워크플로우 결정성 판단 메커니즘☆

A Workflow Determinacy Decision Mechanism

정 우 진* 김 광 훈**
Woojin Chung Kwang-Hoon Kim

요 약

워크플로우 관리 시스템은 크게 다섯 가지 측면, 즉 자원(조직)측면, 제어흐름측면, 데이터흐름측면, 이력관리측면, 운용(응용프로그램)측면의 핵심규격을 정의하고 분석하는 모델링모듈과 그 규격에 따른 워크케이스를 실행시키는 엔진모듈로 구성된다. 또한, 워크플로우 관리 시스템의 품격을 결정짓는 핵심기준은 "제정된 규격에 따른 워크플로우 실행의 정확성을 얼마나 보장할 수 있는가?"이다. 특히, 워크플로우의 제어흐름측면규격과 데이터흐름측면규격의 상호간섭으로 발생할 수 있는 "워크플로우 결정성 문제"는 워크플로우 실행의 정확성 보장 여부를 결정짓는 핵심문제이며, 이에 대한 해결방안은 모델링모듈 차원의 워크플로우 분석기법과 엔진모듈 차원의 워크플로우 실행 검증기법으로 나뉘어 구현될 수 있다. 본 논문에서는 워크플로우 결정성 문제에 대한 해결방안으로 엔진모듈 차원의 워크플로우 실행 검증기법, 즉 워크플로우 엔진모듈에 의해 실행되는 액티비티들간의 상호간섭 조건(제어흐름과 데이터흐름)을 검출할 수 있는 메커니즘을 제안한다. 또한, 제안된 메커니즘을 저자 소속의 연구실에서 개발한 워크플로우 관리 시스템에 실제 적용한 결과를 제시함으로서 워크플로우 결정성 문제에 대한 실행 가능한 해결방안임을 증명한다.

Abstract

The primary tasks of a workflow management system specify workflow models with respect to resource, control-flow, data-flow, functional, and operational perspectives, and to enact their workcases (workflow instances). In terms of enacting workflow models, the essential criterion grading the quality of the system is "how much is the system able to guarantee the correctness of workflow models' enactment?". Particularly, the workflow determinacy problem, which may be caused by the interference of the control-flow and the data-flow specifications, is the most challenging issue in guaranteeing the correctness of the system. We are able to solve the problem by either of the following two approaches—analysis of workflow model and verification of workflow enactment. In the paper, we propose a technique that guarantee the system's correctness through verifying workflow enactment. In other words, the technique is able to detect the conflicts of control-flow and data-flow enactments existing on a workflow model, which causes the system to be non-determinant in enacting workflow models. Finally, by applying the technique to the e-Chautauque workflow management system developed by the authors' research group, we prove that the technique is a feasible solution for the workflow determinacy problem.

☞ Keyword : Workflow Engine, Determinacy Analysis, Mutually Interfering Activity,
워크플로우엔진, 결정성 분석, 결정성 검증, 상호간섭액티비티

1. 서 론

정보기술분야에 있어서 최근의 가장 두드러진 변화는 기존의 데이터중심의 정보기술에서 프로세스중심의 정보기술로 빠르게 전이된다는 사실이다. 즉, 1960년대의 파일시스템을 기반으로 한 정보기술의 발전은 1980년대 관계형 데이터베이스 관리 시스템의 개발과 더불어 더욱 발전되었고, 오늘날의 거의 모든 정보기술을 데이터베이스 기술을 기반으로 한다고 해도 과언이 아닐 정도이다. 하지만 데이터베이스기술은 정보화의 핵심을 해당 도메인의 데이터와 그 데이터를 중심으로 한 업무처리 프로그램 중심의 생산성 향상에

* 준 회 원 : 경기대학교 일반대학원 전자계산학과 석사과정
woojin@kyonggi.ac.kr
** 종신회원 : 경기대학교 정보과학부 교수
kwang@kyonggi.ac.kr
[2008/10/24 투고 - 2008/11/01 심사 - 2008/11/28 심사완료]
☆ 이 논문은 2008년도 GRRC (경기도지역협력연구사업) 지원에 의한 연구이었음(This work was carried out with financial support from the 2008 GRRC)

