프로세스 자동화를 원활하게 진행시킬 수 있는 잇점을 얻고, 워크플로우의 정확성을 향상시키는 연구가 많이 진행되고 있다.

[7, 8, 10, 11]에서는 단계들 간의 제어 흐름, 데이타 흐름 관점에서 장애 극복에 대한 해결책을 제시했고, [9]에서는 제어 흐름, 데이타 흐름뿐만 아니라 데이타를 공유하는 단계들 간의 문제를 데이타 종속성이라는 용어로 정의하여 데이타 종속성과 관련된 장애 극복에 대한 해결책을 제안하고 있다.

그러나, 각 논문에서 제시한 프로토타입이나 상업적인 워크플로우 관리 시스템은 단계들 사이의 실행 순서의 종속성을 지원하고는 있지만, 업무 연계 관점에서 나타나는 병행 수행되는 워크플로우들 사이에서 장애 극복을 위해 수행되는 데이타 일관성을 유지시키는 철회 종속성에 대해서는 연구가 진행되고 있지 않다.

2.2 협력 트랜잭셔널 워크플로우의 정의

워크플로우 스키마가 미리 잘 정의되어 있고 핵심 프로세스를 상시적으로 반복 수행하는 워크플로우를 생산적 워크플로우(Production Workflow)라 한다[7, 9, 12]. 이와 같은 워크플로우는 정보의 정확성 유지에 초점을 둔다. 추가로, 프로세스 처리를 트랜잭션 처리 관점에서 이해하고 트랜잭션이 요구되는 업무 프로세스를 지원하는 워크플로우를 트랜잭셔널 워크플로우(transactional workflow)라 한다[10, 11, 12].

그림 3과 같이 트랜잭셔널 워크플로우 기반의 비즈니스 프로세스는 이질성, 분산성, 자치성을 관리하기 위해

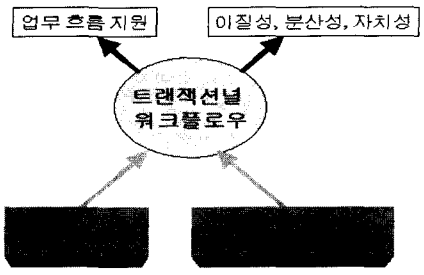


그림 3 트랜잭셔널 워크플로우

전통적인 트랜잭션에서 제시되는 트랜잭션 관리 기법과 실패 복구 기능을 이용한다[10, 11, 12].

본 논문은 다수의 참여자간의 협업, 의사결정 및 공동작업이 원활하게 이루어질 필요가 있는 기업의 핵심 프로세스를 표현하는 생산적 워크플로우들 사이에서 공유 데이타에 대한 트랜잭션 처리의 기능을 적용시킨 워크플로우를 기준으로 하고 있으며 이를 협력 트랜잭셔널 워크플로우라 한다.

표 1 협력 트랜잭셔널 워크플로우에서의 완화된 ACID 기준

원자성 (atomicity)	적절한 진행을 보장하는 완화된 원자성, no all or nothing (forward recovery, compensation, 2PC)
일관성 (consistency)	트랜잭션 수행 후의 새로운 데이타베이스 상태가 일관성 제약조건을 만족 (data integrity, partial rollback)
고립성 (isolation)	병행수행의 결과는 독립적인 수행의 결과들과 일치
영속성 (durability)	각 단계의 최종 완료(commit) 상태는 실패가 발생하여 회복되었을 때 반영 (forward recovery, compensation, 2PC)

표 1은 협력 트랜잭셔널 워크플로우에서 발생되는 트랜잭션의 특징을 보여주고 있는데, 전진 회복 정책(forward recovery policy), 부분적인 철회 정책(partial rollback policy), 보상 정책(compensation policy) 등을 사용하여 실패에 대한 원자성(all or nothing)이 아닌 실패를 복구하여 최종 기간까지 완료할 수 있게 하는 기존의 ACID 기준을 완화한 ACID 기준을 나타내고 있다.

2.3 LAWS(LAnguage for Workflow Specification)

워크플로우의 실행 요구사항을 표현하기 위해 CREW 프로젝트에서는 LAWS라는 워크플로우 스키마 정의 언어를 개발했다[9]. 이 언어로 전통적인 워크플로우 스키마를 기술하는 기능 외에 추가로, 데이타 종속성 컴파일러(data dependency compiler)에 의해 다른 장애 극복 및 협력적인 대등관계의 실행 요구사항을 추출하여 표현하고 있다. 그림 4는 LAWS를 이용한 워크플로우 스키마로서 총 아홉 가지의 명세(specification)를 기술하고 있는데 이들을 간단히 설명하면 다음과 같다.

① 워크플로우 명세(Workflow Specification)

WF1 : I1 I2 I3 I4 I5 : O1 O2;

워크플로우 WF1은 입력 데이타로 I1, I2, I3, I4, I5를 받고, 출력되는 데이타는 O1, O2라는 것을 표현하고 있다.

```
WF-SCHEMA-NAME;
WF1 : I11 I12 I13 I14 I15 : O11 O12;
STEP-INFO ;
S11 : eternity : sum comp1 : I11 I12 I13 I14 : O11;
S12 : era : prod comp2 : I11 I12 I19 : O11;
...
DATA-FLOW;
S11.I1 = WF1.I1;
...
S23.I1 = S12.O13;
S23.I2 = S22.O23;
...
CONTROL-FLOW;
S11: S12;
...
S12: S23;
...
S22: S23;
...
STEP-FAILURE-SPEC;
S11: RE-START S11;
...
S13: COMPENSATE S11 RE-START S11;
...
COMPENSATION-DEPENDENT-SETS;
CDS1: WF1.S12 WF2.S23;
STEP-CONFLICT-SPEC;
IF(WF1.S12 == WF2.S23) THEN WF1.S12
CONFLICTS_WITH WF2.S23;
...
COORDINATED-EXECUTION-SPEC;
ON WF1.S12 COMPENSATE
    IF(WF2.S23 CONFLICTS_WITH WF1.S12)
            AND (WF2.S23 HAPPENED_AFTER WF1.S12)
    THEN COMPENSATE WF2.S23 RE-START WF2.S23;
```

