워크플로우 임계 경로에 관한 분석
(Analyses on the Workflow Critical Path)

손 진 현 † 장 덕 호 †† 김 명 호 †††
(Jin Hyun Son) (Duk Ho Chang) (Myoung Ho Kim)

요 약 임계 경로는 방향성 비순환 그래프 분야를 포함하여 많은 컴퓨터 공학 분야들에서 널리 활용되어 왔다. 워크플로우에서 임계 경로는 여러 개의 예상 수행 경로들 중에서 가장 긴 평균 수행 시간을 가지는 하나의 경로로 정의되며, 이 개념은 워크플로우 응용 영역들에서 유용하게 사용될 수 있다. 일반적으로 특정 워크플로우에 대해 동시에 수행되는 워크플로우 인스턴스들이 여러 개가 존재하므로, 워크플로우 환경에 적합한 새로운 임계 경로 결정 방법의 개발이 필요하다.

본 논문에서 우리는 먼저 워크플로우 특징들을 쉽게 분석할 수 있는 워크플로우 대기 행렬 네트워크 모델에 대해서 언급한다. 그리고, 이 모델을 기반으로 워크플로우 임계 경로를 결정하는 방법을 제안한다. 추가적으로, 워크플로우 임계 경로 개념을 효율적으로 활용할 수 있는 워크플로우 응용 영역들을 몇가지 소개한다.

Abstract The critical path has been widely applied to many areas of computer engineering, especially a directed acyclic graph. Its concept can also be useful in the context of a workflow. The workflow critical path is defined as a path which has the longest average execution time from the start activity to the end activity of a workflow. Because there can be several concurrently executed workflow instances for a specific workflow, a new method to determine the critical path should be developed.

In this paper we specify our workflow queuing network model from which we can easily analyze many workflow characteristics. Based on this workflow model, we propose a method to identify the critical path. In addition, we show some workflow areas which can utilize the critical path.

1. 서 론

업무 프로세스는 규정된 업무 절차에 따라 문서, 정보, 혹은 작업 내용이 해당 담당자들에게 전달되면서 수행된다. 이러한 업무 프로세스의 일부 혹은 전체를 자동화하고자 하는 목적에서 워크플로우 개념이 도입되었다[1]. 업무 프로세스를 자동화하기 위해서 실제 업무는 워크플로우 정의(workflow definition)를 통해서 표현된다. 워크플로우 정의는 워크플로우 액티비티(workflow activity)들과 이들 액티비티들의 수행 관계를 규정하는 워크플로우 제어 조건(workflow control condition)들로 구성된다. 그러므로, 워크플로우 정의는 업무 프로세스에 대응되고 워크플로우 액티비티는 업무 프로세스를 구성하는 각 작업 단계에 대응한다. 그리고, 워크플로우 제어 조건은 업무 프로세스의 단위 작업들 사이의 업무 절차를 표현한다. 각 워크플로우 액티비티에게는 워크플로우 전체의 목적을 달성하기 위해 자신의 역할(role)이 정의되며, 이는 소프트웨어 프로세스 혹은 사람과 같은 에이전트(agent)에 의해서 수행된다. 워크플로우 인스턴스(workflow instance)는 워크플로우 정의에 대한 사용자의 구동 요청에 의해서 생성되고 워크플로우 관리 시스템(workflow management system)에 의해서 관리된다. 워크플로우 관리 시스템은 워크플로우 정의에서 표현된 제어 조건에 따라 워크플로우 참여자(workflow participant)들과의 상호 작용 및 워크플로우 애플리케

· 본 연구는 한국과학재단 특정기초연구(과제번호1999-1-303-007-3) 지원으로 수행되었음.
† 비 회 원 : 한국과학기술원 전자전산학과 연구원
jhson@dbserver.kaist.ac.kr
†† 비 회 원 : (주)디지탈아리아 대표
doung@digitalaria.com
††† 종신회원 : 한국과학기술원 전자전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2000년 11월 3일
심사완료 : 2001년 9월 6일

이션들을 활용하면서 워크플로우 인스턴스들의 수행을 완료한다[1][2].

최근에, 워크플로우에 대한 관심의 증가는 한편으로 현재의 분산되고 복잡한 업무 프로세스를 지원할 수 있는 고성능, 확장성, 신뢰성, 유연성, 상호 가용성 등과 같은 요구 사항들을 워크플로우에서 지원하기를 기대하고 있다. 이를 위해 많은 워크플로우 요소 기술들이 보다 체계적으로 연구되어야 할 필요성이 있다. 이러한 맥락에서, 본 논문에서는 워크플로우 환경에서 유용하게 활용될 수 있는 임계 경로(critical path)의 개념을 정의하고 그에 대해 면밀히 분석하고자 한다.

임계 경로의 개념은 컴퓨터 공학 분야에서 널리 연구되어 왔다. 예를 들면, 최근의 마이크로프로세서 설계에서 동작 속도를 높이기 위해 일반적으로 동적 회로(dynamic circuit)를 많이 사용한다. 그러나, 마이크로프로세서의 설계가 점점 복잡해짐에 따라 전체 설계에 대한 시간 검증이 쉽지 않다. 임계 경로의 개념은 이러한 복잡한 동적 회로 설계의 성능을 측정하는데 사용될 수 있다[3]. 병렬 분산 프로그램의 성능 측정 도구들에 대한 연구 분야에서 프로그램의 성능을 자동으로 알려주는 기술들이 많이 개발되었다. 이러한 기술들 중의 한 예는 프로그램의 실행 과정을 그래프로 표현하여 임계 경로를 찾는 방법을 사용한다[4]. 한 지점에서의 최단 경로(single-source shortest path)를 찾는 알고리즘은 방향성 비순환 그래프(directed-acyclic graph) 영역에서 잘 알려진 문제이다. 이 알고리즘의 한 응용 예는 프로젝트 관리를 위한 PERT 챠트(chart)에서 임계 경로를 결정하여 프로젝트의 시간 및 비용을 효과적으로 관리하는 것이다[5].

본 논문에서 워크플로우 임계 경로는 워크플로우 정의의 시작 액티비티에서 종료 액티비티까지의 가장 긴 평균 수행 시간을 가지는 하나의 경로로서 정의된다. 이러한 임계 경로의 분석은 워크플로우 자원 관리(workflow resource management), 워크플로우 시간 관리(workflow time management) 등과 같은 다양한 워크플로우 분야들에서 유용하게 사용될 수 있기 때문에 아주 중요하다. 예를 들면, 워크플로우 자원 관리 분야에서 임계 경로에 속하는 액티비티들에게 워크플로우 자원을 효율적으로 할당함으로써 전체 워크플로우의 성능을 개선시킬 수 있다. 그리고, 워크플로우 시간 관리 측면에서 임계 경로는 워크플로우 전체 완료 시간에 직접적으로 영향을 주기 때문에 임계 경로에 속하는 액티비티들에게 효율적으로 마감 시간을 할당함으로써 워크플로우 처리량(workflow throughput)을 증가시킬 수 있다.