초점을 두고 있다. 그러나 조직내의 업무처리의 생산성을 분석한 결과 업무처리의 전체시간중에 단지 10%만이 업무자체에 소요되고 나머지 90%의 시간은 업무간의 전이 또는 전달시간에 소요된다는 것을 알게 되면서, 업무처리 프로세스에 대한 생산성 및 신뢰성 향상 문제로 정보기술의 초점이 바뀌게 된다. 이러한 사실이 곧 비즈니스 프로세스 리엔지니어링과 자동화를 통한 업무생산성 향상에 초점을 두게 되었고, 최근 2000년대에는 프로세스 중심의 정보기술(BPM/Workflow)이 핵심으로 등장하고 있다. 특히, 전통적인 산업자원 분야의 급속한 디지털화로 인한 전자거래 및 전자무역, 전자물류 등의 활성화와 더불어 높은 신뢰성을 요구하는 금융정보산업분야의 전자금융결재 및 전자지불, 그리고 행정정보분야의 전자정부 등에서 조직간의 프로세스 연동 및 통합의 필요성이 강조되고 있기 때문에 워크플로우관리 또는 비피엠(비즈네스프로세스관리) 시스템(이하 워크플로우관리 시스템)의 프로세스 실행에 대한 정확성과 신뢰성이 주요연구개발 이슈로 부각되고 있다.[11]

기능적 관점에서의 워크플로우관리 시스템은 크게 다섯 가지 측면, 즉 자원(조직)측면, 제어흐름측면, 데이터흐름측면, 이력관리측면, 운용(응용프로그램)측면의 핵심규격을 정의하고 분석하는 모델링모듈과 그 규격에 따른 워크케이스를 실행시키고 그의 실행이력과 감시 및 모니터링을 수행하는 엔진모듈로 구성된다. 특히, 워크플로우관리 시스템의 엔진모듈의 품격을 결정짓는 핵심기준은 “제정된 규격에 따른 워크플로우 실행의 정확성을 얼마나 보장할 수 있는가?”이다. 특히, 워크플로우의 제어흐름측면규격과 데이터흐름측면규격의 상호간섭으로 발생할 수 있는 “워크플로우 결정성 문제”는 워크플로우 실행의 정확성 보장 여부를 결정짓는 핵심문제이며, 이에 대한 해결방안은 모델링모듈 차원의 워크플로우 분석기법과 엔진모듈 차원의 워크플로우 실행 검증기법으로 나뉘어 구현될 수 있다. 본 논문에서는 워크플로우 결정성 문제에 대한 해결방안으로 엔진모듈 차원의 워크플로우 실행 검증기법, 즉 워크플로우 엔진모듈에 의해 실행되는 액티비티들간의 상호간섭 조건(제어흐름과 데이터흐름)을 검출할 수 있는 메커니즘을 제안한다. 또한, 제안된 메커니즘을 저자소속의 연구실에서 개발한 워크플로우 관리 시스템에 실제 적용한 결과를 제시함으로서 워크플로우 결정성 문제에 대한 실행가능한 해결방안임을 증명하고자 한다.

특히, 복합적인 처리 구조를 가진 워크플로우 관리 시스템에서 각 워크플로우 모델(이하 프로세스)의 수행에 대한 결정성 보장여부는 시스템의 신뢰성과 정확성을 판단하는데 있어서 매우 중요한 판단요소이다. 프로세스 인스턴스가 수행되는 동안 어떤 순간에 참조되는 데이터의 값에 따라 처리 결과가 매번 달라지고, 각각의 업무들을 처리하면서 서로 다른 업무들의 진행에 간섭하게 된다면 이것은 기업이나 조직의 입장에서는 원활한 업무의 진행을 방해하는 요소가 될 수 있다. 이것은 곧 워크플로우관리 시스템의 정확성 보장 뿐 만 아니라 기업의 신뢰도에 큰 영향을 미치기 때문에 시스템 내부적으로 비즈니스 프로세스가 올바르게 운용될 것인지에 대한 검증이 필요하다. 이러한 이유로 본 논문에서는 실행 시점에서 워크플로우 프로세스 인스턴스의 결정성을 판단할 수 있도록, 액티비티 간의 상호 간섭 검출 메커니즘을 개발하고자 한다. 이를 통해 잘못된 프로세스 운용에 대한 오류를 막을 수 있을 것으로 기대할 수 있다.