그림 4 워크플로우 스키마

② 단계 명세(Step Specification)

S3 : eternity : prod comp3 : I1 I2 I3 O1;

단계 S3는 eternity라는 응용 에이전트를 이용하고 응용 프로그램은 prod, 보상 프로그램은 comp3이며 입력데이타는 I1, I2, I3이고 출력 데이타는 O1이라는 것을 표현하고 있다.

③ 데이타 흐름 명세 (Data-Flow Specification)

S3.I1 = WF1.I1;

S3.I2 = S2.O1;

단계 S3의 입력 데이타 I1는 WF1의 입력 데이타 I1과 동일하고, S3의 입력 데이타 I2는 S2의 출력 데이타 O1이라는 것을 표현하고 있다.

④ 제어 흐름 명세 (Control-Flow Specification)

S2 : IF S2.O1 > 50 THEN S3(conditional control branching)

ELSE IF S2.O1 < 50 THEN S4

ELSE S5;

S3 : S7; (direct control flow)

단계 S2는 결과 데이타 O1이 50보다 크면 S3으로 이동하고, 50보다 작은 경우에는 S4로 이동하는 분기 프로세스를 표현할 수 있고, 직접 단계 S3의 다음 실행 단계는 S7이라는 것을 표현하고 있다.

⑤ 단계 실패 명세 (Step Failure Specification)

S7 : COMPENSATE S2 RE-START S2;

단계 S7에서 장애가 발생하면 S2로 보상하여, 재실행함을 표현하고 있다.

⑥ 단계 재수행 명세 (Step Re-Execution Specification)

```
S2 : IF (I1.NEW > I1.OLD)
        THEN  MODIFY(I1.NEW=I1.NEW - I1.OLD)
     RE_ EXECUTE
     ESLEIF (I1.OLD > I1.NEW)
       THEN   MODIFY(I1.NEW=I1.OLD-I1.NEW)
     COMPENSATE
     ELSEIF(I2.NEW   !=I2.OLD)   COMPENSATE
  RE_EXECUTE
```

단계 S2에서 재수행할 경우, 신버전의 입력 데이타가 구버전의 입력 데이타보다 큰 경우, 작은 경우, 같은 경우에 따라 다르게 수행되는 것을 표현하고 있다.

⑦ 보상 종속 집합 (Compensation Dependent Set)

CDS1 : S3 S6;

S6이 보상되면 항상 S3 역시 보상되어야 한다는 것을 표현하고 있다.

⑧ 단계 충돌 명세 (Step Conflict Specification)

```
IF (S12.I1 == WF2.S23.I2)
  THEN S12 CONFLICTS_WITH WF2.S23;
```

단계 S12의 입력 데이타 I1과 워크플로우 WF2의 단계 S23의 입력 데이타 I2가 같고 S12와 WF2의 S23이 충돌 모드를 나타낸다면 단계 S12와 WF2의 S23은 충돌 관계임을 표현하고 있다.

⑨ 협력적인 대등 관계 수행 명세(Coordinated Execution Specification)

```
IF (S12 CONFLICTS_WITH WF2.S23)
  THEN S12 MUTUALLY_EXCLUDE WF2.S23;
ON S12 COMPENSATE
  IF(S23 CONFLICTS_WITH WF1.S12)
      AND (S23 HAPPENED_AFTER WF1.S12)
  THEN  COMPENSATE  WF2.S23  RE-START
  WF2.S23;
```

단계 S12와 WF2의 S23이 충돌 관계라면 상호 배타적으로 실행하고, 만약 S12가 보상되면, 충돌 관계에 있고, 제어 흐름 종속성을 갖고 있는 WF2.S23 역시 보상되어야 함을 표현하고 있다.

이와 같이 직관적으로 파악될 수 있는 장애 극복이나 협력적인 대등 관계의 워크플로우 수행에서 찾을 수 있는 문제점들을 구축 단계인 프로세스 모델링 때 미리 정의함으로써 런타임 시에 발생하는 문제를 런타임에 수정하여 프로세스 자동화를 원활하게 진행시킬 수 있는 잇점을 얻고, 워크플로우의 정확성을 향상시키게 된다.

본 논문에서는 1장에서 문제점으로 제시했던 것처럼, 데이타 일관성을 고려한 철회 종속성의 중요성을 강조하는 입장에서 암시적 협력 트랜잭셔널 워크플로우-공유 데이타를 사용하는 경우- 뿐만 아니라 명시적 협력 트랜잭셔널 워크플로우-공유 데이타를 사용하면서 제어 흐름 종속성이 모두 있는 경우-에서 생길 수 있는 철회 종속성을 구분하는 철회 종속성 명세(rollback dependency specification)를 추가하여 문제 발생시 런타임에 수정되는 과정에서 데이타 일관성을 유지하여 더욱더 워크플로우의 정확성을 향상시키고자 한다.

