확장된 구조적 워크플로우 스키마에서 워크플로우 임계 경로의 결정

(Finding the Workflow Critical Path in the Extended Structural Workflow Schema)

손 진 현 † 김 명 호 ††

(Jin Hyun Son) (Myoung Ho Kim)

요 약 워크플로우에서 임계 경로의 개념은 워크플로우 자원 및 시간 관리 등과 같이 워크플로우의 여러 분야에서 유용하게 활용될 수 있다는 면에서 중요하다. 그럼에도 불구하고 지금까지 임계 경로에 대한 연구가 많이 이루어지지 않았다. 이는 워크플로우에서의 제어 흐름 구조는 기존의 전형적인 그래프 혹은 네트워크 보다 더 복잡한 구조를 가지고 있기 때문이다. 본 논문에서는 먼저 복잡한 업무 흐름을 워크플로우로 표현할 수 있도록 지원하는 다양한 워크플로우 제어 구성자들을 정의한다. 그리고 이를 기반으로 정의된 구조적 워크플로우 스키마에서 임계 경로를 결정하는 방법을 제안한다.

키워드 : 임계 경로, 워크플로우, 대기 행렬 네트워크

Abstract *The concept of the critical path in the workflow is important because it can be utilized in many issues in workflow systems, e.g., workflow resource management and workflow time management. However, the critical path in the context of the workflow has not been much addressed in the past. This is because control flows in the workflow, generally including sequence, parallel, alternative, iteration and so on, are much more complex than those in the ordinary graph or network. In this paper we first describe our workflow model that has considerable workflow control constructs. They would provide the sufficient expressive power for modeling the growing complexities of today's most business processes. Then, we propose a method to systematically determine the critical path in a workflow schema built by the workflow control constructs described in our workflow model.*

Key words : Critical path, Workflow, Queuing Network

1. 서 론

오늘날 산업체 및 공공부문에서는 효율적인 업무 처리가 조직체의 경쟁력과 밀접한 관련이 있다. 이러한 경향에 따라 업무 프로세스의 전체 또는 일부를 자동화하고자 하는 목적에서 워크플로우 개념이 도입되고 있으며, 업무 흐름의 구조적 효율성, 성능 향상, 높은 유연성, 종합적인 업무 처리 및 관리 환경 제공 등과 같은 장점이 있다.

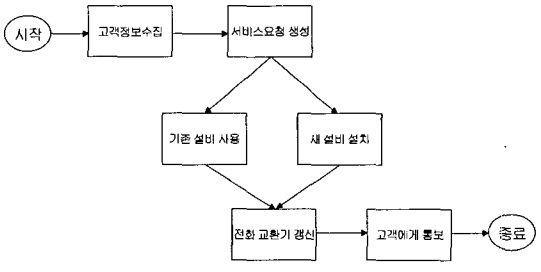


그림 1 전화 설치 워크플로우 스키마

워크플로우는 업무 목표를 달성하기 위해 규정된 업무 흐름에 따라 문서, 정보 및 작업 내용이 해당 담당자들에게 전달되면서 수행되는 자동화된 업무 절차이다[1]. 즉, 다양한 워크플로우 제어 흐름(workflow control flow)으로 연결된 액티비티(activity)들의 집합이다. 여기서 액

· 본 연구는 BK21 산학협력자금의 지원을 받았음.
† 비 회 원 : 한국과학기술원 전자전산학과
jhson@dbserver.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과
mhkim@dbserver.kaist.ac.kr
논문접수 : 2001년 4월 19일
심사완료 : 2002년 2월 14일

티비티는 워크플로우로 표현된 업무의 세부 단위 작업을 의미한다. 업무 프로세스를 워크플로우로 표현하는 과정에서 생성되는 액티비티들의 집합, 각 액티비티의 속성, 액티비티들 사이의 연결 등과 같은 정보를 해당 업무 프로세스의 워크플로우 스키마(workflow schema)라고 부르고 이는 워크플로우 업무의 모든 구조적 및 논리적 정보를 포함한다. 워크플로우 업무에 대한 요청이 발생하면 워크플로우 스키마를 기반으로 워크플로우 인스턴스(workflow instance)가 생성되고 업무의 수행이 완료되면 이 인스턴스는 소멸된다. 워크플로우 관리 시스템(workflow management system : WFMS)은 워크플로우를 정의 및 관리하고, 규정된 논리에 따라 프로그램을 수행시켜 주는 역할을 담당한다[1]. 그림 1은 전기 통신 회사에서 전화 설치 서비스와 같은 새로운 서비스 제공에 대한 워크플로우 스키마의 예이다.