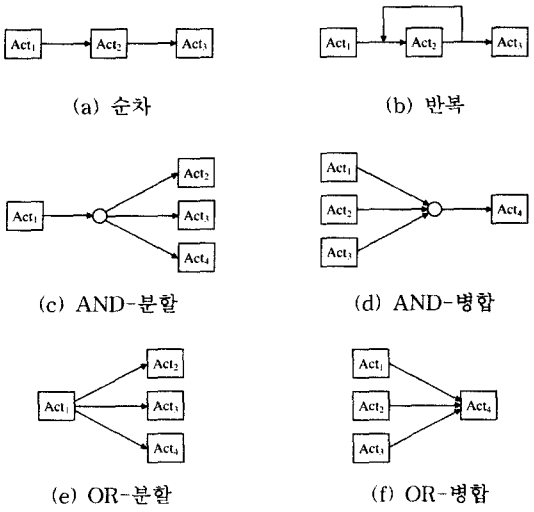
한편, 지금까지 제안된 임계 경로 결정 방법들을 그대로 워크플로우 환경에 적용하는데는 많은 제한이 있다. 프로젝트 계획, 수행 및 제어를 지원하는 PERT-CPM에서는 액티비티들이 순차와 병렬 구조로만 연결된 네트워크 환경에서 임계 경로를 고려하고 있지만[6], 워크플로우는 순차, 선택, 병렬, 반복 등과 같은 다양한 제어 구조들로 구성된 액티비티 네트워크이다. 그리고, PERT-CPM에서는 단지 하나의 수행 인스턴스만이 존재하는 건설과 같은 일회 수행 프로젝트를 고려하고 있다[6]. 그러므로, 임계 경로의 결정은 동시에 수행되는 인스턴스의 개수에 의해서 많은 영향을 받으므로 다수의 워크플로우 인스턴스들이 병렬적으로 실행되는 워크플로우 환경에서는 다른 방법의 제안이 필요하다. 그러므로, 본 논문에서는 워크플로우 특성에 적합한 임계 경로 결정 방법을 제안하고자 한다.

최근에 워크플로우 자원 관리와 관련하여 많은 연구가 이루어지고 있다. [7]에서는 워크플로우 정의에서 병목 지점으로 고려되는 특정 액티비티들에게 보다 많은 자원을 할당하여 액티비티들의 처리 용량을 증가시킴으로써 시간 제약성(time constraints)을 가진 워크플로우의 처리 성능을 개선시키는 방법에 대해 연구했다. [8]에서는 분산 워크플로우 환경에서 워크플로우 수행 중에 발생하는 통신 비용을 최소화하는 효율적인 워크플로우 작업 할당 방법이 제안되었다. 그리고, [9]는 분산 트랜잭션 워크플로우 시스템에 관하여 연구하였다.

워크플로우 액티비티 마감시간(deadline)의 효율적인 계산, 시간 제약성 관리, 그리고 에스컬레이션(escalation) 지원 등과 같은 워크플로우 시간 관리 분야들은 [10], [11], [12] 등에서 언급되고 있다. [13]에서는 만약 특정 워크플로우 인스턴스가 최종적으로 전체 워크플로우 마감시간을 만족하지 못할 것으로 예측되면, 그 워크플로우 인스턴스의 수행을 가능한 한 빨리 종료시키는 것이 바람직하다는 생각을 가지고 예측 워크플로우(predictive workflow)에 대해 언급하고 있다. [14]와 [15]는 액티비티 마감시간을 동적으로 조정하는 방법에 대해서 연구했다. 이들 연구들은 모두 분산 소프트 실시간(Distributed soft real-time) 환경에서의 마감시간 할당 방법을 언급한 [16]에 기반을 두고 있다. 한편, [17]는 워크플로우 환경에서 정적 마감시간을 효율적으로 할당하는 것이 필요함을 언급하고 그의 방법을 제안하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 기존의 관련 논문들에서 정의된 워크플로우 제어 조건들을 기반으로 본 논문에서 고려하는 워크플로우 모델을 언급한다. 이 모델을 기반으로 3장에서는 워크플로우 임계

경로를 찾는 방법을 제안한다. 4장에서는 워크플로우 임계 경로에 대한 분석과 실험을 통하여 본 논문에서 제시한 방법의 타당성을 검증하며, 마지막으로 5장에서는 본 연구의 공헌과 앞으로 해야할 일에 대해 언급을 하면서 결론을 맺는다.



(a) 순차 (b) 반복

(c) AND-분할 (d) AND-병합

(e) OR-분할 (f) OR-병합

그림 1 워크플로우 제어 조건들

2. 워크플로우 모델

2.1 워크플로우 실행 경로 제어

워크플로우 정의에서 액티비티들은 업무 수행 절차에 따라 서로 상관 관계를 가지며, 워크플로우 관리 시스템에 의해서 생성된 워크플로우 인스턴스는 이 절차에 따라 실행된다. 워크플로우 인스턴스는 그의 생명주기 동안 크게 순차적 실행 경로와 병렬적 실행 경로에 의해서 제어된다. 순차적 실행 경로는 액티비티들을 하나의 실행 제어에 의해서 순차적으로 실행시킨다. 즉, 선행 액티비티의 역할이 완전히 수행된 이후에만 다음 액티비티가 수행될 수 있다. 병렬적 실행 경로에서는 두개 이상의 액티비티들이 동시에 수행될 수 있어 한 워크플로우 인스턴스에 여러 개의 실행 제어가 존재하게 된다. 이러한 두가지 종류의 워크플로우 실행 경로들은 순차(Sequence), 반복(Iteration), AND-분할(AND-Split), AND-병합(AND-Join), OR-분할(OR-Split), OR-병합(OR-Join) 등과 같은 워크플로우 제어 조건(workflow control condition)들에 의해서 결정된다[1]. 다시 말해서, 병렬적 실행 경로는 AND-분할 제어 조건에서 시작하여 AND-병합 제어 조건으로 종료된다. AND-분할과 AND-병합을 제외한 순차, 반복, OR-분할 그리고 OR-병합 제어 조건들은 순차적 실행 경로를 만든다. 지금까지 대부분의 워크플로우 연구 분야들에서는 위에서 언급한 여섯가지 워크플로우 제어 조건들에 의해서 정의된 워크플로우 모델을 바탕으로 하고 있다[2][7][14][15][17][18].

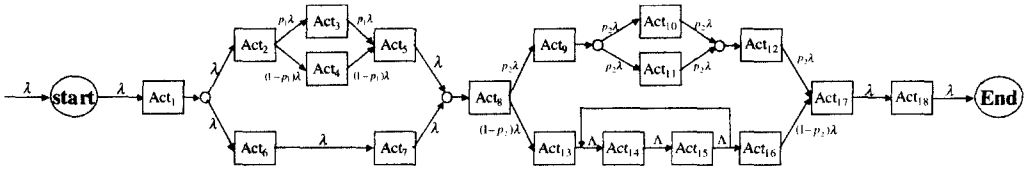
그림 1은 WfMC의 워크플로우 참조 모델에서 정의된 워크플로우 제어 조건들에 대한 도식적인 표현들이다[1]. 이들 제어 조건들은 표 1과 같은 의미를 가진다[19].

표 1 워크플로우 제어 조건들의 의미

제어 조건	설 명
순차	액티비티들은 하나의 실행 제어를 통해서 순서대로 수행된다. 즉, 선행 액티비티의 수행이 완료되기 전에 후행 액티비티가 수행될 수 없다.
AND-분할	하나의 실행 제어가 두개 이상의 실행 제어로 분할되어 동시에 여러 개의 액티비티들이 병렬적으로 수행되게 한다.
AND-병합	두개 이상의 병렬적으로 실행되고 있는 액티비티들을 동기화시켜 하나의 실행 제어를 형성하게 한다.
OR-분할	선택할 수 있는 워크플로우 경로들이 여러 개가 존재하여 이들 중의 한 경로로만 실행 제어를 전달하게 한다.
OR-병합	두개 이상의 선택적인 워크플로우 경로들을 하나의 액티비티에 모아 워크플로우 실행을 계속하게 한다. 병렬적인 액티비티 실행이 없기 때문에 동기화가 이루어지지 않는다.
반복	조건을 만족할 때까지 특정 워크플로우 액티비티들을 반복적으로 수행하게 한다.