3. 종속성 관점에서의 협력 트랜잭셔널 워크플로우의 정확성

협력 트랜잭션널 워크플로우의 정확한 수행을 위해서는 전통적인 트랜잭션에서 요구하는 전역 직렬성이나 실패에 대한 원자성보다는 종속성 관점에서 다른 어떤 정확성 요구사항들을 필요로 하게 된다. 3.1절에서는 그런 종속성 관점에서의 정확성 요구사항들을 분류하여 설명하고, 3.2절에서는 정의한 정확성 요구사항들에 대한 형식적인 표기를 제시한다.

3.1 네 가지의 종속성에 대한 정확성 요구사항

협력 트랜잭셔널 워크플로우는 자율적인 로컬 사이트에 있는 데이타베이스 관리시스템의 데이타에 대한 판독이나 혹은 갱신 연산을 실행하는 단계들로 구성된 장기간 수행되는 응용이기 때문에 정확성 기준은 데이타 일관성을 고려한 데이타 흐름과 제어 종속성이 유지되는가에 초점을 두게 된다. 따라서, 기존 워크플로우 연구에서 제시한 [정확성 요구사항 1~3]뿐만 아니라 [8, 9, 10], 협력 트랜잭션의 정확한 수행을 위해 필요한 요구사항인 데이타 일관성을 고려한 철회 종속성을 본 논문에 추가한다.

[정확성 요구사항 1] 제어 흐름 종속성 유지

워크플로우 실행 동안 수행되는 단계들의 순서를 말하며, 임의의 단계들의 쌍이 제어 흐름 종속성을 가졌다면 단계의 수행은 전 단계의 종료를 요구한다. 워크플로우는 제어 흐름 종속성을 유지하며 수행되어야만 정확하다.

[정확성 요구사항 2] 데이타 흐름 종속성 유지

임의의 단계들 쌍이 데이타 흐름 종속성을 가졌다면

한 단계는 다른 단계의 결과 데이타를 요구한다. 워크플로우는 데이타 흐름 종속성을 유지하며 수행되어야만 정확하다.

[정확성 요구사항 3] 데이타 종속성 유지

원격 데이타베이스 관리시스템의 데이타에 접근하는 단계들간의 접근 형태에 의해 발생되는 것으로, 충돌 모드(판독-갱신)의 단계들간에는 데이타 종속성이 존재하며 트랜잭셔널 워크플로우는 이런 데이타 종속성을 유지하며 수행되어야만 정확하다.

[정확성 요구사항 4] 데이타 일관성을 고려한 철회 종속성 유지

원격 데이타베이스 관리시스템의 데이타에 접근하는 단계들의 응용이 실패했을 경우 수행되는 철회 정책은 데이타 일관성을 고려한 철회 종속성을 유지하며 수행되어야만 정확하다.

3.2 정확성 요구사항들의 형식적인 표기

이번 절에서는 전 절에서 언급했던 정확성 요구사항들에 대해 형식적인 표기를 제시한다. [9]의 LAWS의 형식적인 표기법인 LITES(Language for Implementation TEchniques Specification)을 기반으로 정확성 요구사항 네 가지를 표현한다.

표 2와 표 3에 있는 표기 방법을 사용하여 정확성 요구사항들을 표현하면 다음과 같다.

[정의 1] 제어 흐름 종속성

$\forall(a, b), a \ll b \Rightarrow e(a) (e(b), (s(a) (f(a) (s(b) (f(b))$

[정의 2] 데이타 흐름 종속성

$\forall(a, b), dfd(a, b) \Rightarrow f(a)$ (o $s(b)$

[정의 3] 데이타 종속성

단계 a, b에 대한 충돌 조건을 C라 하자.

$\forall(a, b), sd(a, b) \equiv C(a, b) \wedge dfd(a, b)$

[정의 4] 데이타 일관성을 고려한 철회 종속성

$\forall a \in X, \forall b \in Y, drd(a, b) \Rightarrow sd(a, b) \wedge (c(b) (c(a) (t(a) (t(b))$

다음 4장에서는 협력 트랜잭셔널 워크플로우 환경에서 빈번하게 발생될 수 있는 데이타 공유에 따른 [정확성 요구사항 4]의 중요성을 인지하고, LAWS에서 제공했던 기존의 아홉 가지 명세에 철회 종속성 명세를 추가한다.

4. 협력 트랜잭셔널 워크플로우에서의 철회 종속성 관리

4.1절에서는 데이타 일관성을 고려한 철회 종속성을 하나의 워크플로우에서의 암시적 철회 종속성, 협력 트랜잭셔널 워크플로우 환경에서의 명시적 철회 종속성과 암시적 철회 종속성, 이렇게 세 가지로 구분하여 이들을 자동으로 생성하는 알고리즘을 제시하고, 여러 시나리오에 대한 정확성을 보인다. 4.2절에는 평가 분석하고, 4.3절에서는 추가 고려사항을 설명한다.

4.1 데이타 일관성을 고려한 철회 종속성 명세에 대한 자동 생성

본 논문에서 언급하고 있는 데이타 일관성을 고려한 철회 종속성 명세는 하나 이상의 워크플로우들의 대등 수행 명세(Coordinated Execution Specification)에서 고려하지 않은 사항에 대한 추가이다. 따라서, 데이타 일관성을 고려한 철회 종속성 명세 역시 2.3절에서 설명했던 LAWS 언어를 이용하며, 이런 데이타 일관성을 고려한 철회 종속성을 자동적으로 생성할 수 있는 방법에 대해서 설명한다.

