워크플로우 로그에서 워크플로우 명세 탐사

정희택*

Workflow Specification Mining on Workflow Logs

Hee-Taek Jeong*

이 논문은 2000년도 여수대학교 학술 연구지원비를 지원받았음

요 약

자동화된 업무 처리를 위한 워크플로우 시스템이 일반화되었다. 본 연구는 워크플로우 로그로부터 워크플로우 명세를 탐사하는 방안을 제안한다. 제안한 기법은 과업간 순환, AND 그리고 OR 제어 흐름을 고려하여 워크플로우 명세를 탐사한다. 또한 로그 생성 시에 워크플로우 명세를 탐사할 수 있는 동적 탐사 기법을 제안한다.

ABSTRACT

Workflow systems, automated business processing, have been generalized. In this paper, we propose a method to mine workflow specification on workflow logs. The method detects workflow specification considering cycle, AND and OR control flow between tasks. Also, we provide dynamic mining method to detect workflow specification in which log is generated.

키워드

Workflow system, Workflow log, Workflow graph, Workflow specification mining

Ⅰ. 서 론

현대사회에서 조직의 규모가 방대해지고 수행하는 업무의 형태가 복잡해지면서 서류화된 업무 지침서나 실무 경험에 의존하는 업무 처리 방식은 한계를 드러내고 있다. 또한 급변하는 업무 환경 변화에 적응하기 위해, 컴퓨터와 통신망을 이용하여 복잡한 업무 흐름을 정의하고, 정의된 흐름에 맞게 업무가 수행할 수 있도록 제어하는 워크플로우 시스템이 일반화 되었다[1,2,3].

업무 처리의 자동화를 위해 채용되고 있는 워크플로우 시스템은 업무 흐름을 의미하는 워크플로우를 제어한다. 워크플로우는 다양한 개별 과업과 그들 간의 제어 흐름을 포함한다[3,4]. 워크플로우 관리 시스템에 의한 워크플로우 수행은 여러 과업들의 수행 및 제어 전이를 만족하는 일련의 과업 수행을 통해 목적을 달성하게 한다. 이때, 각 워크플로우의 수행 상태 정보는 보조 기억장치에 로그(log) 형태로 유지

*여수대학교 정보기술학부 정보과학전공 조교수

접수일자 : 2002 7. 16

함으로써 수행 상태에 대한 모니터링 정보를 제공한다[5].

모니터링은 현재 수행중인 워크플로우 상태를 관리자나 일반 사용자에게 제공함으로써 작업 부하의 균등분배나 작업의 진척상황을 제공할 수 있다. 그러나 워크플로우가 수백 개 이상의 과업들과 그들 간의 제어 흐름으로 구성되기 때문에, 모니터링은 현재의 정보만을 제공한다. 즉, 수백 개의 과업 중 몇몇 과업의 수행 상태만을 제공할 뿐이다.

한편, 워크플로우 로그 정보를 기반으로 워크플로우 명세를 탐사하지 위한 프로세스 탐사(process mining) 연구가 이루어졌다[6,7,8,9,10, 11,12,13,14]. 기존 연구 중 [6,7,8,9,10]은 단순히 워크플로우 로그로부터 상태변화의 빈도를 근간으로 워크플로우 명세 발견한다. 제안된 연구에서는 빈도가 낮은 사건을 고려대상에서 제외하거나, 과업 간에 존재할 수 있는 모든 제어흐름을 고려하지 않는다. 다음으로, 워크플로우 명세를 생성하는 과정에서 발생할 수 있는 많은 부담과 실패의 위험을 최소화하기 위한 방안으로써 탐사 방안이 제안되었다[11,12,13]. 즉, 초기 명세시에 최소의 과업들 간 제어흐름을 제시하고 수행을 통해 전체적인 제어흐름의 명세를 달성하는 적응적 모델링 방안으로써 제안되었다. 마지막으로 [14,15]에서는 로그정보를 기본으로 통계정보를 산출하거나 워크플로우 수행이 명세를 만족하는지 검증하기 위한 방안을 제안하였다. 그러나 제안된 연구들은 제한된 모델링 요소들만을 고려한 프로세스 탐사 방안을 제안하였고 정적인 탐사 기법만을 제공한다.

본 연구에서는 로그에 저장된 수행정보로부터 워크플로우 명세를 추출할 수 있는 방안을 제안한다. 첫째, 과업간 부분순서 관계와 그들간의 제어 흐름을 고려한 워크플로우 명세의 탐사 방안을 제안한다. 즉, 각 과업간 부분 순서관계를 탐사하고 이를 기반으로 과업간 제어 흐름을 식별하며, 식별된 제어 흐름을 순환, AND 그리고 OR 제어흐름으로 구별함으로써 워크플로우 명세를 탐사 할 수 있다. 둘째, 동적인 워크플로우 명세 탐사방안을 제안한다. 워크플로우 수행 중에 발생하는 로그 정보로부터 워크플로우 명세를 탐사할 수 있는 방안이다.

본 연구의 구성은 먼저 2장에서 워크플로우 로그를 기반으로 수행된 관련연구를 분석하고, 3장에서는 로그로부터 명세의 추출하기 위한 방안을 제시한다. 4장에서는 본 연구와 기존연구를 비교 분석한다. 5장에서는 본 연구의 결론을 기술한다.

II. 관련연구

워크플로우 시스템은 업무 흐름을 표현하는 워크플로우 명세와 이를 실행하는 실행 시스템으로 구분되며 다음 그림과 같다. 워크플로우 명세는 목적을 달성하기 위한 과업과 그들간의 제어흐름을 포함하고 있으며, 실행 시스템은 워크플로우 관리 엔진, 작업 관리자(worklist handler) 등으로 구성된다. 워크플로우 시스템에서 유지되는 데이터는 제어데이터, 관련데이터로 구분한다. 제어데이터에는 워크플로우 로그나 히스토리(history) 정보를 유지한다. 관련데이터는 워크플로우 관리 엔진과 응용 프로그램이 접근할 수 있는 데이터로 수행할 과업의 결정을 위한 정보를 포함한다.