일반적으로 분할 제어 조건은 병합 제어 조건과 결합하여 논리적으로 의미있는 워크플로우 제어 흐름을 가진다. 대부분 AND-분할은 AND-병합과 그리고 OR-분할은 OR-병합과 결합하게 되므로 본 논문에서는 (AND-분할, AND-병합) 쌍을 ***AND 제어 구조***라고 부르고 (OR-분할, OR-병합) 쌍을 ***OR 제어 구조***라고 부르기로 한다. 용어의 일관성을 유지하기 위해, 순차와 반복 제어 조건들은 각각 ***순차(Sequence) 제어 구조***와 ***반복(LOOP) 제어 구조***라고 부르기로 한다. 결과적으로, 워크플로우는 그림 2와 같이 본 논문에서 정의한 워크플로우 제어 구조들에 의해서 연결된 액티비티 네트워크가 되며, 한 워크플로우 제어 구조는 다른 워크플로우 제어 구조들을 포함하기도 한다. 그리고 순차 제어 구조를 제외한 나머지 제어 구조들(AND, OR, 반복)은 비-순차 제어 구조이다.

한편, [18]에서는 본 논문에서 정의한 워크플로우 제



λ, Λ : 도착률 p_i : 분기확률

그림 2 워크플로우 대기 행렬 네트워크

어 구조들 이외에 그림 3과 같이 병렬 실행에서 서로 다른 경로에 속하는 임의의 두 액티비티의 동기화를 지원하는 동기화 에지(synchronization edge)의 필요성에 대해서 언급했다. 즉, 동기화 에지 Act_i Act_j는 Act_i가 전혀 실행되지 않거나 혹은 실행이 완료되었을 때에만 Act_j가 실행될 수 있음을 의미한다. 그림 3은 Act_4에서 Act_8로의 동기화 에지를 보여준다.

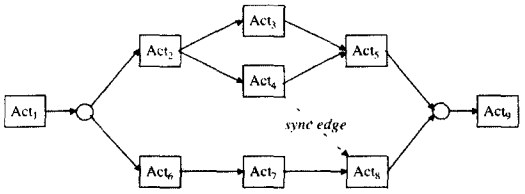


그림 3 동기화 에지

2.2 워크플로우 대기 행렬 네트워크

워크플로우 설계자가 워크플로우 제어 구조들에 대한 정확한 분석 없이 복잡한 업무 프로세스를 워크플로우로 정의할 때, 각종 논리적 및 구조적 오류들이 발생하기 쉽다[7][13]. 예를 들면, 그림 2에서 만약 반복 제어 구조의 피드백(feedback)이 Act_{14}대신에 Act_7이면 이 워크플로우 정의는 AND 제어 구조의 논리적 의미에 의해서 올바른 제어 흐름을 가지지 못한다. 다시 말해서, 반복 제어 구조의 피드백이 Act_7이면, 병렬적으로 수행되어 온 두개의 실행 제어를 동기화 시키는 AND-병합 액티비티 Act_8에서 오류가 발생한다. 이러한 면에서, 잘 정의된 워크플로우 구조(well-formed workflow structure)가 되기 위해서는 다음의 세가지 조건들을 만족해야 한다. 첫째, 임의의 AND-분할 조건은 반드시 AND-병합 조건과 쌍을 이루어야 한다. 둘째, 임의의 OR-분할 조건은 반드시 OR-병합 조건과 쌍을 이루어야 한다. 셋째, 비순차 제어 구조는 다른 비순차 제어 구조안에 완전히 포함될 수 있다. 그러나, 임의의 두개의 비순차 제어 구조들이 부분적으로 서로 겹쳐져서는 안된다. 여기서, 두개의 비순차 제어 구조들이 부분적으로 겹친다는 것은 한 제어 구조에 속하는 액티비티가 다른 제어 구조의 내부에 존재하는 것을 의미한다.

그림 2의 예에서, AND-분할점 Act_1에 대해 대응되는 AND-병합점 Act_8이 있고, OR-분할점 Act_2에 대해 대응되는 OR-병합점 Act_5가 존재한다. 그리고, 반복 제어 구조의 피드백 액티비티는 Act_7과 같은 액티비티가 될 수 없다. 왜냐하면, AND 제어 구조와 반복 제어 구조가 서로 부분적으로 겹치게 되기 때문이다.

정의 2.1 워크플로우에서 하나의 비순차 제어 구조가 다른 워크플로우 제어 구조를 완전히 포함할 때, 이러한 워크플로우를 **중첩 워크플로우(nested workflow)**라고 부른다. 중첩 워크플로우에서 가장 외부의 비순차 제어 구조를 비순차 제어 블록 혹은 간단히 **제어 블록(control block)**이라고 부른다.

그림 2에서 묘사된 워크플로우는 AND 제어 블록(Act_1에서부터 Act_8까지)과 OR 제어 블록(Act_8에서부터 Act_{17}까지)의 두개의 제어 블록들을 가진 중첩 워크플로우이다. AND 제어 블록은 OR 제어 구조를 포함하고, OR 제어 블록은 AND와 반복 제어 구조를 포함한다. 본 논문에서는 그림 2에서와 같은 중첩 워크플로우가 항상 하나의 시작점과 하나의 종료점을 가지게 하기 위해 Start와 End의 두개의 가상 노드를 추가한다. 지금부터 다루어 지는 워크플로우는 특별한 언급이 없는 한 위에서 정의된 중첩 워크플로우임을 가정한다.

우리의 일상 생활 주변에 존재하는 많은 물리적 혹은 유기적 프로세스들은 포아송(Poisson) 프로세스로 잘 모델링 될 수 있는 현상을 가지고 있다. 그리고, 지수(Exponential) 분포는 서비스 시설에서의 서비스 시간 분포를 적절히 모델링할 수 있다. 이러한 이유로 포아송 도착 프로세스와 지수 서비스 시간 분포는 지금까지 많은 전산 분야들의 현상을 분석하는데 사용되어 왔다[20]. 본 논문에서도 역시 워크플로우 요청은 포아송 프로세

스로 그리고 각 액티비티 에이전트는 지수 서비스 시간 분포로 모델링 하며, 아래와 같이 두가지 가정을 한다.

가정 2.1 각 워크플로우 액티비티의 큐(queue)는 FCFS(First Come-First Served) 정책을 가진다. 즉, 워크플로우 인스턴스들은 그들이 도착한 순서대로 수행된다.

가정 2.2 워크플로우 액티비티의 큐는 많은 워크플로우 인스턴스들을 수용할 수 있을 만큼 충분히 크다.

결과적으로, 우리는 워크플로우를 기존의 전화 혹은 컴퓨터 통신 네트워크와 같이 M/M/1 대기 행렬 네트워크로 모델링할 수 있다. M/M/1 워크플로우 대기 행렬 네트워크에서 각 액티비티는 하나의 독립된 M/M/1 대기 행렬 시스템임이 알려져 있다 [21]. 다시 말해서, 액티비티의 역할을 수행하는 액티비티 에이전트는 단일 쓰레드(single-threaded) 에이전트를 의미하는 M/M/1 대기 행렬 시스템으로 모델링된다. 그러나, 우리는 다음과 같은 두가지 상황을 고려해야 한다. 하나는 액티비티들이 다중 쓰레드(multi-threaded)를 가지는 다중 에이전트들에 의해서 수행될 수 있다는 것이고, 다른 하나는 한 에이전트가 서로 다른 워크플로우에 속하는 여러 액티비티들의 역할을 수행하는데 사용될 수 있다는 것이다. 실제로, 다중 쓰레드를 가지는 다중 에이전트들은 동시에 여러 개의 워크플로우 인스턴스들을 지원할 수 있기 때문에, 하나의 다중 쓰레드 에이전트는 일반적으로 M/M/c로 모델링 되고 다중 쓰레드-다중 에이전트들은 y개의 M/M/c로 모델링 된다. 여기서 c는 쓰레드의 개수이고 y는 다중 에이전트들의 개수이다. 그리고, 한 에이전트가 서로 다른 워크플로우에 속하는 액티비티들을 지원할 때, 이 에이전트의 서비스율(service rate)은 다음에 수행되는 액티비티에 따라 다르기 때문에 M/M/1으로 모델링되기가 어렵다. 이러한 경우는 에이전트에서의 평균 실행 시간의 계산과 같은 분석을 어렵게 만든다. 그러나, 이상에서 언급된 두가지 경우를 근사적으로 M/M/1 단일 쓰레드 에이전트로 쉽게 변환할 수 있기 때문에 본 논문에서는 각 액티비티를 하나의 독립된 M/M/1 대기 행렬 시스템으로 간주한다.