4.1.1 철회 종속성 자동 생성 컴파일러의 알고리즘

충돌 모드로 데이타를 공유하는 단계들 사이에 제어 흐름 종속성이 형성되어 워크플로우 스키마에 제어 흐름

표 2 다양한 이벤트 및 연결자에 대한 표기

Event	Notation
Start of Step Execution	s(i)
Finish of Step Execution	f(i)
Failure of Step Execution	Fa(i)
Start of Step Compensation	Ss(i)
Finish of Step Compensation	Cf(i)
Abort of workflow X	a(X)

Connector Purpose	Symbol
Implication	⇒
Equivalence	≡
Happens before(at run time)	(
Precedes(in schema definition)	≪
Membership connectors	∈, ∀, ∋
Logical connectors	∧, ∨

표 3 복잡한 이벤트 및 단계들 사이의 관계에 대한 표기

Event Description	Original Notation	New Notation
Execution of Step	s(a) (f(a)	e(a)
Compensation of Step	sc(a) (fc(a)	c(a)
Attempt (try) of Step	s(a) (fa(a)	t(a)

Data based step Relationship	Notation
Data flow dependency between steps	dfd(x, y)
Step relationship contained shared data	sd(x, y)
Happens Before with output data	(o

종속성 명세뿐만 아니라 단계 충돌 명세가 포함되게 된다. 또한, 하나의 워크플로우를 구성하는 단계들 중에서 서로들에게 영향을 주는 공유 데이타를 이용하는 단계들을 묶어 표현한 보상 종속 집합(compensation dependent sets) 명세 역시 포함된다. 따라서, 이들을 이용하여 데이타 일관성을 고려한 철회 종속성을 추출할 수 있게 된다.

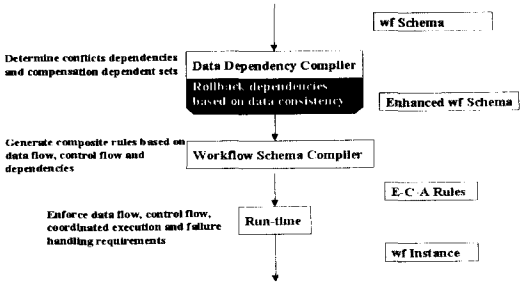


그림 5 데이타 일관성을 고려한 철회 종속성을 포함한 워크플로우 컴파일

그림 8은 협력 트랜잭셔널 워크플로우 환경에서의 데이타 일관성을 고려한 철회 종속성 생성 알고리즘으로서 LAWS로 표현된 워크플로우 스키마를 기반으로 하고 있다.

여기서 IRD는 암시적 철회 종속성(Implicitly Rollback Dependency)을 나타내고, ERD는 명시적 철회 종속성(Explicitly Rollback Dependency)을 나타낸다.

암시적 철회 종속성은 그림 6처럼, 임의의 두 단계 S12와 S23은 제어 흐름 종속성은 없지만, 공유 데이타를 이용하는 데이타 종속성을 가진 협력 트랜잭셔널 워크플로우 환경에서의 철회 종속성을 의미하며 공유 데이타에 대한 실제 접근 순서는 워크플로우 인스턴스가 생성되고, 런타임 때 알 수 있기 때문에 철회 종속성도 런타임 때 생성된다.

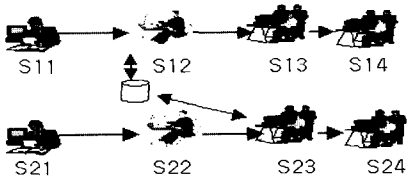


그림 6 암시적 협력 트랜잭셔널 워크플로우

명시적 철회 종속성은 그림 7처럼, 임의의 두 단계 S12와 S23에 제어 흐름 종속성이 명시적으로 표현되어 있고 공유 데이타를 이용하는 데이타 종속성을 가진 협력 트랜잭셔널 워크플로우 환경에서의 철회 종속성을 의미한다. 이와 같은 경우는 워크플로우 스키마에 제어 흐름 명세와 단계 충돌 명세가 명시적으로 표현된다. 따라서, 구축 단계시 데이타 일관성을 고려한 철회 종속성을 생성할 수 있게 된다.

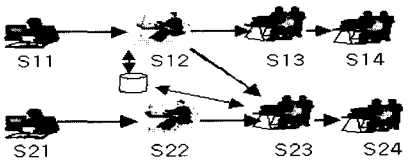


그림 7 명시적 협력 트랜잭셔널 워크플로우

```
case1 : IRD in a transactional workflow
    while(search COMPENSATION-DEPENDENT-SETS)
        make pairs of compensation-dependent-sets;
    make rollback dependency spec;
        { while((Si, Sj) exists in pairs of CDS)
            compensate Sj Si;
            re-start Si Sj;}

case2 : ERD in collaborative transactional workflows with CF
    while ( search CONTROL-FLOW)
        make pairs of control flow steps;
    while ( search STEP-CONFLICT-SPEC)
        make pairs of conflicting steps;
    make rollback dependency spec;
        { while ((Si, Sj) exists in pairs of control flow steps)
            if ((Si,Sj) exists in pairs of conflicting steps)
                compensate Sj, Si;
                re-start    Si, Sj;}

case3 :IRD in collaborative transactional workflows without CF
    while ( search STEP-CONFLICT-SPEC except case2)
        make pairs of conflicting steps;
    make rollback dependency spec;
        { while((Si, Sj) exists in pairs of conflicting steps)
            find_application_finish_time(Si, Sj);}
```