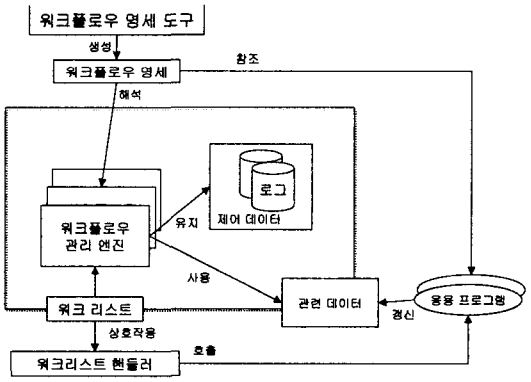

그림 1. 워크플로우 시스템 구조
Fig. 1. Workflow system architecture

워크플로우 로그를 이용한 명세 탐사는 두 가지 의미를 갖는다. 첫째, 워크플로우 명세가 정의되어 있을 때, 탐사된 워크플로우 명세와의 비교를 통해 정의된 명세를 분석하여 효과적인 명세 방안을 제공할 수 있다. 둘째, 초기 워크플로우 명세를 정의하는 부담으로 인해 명세를 정의할 수 없을 때, 워크플로우 시스템 수행 후 로그 정보로부터 명세를 탐사하

여 워크플로우 명세를 정의하는 방안이다. 이는 워크플로우 명세가 없거나 명세할 수 없는 환경에서 점진적으로 명세를 발견 또는 탐사하기 위한 방안이다.

페트리 넷을 기반으로 명세된 워크플로우로부터, 로그 정보를 이용한 워크플로우 명세 탐사 방법이 제안되었다[6,7]. 제안된 방안은 페트리 넷의 구성요소인 플레이스를 과업으로 과업들 간 제어흐름을 트랜지션으로 구분하여, 로그에 존재하는 트랜지션의 발생 빈도를 근간으로 수행한다. 로그 상에 존재하는 트랜지션의 순서와 발생 빈도를 기반으로 과업 간 순서 관계를 구분함으로써 프로세스 탐사를 수행한다. 그러나 페트리 넷을 이용한 워크플로우 명세의 수행에 의해 유지될 로그 정보는, 페트리 넷 특성상 많은 상태 정보의 유지를 요구한다. 즉, 많은 과업들로 구성된 워크플로우에 대한 로그 정보의 양은 대량의 데이터를 이루게 되며 그에 따라 노이즈(noise)의 증가를 야기 할 수 있고, 이는 부정확한 프로세스 탐사를 수행할 수 있다.

[8,9,10]에서는 순차 패턴(sequential patterns) 탐사 방법[8]을 확장하여 워크플로우 명세를 탐사할 수 있는 방안을 제안하였다. 순차 패턴 탐사 기법은 일련의 순차적인 사건들 간의 전체 순서관계를 탐사하는 기법으로 워크플로우에 존재하는 병행 수행 관계를 고려하지 않는다. 제안된 탐사 방안에서는 빈번하게 발생하는 과업들 간의 부분 순서 관계를 근간으로 순서관계를 결정하고 각 순서관계를 통합함으로써 워크플로우 명세를 탐사한다. 임의의 두 과업간의 부분 순서 관계들을 조합하여 전체 순서 관계로 결정함으로써 달성한다. 그러나 제안된 과업들 간 제어흐름 중 OR 형태만을 고려함으로써 AND와 같은 제어 흐름을 탐사할 수 없다.

[11,12,13]에서는 워크플로우 명세를 생성하는 과정에서 발생할 수 있는 많은 부담과 실패의 위험을 최소화하기 위해 제안된 방안이다. 즉, 초기 명세 시에 최소의 과업들 간 제어흐름을 제시하고 로그 정보로부터 전체적인 제어흐름의 명세를 달성하는 적응적 워크플로우 명세 방안이다. 제안된 방안에서는 AND/OR의 제어 흐름을 고려하고 있으며 전체적인 명세 정의가 어려운 환경에서 워크플로우 명세를 생성하기 위해 적합한 방안이다. 그러나 제안된 방법은 과업 간 부분 순서관계를 확률을 기반으로 발견적(heuristic)으로 결정하고, AND 제어 흐름의 결정은 빈도수에 의한 확률 값으로 결정한다. 한편, 탐사과정에서 생성된 워크플로우 명세를 전문가에 의해 검증하고 이를 반영하여 최종적인 워크플로우 명세를 정의한다.

한편, [14]에서는 로그 정보를 분석함으로써, 최종 목적의 달성 시간, 시스템 부담, 작업부하 분석과 같은 통계적 정보를 산출할 수 있으며 과업에 기반을 둔 수행 경비의 산출이 가능함을 제시하였다. 제시한 방안은 단순히 수치적 통계정보만을 제공한다.

마지막으로, [15]에서는 로그정보를 기본으로 워크플로우 수행이 명세를 만족하는지 검증하기 위한 방안을 제안하였다. 로그 상에 존재하는 과업들 간 순서관계를 기반으로 순환(cycle)이 존재하는지 검증하고, 이 정보를 시스템의 기반지식으로 유지함으로써 워크플로우 수행을 검증할 수 있는 방안을 제안하였다. 그러나 제안된 방안은 과업들 간에 순환 수행의 존재여부만을 검증한다.

III. 워크플로우 로그로부터 워크플로우 그래프 탐사

워크플로우 명세는 그래프를 기반으로 모델링 한다, 워크플로우를 구성하는 과업은 노드로 표현하고 그들 간의 종속성은 간선으로 표현한다. 그래프에는 초기 과업을 의미하는 노드와 마지막으로 목적의 달성을 의미하는 성공 종료 과업과 과업들의 수행 중에 목적을 달성하지 못하여 도달하는 실패 종료과업이 존재한다. 워크플로우 그래프는 다음과 같이 정의한다.