M/M/1 대기 행렬 네트워크에서 워크플로우의 초기 도착률, 각 액티비티 에이전트의 서비스율, 그리고 OR와 반복 제어 구조에서의 분기 확률 등의 정보를 활용하여 그림 2에서 처럼 각 액티비티에서의 도착 및 출발 프로세스를 명시할 수 있다. 이는 [21]에서 언급된 시간 역행성(time reversibility)의 성질과 아래의 사실들에 바탕을 두고 있다.

M/M/1 시스템의 도착 프로세스가 포아송 프로세스라면 이 시스템의 출발 프로세스 역시 포아송 프로세스임이 알려져 있다[20]. 그러므로, 순차 제어 구조에 속하는 모든 액티비티들은 포아송 도착 및 출발 프로세스를 가진다. 그리고, 그림 2의 Act_2, Act_5와 같이 OR 제어 구조에서 서로 독립적인 포아송 프로세스들의 분할과 병합 역시 포아송 프로세스라고 알려져 있다[21]. 그러므로, OR 제어 구조에 속하는 모든 액티비티들도 역시 포아송 도착 및 출발 프로세스를 가진다. AND 제어 구조의 경우, Act_1과 같은 AND-분할 액티비티에서의 도착과 출발 프로세스는 명백히 포아송 프로세스이지만, Act_N과 같은 AND-병합 액티비티에서는 병렬적으로 수행되어온 각각의 요청들이 AND-병합 액티비티에서 동기화가 되기 때문에 AND-병합 액티비티에서의 도착은 실제로 포아송 프로세스로 모델링 되기가 어렵다. 그러나, 아래의 사실에 주목할 필요가 있다. n개의 동시 실행 경로를 가지는 AND-분할 액티비티에서 한 워크플로우 인스턴스는 병렬적으로 수행되는 n개의 하위 요청들로 분할된다. 이들 n개의 하위 요청들은 AND-병합 액티비티에서 서로 동기화되어야 한다. AND-병합 액티비티에서의 이러한 동기화는 n개의 하위 요청들 중에서 가장 늦게 AND-병합 액티비티에 도착하는 하나의 하위 요청에 의해서 결정된다. 이러한 이유로 AND-병합 액티비티의 도착 프로세스는 n개의 경로들 중에서 가장 긴 평균 실행 시간을 가지는 경로를 통해 도착하는 요청들의 프로세스에 근사적으로 접근한다. 그러므로, 본 논문에서 AND-병합 액티비티에서의 도착 프로세스는 위의 경로를 통하는 요청들의 포아송 프로세스로 간주할 수 있다. 한편, 반복 제어 구조는 이전에 수행한 액티비티들을 반복적으로 재 수행하게 하는 피드백(feedback)을 가지고 있다. 이러한 피드백으로 인해서 반복 제어 구조의 내부 흐름은 포아송 프로세스가 아님에도 불구하고, 반복 제어 구조에 속하는 각 액티비티들은 마치 독립적인 M/M/1 시스템들 처럼 동작한다고 알려져 있다[22]. 그러므로, 반복 제어 구조 역시 M/M/1 대기 행렬 네트워크의 한 부분으로 동작할 수 있음을 알 수 있다. 결론적으로, 본 논문에서 정의한 중첩 워크플로우는 모든 액티비티들이 포아송 도착 및 출발 프로세스를 가지는 M/M/1 대기 행렬 네트워크로 모델링될 수 있다. 다음 장에 설명하는 임계 경로 결정 알고리즘은 이러한 M/M/1 워크플로우 대기 행렬 네트워크 모델을 기반으로 하고 있다.

3. 임계 경로

3.1 CPI 알고리즘

워크플로우 임계 경로는 워크플로우의 시작 노드에서 종료 노드까지의 많은 예상 경로들 중에서 가장 긴 평균 수행 시간을 가지는 하나의 경로로 정의된다. 본 논문에서는 2장에서 언급된 M/M/1 대기 행렬 네트워크로 모델링되는 워크플로우 정의에서 임계 경로를 결정하는 CPI(Critical Path Identification) 알고리즘을 제안한다. 일반적으로 한 워크플로우 정의에는 AND 제어 구조에 의해서 병렬적으로 수행되는 경로들이 존재한다. 또한 동시에 많은 워크플로우 인스턴스들이 수행되기 때문에, 임의의 액티비티에서 어떤 워크플로우 인스턴스들은 그 액티비티에 먼저 도착한 워크플로우 인스턴스들이 완전히 수행될 때 까지 기다려야 한다. 이러한 사실들은 한 액티비티에서 임의의 워크플로우 인스턴스의 평균 수행 시간은 액티비티 에이전트의 평균 서비스 시간과 액티비티의 큐에서 기다리는 평균 대기 시간의 합으로 결정됨을 의미한다. 그러므로, 본 논문에서 제안된 CPI 방법은 워크플로우 제어 구조들의 특성을 분석하면서 액티비티에서의 평균 수행 시간에 기반을 두고 있다.

그림 4는 CPI 방법에 의해서 임계 경로를 결정하는 전체 예제를 보여준다. 임계 경로의 정의에 따라 워크플로우 제어 블록(control block)에 속하지 않는 액티비티들과 각 제어 블록에서 가장 긴 평균 수행 시간을 가지는 경로는 임계 경로에 속함을 쉽게 알 수 있다. 그러므로, 그림 4-(a)에서 Act_{18}은 기본적으로 임계 경로에 속한다. CPI 알고리즘의 주요 기능은 각 제어 블록에서 가장 긴 수행 시간을 가지는 부임계 경로(sub-critical path)을 결정하고 이들 부임계 경로들과 제어 블록에 속하는 않는 액티비티들을 연결함으로써 최종적으로 임계 경로를 결정하는 것이다. 임의의 제어 블록에서 부임계 경로의 결정은 제어 블록에 속하는 가장 내부의 제어 구조에서부터 시작하여 가장 외부의 제어 구조들을 분석함으로써 이루어진다.