본 논문의 구성은 2장에서 결정적 워크플로우 프로세스와 상호 비 간섭 액티비티에 대해 기술하고, 3장에서는 워크플로우 프로세스의 결정성 판단을 위한 상호 간섭 액티비티 검출 메커니즘에 대해 기술한다. 4장에서는 상호 간섭 액티비티 검출 메커니즘을 워크플로우 엔진에 직접 적용하여 구현 후, 실제의 프로세스를 수행함으로서 본 메커니즘의 구현 가능성을 증명한다. 마지막으로 5장에서는 결론과 향후 연구에 대해 기술한다.

2. 관련 연구

워크플로우 프로세스 결정성[7][9][10]은 병행 워크플로우[5][6][8](Concurrent Workflow)와 관련이 깊다. 병행 워크플로우는 워크플로우 프로세스의 실행에 있어서 프로세스의 다른 액티비티들이 동일한 자원에 접근하여 관련되어지는 것을 의미한다. 즉, 프로세스의 AND-SPLIT에 의해 워크플로우 프로세스를 구성하는 액티비티들이 병행적으로 동시에 수행될 때 수행되어지는 액티비티들이 동일한 자원에 접근 되어 지는 것을 의미한다.

2.1 결정적 워크플로우 프로세스

워크플로우 프로세스는 실행 시마다 워크케이스를 생성하고, 생성된 각각의 워크케이스들은 자체의 실행 데이터를 가지고 그에 따라 적합하게 수행된다. 만약 하나의 워크플로우 프로세스에 의해 생성된 워크케이스들이 동일한 환경, 동일한 수행자, 동일한 데이터를 가지고 프로세스를 수행한다면, 항상 같은 결과를 산출해야 한다. 만약 프로세스가 수행시마다 다른 결과를 산출한다면 이것은 워크플로우 프로세스가 잘못 모델링된 것이다.

다시 말해 워크플로우 프로세스의 관련데이터 관점에서 볼 때, 동일한 입력 데이터를 가지고 프로세스 인스턴스를 생성하여 수행할 때, 모든 프로세스 인스턴스가 항상 같은 결과 데이터를 산출 하면, 이 프로세스를 결정적 워크플로우 프로세스(determinacy workflow process)라고 한다.

상호 비 간섭 액티비티와 결정적 프로세스의 관계를 볼 때, 워크플로우 프로세스의 모든 액티비티가 상호 비 간섭 액티비티들로만 구성되어 있으면 그것은 결정적인 프로세스이다.[1] 즉, 프로세스를 구성하는 액티비티들 간의 상호 간섭 관계를 조사함으로서 프로세스의 결정성 판단이 가능하다.

2.2 상호 비 간섭 액티비티

상호 비 간섭 액티비티(mutually non-interfering activities)는 다른 액티비티로부터 데이터에 대한 입력 또는 출력의 방해 또는 간섭이 존재하지 않는 액티비티이다. 이는 다음과 같이 정의 할 수 있다.

액티비티 α', α''가 워크플로우 프로세스 P의 구성요소일 때, α', α''가 다음 조건을 만족하면 α', α''는 P에서 상호 비 간섭 액티비티이다. [1][2]

1. 워크플로우 프로세스 P에서 액티비티 α'와 α''는 병행 처리 되지 않는다. 또는

2. 워크플로우 프로세스 P에서 액티비티 α'와 α''는 다음 조건을 만족 한다.

(Iα' : 액티비티 α'가 데이터를 읽어오는, 관련 데이터 저장소(relevantdata repository)들의 집합, Oα' : 액티비티 α'가 데이터를 기록하는, 관련 데이터 저장소들의 집합)

1) $O\alpha' \cap O\alpha'' = \varnothing$, and
2) $O\alpha' \cap I\alpha'' = \varnothing$, and
3) $I\alpha' \cap O\alpha'' = \varnothing$

위의 정의는 워크플로우 프로세스를 구성하는 액티비티들 간의 관계를 통해 상호 비 간섭 액티비티를 정의하고 있다. 첫 번째 조건에서, 액티비티들이 병행 처리 되지 않으면, 액티비티들 간에 상호 간섭이 존재하지 않는다는 것을 정의하고 있다. 워크플로우에서 병행 처리는 AND-SPLIT과 AND-JOIN 사이에서만 발생 가능함으로, 액티비티들 간의 상호 간섭은 AND-SPLIT과 AND-JOIN 사이에서만 발생 가능하다.