정의 1. 워크플로우 그래프

워크플로우 그래프 WGF=(V,E)는 다음과 같이 정의되는 방향 그래프이다. V는 과업의 집합이고 E는 제어 연결자의 집합이다. 과업 t∈V인 t = (TID, Opers, RDatas, WDatas, ANDCIDs, ORCIDs). 제어 연결자 e ∈ E인 e = (EID, BTID, ATID, EC). TID는 과업식별자로 과업 t를 유일하게 구별하게 하며, Opers는 과업이 수행할 연산집합으로 해당과업이 수행할 활동을 기술한다. RDatas와 WDatas는 각각 판독 및 갱신 데이터 집합으로 과업에 의해 판

독 및 갱신되는 워크플로우 관련 데이터 집합을 의미한다. ANDCIDs(또는 ORCIDs)는 AND(OR) 연결 형태를 갖는 제어 연결자의 식별자 집합으로 선행과업 모두(하나)가 완료될 때 과업 t로의 전이를 가능하게 하는 제어 연결자들의 집합이다.

EID는 제어연결자식별자로 연결자 e를 유일하게 구별하게 하며, BTID와 ATID는 각각 시작 및 종료 과업 식별자로 제어 연결자 e가 연결하는 과업의 식별자이다. EC는 가능 조건(enable condition)으로 제어 연결자에 부가된 조건식으로서 워크플로우 데이터, 과업 수행상태를 인수로 갖는 논리식이다.□

[예제 1] 워크플로우 그래프를 이용한 워크플로우 명세는 다음과 같다. 과업 집합 V = {sT,T1,T2,T3,T4,T5,T6,T7,gtT,ftT}이며, 제어 연결자 집합 E={e1,e2,e3,e4,e5,e6,e7,e8,e9,e10}이다. sT는 시작 과업을, gtT는 워크플로우 목적을 달성하여 마지막으로 수행하는 성공종료 과업을, ftT는 워크플로우 수행 중에 임의 과업 수행 실패에 의해 목적을 달성하지 못하고 종료하였을 때 수행하는 실패종료 과업이다. 과업 간 연결은 성공종료 과업으로 전이를 제공하는 연결자 집합만을 제시하였으며 실패종료 과업으로의 전이는 각 과업에서 연결만으로 간단히 정의할 수 있다.□

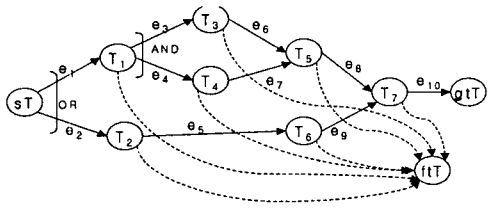

그림 2. 워크플로우 그래프의 예
Fig. 2. The example of workflow graph

워크플로우 명세에 대한 수행은 여러 워크플로우 인스턴스를 이루며, 각 인스턴스는 유일한 식별자에 의해 구분된다. 인스턴스의 수행은 과업들의 수행들로 이루어진다. 이때 각 과업은 시작 및 종료 사건의 상태 변이를 갖는다. 각 사건에 대한 정보를 워크플로우 엔진은 로그에 유지한다. 그러나 본 연구에서는 일반성에 위배됨이 없고 표현의 간략화를 위해, 로그에는 각 과업의 종료 때 발생하는 사건 정보를 유지한다고 가정한다. 워크플로우 로그를 정의하면 다음과 같다.

정의 2. 워크플로우 로그

워크플로우 로그 WL=(WID,TID,T,ODatas)이며, WID는 워크플로우 인스턴스를 유일하게 구별하는 식별자이고 TID는 과업 식별자, T는 해당 사건이 발생한 시간, ODatas는 과업이 수행에 의해 생성된 출력 데이터이다.□

하나의 워크플로우 인스턴스에 대한 로그는 일련의 수행 상태에 대한 정보를 포함한다. 각 워크플로우 인스턴스에 대한 식별자나 과업에 대한 식별자는 워크플로우 엔진에 의해 유일하게 할당된다. 하나의 워크플로우 인스턴스는 하나이상의 과업 수행을 포함하고 있기 때문에 다수의 로그 데이터들로 구성하며 다른 워크플로우 인스턴스와 구별은 WID에 의해 이루어진다.

[예제 2] 그림 2에서 제시한 워크플로우 명세에서, 워크플로우 인스턴스 식별자 WID를 "i3" 로 가정하고 생성될 수 있는 워크플로우 로그는 ("i3","sT","10시10분",{A,B,C}), ("i3", "T1","11시",{C,D})와 같다. 본 연구에서는 무의미한 사건발생 시간 T와 출력 데이터 집합 ODatas를 제외하여 ("i3","sT"), ("i3", "T1")로 간략히 표현할 수 있다. 동일한 인스턴스에서 발생한 로그는 동일한 인스턴스 식별자를 갖기 때문에, 그림 2의 워크플로우 명세로부터 생성될 수 있는 인스턴스에 대한 로그 집합은 {(sT,T1,T3,T4,T5,T7,gft), (sT,T2,T6,T7,gtT), (sT,T1,T3,ftT), (sT,T1,T4,T3,T5, T7,ftT)}와 같다. 이는 네 가지의 인스턴스가 생성되었고 각 인스턴스 내에서 수행한 과업 식별자만을 표현한 것이다. □

워크플로우 로그로부터 워크플로우 그래프를 탐사 과정을 간략하게 표현하기 위해 로그 집합의 각 데이터를 문자열로 가정한다. 즉 예제 2의 로그는 {(sT,1,3,4,5,7,gfT), (sT,2,6,7,gtT), (sT,1,3,ftT), (sT,1,4,3,5,7,ftT)}과 같이 표현하고, 각 과업 간 구분 문자 콤마는 의미적 구분수단으로써 사용한다.