그림 4-(a)에는 AND와 OR의 두개의 제어 블록들이 존재한다. AND 제어 블록은 OR 제어 구조를 포함하고, OR 제어 블록은 AND와 반복 제어 구조들을 포함하고 있다. Act_2에서부터 분할되는 가장 내부의 OR 제어 구조에 대해서, 가장 긴 수행 시간을 가지는 하나의 경로를 선택하고 이 경로를 하나의 액티비티로 변환한다. 이때 OR 제어 구조에 대응되는 이 액티비티는 OR-분할 액티비티의 도착률과 동일한 도착률을 가져야 한다. 이는 그림 4-(b)의 Act_4'가 된다. Act_9에서부터 분할되는 가장 내부의 AND 제어 구조에 대해서 가장 긴 수행 시간을 가지는 하나의 경로를 선택하고, 가장 내부의 반복 제어 구조에 대해서는 대응하는 순차 제어 구조로 변환하여 그림 4-(b)에서 처럼 Act_{14}'와 Act_{15}'를 얻는다. 최종적으로, 각 제어 블록에서 부임계 경로

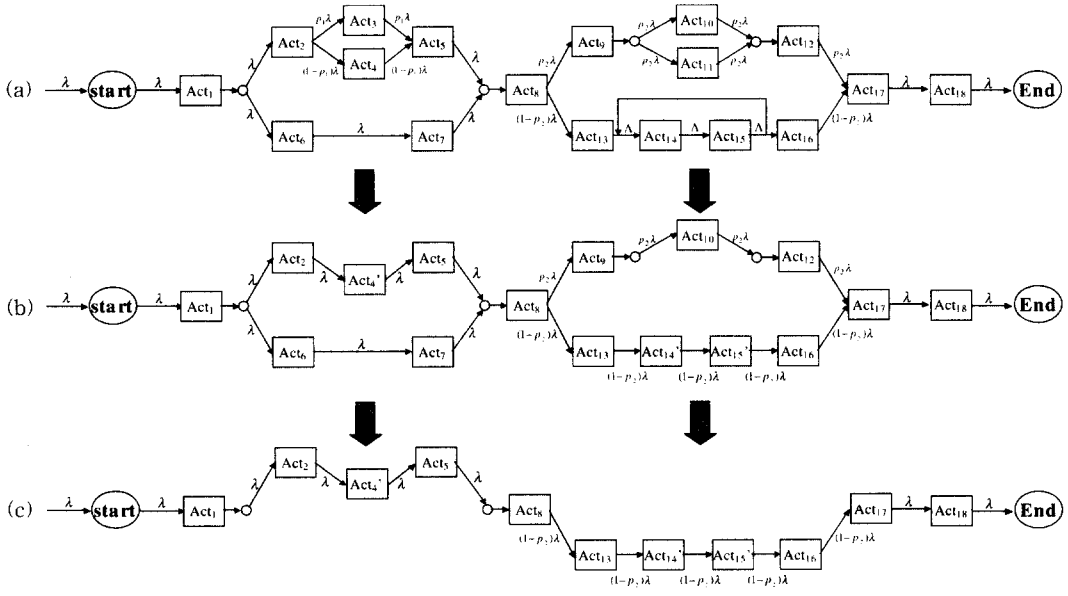


그림 4 임계 경로를 결정하는 예제

를 찾게되고 이들을 서로 연결하여 그림 4-(c)에서 처럼 워크플로우 임계 경로를 결정하게 된다.

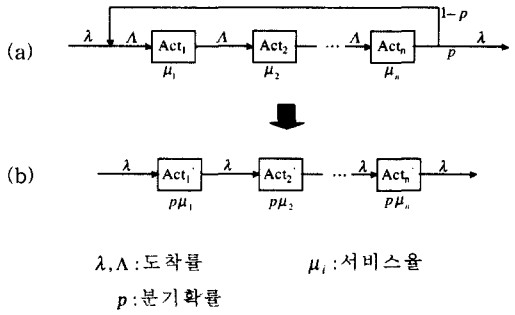
```
1. 제어 블록에 속하지 않는 모든 액티비티들은 항상 임계 경로에 속한다.
2. For (각각의 제어 블록)
   {
   3. WHILE (중첩 제어 블록의 존재)
      /* 이 WHILE 문의 종료시에 중첩 제어 블록은 하나의 단일 제어 구조로 변환된다. */
      {
      /* 중첩 제어 블록에 가장 내부의 제어 구조들이 여러 개가 존재할 때, 이들의 변환 순서는 무관하다. */
      4. IF (가장 내부의 제어 구조가 반복 제어 구조일 때)
         4.1 반복 제어 구조를 순차 제어 구조로 변환.
      5. ELSE IF (가장 내부의 제어 구조가 AND일 때)
         5.1 AND 제어 구조 내부의 여러 경로들 중에서 가장 긴 평균 수행 시간을 가지는 하나의 경로를 선택.
      6. ELSE IF (가장 내부의 제어 구조가 OR일 때)
         6.1 OR 제어 구조 내부의 여러 경로들 중에서 가장 긴 평균 수행 시간을 가지는 하나의 경로를 선택.
         6.2 이 경로를 한 액티비티로 변환.
   }
   /* 여기서, 중첩 제어 블록은 하나의 제어 구조로 변환됨. */
   7. 중첩 제어 블록에 대응되는 제어 구조에서 가장 긴 수행 시간을 가지는 경로를 선택하면 이 제어 블록의 부임계 경로가 된다.
}
8. 제어 블록에 속하지 않는 액티비티들과 부임계 경로들을 결합하여 워크플로우 임계 경로를 결정한다.
```

그림 5 CPI 알고리즘

CPI 알고리즘은 그림 5에서 설명하고 있다. 이 알고리즘의 주요 부분은 임의의 제어 블록에서부터 하나의 순차 경로를 결정하는 것으로 각 제어 구조에 대한 구체적인 방법들은 아래의 3.2, 3.3, 3.4장에서 설명될 것이다. 제어 블록에서 부임계 경로의 추출은 제어 블록의 가장 내부의 제어 구조에서부터 가장 외부의 제어 구조로 진행된다. 모든 부임계 경로들과 제어 블록들에 속하지 않는 액티비티들을 연결하면 워크플로우 임계 경로가 결정된다. 임의의 워크플로우 정의에서 제어 블록의 수를 c라고 하고 d는 제어 블록의 중첩 깊이라고 할때, CPI 알고리즘의 복잡도는 대략적으로 O(cd)이다. 워크플로우 정의에서 액티비티의 수를 n이라고 할 때, c * d는 일반적으로 n보다 작기 때문에 실제로 CPI 알고리즘의 계산 복잡도는 크지 않다.

다음 장에서는 부임계 경로를 결정하는 알고리즘에서 각 워크플로우 제어 구조들에게 적용되는 과정들, 즉 CPI 알고리즘에서 4, 5, 6 단계들에 대해서 상세히 설명한다.

3.2 반복 제어 구조



λ, Λ : 도착률 μ_i : 서비스율

p : 분기확률

그림 6 LOOP 제어 구조의 변환

그림 6-(a)에서와 같이 제어 블록에서 가장 내부에 존재하는 반복 제어 구조에 대해서 고려하자. 제어 블록에서 가장 내부의 제어 구조는 다른 비 순차 제어 구조들을 포함하지 않기 때문에, 가장 내부의 제어 구조는 순차 경로들에 의해서 연결된 액티비티들로만 구성됨을 주목할 필요가 있다. 반복 제어 구조에 속하는 각 액티비티는 2.2장에서 언급한 것처럼 하나의 독립적인 M/M/1 시스템으로 간주될 수 있기 때문에, 반복 제어 구조는 그림 6에서와 같이 하나의 순차 제어 구조로 변환될 수 있다. 이 과정은 CPI 알고리즘의 단계 4에 속한다. 반복 제어 구조의 피드백을 고려하여 그의 총 도착률을 Λ라고 하자. 반복 제어 구조의 최초 도착률이 λ이고 피드백 도착률이 (1-p)Λ이기 때문에, 총 도착률 Λ은 다음과 같이 계산된다.

$$\Lambda = \lambda + (1-p)\Lambda$$
$$\Lambda = \frac{\lambda}{p}$$

여기서 1-p는 Act_n 이후에 발생하는 반복 제어 구조의 피드백 확률이다.