일반적으로 워크플로우 스키마에는 여러 개의 수행 경로들이 존재하는데 이 중에서 가장 긴 수행 시간을 가지는 경로를 임계 경로(critical path)라고 하며, 이 경로는 워크플로우의 수행 완료 시간에 직접적인 영향을 미친다. 그리고 임계 경로 상의 액티비티들을 임계 액티비티(critical activity)라고 부른다. 이러한 임계 경로 및 임계 액티비티 개념은 워크플로우 자원 관리(workflow resource management), 워크플로우 시간 관리(workflow time management) 등과 같은 다양한 워크플로우 응용 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 개선시 킬 수 있다. 실제로 워크플로우 자원 관리에 대해 연구한 [4]에서는 본 논문에서 언급한 워크플로우 임계 경로의 개념을 효율적으로 이용하여 워크플로우의 성능을 개선시킨다. 그리고, 워크플로우 시간 관리 측면에서 워크플로우 임계 경로는 워크플로우 전체 완료 시간에 직접적으로 영향을 주기 때문에 임계 경로에 속하는 임계 액티비티들에게 효율적으로 마감 시간을 할당함으로써 워크플로우 처리량(workflow throughput)을 증가시킬 수 있다. [3]과 [5]에서는 워크플로우 임계 경로 개념을 이용하여 워크플로우 시간 관리를 효율적으로 지원하는 방법을 제안하고 있다. 이상과 같이 워크플로우 임계 경로 및 임계 액티비티의 개념은 많은 워크플로우 응용 연구 분야에 활용될 수 있는 기반 개념임에도 불구하고 위의 관련 연구들은 워크플로우 임계 경로를 결정하는 방법에 대해 구체적으로 언급하고 있지 않다.

프로젝트 계획, 스케줄링, 제어를 위한 관리 분석 기법으로 지금까지 임계 경로법(critical path method : CPM)과 프로젝트 계획 관리법(project evaluation and review technique : PERT)이 많이 적용되어 왔다[6]. 그러나, PERT-CPM 방법은 기본적으로 순차 혹은 병렬적으로 수행되는 액티비티들의 환경을 고려하고 있다. 그러므로, 순차, 선택, 병렬, 반복 등과 같은 복잡한 업무 흐름으로 구성되는 워크플로우 스키마에서 임계 경로를 결정하는데 PERT-CPM을 적용하기에는 많은 제약이 있다. 본 논문에서는 먼저 현재의 복잡한 업무를 효과적으로 표현할 수 있는 확장된 구조적 워크플로우 스키마에 대해 정의하고, 이러한 워크플로우 환경하에서 임계 경로를 결정하는 CPWS(Critical Path in a Workflow Schema) 방법을 제안하고자 한다. CPWS는 대기 행렬(Queuing) 이론을 바탕으로 워크플로우 액티비티들의 평균 수행 시간을 계산함으로써 워크플로우 임계 경로를 찾는 방법이다.

워크플로우 자원 관리 분야에 대해 [4]에서는 특정 액티비티들에게 많은 자원을 할당하여 이들의 처리 용량을 증가시킴으로써 시간 제약성을 가진 워크플로우의 처리 성능을 개선시키는 방법에 대해 연구했다. [7]에서는 분산 워크플로우 환경에서 워크플로우 수행에 필요한 비용을 줄이고자 하는 목적에서 효율적인 작업 할당 방법이 제안되었다.