1. 워크플로우 그래프 탐사

워크플로우 로그 집합에서 워크플로우 그래프를 탐사하기 위한 과정은 세 단계로 구분한다. 첫째, 로그 데이터에 존재하는 순환을 제거한다. 이는 순환 수행에 일련의 과업식별자의 중복을 제거한다. 둘째, 로그 데이터에 대해 문자열 비교과정을 통해 각 과업들 간의 선후 관계를 결정한다. 이는 각 과업 간 부분 순서 관계를 결정하기 위한 과정이다. 셋째, 앞 과정에서 생성된 정보를 기반으로 워크플로우 그래프를 생성한다.

첫째, 과업들 간에 직접 순환이나 전이적 순환에 의해 동일 과업이 두 번 이상 수행할 수 있다. 직접 순환은 자신에서 자신으로의 제어 흐름에 의해 임의의 조건이 만족할 때까지 반복 수행하는 것이다. 전이적 순환은 과업들 간에 전이적(transitive) 제어 흐름에 의해 임의의 과업들이 반복수행하는 것이다. 이러한 반복 수행에 의한 과업식별자의 중복 정보를 정제한다. 이는 다음 알고리즘에 의해, 각 로그 데이터에 대해 문자열 비교를 통해 수행한다. 과업식별자가 n번 중복되었을 때 n-1번 삭제한다.

```
Algorithm CycleFiltering
Input : Workflow Log Set WLS
Output : Workflow Log Set nWLS,
         Cycle Tid Set CTS

Each wl ∈ WLS {
  Each tid ∈ wl {
    if( count(wl,tid) > 1 ) {
      concatenate(cycleSequence,tid)
      while(true) {
        tid = follow(wl,tid)
        if( count(wl,tid) > 1 )
            concatenate(cycleSequence,tid))
        else
            break
      }
    }
    eraseSequence(wl,cycleSequence)
    nWLS = nWLS ∪ wl
    CTS = CTS ∪ cycleSequence
  }
}
/* wl:워크플로우로그집합에 속하는 임의의로그
   tid : wl에 존재하는 임의의 과업식별자
   count(wl,tid):wl에 해당 tid의 개수를 반환
   concatenate(cycleSequence,tid)
    :cycleSequence문자열의 마지막에 tid 첨가
   follow(wl,tid)
    :wl상에서 tid 이후에 존재하는 tid를 반환
   eraseSequence(wl,cycleSequence)
    :wl에서 cycleSequence에 속한 tid 리스트삭제 */
```

[예제 3] 다음 그림과 같은 워크플로우 명세에 대한 로그 데이터는 순환에 의해 과업 식별자가 중복된다. (a)는 (....4,4,4,,...) 형태로 제시한 알고리즘에 의해 (...,4,...)로 정제되고 (b)는 (...,1,3,1,3 ,...)은 (...,1,3,...)으로 정제한다. 순환 과업 식별자 집합은 {(4), (1,3)}을 생성한다.□

(a) (b)

그림 3. 순환이 있는 워크플로우 명세

Fig. 3. Workflow specification with cycle

둘째, 로그 데이터에 대해 문자열 비교과정을 통해 각 과업들 간의 부분 순서 관계를 결정한다. 직관적으로, 로그 데이터는 하나의 워크플로우 명세로부터 생성될 수 있는 일련의 문자열들을 생성한다고 가정할 수 있다. 문자열들 간에는 공유하는 문자열과 해당 로그에만 존재하는 개별적 문자열 정보를 포함한다. 공유하는 문자열은 워크플로우 명세 관점에서는 수행이 필수적인 과업집합을 의미한다. 개별적 문자열은 선택적 수행을 갖는 과업집합을 의미한다. 즉, 전자는 순차적 수행이나 AND 제어 흐름에 의한 과업 집합을 의미한다. 후자는 순차적 수행과정에서 OR 제어 흐름에 의해 수행될 수 있는 과업 집합을 의미한다.

문자열들 간에 동일성과 이질성을 구별하기 위해서는, 분자 생물학 연구에서 사용하는 두 서열 비교 방안[16]을 고려할 수 있다. 그러나 [16]에서는 문자열들 간에 동일성과 이질성이 아닌 유사도 또는 차이도를 구분하여 유사도가 높고 차이도가 낮은 문자열을 구별하는데 목적이 있다. 특히, AND 제어 흐름에 속한 과업들의 순서는 수행 상황에 따라 결정되기 때문에 동일한 문자 집합 내에서도 순서가 동일한 것과 불일치하는 문자열이 존재한다. 그러나 [16]에서 제안된 두 문자열간 정렬(alignment) 과정을 응용함으로써 과업들 간 부분순서 관계를 결정할 수 있다.

로그 데이터들 중 가장 긴 문자열은 가장 많은 과업 정보를 갖는다. 가장 긴 문자열은 워크플로우 명세에 대한 정보-제어 흐름 및 과업들-를 가장 많이

포함하고 있다. 가장 많은 정보를 가지고 있는 로그를 기준으로 다른 로그 데이터들과 정렬함으로써 과업들 간 부분순서 관계를 결정할 수 있다. 두 문자열의 정렬은 동일한 문자열을 기준으로 정렬하고 이질적인 문자열에 대해서는 널 문자'-'를 이용하여 이질성을 표시한다. 임의의 문자열, 즉 로그 데이터가 {(1,2, 3,4),(1,3,4)}일 때 다음과 같이 정렬할 수 있다. 워크플로우 명세 상에서 의미는, 과업 1 수행한 후 2나 3을 수행할 수 있으며, 2를 수행한 후 3을 수행하고, 3을 수행한 후 4를 수행한다는 의미이다.