그림 6-(a)에서 $\rho_1, \rho_2, \ldots, \rho_n$을 각각 Act_1, Act_2, ..., Act_n 액티비티 앞에서 기다리는 서비스 요청들의 평균 수라고 하면, ρ_i은 $\Lambda/\mu_i = \lambda/p\mu_i$(i=1, ..., n)가 된다. 반복 제어 구조의 평균 수행 시간 WLOOP은 반복 제어 구조에 속하는 모든 액티비티들의 평균 수행 시간들의 합이기 때문에,

$$W_{LOOP} = \left(\frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2} + \ldots + \frac{\rho_n}{1-\rho_n}\right)\frac{1}{\lambda} = \frac{1}{p\mu_1 - \lambda} + \frac{1}{p\mu_2 - \lambda} + \ldots + \frac{1}{p\mu_n - \lambda} \quad (1)$$

여기서 Act_i에서의 평균 수행 시간은 $\rho_i/(1-\rho_i)\lambda$ ($\rho_i = \lambda/p\mu_i$)이다. 식 (1)은 반복 제어 구조의 평균 수행 시간 W_{LOOP}이 그림 6-(b)에서 처럼 도착률 λ와 서비스율 μ_i을 가지는 n개의 액티비티로 구성된 순차 제어

구조의 평균 수행 시간과 동일함을 의미한다. 그러므로, 반복 제어 구조는 CPI 알고리즘의 단계 4에서 하나의 순차 제어 구조(그림 6-(b))로 변환된다.

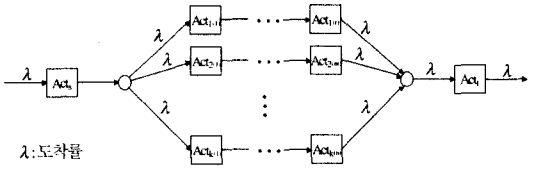


그림 7 AND 제어 구조

3.3 AND 제어 구조

그림 7과 같이 여러 개의 순차 경로들로 구성되는 AND 제어 구조가 제어 블록의 가장 내부의 제어 구조인 경우를 생각하자. 이는 CPI 알고리즘의 단계 5에 해당되는 것으로, AND 제어 구조에서 가장 긴 수행 시간을 가지는 하나의 경로를 선택한다. 서비스율 μ와 도착률 λ를 가지는 액티비티의 평균 수행 시간인 $W=1/(\mu_i-\lambda)$로부터, 총 평균 수행 시간 $MAX(\sum_i 1/(\mu_i-\lambda))$인 경로를 선택한다. 여기서 λ는 도착률이고 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다.

2장에서 언급한 것처럼 AND 제어 구조에서 그림 3의 Act_4 ➡ Act_N와 같은 동기화 에지를 고려해보자. 여기서 액티비티 Act_1와 Act_N는 AND 제어 구조에서 병렬적으로 수행되는 서로 다른 경로에 속한다. 이때, 액티비티 Act_N을 포함하는 경로의 평균 수행 시간은 액티비티 Act_1을 포함하는 경로의 평균 수행 시간에 영향을 받게 된다. 만약 Act_N의 바로 이전 액티비티(Act_7)가 완료되기까지의 총 평균 수행 시간이 동기화 에지에서 Act_1가 완료되기 까지의 총 평균 수행 시간보다 작다면, 전자의 평균 수행 시간은 후자의 평균 수행 시간으로 재설정된다. 그 이외의 경우에는, 동기화 에지가 Act_N을 포함하는 경로의 총 평균 수행 시간에 전혀 영향을 미치지 않는다. 그러므로, 본 논문에서 제안된 CPI 알고리즘은 이러한 동기화 에지의 특성을 쉽게 지원하도록 확장될 수 있다.

3.4 OR 제어 구조

앞에서 언급한 AND 제어 구조에서와 같이 k개의 순차 경로들로 구성된 OR 제어 구조가 제어 블록의 가장 내부의 제어 구조인 경우를 고려하자. 그림 8-(a)에서처럼 각 경로의 도착률은 $p_i\lambda\left(\sum_{i=1}^{k} p_i=1\right)$이다. CPI 알고리즘의 단계 6.1에서 언급한 것과 같이, OR 제어 구조

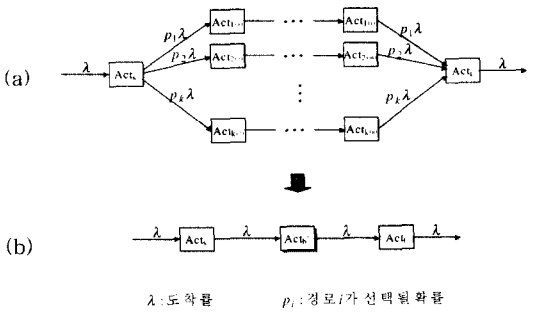


그림 8 OR 제어 구조

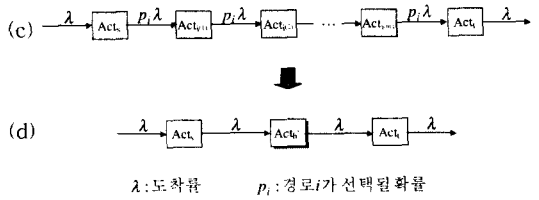


그림 9 순차 경로를 하나의 액티비티로의 변환

에서 총 평균 수행 시간이 $MAX\left(\sum_{i=1}^{k}\frac{1}{\mu_i-p_i\lambda}\right)$인 가장 긴 평균 수행 시간을 가지는 경로를 선택한다. 여기서 $p_i\lambda$는 이 경로의 도착률이고 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다.

위의 경로를 선택한 후, CPI 알고리즘의 단계 6.2에서 언급한 것처럼 이 경로를 하나의 액티비티로 변환한다. 이러한 변환의 필요성은 다음과 같은 이유에 근거를 둔다. $Act_{i(1)}$, ..., $Act_{i(m)}$을 OR 제어 구조에서 가장 긴 평균 수행 시간을 가지는 경로라고 하고, $p_i\lambda$을 이 경로에 속하는 각 액티비티에서의 도착률이라고 하자. 그림 9-(a)은 이러한 경우를 보여주고 있다. 순차 경로에 속하는 모든 액티비티들은 동일한 도착률을 가져야 하기 때문에, 그림 9-(a)은 그림 9-(b)로 변환되어야 한다. 이 과정에서 우리는 액티비티 Act_b'의 도착률이 OR-분할 액티비티의 도착률 λ와 동일하도록 액티비티 Act_b'의 서비스율 μ'을 아래와 같이 계산할 수 있다. 물론 변환 과정에서 액티비티 Act_b'는 순차 경로 $Act_{i(1)}$, ..., $Act_{i(m)}$와 동일한 평균 수행 시간을 가지도록 해야 한다.

그림 9와 같이 액티비티 Act_b'의 서비스율을 μ'라 하고 OR 제어 구조에서 가장 긴 평균 수행 시간을 가지는 경로에 속하는 액티비티 $Act_{i(j)}$(j=1,...,m)의 서비스율을

$\mu_{k(j)}$라고 하자. 그리고, OR 제어 구조에서 이 경로가 선택될 확률을 p_i라 하자. 이때, 액티비티 Act_b'의 평균 수행 시간 $\frac{1}{\mu'-\lambda}$은 이 경로의 총 평균 수행 시간인 $\sum_{j=1}^{m}\frac{1}{\mu_{k(j)}-p_i\lambda}$와 같아야 하므로 $\mu'-\frac{1}{\sum_{j=1}^{m}\frac{1}{\mu_{k(j)}-p_i\lambda}}+\lambda$이다.