그림 8 데이타 일관성을 고려한 철회 종속성 추출 알고리즘

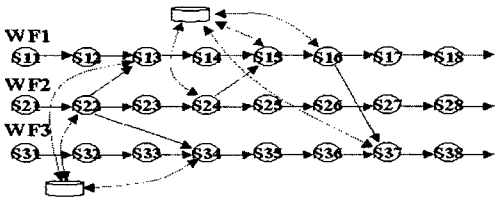
그림 8에서 제시하고 있는 알고리즘의 첫 번째는 하나의 워크플로우에서의 암시적 철회 종속성을 추출하는 경우로서, 보상 종속 집합을 통해 철회가 될 때 보상 순서와 재실행 순서를 추출할 수 있다. 두 번째는 제어 흐름 종속성을 갖고 있는 협력 트랜잭셔널 워크플로우에서의 명시적 철회 종속성을 추출하는 경우로서, 제어 흐름 명세와 단계 충돌 명세를 통해 보상 순서와 재실행 순서를 추출할 수 있다. 즉, 첫 번째 경우와 두 번째 경우는 워크플로우 스키마를 입력으로 받아 철회 종속성 명세가 명확히 표현된다. 그렇지만, 세 번째는 제어 흐름 종속성이 없는 협력 트랜잭셔널 워크플로우에서의 암시적 철회 종속성을 추출하는 경우로서, 단계 충돌 명세의 두 번째 경우를 제외한 부분에서 단계의 보상 시 다른 단계와의 관계를 기술하게 된다. 이 경우에 철회 종속성 명세에 명확히 보상 순서와 재실행 순서를 기술하지 못하는 이유는 공유 데이타를 사용하는 단계들이 어느 단계가 먼저

수행되는가에 대한 사실은 런타임 때 알 수 있기 때문이다. 따라서, 기술되는 철회 종속성 의미는 어느 단계가 철회될 때, 그 단계와 충돌 모드의 다른 모든 단계들의 수행시간을 확인할 수 있는 함수(find_application_finish_time)를 수행하여 최근에 수행된 단계부터 철회할 수 있게 한다.

4.1.2 예제

[예제 1] 복잡한 명시적 협력 트랜잭셔널 워크플로우 시나리오

셋 이상의 워크플로우가 협력 관계를 나타내고 있고, 이들을 구성하는 단계들이 여러 데이타베이스를 공유하고 있는 경우이다.



롤백 순서 :

S37 → S16 → S15 → S24 → (S13, S34) → S22

그림 9 복잡한 명시적 협력 트랜잭셔널 워크플로우 모델

그림 9는 {S22, S13, S34}, {S24, S15}, {S16, S37}이 공유 데이타를 사용하고 있으며 이들간의 제어 흐름 종속성이 있는 명시적 협력 트랜잭셔널 워크플로우 경우이다. 두 번째 워크플로우 스키마를 기준으로 4.1.1절의 알고리즘을 적용한 데이타 일관성을 고려한 철회 종속성 명세는 그림 10과 같다.

여기서, {S16, S37}의 철회 종속성은 두 번째 워크플로우 스키마에 표현되어 있지 않기 때문에 철회 종속성 명세에 나타나지 않는다. 그렇지만, 이것에 대한 표현은 첫 번째 워크플로우 스키마에 표현될 것이다. 따라서, 데이타 일관성을 고려한 보상 순서는 그림 9에 표현되어 있는 것처럼 S37 →S16 →S15 →S24 →(S13, S34) →S22이고, 재실행 순서는 역순이 될 것이다. 따라서, 그림 9와 같은 모델은 4.1.1절에서 제시한 알고리즘을 적용했을 경우 데이타 일관성을 고려한 철회 종속성 명세는 정확성을 보이고 있다.

[예제 2] 혼합된 협력 트랜잭셔널 워크플로우 시나리오

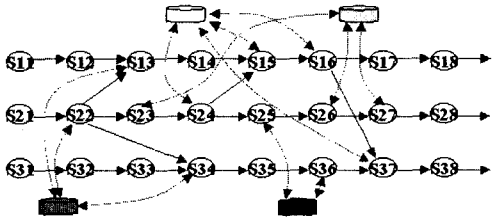
그림 11은 {S22, S13, S34}, {S24, S15}, {S16, S37}이 공유 데이타를 사용하고 있으며 명시적으로 제어 흐름 종속성을 갖고 있고, {S25, S36}은 공유 데이타를 사용하고 있지만, 제어 흐름 종속성은 없고, {S23, S26, S27}은 공유 데이타를 사용하고 있으며 명시적 제어 흐름 종속성은 없지만 이행 규칙(transitive rule)을 적용하여 제어 흐름 종속성을 알 수 있는 혼합된 복잡한 협력 트랜잭셔널 워크플로우 경우이다.

```
ON S22 COMPENSATE
 IF((WF1.S13 CONFLICTS_WITH S22)
     AND (WF1.S13 HAPPENED_AFTER S22))
 THEN COMPENSATE WF1.S13 S22
      RESTART S22 WF1.S13 ;

ON S22 COMPENSATE
 IF((WF3.S34 CONFLICTS_WITH S22)
     AND (WF3.S34 HAPPENED_AFTER S22))
 THEN COMPENSATE WF3.S34 S22
      RESTART S22 WF3.S34 ;

ON S24 COMPENSATE
 IF((WF1.S15 CONFLICTS_WITH S24)
     AND (WF1.S15 HAPPENED_AFTER S24))
 THEN COMPENSATE WF1.S15 S24
      RESTART S24 WF1.S15 ;
```

그림 10 복잡한 명시적 협력 트랜잭셔널 워크플로우 예제에 대한 철회 종속성 명세



롤백 순서 :