1 2 3 4
1 - 3 4

하나의 문자열에서 과업들 간의 순서관계는 순차수행 흐름을 의미한다. 그러나 둘 이상의 문자열에서 동일한 과업집합이나 순서가 상이한 제어 흐름이 존재할 수 있다. 이는 AND 제어 흐름에 의한 결과로, 이를 다음과 같이 병행 관계라 정의하고 병행 관계에 있는 과업 집합을 병행수행 과업집합이라 한다. 병행수행 과업집합에 속한 과업들은 AND 제어 흐름을 갖는 과업들이다.

정의 3. 병행 관계

임의의 문자열 S1=S11S12...S1i...S1n과 S2=S21S22... S2i...S2m에서, S1i와 S2i까지 정렬을 수행하였고 S1i와 S2i가 동일하고 S2k ∈ prefixS1 이면, S1과 S2는 병행 관계라 정의한다. 여기서 i<k ≤m이고, prefixS1 = {S11,S12,...,S1i-1}이다.□

정의 4. 병행수행 과업집합

임의의 문자열 S1=S11S12...S1i...S1n과 S2=S21S22... S2i...S2m에서, S1i와 S2i에서 병행 관계이면, S2k ∈ prefixS1을 만족하는 ∪S2k와 S1i를 병행수행 과업 집합이라 정의한다. 여기서 i<k≤m이고, prefixS1 = {S11,S12,...,S1i-1}이다.□

각 과업 간 부분 순서관계를 결정하기 위해, 정렬된 문자열들로부터 각 과업의 후순위 과업을 결정한다. 정렬된 문자열로부터 특정 문자-과업식별자- 이후에 오는 문자는 널 문자나 과업식별자인 임의의 문자이다. 널 문자는 후순위 과업에 대한 부분순서 관계를 제시하지 못하기 때문에, 널 문자가 아닌 문자가 나올 때까지 해당 문자열을 검색하여, 후순위 과업을 결정한다. 각 과업에 대한 후순위 과업을 다음과 같이 정의한다.

정의 5. 후순위과업

정렬된 문자열 S1=S11S12...S1i...S1n과 S2=S21S22... S2i...S2m에 대해서 S1i에 대한 후순위 과업은 S1i+1과 S2i+1이다. 단, S1i+1(또는 S2i+1)이 널문자이면 S1i+2(S2i+2) 이후에 처음으로 널 문자가 아닌 S1j(S2j)를 의미한다.□

각 과업의 후순위를 결정함으로써 과업 간 부분순서 관계를 정의할 수 있다. 이를 알고리즘으로 표현하면 다음과 같다. 로그 데이터는 문자열로 변환되고 문자열에 나타난 모든 문자 집합이 입력된다고 가정한다.

```
Algorithm DetectFollowTasks
Input : Log Set LS, TaskIDs TIDs
Output:Following TasksIDs on each task FTIDS

Alignment Log Set ALS
ALS = Alignment(LS)
/* 병행 관계인 경우 병행수행 과업집합을 구성하고
정렬에서 제외 */
Each tid ∈ TIDs {
  FTIDs = ∅
  Each als ∈ ALS {
    fid = SearchNext(als,tid)
      if( fid == '-' ) {
        do {
          fid = searchNext(als,fid)
        }while(fid != '-')
      }
      FTIDs = FTIDS ∪ fid
  }
  Output(tid,FTIDs)
}
/*  Aligment(LS):로그 집합 LS를 정렬하여 반환
    tid, fid : 과업 식별자
    als : 정렬된 로그 집합에서 임의의 로그
    searchNext(als,tid):als상에서 tid이후에
       존재하는 과업 식별자를 반환
    Output(tid,FTIDS) : tid의 후순위 과업식별
       자 집합 FTIDS를 반환                  */
```

[예제 4] 예제 2에서 제시된 로그 정보에 대한 정렬 및 후순위 결정 과정을 제시한다. 먼저, 로그 정보에서 실패종료 과업 ftT은 명세를 수행하는 과정

에 대한 정보를 포함하고 있기 때문에 ftT 문자를 제거하고 문자열을 구성한다. {(sT,1,3,4,5,7,gtT), (sT,2,6,7,gtT),(sT,1,3,ftT),(sT,1,4,3,5,7,ftT)}를 {sT13457gtT,sT267gtT,sT13, sT14357}로 구성한다. 다음으로 가장 많은 명세정보를 가지고 있는 가장 긴 문자열을 기준으로 정렬을 수행한다.

S_1	sT	1	-	-	3	4	5	7	gtT
S_2	sT	-	2	6	-	-	-	7	gtT
S_3	sT	1	-	-	3				
S_4	sT	1	-	-	-	4 3	5	7	

	후순위 과업
sT	1,2
1	3,4
2	6
3	5
4	5
5	7
6	7
7	gtT
gtT	

먼저, 정렬과정에서 S4는 S1과 병행 관계를 갖게 되며, 병행 수행 과업집합은 {3,4}이다. 각 과업의 후순위 과업은 오른쪽 표와 같다. 병행수행 과업 집합에 속한 과업들의 후순위 과업은 AND 제어 흐름 특성에 의해 동일하다.□

셋째, 순환 수행 과업 정보와 과업 간 부분 순서 정보를 이용하여 워크플로우 그래프를 생성한다. 워크플로우 그래프를 생성하기 위해 먼저, 각 과업에 대해 노드로 정의하고 과업 간 제어 흐름은 후순위 과업 관계를 기반으로 간선을 지정한다. 부분 순서 관계에서 둘 이상의 후 순위 과업을 갖는 제어 흐름은 OR 제어 흐름으로 정의한다. 이는 로그 상에 선택적인 제어 흐름이 존재함을 의미하기 때문이다. 병행수행 과업집합에 속한 노드들은 AND 제어 흐름으로 정의한다. 이러한 과정을 통해 그림 1과 같은 워크플로우 그래프를 생성한다.