두 번째 조건에서, 액티비티들이 관련데이터 저장소에 데이터 읽기, 쓰기에 대한 조건을 통해 정의가 가능하다. 병행 액티비티들은 그들의 접근 순서가 상황에 의존적이기 때문에, 어떤 순서대로 공유 데이터에 접근 할지 알 수가 없다. 그러나 병행 액티비티들의 공유데이터에 비결정적인 접근은 워크플로우 실행으로부터 기대되어지지 않은 결과를 가져온다. 다음의 레이스 컨디션(race

condition)은 병행 액티비티들의 실행으로부터 고려되어질 수 있다.[3]

(1) READ-WRITE 충돌

액티비티 α'가 관련데이터 R로 부터 데이터를 읽으려고 시도할 때, 액티비티 α''가 같은 관련데이터 R에 데이터를 쓰려고 시도하는 상황, 그 때, α'와 α''는 병행 액티비티이다

(2) WRITE-WRITE 충돌

액티비티 α'가 관련데이터 R에 데이터를 쓰려고 할 때, 액티비티 α''가 관련데이터 R에 데이터를 쓰려고 시도하려는 상황, 그 때 α'와 α''는 병행 액티비티이다.

액티비티들이 병렬로 처리 될 때, 위의 두 가지 레이스 컨디션이 발생 하지 않아야 상호 비 간섭 액티비티이다. 즉 액티비티들이 관련 데이터와 READ-READ 조건일 경우에만 상호 비 간섭 액티비티 조건을 만족한다.

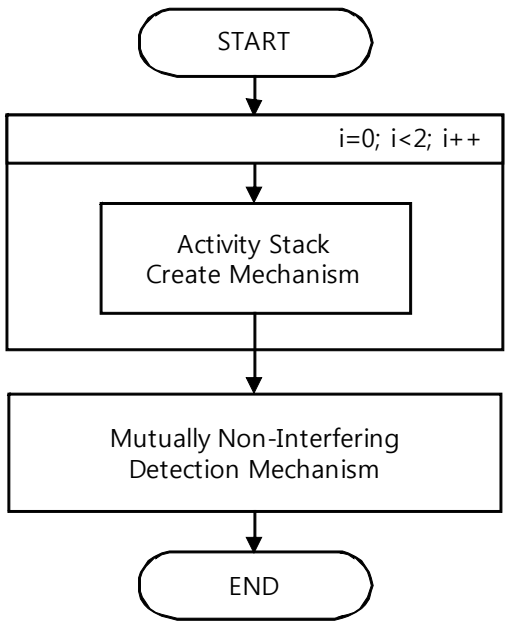
3. 상호 간섭 액티비티 검출 메커니즘

관련데이터 정보에는 LAST_READ_ACTIVITY_ID (관련데이터로부터 마지막으로 데이터를 읽은 액티비티)와 LAST_WRITE_ACTIVITY_ID (관련데이터에 마지막으로 데이터를 기록한 액티비티)가 포함되어 있다. 이 정보들은 실행 시점에서 기록되어 지고, 액티비티간의 상호 간섭 관계를 조사하는데 중요한 역할을 한다.

본 논문에서는 실행 시점, 즉 워크플로우 엔진에서 결정성을 판단하기 위해, 액티비티들 간의 상호 간섭을 조사한다. 상호 간섭 관계는 두 액티비티를 기준으로 수행한다.

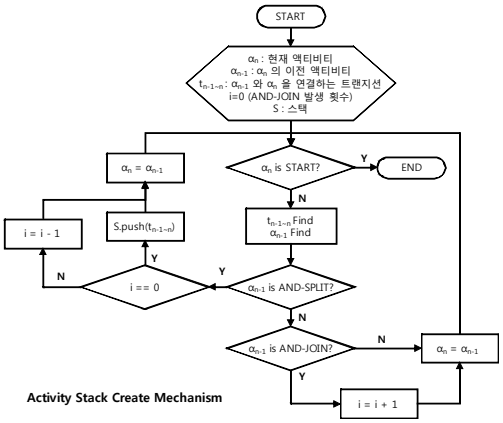
실행시점에 검출하기 때문에 현재 수행하고 있는 액티비티와 관련 데이터 저장소에 기록되어 있는 LAST_READ_ACTIVITY, 또는 LAST_WRITE_ACTIVITY 와 상호 간섭 관계를 검사한다.