S27 → S26 → (S25, S36) ... S37 → S16 → S15 → S24 → S23 → (S13, S34) → S22

그림 11 혼합된 협력 트랜잭셔널 워크플로우 모델

두 번째 워크플로우 스키마를 기준으로 4.1.1절의 알고리즘을 적용한 데이타 일관성을 고려한 철회 종속성 명세는 그림 12와 같다. 좌측 하단은 명시적 협력 트랜잭셔널 워크플로우에 대한 철회 종속성 부분으로서, [예제 1]

```
ON S25 COMPENSATE
 FIND_APPLICATION_FINISH_TIME(S25,WF3.S36) ;

ON S23 COMPENSATE
   COMPENSATE  S27 S26 S23
   RESTART  S23 S26 S27 ;

ON S22 COMPENSATE
 IF((WF1.S13 CONFLICTS_WITH S22)
     AND (WF1.S13 HAPPENED_AFTER S22))
 THEN COMPENSATE WF1.S13 S22
      RESTART S22 WF1.S13 ;
 ON S22 COMPENSATE
 IF((WF3.S34 CONFLICTS_WITH S22)
     AND (WF3.S34 HAPPENED_AFTER S22))
 THEN COMPENSATE WF3.S34 S22
      RESTART S22 WF3.S34 ;
 ON S24 COMPENSATE
 IF((WF1.S15 CONFLICTS_WITH S24)
     AND (WF1.S15 HAPPENED_AFTER S24))
 THEN COMPENSATE WF1.S15 S24
      RESTART S24 WF1.S15 ;
```

그림 12 복잡한 협력 트랜잭셔널 워크플로우 예제에 대한 철회 종속성 명세

에서 설명한 그대로이고, 우측 상단은 하나의 워크플로우에서의 암시적 철회 종속성 부분으로서, 워크플로우 스키마에 철회 종속성 명세에 명확히 보상 순서와 재실행 순서를 알 수 있는 경우이다. 좌측 상단은 협력 트랜잭셔널 워크플로우에서의 암시적 철회 종속성 부분으로서, S25와 WF3.S36은 워크플로우 스키마 레벨에서 어느 단계가 먼저 수행되었는지 알 수 없다. 런타임 때 그 순서를 알 수 있게 된다. 따라서, 워크플로우 스키마 레벨에서는 단지 수행시간에 대해 비교 대상이 되는 단계들이 무엇인가에 대한 확인만을 나타내게 된다.

두 번째 워크플로우 스키마를 기준으로 S22까지 철회가 일어나야 하는 상황에서 데이타 일관성을 고려한 보상 순서는 그림 10에 표현되어 있는 것처럼 S27 →S26 →(S25, S36) …S37 →S16 →S15 →S24 →S23 → (S13, S34) →S22이고, 재실행 순서는 역순이 될 것이다. 따라서, 그림 11과 같은 모델 역시 4.1.1절에서 제시한 알고리즘을 적용했을 경우 데이타 일관성을 고려한 철회 종속성 명세는 정확성을 보이고 있다.

4.2 평가 분석

워크플로우 스키마를 비교할 때, 여러 가지 비교 기준이 제시될 수 있겠지만, 단계들을 수행하는 많은 응용들에서 서로 충돌 모드로 공유 데이타를 사용하고 협력 트랜잭셔널 워크플로우 환경이라면, 데이타 일관성을 고려하고 있는가, 그렇지 않은가는 아주 중요한 사항이다. 왜냐하면, 데이타 비일관성을 초래하고 있는 데이타를 사용한 워크플로우의 최종 결과를 갖고 수행된 비즈니스는 바람직한 결과를 생성할 수 없기 때문이다.

표 4는 기존의 상업적인 워크플로우 관리 시스템의 워크플로우 스키마와 본 논문의 동기가 되었던 [3]에서 제시한 워크플로우 스키마, 그리고 본 논문에서 제안한 워크플로우 스키마를 대상으로 데이타 일관성, 임의의 트랜잭셔널 워크플로우에서의 암시적 철회 종속성, 협력

표 4 제안한 스키마와 기존 스키마와의 비교

비교 기준	상업적인 WFMS에서의 wf 스키마		Mohan Kamath wf 스키마		제안한 wf 스키마	
철회 종속성 (RD)	고려 유무	장애 극복 시점	고려 유무	장애 극복 시점	고려 유무	장애 극복 시점
데이타 일관성을 고려한 철회 종속성	×	런타임 수작업	×	런타임 수작업	○	런타임 자동
IRD in a TWf	×	런타임 수작업	×	런타임 수작업	○	런타임 자동
IRD in Collaborative TWf without CF	×	런타임 수작업	×	런타임 수작업	○	런타임 때 철회 관련 단계들 자동 추출
ERD in Collaborative TWf with CF	×	런타임 수작업	△	런타임 자동	○	런타임 자동

트랜잭셔널 워크플로우에서의 암시적, 명시적 철회 종속성에 대한 고려 유무와 실제로 런타임 환경에서 장애가 발생했을 때의 대처 방법이 무엇인가를 비교 기준으로 설정해 분석한 것이다.

표 4에 의하면, 제안한 워크플로우 스키마는 다음과 같은 장점을 갖고 있다.

- 데이타 일관성을 고려한 철회 종속성
- 임의의 트랜잭셔널 워크플로우에서 철회 종속성을 고려함으로써 런타임 시 발생하는 데이타 일관성을 고려한 철회 정책의 자동 생성
- 협력 트랜잭셔널 워크플로우에서 암시적 철회 종속성을 고려함으로써 런타임 시 발생하는 철회 정책과 관련된 단계들의 자동 추출
- 협력 트랜잭셔널 워크플로우에서 명시적 철회 종속성을 고려함으로써 런타임 시 발생하는 데이타 일관성을 고려한 철회 정책의 자동 생성

4.3 데이타 일관성을 고려한 철회 종속성에 대한 추가적인 고려 사항

마지막으로 언급하고자 하는 문제는 협력 트랜잭셔널 워크플로우 환경에서 데이타 일관성을 고려한 철회 종속성이 워크플로우 종료에 의해 제한을 받게 된다는 점이다. 왜냐하면 단계들은 완료 후에도 보상 프로그램을 이용하여 철회가 가능하지만, 워크플로우의 완료는 더 이상 철회할 수 없기 때문이다.