지금까지 3.2, 3.3, 3.4장을 통하여 우리는 각 제어 블록에서 부임계 경로를 결정하는 주요 과정으로 CPI 알고리즘의 단계 4, 5, 6을 설명하였다. 이를 통하여 모든 제어 블럭에서 부임계 경로를 찾을 수 있고 최종적으로 워크플로우 임계 경로를 결정할 수 있다.

4. 분석 및 실험

본 논문에서 정의한 워크플로우 영역에서 임계 경로는 워크플로우의 시작 노드에서 종료 노드까지의 여러 수행 경로들 중에서 가장 긴 평균 수행 시간을 가지는 경로로 정의된다. 임계 경로의 특징은 워크플로우 실행시에 임계 경로를 지나가는 워크플로우 인스턴스들이 다른 경로를 지나가는 워크플로우 인스턴스들보다 워크플로우 마감 시간과 같은 시간 제약성을 만족시키지 못할 확률이 높다는 것이다. 이러한 워크플로우 임계 경로의 속성은 워크플로우 시간 관리(Workflow Time Management), 워크플로우 자원 관리(Workflow Resource Management) 등과 같은 연구 분야에 효과적으로 활용될 수 있을 것이다. 여기서 우리는 워크플로우 임계 경로에 대한 아래 두가지 사항을 주목할 필요가 있다.

1. 한 워크플로우 정의(workflow definition)로부터 생성되는 모든 워크플로우 인스턴스들이 항상 워크플로우 임계 경로를 지나가는 것은 아니다.
2. 워크플로우 마감 시간을 만족시키지 못하는 워크플로우 인스턴스들 중에서 워크플로우 임계 경로를 지나는 워크플로우 인스턴스 수가 다른 경로들과 비교하여 항상 가장 많은 것은 아니다.

일반적으로 워크플로우는 순차, 병렬, 선택, 반복 등과 같은 다양한 제어 구조들로 구성된다. 여기에서 선택 제어 구조는 여러 경로들 중에서 조건을 만족시키는 특정 경로만을 선택하여 수행시킨다. 그러므로 위의 첫번째 언급한 항목은 이러한 선택 제어 구조의 속성으로 인하여 발생하게 된다. 이러한 사항은 기존의 임계 경로 분석 알고리즘인 CPM/PERT를 워크플로우 임계 경로의 결정에 사용되는데 제한점이 있음을 보여주고 있다. 다시 말해서, CPM/PERT는 모든 단위 작업(워크플로우에서는 액티비티에 해당됨)들이 항상 수행되는 환경에 대해서 임계 경로를 분석하는 방법이기 때문에 워크플로우 환경에 그대로 적용되기는 어렵다. 두번째 언급한 항목은 첫번째 항목에 대한 영향으로서 워크플로우 임계 경로보다 전체 수행 시간은 짧은 경로이지만 많은 워크플로우 인스턴스들이 지나가게 되어 상대적으로 마감 시간을 만족시키지 못하는 워크플로우 인스턴스(즉, 에스컬레이션되는 인스턴스)의 수가 많이 발생할 수 있다. 그러므로 워크플로우 성능 향상과 관련하여 전체 에스컬레이션되는 인스턴스의 수를 감소시키고자 하는 연구 분야에서는 본 논문에서 제안한 워크플로우 임계 경로의 개념에 대한 보완이 필요하다. 그러나, 워크플로우 임계 경로의 정의에 의해서 워크플로우 임계 경로를 지나는 인스턴스들이 마감 시간을 어기는 비율은 다른 어떤 경로들보다 높음을 기대할 수 있으며 이는 아래의 실험에서 검증된다.

워크플로우 임계 경로에 관한 실험은 DEVSim++을 사용한다. DEVSim++은 통신 네트워크 설계, 병렬 컴퓨터 구조 설계 등과 같은 다양한 시스템 설계 분야에서 많이 사용되어 온 C++ 기반의 이산 사건 모델링 프레임워크이다. 본 논문에서는 DEVSim++을 이용하여 그림 10과 같은 워크플로우 정의에서 포아송 도착률 3으로 생성된 10,000 개의 워크플로우 인스턴스들이 지나가는 경로에 따라 에스컬레이션되는 인스턴스들의 수와 그 비율을 실험한다. 여기서 워크플로우 마감 시간은 6sec로 하고, 그림 10의 각 워크플로우 액티비티의 서비스율은 표 2를 기반으로 한다.

그림 10의 워크플로우 정의는 두개의 OR 제어 블록들로 구성되며 6가지의 서로 다른 수행 경로를 가진다. 첫번째 OR 제어 블록은 3가지의 수행 경로가 존재하는데 액티비티 1, 2, 5를 경로 a, 액티비티 1, 3, 5를 경로

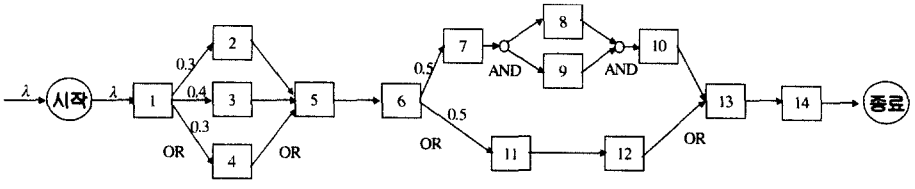


그림 10 워크플로우 정의

표 2 워크플로우 액티비티들의 서비스 시간

액티비티	서비스 시간	액티비티	서비스 시간
1	0.15 sec	8	0.05 sec
2	0.2 sec	9	0.15 sec
3	0.3 sec	10	0.1 sec
4	0.05 sec	11	0.3 sec
5	0.25 sec	12	0.35 sec
6	0.2 sec	13	0.15 sec
7	0.05 sec	14	0.25 sec

표 3 에스컬레이션되는 워크플로우 인스턴스 비율

수행경로	지나가는 인스턴스 수	에스컬레이션되는 인스턴스 수	에스컬레이션 비율
경로a-d	1439	115	7.9%
경로a-c	1478	248	16.8%
경로b-d	2095	202	9.6%
경로b-e	2056	458	22.3%
경로c-d	1505	109	7.2%
경로c-e	1427	221	15.5%
총계	10,000	1,353	

b, 그리고 액티비티 1, 4, 5를 경로 c라 하자. 두번째 OR 제어 블록은 2가지의 수행 경로가 존재하며 액티비티 8과 9는 병렬적으로 수행되므로 보다 긴 수행 시간을 요구하는 액티비티에 의해서 동기화된다. 여기서 액티비티 6, 7, 8(혹은 9), 10, 13을 경로 d라 하고, 액티비티 6, 11, 12, 13을 경로 e라 하자. 결국, 그림 10의 워크플로우 정의는 6가지의 수행 경로를 가지며 이들 중에서 본 논문에서 제안한 워크플로우 임계 경로는 경로 b-e가 된다. 추가적으로, 그림 10에는 OR 제어 구조에서의 분기 확률들이 명시되어 있다. 표 3은 6가지 경로들 각각에 대해서 수행되는 워크플로우 인스턴스 수, 에스컬레이션되는 워크플로우 인스턴스 수, 그리고 에스컬레이션되는 비율에 대한 실험 결과이다. 이 표에서 워크플로우 임계 경로(경로 b-e)를 지나는 워크플로우 인스턴스들의 에스컬레이션 비율이 가장 높음을 알 수 있다. 이는 본 논문에서 정의한 워크플로우 임계 경로의 속성과 본 논문에서 제안한 임계 경로 결정 방법이 타당함을 보여준다.