두 액티비티가 상호 간섭 관계에 있는지 검사하기 위해서는 두 개의 메커니즘을 수행 한다. 하나는 액티비티들이 병렬 관계에 있는지 검사하기 위해 액티비티 스택을 만드는 '액티비티 스택 생성 메커니즘(Activity Stack Create Mechanism)'이다. 다른 하나는 액티비티 스택을 사용하여 검사한 액티비티들이 병렬관계인지 확인하고, 그것들이 관련데이터와 어떤 관계를 가지고 있는지 확인을 통해 상호 관섭관계에 있는지 검사하는 '상호 비간섭 검출 메커니즘(Mutually Non-Interfering Detection Mechanism)'이다. 현재 수행하는 액티비티와 관련데이터 저장소에 기록된 액티비티 각각의 액티비티 스택을 생성해야 하기 때문에 액티비티 스택 생성 메커니즘은 두 번 수행한다.



(그림 1) 상호 간섭 액티비티 검출 메커니즘

다음의 메커니즘은 두 액티비티의 액티비티 스택을 만드는 메커니즘이다.

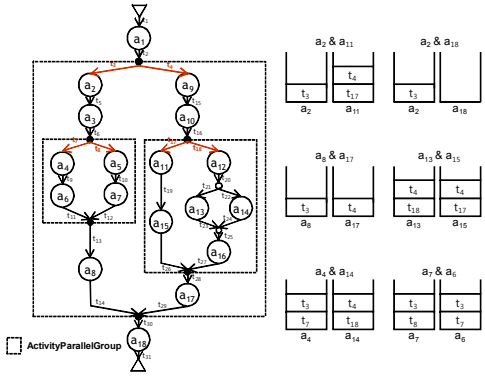


(그림 2) 액티비티 스택 생성 메커니즘

(표 1) 액티비티 스택 생성 알고리즘

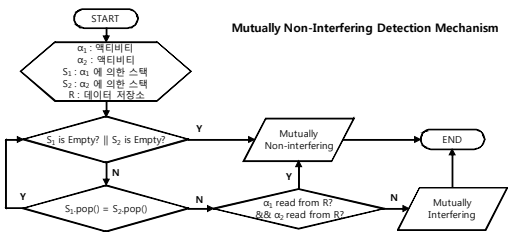
```
while αn ≠ START
    do tn-1~n find; αn-1 find;
    if αn-1 = AND-SPLIT
        if i=0
            then tn-1~n push up Stack; αn = αn-1;
        else i=i-1; αn = αn-1;
    else
        if αn-1 = AND-JOIN
            then i=i+1; αn = αn-1;
        else do αn = αn-1;
```

워크플로우 프로세스에서 병렬 관계는 AND-SPLIT시 분기되는 트랜지션을 통해 확인이 가능하다. 트랜지션은 액티비티와 액티비티를 연결해 주는 간선으로, 전·후 액티비티의 정보를 가지고 있다. 이 트랜지션 정보를 통해 검사할 액티비티의 트랜지션을 진행된 경로를 뒤로 따라 이동하면서 AND-SPLIT 발생 시 해당 트랜지션을 액티비티 스택에 담는다. 이 때, 트랜지션을 스택에 담을 때는 알고리즘에 따라 항상 짝이 맞는 트랜지션만 담게 된다. 워크플로우에서 병렬로 수행 가능한 부분은 AND-SPLIT과 AND-JOIN 내부이다. 이처럼 하나의 병렬 수행 부분을 묶어 액티비티 병렬 그룹(ActivityParallelGroup)이라 한다.



(그림 3) 액티비티 스택

그림 3은 워크플로우 프로세스의 액티비티 스택 생성 메커니즘 수행 결과를 보여주고 있다. α_8의 경우 액티비티 생성 알고리즘을 수행하면, 액티비티 스택에는 t_3이 삽입된다. 그림 3의 왼쪽 프로세스 모델에서 보면 t_7와 t_3이 액티비티 스택에 삽입 가능하지만, t_7에 의해 분기되는 트랜지션은 α_8에 영향을 주지 못하기 때문에 t_3만 삽입된다.