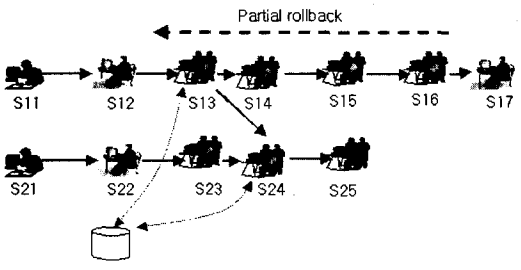


그림 13 워크플로우 종료에 따른 데이타 일관성을 고려한 철회 종속성

그림 13과 같은 협력 트랜잭셔널 워크플로우들이 있다고 하자. 단계 S13은 다른 워크플로우 단계 S24와 제어 흐름 관계를 형성하며 공유 데이타를 사용하고 있다. 이런 환경에서 S16에서 장애가 발생하여 보상 정책이 S13까지 영향을 준다면 S13은 철회될 것이고, S24에게 이 사실을 통보할 것이다. 그런데, 만약 이미 S25까지 수행하여 워크플로우를 종료하게 되면, 더 이상 보상할 수 없게 된다. 따라서, S24에서 철회는 진행되지 않고, 이미 수행된 하위 워크플로우의 결과는 잘못된 데이타를 갖고 수행된 바람직하지 않은 결과이다.

이 문제에 대한 해결하는 방법은 세 가지 경우로 접근해 볼 수 있다.

① 상위 워크플로우의 종료까지 하위 워크플로우의 종료를 지연시키는 방법

상위의 워크플로우의 단계 S13의 응용이 하위 워크플로우 단계 S24의 응용보다 먼저 데이타에 접근했기 때문에 상위의 워크플로우가 종료될 때까지 무작정 하위 워크플로우의 종료를 기다리게 하는 방법이다. 이 방법의 장점으로는 상위 워크플로우가 어느 시점에서 장애가 발생하여 철회되더라도 하위 워크플로우를 철회시킬 수 있어 데이타 일관성을 고려한 철회 정책을 쉽게 만들 수 있다. 그러나, 하위 워크플로우의 많은 종료 지연시간의 낭비를 초래한다.

② 임의의 시점까지 하위 워크플로우의 종료를 지연시키는 방법

상위 워크플로우의 종료시점까지의 지연으로 하위 워크플로우의 많은 종료 지연시간의 낭비를 피하기 위해 상위 워크플로우 스키마에서 단계 실패 명세를 이용하여 어느 시점까지 지연을 해야 하는가를 명시적으로 표현하는 방법이다.

그림 13과 같은 경우, S16의 철회 정책은 S12까지 철회되고, S17부터는 더 이상 S12까지의 철회 정책이 없다고 가정한다면, 하위 워크플로우는 S17이 시작할 때까지만 지연시키고, S17이 수행되면 종료하는 정책을 사용할 수 있다. 이와 같은 방법은 하위 워크플로우의 종료 지연시간이 첫 번째 방법보다는 짧아진다는 장점이 있지만, 이 방법은 S17에 장애가 발생하여 S13까지 보상되어 재실행하다가 다시 S16에서 발생하면 S12까지 보상되어 재실행되어야 하는 경우가 발생할 수 있다. 이 경우, 하위 워크플로우는 종료됐기 때문에 위에서 언급했던 문제는 계속 반복되게 된다.

③ 하위 워크플로우의 종료와 함께 단계 실패 명세의 규칙을 동적으로 변경하는 방법

하위 워크플로우가 종료되었을 때, 동적으로 상위 워크플로우의 보상정책에 대한 규칙을 동적으로 변경하는 방법이 있다. 그림 13의 하위 워크플로우가 종료되면, 상위 워크플로우의 단계 실패 명세와 관련된 규칙들 중 S13까지 보상정책을 갖고 있는 규칙은 S14로 바꿔 기존의 S13까지의 실행은 안전하다는 것을 보장하는 것이다. 따라서, 하위 워크플로우가 종료된 후 S13 이후의 하부 단계들에서 철회가 발생하여도 S13까지는 영향을 미치지 않게 되어서 종료된 하위 워크플로우의 정확한 종료를

보장하게 된다. 이 방법은 하위 워크플로우의 종료를 지연시키지 않는다는 장점을 갖고 있지만, 어떤 환경에서도 적용될 수 있는 범용적인 방법은 아니다. 또한, 워크플로우 설계자에게는 많은 부하(overload)를 준다.

이번 절에서 제시한 고려사항은 워크플로우 인스턴스를 수행하는 런타임 시에 발생되는 문제이기 때문에 워크플로우 스키마에 기술하여 정확성을 표현할 수 있는 문제는 아니다. 워크플로우가 복잡한 많은 업무에 적용되기 때문에 위에서 언급한 세 가지 접근 방법 중 어느 것이 최상의 방법이라고 말할 수는 없고, 적용되는 환경에 따라 워크플로우 설계자가 서로 절충한 적합한 방법을 모색해야 할 것이다.