5. 결 론

임계 경로의 분석은 워크플로우 정의에 대한 중요한 정보를 제시한다. 무엇보다도, 임계 경로는 워크플로우의 많은 병목 지점들을 포함하기 때문에, 임계 경로를 효율적으로 관리함으로써 고성능 워크플로우 시스템을 지원할 수 있다. 이는 워크플로우 환경에서 최근에 가장 중요하게 관심을 가지는 문제이기도 하다. 특히, 시간 제약성을 가지는 워크플로우는 임계 경로를 보다 잘 활용할 수 있는 아주 좋은 예가 될 수 있다. 현재 대부분의 업무 프로세스들은 마감 시간(deadline)과 같은 시간 제약성을 가진다. 워크플로우에서 시간 관리 문제는 워크플로우 성능과 관련하여 많이 언급되고 있다. 만약 한 워크플로우 인스턴스가 워크플로우 수행 도중에 액티비티 마감 시간을 만족시키지 못하면, 워크플로우 관리 시스템들은 이 인스턴스에 대해 에스컬레이션(escalation)이라는 예외 처리를 수행할 것이다. 비록 에스컬레이션의 영향이 액티비티에 따라 다르지만, 에스컬레이션 과정은 보통 새로운 액티비티들의 수행, 이미 수행된 액티비티들에 대한 보상 수행, 혹은 사람 개입 등과 같은 추가적인 워크플로우 수행을 유발시킨다. 그러므로 에스컬레이션 과정 자체의 수행 비용을 줄이는 것도 중요하지만 에스컬레이션되는 워크플로우 인스턴스들의 수를 가능한 한 줄이는 것이 더욱 필요하다. 이러한 경우, 임계 경로의 개념을 적절히 활용한다면, 워크플로우 성능을 향상시키기 위한 효율적인 방법들이 개발될 수 있다.

본 논문에서는 먼저 업무 프로세스를 적절히 워크플로우 개념으로 묘사하는데 사용되는 워크플로우 제어 조건들을 명시하였다. 그리고, 중첩 워크플로우에서 임계 경로를 결정할 수 있는 CPI 알고리즘을 제안하였다. 워크플로우 정의에서 임계 경로 분석은 각 액티비티 에이전트의 서비스율 뿐만아니라 워크플로우 인스턴스들의 도착률을 고려하면서 이루어져야 하기 때문에, 본 논문에서 제안된 방법은 각 액티비티에서 발생되는 대기 시간을 수용하기 위해 워크플로우 대기 행렬 네트워크에 기반을 두고 있다. 이 과정에서 각 제어 블록의 부임계 경로 선정을 위해 비순차 제어 구조에서 하나의 순차 경로를 추출하는 과정을 설명하였다.

본 알고리즘은 순차(Sequence), 반복(Iteration), AND-분할, AND-병합, OR-분할, OR-병합 제어 조건들과 동기화 에지에 의해서 정의된 중첩 워크플로우를 대상으로 하고 있다. 이러한 제어 조건들은 워크플로우 기술들에 대한 표준을 제정하는 WfMC(Workflow Management Coalition)에서 기본적으로 언급하고 있기 때문에, 본 논문에서 역시 동일하게 받아들였다. 그러나, 워크플로우 정의의 표현력을 확장하는데 유용한 다른 제어 조건들이 고려될 수 있기 때문에, 확장된 워크플로우 제어 조건들로 구성된 워크플로우 정의에서 임계 경로에 대한 분석이 이루어질 필요가 있다.

참 고 문 헌

[1] Lawrence, P., Workflow Handbook 1997, John Wiley & Sons Ltd, 1997.

[2] Leymann, F., and Roller, D., Production Workflow: Concepts and Techniques, Prentice Hall, N.J., 1999.

[3] Kyung Tek Lee and Jacob A. Abraham, ritical Path Identification and Delay Tests of Dynamic Circuits,Proceedings of International Test Conference, (1999) 421-430.

[4] Cui-Qing Yand and Barton P. Miller, Critical Path Analysis for the Execution of Parallel and Distributed Programs, Proceedings of 8th International Conference on Distributed Computing Systems, (1988) 366-373.

[5] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, Introduction to Algorithms, The MIT Press, 1994.

[6] Hamdy A. Taha, Operations Research, Macmillan Publishing Company, 1992.

[7] Son, J. H. and Kim, M. H., Improving the Performance of Time-Constrained Workflow Processing, Journal of Systems and Software, Vol 58/3, pp 209-217, Sep 2001.

[8] Oh, S. K., Son, J. H., Lee, Y. J., and Kim, M. H., An Efficient Method for Allocating Workflow Tasks to Improve the Performance of Distributed Workflows, International Conference on Computer Science and Informatics, 2000.

[9] Dongsoo Han, Jaeyong Shim, and Chansu Yu, ICU/COWS: A Distributed Transactional Workflow System Supporting Multiple Workflow Types, IEICE Transactions on Information and Systems, E83-D(7) (2000) 1514-1525.

[10] Hagen, C., and Alonso, G., Flexible exception handling in the Opera process support system, In Proceedings of the 18th IEEE International Conference on Distributed Computing Systems, 1998.

[11] Heinl, P., Exceptions during workflow execution, In Proceedings of the Sixth International Conference on Extending Database Technology, 1998.

[12] Pozewaunig, H., Eder, J., and Liebhart, W., ePERT: Extending PERT for workflow management systems, The 1st European Symposium in ADBIS, (1997) 217-224.

[13] Panagos, E. and Rabinovich, M., Predictive Workflow Management, The 3th International Workshop on NGITS, 1997.

[14] Eder, J., Panagos, E., and Rabinovich, M., Time Constraints in Workflow Systems, Conference on Advanced Information Systems Engineering, (1999) 286-300.

[15] Panagos, E. and Rabinovich, M., Reducing Escalation-Related Costs in WFMSs, In Proceedings of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability, (1997) 106-128.

[16] Kao, B. and Garcia-Molina, H., Deadline assignment in a distributed soft real-time system, In Proceedings of the 13th International Conference on Distributed Computing Systems, (1993) 428- 437.

[17] Son, J. H., Kim, J. H., and Kim, M. H., Hard/Soft Deadline Assignment for High Workflow Throughput, In Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments, (1999) 272-278.

[18] Reichert, M. and Dadam, P., ADEPT$_{flex}$-Supporting Dynamic Changes of Workflows Without Loosing Control, Journal of Intelligent Information Systems, 10(2) (1998) 93-129.

[19] Workflow Management Coalition: Terminology & Glossary, Document No. WFMC-TC-1011, Issue 3.0, 1999.

[20] Kleinrock, L., Queueing Systems: Computer Applications, Prentice Hall, 1989.

[21] Wolff, R. W., Stochastic Modeling and the Theory of Queues, Prentice Hall, 1989.

[22] Disney, R. L., Queueing Networks, American Mathematical Society Proceedings of Symposium in Applied Mathematics, (1981) 53-83.

손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 2001년 한국과학기술원 전자전산학과 박사. 2001년 ~ 현재 한국과학기술원 전자전산학과 박사후 연구원. 관심분야는 분산 데이타베이스, 미들웨어, 워크플로우, 컴포넌트, CORBA, 객체지향모델링

장 덕 호

1981년 성균관대학교 전기공학과 학사. 1983년 성균관대학교 대학원 전기공학과 석사. 2001년 현재 한국과학기술원 전산학과 박사 수료. 1983년 ~ 2000년 한국전자통신연구원 책임연구원. 1994년 ~ 1995년 미국 Transarc Corp. 초빙연구원. 2000년 ~ 2001년 현재 (주)디지탈아리아 대표. 관심분야는 워크플로우, 분산트랜잭션 처리, 모바일 멀티미디어

김 명 호

정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조