(그림 4) 상호 비 간섭 조건 검출 메커니즘

(표 2) 상호 비 간섭 조건 검출 메커니즘

```
while true
    do if S1.isEmpty()=true || S2.isEmpty()=true
        then Mutually Non-interfering; break;
    else
        do if S1.pop() = S2.pop()
            then continue;
        else
            do if α1 read from R && α2 read from R
                then Mutually Non-interfering; break;
            else do Mutually Interfering; break;
```

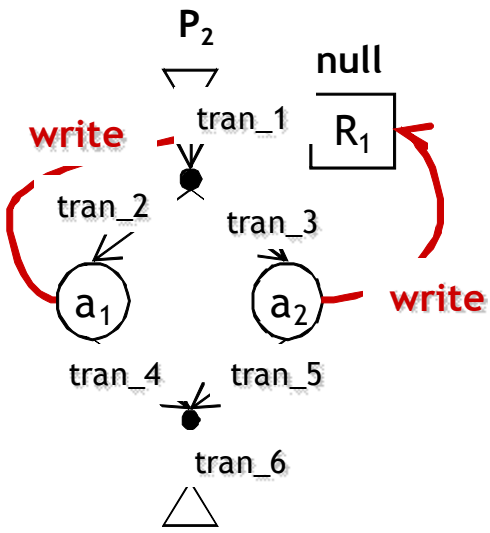
액티비티 생성 메커니즘으로 생성된 두 액티비티의 액티비티 스택을 통해 그림 4와 표 2가 나타내는 상호 비 간섭 조건 검출 메커니즘에서 액티비티들 간의 병렬관계를 검사한다. 액티비티 스택이 비어있는 경우는 분기를 하지 않는 경우이기 때문에, 어느 한쪽의 액티비티 스택이 빌 때 까지 각각의 스택에서 하나씩 꺼내서 비교한다. 두 개의 액티비티 스택에서 꺼낸 트랜지션이 모두 같으면 두 액티비티는 상호 비 간섭 관계에 있다. 만약 다를 경우 꺼낸 액티비티들과 관련데이터와의 관계를 파악한다. 이전에서 상호 비 간섭 액티비티 정의에 따라 두 액티비티와 관련데이터 저장소의 관계가 READ-WRITE 충돌 또는 WRITE-WRITE 충돌인 경우 두 액티비티는 상호 간섭 관계이다. 만약 두 액티비티가 동일한 데이터 저장소에 READ-READ 일 경우 두 액티비티는 상호 비 간섭 관계이다.

이상의 메커니즘을 통해 액티비티의 상호 비 간섭 여부 검출이 가능하다. 워크플로우 프로세스 인스턴스가 종료될 때 까지 메커니즘에 적용되는 모든 액티비티들이 상호 비 간섭일 경우 해당 워크플로우 프로세스는 결정적 프로세스 이고, 간섭 관계가 발생할 경우 해당 워크플로우 프로세스는 비 결정적 프로세스이다.

4. 결정성 보장 워크플로우 엔진

지금까지 결정성 판단을 위한 액티비티 상호 간섭 검출 메커니즘에 대하여 기술하였다. 실행 시점에서 간섭 조건을 판단할 액티비티들이 병렬 관계에 있는지 액티비티 스택을 통해 조사하고, 만약 병렬 관계에 있을 경우 액티비티들과 관련 데이터와의 값 읽기, 쓰기 조건을 판단하여 상호 간섭을 판단한다. 본 장에서는 상호 간섭 액티비티 검출 메커니즘을 워크플로우 엔진에 직접 적용하여 구현 후, 실제의 프로세스를 수행함으로서 본 메커니즘의 구현 가능성을 증명한다.

테스트 프로세스는 하나의 관련데이터 저장소로부터 병렬 관계에 있는 모든 액티비티들이 데이터를 기록하는 모델이다.



(그림 5) WRITE-WRITE 프로세스

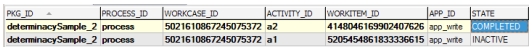
그림 5의 워크플로우 프로세스 모델은 액티비티 α_1 과 α_2 모두 관련데이터 R_1에 데이터를 기록하기 때문에 어느 것이 먼저 수행되어도 동일한 결과를 얻을 수 있다. 그렇기 때문에 한 경우에 수행한다. 그림 6은 워크플로우 엔진에서 프로세스가 시작되면, 모든 액티비티의 워크아이템이 생성된 상태를 나타낸다. 워크아이템이 생성되면 업무를 수행하는 사람이 해당 업무를 시작하기 전까지 INACTIVE 상태를 유지한다.