2. 동적인 워크플로우 그래프 탐사

본 연구에서 제안한 방안으로, 워크플로우 로그에서 워크플로우 그래프를 탐사할 때, 필요한 정보의 완전성(completeness)을 고려해야 한다. 워크플로우 로그에 모든 과업 정보 및 제어 흐름에 대한 정보를 포함하고 있느냐는 문제다. 모든 정보가 워크플로우 로그에 존재하지 않는다면 완전한 워크플로우 그래프의 생성은 불가능하기 때문이다. 즉, 예제 4에서 S1 문자열만으로는 그림 1과 같은 워크플로우 명세를 탐사할 수 없다. 본 연구에서도 모든 과업 정보와 제어 흐름이 워크플로우 로그에 포함되어 있다고 가정한다. 그러나 이러한 제약은 동적인 탐사를 통해 완화할 수 있다.

동적인 탐사 방법은 워크플로우 로그가 생성될 때마다 워크플로우 그래프를 재정의 함으로써 최종 명세를 생성하는 것이다. 워크플로우 그래프의 동적 탐사는 제안한 세 단계를 수행함으로써 달성한다. 먼저, 처음 생성된 로그에 대한 순환 여부를 정제하고, 정제된 로그에서, 각 과업에 대한 후순위 과업을 결정한다. 다음으로, 앞선 단계에서 생성된 그래프에 대해 후순위 과업과의 병렬 관계 존재여부를 검사한다. 병렬관계가 존재하면 병행수행 과업 집합을 결정하고, 병렬 관계가 없으면 후순위 과업간의 관계를 결정한다. 마지막으로, 후순위 과업 간 관계는 OR 제어 흐름으로 표현하고 병행수행 과업집합은 AND 제어 흐름으로 표현한다. 순환과업이 존재하면 이를 간선으로 그래프에 표현한다.

워크플로우 로그가 생성될 때마다 탐사 과정이 수행하기 때문에, 비연결성 그래프가 생성될 수 있다. 실패 종료 과업을 포함한 로그 데이터에 대한 수행은 비 연결성 그래프를 생성하게 한다. 이를 해결하기 위해, 모든 과업은 성공 종료 과업으로의 제어 전이를 기본으로 하며 실패 종료과업을 포함한 로그에는 실패 종료과업대신 수행해야 할 또 다른 과업이 존재한다고 가정한다. 이는 워크플로우의 수행 목적이 사용자의 요구를 달성-성공-하기 위함이고, 실패 종료과업으로의 전이는 해당 과업을 수행하다 성공할 수 없었기 때문에 발생하는 전이로 그 이후에도 수행할 과업이 존재함을 의미하기 때문이다. 이런 이유로 비연결성 그래프에 대해 최후 과업은 후순위 과업으로 성공 종료 과업을 포함하며 실패 종료 과업에 대한 전이는 임시 과업으로의 전이를 포함하게 명세 한다. 이를 포함한 알고리즘은 다음과 같다.

```
Algorithm DynamicWorkflowGraphMining
Input : Workflow Log WL
Output : Workflow Graph WG

Cycle tid Set CTS
Following Tasks on each task FTs
CTS = CycleFiltering(WL)
FTs = DetectFollowTasks(CTS)
Each ft ∈ FTs {
  /* 실패종료 과업으로의 전이를 임시 과업 tempT로
     전이하게하고 이를 성공종료 과업으로 전이함*/
  if( "ftT" ∈ ft ) {
    ft = ft - "ftT"
    FTs = FTs ∪ "tempTi"
    ft = ft ∪ "tempTi"
    MakeFollowingTask("tempTi","gtT",FTs)
  }
}
MakeWorkflowGraph(FTs,CTS)
/* ft:각 과업에 대한 후순위 과업 집합 FTs에
      속하는 임의의 후순위 과업
   MakeFollowingTask("tempTi","gtT",FTs)
     :FTs 집합에 임시과업 "tempTi"에서 "gtT"
      로의 후순위 관계 입력
   MakeWorkflowGraph(FTs,CTS)
     :FTs에 순환 과업 식별자 집합 CTS를 반영
      하여 그래프 생성          */
```

동적 탐사 중에 생성된 임시 과업은 어떤 작업도 수행하지 않는 과업으로 제어 구조의 연결을 위한 수단일 뿐이다. 임의의 과업이 후순위 과업으로 임시 과업과 또 다른 과업을 포함하고 있을 때, 임시 과업을 삭제한다. 이는 다른 과업으로의 전이가 발견되었음을 의미하기 때문이다.

[예제 5] 앞서 제시한 예제 4의 로그 데이터가 S3, S2, S1, S4 순으로 생성되었다고 가정할 때, 동적 탐사 과정은 다음과 같다.