5. 결 론

데이타를 공유하며 협력적인 공동 작업을 필요로 하는 기업들간의 중요한 일을 수행하는 업무 연계와 같은 분야에 적용되는 협력 트랜잭셔널 워크플로우는 네트워크 인프라의 부족과 기업간의 정보 교환을 위한 표준의 부재, 데이타 일관성 유지 문제 등 많은 제약 요소에 의해 워크플로우 적용에 많은 애로사항이 있다. 특히, 데이타 일관성을 고려한 현재의 프로토타입이나 상업적인 워크플로우 관리시스템은 개별적인 워크플로우의 단계들 사이의 실행 순서의 종속성은 지원하고 있지만, 공유 데이타를 사용하고 있는 상황에서 수행되는 철회 정책은 런타임 시 수작업에 의존하고 있어 오류 가능성뿐만 아니라 많은 비용을 초래한다. 또한, 철회 정책을 지원한다 하더라도 데이타 일관성을 고려한 철회 정책이 지원되고 있지 않기 때문에 워크플로우가 정확한 데이타를 사용하면서 수행되고 있다고 보장할 수 없다.

본 논문에서는 협력 트랜잭셔널 워크플로우들 사이에서 임의의 워크플로우와 다른 워크플로우들의 단계들 사이에서 발생될 수 있는 철회 종속성을 명시적 철회 종속성, 하나의 워크플로우에서의 암시적 철회 종속성, 협력 워크플로우 사이에서의 암시적 철회 종속성, 이렇게 데이타 일관성을 고려한 세 가지 철회 종속성을 워크플로우 스키마에 명시적으로 표현하는 자동 생성 알고리즘을 제안하였고, 이에 대한 컴파일러를 작성하였다. 여러 가지 시나리오를 컴파일러에 입력하여 나온 결과를 살펴보면서 철회 정책에 따른 보상 순서와 재실행 순서를 확인하였고 데이타 일관성을 유지하고 있음을 보였다.

따라서, 데이타 공유와 관련하여 데이타 일관성을 고려해서 런타임에 발생되는 철회 정책을 미리 워크플로우 스키마 정의 시에 기술함으로써 회복 정책이 런타임에 수립되어 그에 따른 비용을 절약하고, 수작업의 오류 가능성을 사전에 방지할 수 있기 때문에 워크플로우 관리 시스템의 정확성을 한층 향상시킬 것이라 기대된다.

참 고 문 헌

[1] WfMC, "Reference Model - The Workflow Reference Model (WFMC-TC-1003, 19-Jan-95, V1.1)", WfMC Specification, 1995.

[2] WfMC, "Interface 1 - Processing Definition Interchange V1.0 Final (WfMC-TC-1016-P)", WfMC Specification, 1998.

[3] WfMC, "Interface 2 - Workflow Client Application Programming Interface (Interface 2 & 3) Specification (WFMC-TC-1009-Specification) V2.0", WfMC Specification, 1998.

[4] WfMC, "Interface 4 - Interoperability Abstract Specification (WFMC-TC-1012, 20-Oct-96, V1.0)", WfMC Specification, 1996.

[5] Mike Anderson, "Workflow Interoperability - Enabling E-Commerce", WfMC White Paper, 1999.

[6] Krithi Ramamritham, Panos K. Chrysanthis, "Advances in Concurrency Control and Transaction Porcessing", IEEE Computer Society Press, 1997.

[7] G. Alonso, D. Agrawal, and A. El Abbadi, "Porcess synchronization in workflow management systems", In 8th IEEE Symposium on Parallel and Distributed Processing(SPDS' 97), New Orleans, Louisiana, October 1996.

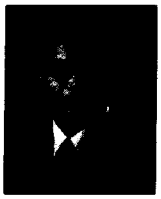
[8] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan, "Functionalities and limitations of current workflow management systems", IEEE Expert: Special Issue on Cooperative Information Systems, 1997.

[9] M Kamath, "Improving Correctness and Failure Handling in Workflow Management Systems", Dissertation written for a Ph. D., 1998.

[10] A. Sheth and M. Rusinkiewics, "On Transactional Workflows", Bulletin of the Technical Committee on Data Engineering, 16(2), June 1993. IEEE Computer Society.

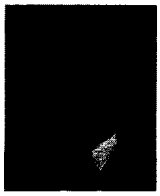
[11] M. Hsu, "Special Issue on Workflow and Extended Transaction Systems", Bulletin of the Technical Committee on Data Engineering, IEEE, 16(2), 1993.

[12] 안승해, 백창현 공저, "워크플로우", 시사컴퓨터, 2000.



변 창 우

1999년 서강대학교 컴퓨터학과 학사. 2001년 서강대학교 컴퓨터학과 공학석사. 2001~현재 서강대학교 컴퓨터학과 박사과정. 관심분야는 트랜잭션 관리, 워크플로우, 웹과 데이타베이스, 접근제어



박 석

1978년 서울대학교 계산통계학과(이학과). 1980년 한국과학기술원 전산학과(공학박사). 1983년 한국과학기술원 전산학과(공학박사). 1983년 9월~현재 서강대학교 컴퓨터학과 교수. 1989년~1991년 University of Virginia 방문교수. 1996년 ~1998년 한국정보과학과 데이타베이스 연구회 운영위원장. 1997년 2월~현재 한국통신정보보호학회 이사. 1999년 1월~현재 한국정보과학회 이사 2000년 4월~현재 DASFAA Steering Committee 멤버. 관심분야는 실시간 데이타베이스, 데이타베이스 보안, 멀티미디어 데이타베이스, 트랜잭션 관리, 데이타웨어하우스, 웹과 데이타베이스, 워크플로우