PKG_ID	PROCESS_ID	WORKCASE_ID	ACTIVITY_ID	WORKITEM_ID	APP_ID	STATE
determinacySample_2	process	502161086724507537	a2	414804616990240762	app_write	INACTIVE
determinacySample_2	process	502161086724507537	a1	520545486183333661	app_write	INACTIVE

(그림 6) 각 액티비티의 워크아이템 생성

아직까지는 관련데이터 R1을 이용하여 작업하는 워크아이템이 활성화되지 않은 상태이기 때문에 현재 워크플로우 관련데이터 R1의 값은 NULL 이다. 업무 처리자가 액티비티 α_2를 수행해서 워크아이템이 처리가 되면 관련데이터 R_1에 작업한 내용이 저장된다. 그림 7에서 나타내는 액티비티

α_2의 워크아이템은 관련데이터 R1에 데이터를 기록하고, 완료 상태로 전이된것을 보여준다.

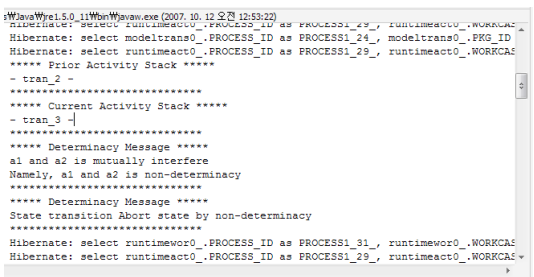


PKG_ID	PROCESS_ID	WORKCASE_ID	ACTIVITY_ID	WORKITEM_ID	APP_ID	STATE
determinacySample_2	process	502161086724507537 2	a2	414804616990240762 6	app_write	COMPLETED
determinacySample_2	process	502161086724507537 2	a1	520545486183333661 5	app_write	INACTIVE

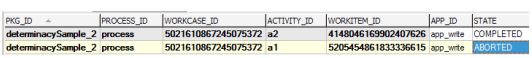
(그림 7) α_2 액티비티의 워크아이템 수행 완료

관련데이터 R_1의 LAST_WRITE_ACTIVITY_ID에는 α_2가 저장되어 있다. 또 다른 액티비티 α1은 관련데이터 R_1에 데이터를 기록해야하기 때문에 관련데이터 R_1에 대해서 상호 간섭관계를 고려해야 한다. 현재 액티비티 α_1과 LAST_WRITE_ACTIVITY α_2 간 에 비결정성 검출 메커니즘을 수행 한다.

그림 8과 그림 9가 나타내는 실행 결과에 따르면, α_1액티비티 스택에는 tran_2 트랜지션이 저장되고, α_2 액티비티 스택에는 tran_3 트랜지션이 저장된다. 이들 간의 관계를 상호 비 간섭 검출 메커니즘에 따라 수행해 보면 액티비티 α1과 α2사이에는 상호 간섭이 존재한다. 그렇기 때문에 그림 4와 표2의 상호 비간섭 검출 메커니즘에 따라 액티비티 α_1의 상태가 처리중단상태(ABORTED)로 상태 전이 되고, 이에 따라 워크케이스의 상태도 처리중단상태(ABORTED)로 상태 전이 된다.



(그림 8) 워크플로우 엔진 수행 화면 - 상호 간섭 조건 검출 및 작업 중지



PKG_ID	PROCESS_ID	WORKCASE_ID	ACTIVITY_ID	WORKITEM_ID	APP_ID	STATE
determinacySample_2	process	502161086724507537 2	a2	414804616990240762 6	app_write	COMPLETED
determinacySample_2	process	502161086724507537 2	a1	520545486183333661 5	app_write	ABORTED