S3 : sT 1 3 ftT

	후순위 과업
sT	1
1	3
3	tempT
tempT	gtT
gtT	

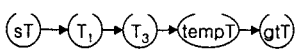

그림 4. 문자열 S3에 대한 워크플로우 그래프

Fig. 4. Workflow graph for string S3

S2 : sT 2 6 7 gtT

	후순위 과업
sT	1,2
1	3
2	6
3	tempT
6	7
7	gtT
tempT	gtT
gtT	

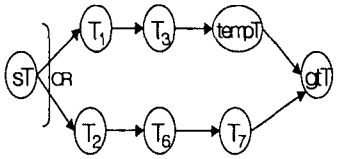

그림 5. 문자열 S3, S2에 대한 워크플로우 그래프

Fig. 5. Workflow graph for string S3 and S2

S1 : sT 1 3 4 5 7 gtT

	후순위 과업
sT	1,2
1	3
2	6
3	4
4	5
5	7
6	7
7	gtT
gtT	

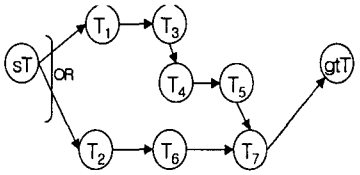

그림 6. 문자열S3, S2, S1에 대한 워크플로우 그래프

Fig. 6. Workflow graph for string S3, S2 and S1

S4 : sT 1 4 3 5 7

	후순위 과업
sT	1,2
1	3,4
2	6
3	5
4	5
5	7
6	7
7	gtT
gtT	

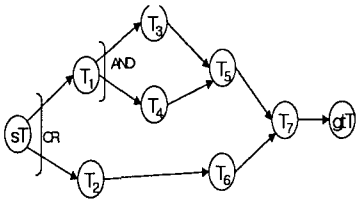

그림 7. 문자열 S3, S2, S1, S4에 대한 워크플로우 그래프

Fig. 7. Workflow graph for string S3, S2, S1 and S4

S1이 입력되었을 때 과업 3의 후순위 과업이 tempT 과업이외에 과업 4가 존재하기 때문에 tempT를 제거한다. S4가 입력되었을 때, 과업 3과 과업 4가 병행 관계에 의해 병행수행 과업으로 구분되고 이들을 AND 제어 흐름으로 표현한다. □

IV. 비교 분석

본 연구에서 제안한 워크플로우 명세 탐사 방법은 AND/OR의 제어 흐름을 탐사 할 수 있고, 워크플로우 시스템 수행 중에 동적으로 명세를 탐사할 수 있다. 이를 기존 연구와 비교하면 표 1과 같다.

본 연구에서 제안한 방안은 부분적인 과업 및 그들간의 제어흐름만으로도 명세를 탐사할 수 있으나, 페트리넷을 기반으로 한 방안[6,7]은, 로그 정보에 페트리넷 요소인 플레이스와 트랜지션의 상태 정보를 포함하고 있을 때 수행될 수 있는 방안이다. 트랜지션의 활성화(fire) 횟수에 대한 정보를 로그에 포함하고 있을 때, 워크플로우 명세를 탐사할 수 있다. 적응적 방안[11,12,13]은 로그로부터 탐사를 수행하는 과정에 더욱 구체화된 명세를 정의하기 위해 전문가의 요구를 반영한다.

로그 정보의 탐사 과정에서 발견할 수 있는 제어 흐름 관점에서, 본 연구에서 제안된 방안 외에는 부분적인 제어 흐름만을 고려한다. [6,7]에서는 순환이 없는 명세로부터 생성된 로그 정보를 기반으로 하고 있으며, 순차패턴 탐사를 응용한 방안[8,9,10]은 과업 간의 제어 흐름을 OR 흐름만으로 가정하고 탐사한다. 그러나, 본 연구에서에서 제안한 방안은 순환, AND 그리고 OR 제어흐름을 발견할 수 있다.

탐사기법 측면에서 제안한 방안과 적응적 방안만이 동적인 탐사를 수행한다. [6,7]에서는 모든 트랜지션에 대해, 각각의 활성화 횟수가 존재할 때만 탐사를 수행 할 수 있으며, [8,9,10]도 모든 과업 정보와 과업 간에 부분 순서관계가 존재할 때만 탐사를 수행할 수 있다. 그러나. 제안된 방안에서는 부분적인 과업과 제어 흐름만을 포함한 로그 정보로부터 워크플로우 시스템 수행 중에 동적으로 또한 점진적으로 명세를 탐사할 수 있다.

로그정보의 분석 방안으로써 제안된 통계적 분석[14]와 수행결과 검증[15]은 단순히 과업 및 워크플로우 시스템 수준의 통계 정보를 탐사하거나, 로그 정보를 기본으로 순환이 존재하는 지를 검증할 뿐, 본 연구에서 제안한 명세를 탐사하지 않는다. 그러나, [14]에서 제시한 각 과업에 대한 부하 분석이나 [15]에서 제안한 순환관계 발견은 명세 탐사를 위해 부가적인 정보로서 의미를 갖는다.

V. 결 론

본 연구에서는 워크플로우 로그에 저장된 수행정보로부터 워크플로우 명세를 추출할 수 있는 방안을 제안한다.

첫째, 수행정보를 기반으로 과업 간 부분순서 관계를 기반으로 다양한 모델링 요소를 포함한 워크플로우 명세의 탐사 방안을 제안한다. 먼저 워크플로우 로그로부터 순환 정보를 추출하고, 각 과업들 간의

표 1. 비교

Table 1. Comparesion

방안 \ 기준		목적	탐사가능 조건	발견하는 제어흐름	탐사기법
제안한 방안		워크플로우 명세 탐사	없음	순환, AND와 OR	정렬과 후순위 관계를 이용한 동적인 탐사기법
페트리넷을 기반으로 한 방안		"	페트리넷 모델을 기반으로한 로그 정보 필요	AND와 OR	발생빈도를 기반으로 한 정적인 탐사기법
순차패턴 탐사를 응용한 방안		"	없음	순환, OR	부분 순서 관계를 이용한 정적인 탐사기법
적응적 방안		"	전문가에 의한 재정의	AND와 OR	확률에 의한 부분순서관계를 이용한 동적인 탐사기법
로그정보 분석	통계적 분석	수치적 통계정보 산출	없음	-	통계적 분석
	수행결과 검증	명세와 탐사된 명세 간에 일치여부 검증	없음	순환	순환 제어 흐름 존재만을 탐사

후순위 관계를 정렬을 통해 정의하였다. 정렬된 정보와 병행수행 과업 집합 정보를 기반으로 워크플로우 명세를 탐사하였다. 본 연구에서는 업무 처리 과정에서 빈번히 발생하는 AND와 OR 제어흐름을 포함한 워크플로우 명세를 탐사 할 수 있다.

둘째, 동적인 워크플로우 명세 탐사방안을 제안한다. 워크플로우 로그가 생성될 때, 해당 정보와 기존에 탐색된 정보를 기반으로 새로운 워크플로우 명세를 탐사할 수 있다. 즉, 후순위 과업 관계에 대한 정보를 동적으로 갱신함으로써 달성한다. 이는 부분적인 과업과 제어 흐름만을 포함한 로그 정보로부터 워크플로우 시스템 수행중에 동적으로 또한 점진적으로 로그 정보를 반영하여 명세를 완성한다는 의미이다.

탐사 과정에 의해 발견된 명세는, 워크플로우 명세 도구에 의해 정의된 워크플로우 명세가 존재할 때 이와 비교함으로써 정확한 명세 탐사가 이루어졌는지 검증할 수 있다.

본 연구는 이질적인 시스템을 기반으로 워크플로우가 통합될 때, 각 지역 시스템의 워크플로우 명세를 탐사할 수 있으며 나아가 다른 명세들과 쉽게 통합할 수 있다. [17,18,19]에서 제안한 히스토리 관리 기법을 기반으로 이질적인 워크플로우 시스템들 간에 로그 정보를 획득할 수 있고 이를 기반으로 워크플로우 명세를 탐사할 수 있다. 더욱이 본 연구에서 제안한 동적인 탐사 기법은 로그 정보를 유지하고 있지 않는 워크플로우 시스템으로부터 현재 수행상태 정보만을 획득하고 이를 기반으로 워크플로우 명세를 탐사할 수 있다.

향후 연구에서는 워크플로우 로그를 성공종료과업과 실패종료과업 수행 여부에 따라 구분하여, 부가적인 제어 정보를 탐사하고자 한다. 사용자 요구에 대한 실패 경우수를 최소화 할 수 있는 제어 정보의 추출을 통해, 장기수행 특성에 의한 비용을 최소화 할 수 있는 연구를 수행중이다.

참고문헌

[1] D. Georgakopoulos, M. Hornik and A. Sheth," An Overview of Workflow Management: From Process Modeling to workflow Automation Infrastructure", Distributed and Parallel Database, Vol. 3, No. 1, pp. 119-154, 1995.

[2] A. Sheth(ed.). Proc. NSF Workshop on Workflow and Process Automation in Information Systems, May 1996.

[3] G. Alonso, D. agrawal, A. EL Abbadi and C. Mohan, "Functionality and Limitations of Current Workflow Management Systems", IEEE Expert, Vol. 12, No. 5, 1997.

[4] A. Dogac, L. Kalinichenko, M. T. Ozsu and A. Sheth(Eds.), Workflow Management Systems and Interoperability, Springer-Verlag, 1998.

[5] WFMC, "Workflow Management Coalition Audit Data Specification", WFMC-TC-1015, 1998.

[6] W.M.P van der Aalst, A.P. Barros, A.H.M. ter Hofstede and B. Kiepuszewski,"Advanced Work flow Patterns", 7th Int'l Conf. on Cooperative Information Systems, pp.198-209, Springer-Verlag, 2000.

[7] A.J.M.M. Weijters and W.M.P van der Aalst,"Process Mining. Discovering Work flow Models from Event-Based Data", Proc. of the 13th Belgium Netherlands Conf. on Artificial Intelligence, pp. 283-290, 2001.

[8] R. Agrawal and R. Srikant,"Mining Sequential Patterns", 11th Int'l Conf. on Dara Engineering, March 1995.

[9] R. Agrawal and D. Gunopulos, "Mining Process Models from Workflow Logs",6th Int'l conf. on Extending Database Technology, pp.469-483, 1998.

[10] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements", Proc. of the 5th Int'l Conf. on Extending Database Technology, pp. 3-17, 1996.

[11] J. Herbst,"A Machine Learning Approach to Workflow Management", Proc. of European Conf. on Machine Learning, pp.183-194, 2000.

[12] J. Herbstand, "Induction: A Solution for the

Acquisition and Adaptation of Workflow Models?", Intl. Process Technology Workshop, 1999.

[13] J. Herbst and D. Karagiannis, "Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models", Proc. of the 9th Int'l Workshop on Database and Expert Systems Applications, pp. 745-752. IEEE, 1998.

[14] M. Muehlen, "Workflow-based Process Controll-ing-Or: What You Can Measure You Can Control", Workflow Handbook 2001(Ed. Layna Fischer), pp. 61-77, 2001.

[15] A. Piramuthu,"Knowledge-based Approach to Validation Workflow Sequences", Int'l Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations, 2001.

[16] S.B. Needleman and C.D. Wunch,"A General Method Applicable to the Search of Similarities in the Amino Acid Sequence of Two proteins", Journal Molecular Biology, Vol.48, pp.443-453, 1970.

[17] P. Koksal, S. N. Arpinar and A. Dogac, "Workflow History Management", SIGMOD Record, Vol. 27, No. 1, 1998.

[18] P. Koksal, S. N. Arpinar and A. Dogac, "History Management in Workflow Systems", Intl. Symposium on Computer and Information Sciences, October, 1998.

[19] P. Muth, J. Weissenfels, M. Gillmann and G. Weikum," Workflow History Management in Virtual Enterprises using a Light-Weight workflow Management Systems", 9th Int'l Workshop on Research Issues in Data Engineering, March, 1999.

저자소개

정희택(Hee-Taek Jeong)

1992년 전남대학교 전자계산학과 졸업(이학사)
1995년 전남대학교 전자계산학과(이학석사)
1999년 전남대학교 전자계산학과(이학박사)
1999년~현재 여수대학교 정보과학전공 조교수
※관심분야: 워크플로우 시스템, 데이터 마이닝, 분산처리 시스템