(그림 9) 워크아이템 중지

5. 결 론

본 논문에서는 최근에 연구 개발 되고 있는 거의 모든 워크플로우 관리 시스템이 직면하고 있는 워크플로우 엔진의 안정성, 결정성 문제를 해결하고자 워크플로우 프로세스의 결정성 문제를 해결할 수 있는 결정성 판단 메커니즘을 제안하고, 이를 실제 워크플로우관리 시스템에 적용하고 실행결과를 자세히 보여줌으로서 그의 구현가능성을 검증하였다. 실행 시점에서 간섭 조건을 판단할 액티비티들이 병렬 관계에 있는지를 액티비티 스택을 통해 조사하고, 만약 병렬 관계에 있을 경우 액티비티들과 관련데이터와의 값 읽기, 쓰기 조건을 판단하여 상호 간섭을 판단할 수 있었다. 그리고 이를 워크플로우 엔진에 적용함으로서 실행 시점에서 워크플로우 프로세스의 결정성을 판단하도록 하였다.

결과적으로, 실행시점의 워크플로우 엔진모듈에서 프로세스의 결정성을 판단함으로서 프로세스 모델링 시점에서 발견할 수 없는 프로세스 결정적 요소들의 검출이 가능하게 했을 뿐 만 아니라 특정 규칙에 따른 비결정적인 요소의 사전검증 및 제거를 가능하게 하여 워크플로우관리 시스템의 신뢰성과 정확성을 보장할 수 있는 가능성을 제시했다는 점이 본 연구의 주요 기여부분이라고 할 수 있다.

참 고 문 헌

[1] Gary J. Nutt, “Centralized and Distributed Operation Systems”, Prentice Hall, 1992

[2] A. J. Bernstein, “Program Analysis for Parallel Processing”, IEEE Transactions on Electronic Computers, Oct-1996, 757-762

[3] Minkyu Lee, Dongsoo Han, "Set-Based Access Conflicts Analysis of Concurrent Workflow Definition", IEEE, 2001, 172-176

[4] Jonathan E. Cook, Zhidian Du, Chongbing Liu,

Alexander L. Wolf, “Discovering models of behavior for concurrent workflows”, Computers in Industry archive, April 2004, 297-319

[5] A.P. Barros, A.H.M. ter Hofstede, “Modelling Extensions for Concurrent Workflow Coordination”, Cooperative Information Systems, 1999, 336-347

[6] Jingfu Zhong, Binheng Song, ″Verification of Resource Constraints for Concurrent Workflows″, Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2005

[7] Stephen F.Mills, Murat M. Tanik ″Determinacy and Concurrency Issues in Process Engineering″, Journal of Systems Integration, 1998, 183-201,

[8] KwangHoon Kim, ″Architecture for Very Large Scale Workflow Management Systems″, February 1998

[9] Anna Philippou and David Walker, ″On Sharing and Determinacy in Concurrent Systems″, Lecture Notes In Computer Science, 1995, 456-470

[10] A. J. Bernstein, ″Analysis of Programs Parallel Processing″, IEEE Transactions on Electronic Computers, 1996, 757-762

[11] 김광훈, “워크플로우 기술1″, TTA(한국정보통신기술협회) 저널, pp.107 - 118, 2003.01

◐ 저 자 소 개 ◑

정 우 진 (Woojin Chung)
2007년 경기대학교 전자계산학과 졸업(학사)
2007년 ~ 2008년 현재 경기대학교 일반대학원 전자계산학과 석사과정
2007년 12월 ~ 현재 (주)아이씨엔아이티 기술연구소 연구원
관심분야 : Workflow, BPM, Business Intelligence
E-mail : woojin@kyonggi.ac.kr

김 광 훈 (Kwang-Hoon Kim)
1984년 경기대학교 전자계산학과 졸업(학사)
1986년 중앙대학교 대학원 전자계산학과 졸업(석사)
1994년 University of Colorado at Boulder, Computer Science, MS
1998년 University of Colorado at Boulder, Computer Science, Ph.D
1986년 2월 ~ 1991년 8월 한국전자통신연구원
1993년 5월 ~ 1994년 8월 American Educational Products, Inc., Processional DB Consultant
1994년 9월 ~ 1995년 8월 Colorado Advanced software Institute, Research Assistant
1995년 9월 ~ 1997년 2월 Aztek Engineering, Inc., Software Engineer
1998년 ~ 현재 경기대학교 정보과학부 교수
관심분야 : Workflow, BPM, Groupware, Advanced Databases, Grid/Cloud Computing
E-mail : kwang@kyonggi.ac.kr