워크플로우 액티비티 마감시간(deadline)의 효율적인 계산, 시간 제약성 관리, 그리고 에스컬레이션(escalation) 지원 등과 같은 워크플로우 시간 관리 분야들에 대해서는 [8], [9], [10]에서 연구가 수행되었다. [11]에서는 특정 워크플로우 인스턴스가 최종적으로 전체 워크플로우 마감시간을 만족하지 못할 것으로 예측되면, 그 워크플로우 인스턴스의 수행을 가능한 한 빨리 종료시키는 것이 바람직하다는 생각에서 예측 워크플로우(predictive workflow)에 대해 제안하였다. [2]와 [3]은 액티비티 마감시간을 동적으로 조정하는 방법에 대해서 연구했다. 이상에서 언급한 연구들은 모두 분산 소프트 실시간(distributed soft real-time) 환경에서의 마감시간 할당 방법을 제안한 [12]에 기반을 두고 있다. 한편, [5]는 워크플로우 환경에서 정적 마감시간을 효율적으로 할당하는 것이 필요함을 언급하고 새로운 정적 마감 시간 할당 방법을 제안하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 다양한 워크플로우 제어 구조와 본 논문에서 고려하는 워크플로우 모델을 언급한다. 이 모델을 기반으로 3장에서는 워크플로우 임계 경로를 찾는 방법을 제안하고, 4장에서는 제안한 방법에 대한 전체적인 예를 보인다. 마지막으

로 5장에서는 본 연구의 공헌과 앞으로 해야할 일에 대해 언급을 하면서 결론을 맺는다.

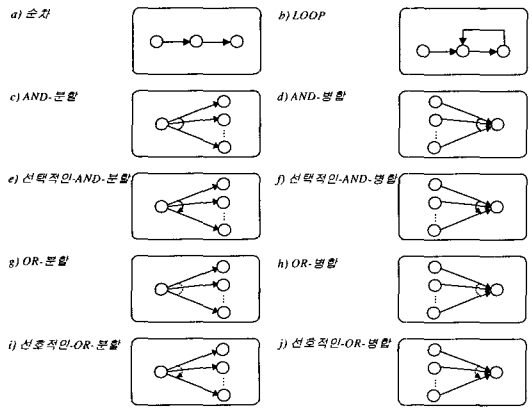


그림 2 워크플로우 제어 구성자

2. 워크플로우 모델

2.1 워크플로우 제어 구조

워크플로우 스키마는 노드와 방향성 에지로 구성된 네트워크로 시작 노드에서 종료 노드까지의 각 노드는 액티비티를 나타내고 각 방향성 에지는 두 노드 사이의 전이를 표현한다. 워크플로우 제어 구성자는 워크플로우 스키마에서 노드의 집합 S에서 노드의 집합 D까지의 경로를 정의한다. 여기서 S와 D는 각각 경로의 시작과 도착 집합을 의미한다. 경로는 방향성 에지들의 집합으로 필요한 경우 적절한 조건과 함께 나타낼 수 있다. 워크플로우 제어 구조는 순차, 반복, 분할, 병합의 네 가지로 분류될 수 있다. $|S|$와 $|D|$가 각각 집합 S와 D에 있는 노드의 개수라고 할 때, 순차 제어 구조와 반복 제어 구조의 경우에는 $|S|=|D|=1$ 이고, 분할 제어 구조는 $|S|=1$, $|D|>1$, 병합 제어 구조는 $|S|>1$, $|D|=1$ 이다.

그림 2는 현재의 복잡한 업무 흐름을 표현할 수 있는 다양한 워크플로우 제어 구성자들에 대해 정의하고 있다. 순차 제어 구조에서 액티비티들은 일련의 순서대로 수행된다. 즉, 선행하는 액티비티의 수행이 완료될 때까지 다음 액티비티가 수행될 수 없다. 반복 제어 구조에서는 하나 이상의 액티비티가 조건이 만족될 때까지 반복적으로 수행된다.

분할 제어 구성자는 한 노드에서 여러 노드들로 제어 흐름을 전달하는 흐름 제어기 역할을 한다. 분할 제어 구성자에는 AND-분할, 선택적인-AND-분할, OR-분할, 선호적인-OR-분할 등이 속한다. AND-분할은 단일 실행 흐름을 병렬 실행 흐름으로 바꾸어 AND-분할의 도착 집합 D에 속하는 모든 액티비티들을 동시에 수행시킨다. 선택적인-AND-분할은 D집합에서 하나 이상의 액티비티들을 동시에 수행시킨다. 만약, 선택적인-AND -분할의 D집합에 속하는 모든 액티비티들이 동시에 수행되는 경우에는 AND-분할과 동일한 기능을 가진다. 반대로, OR-분할은 여러 개의 워크플로우 분기들 중에 한 분기만을 선택하기 때문에 OR-분할의 D집합에 속하는 액티비티들 중에서 오직 한 액티비티만이 수행된다. 선호적인-OR-분할은 전이 조건을 만족하는 여러 개의 분기들 중에서 가장 우선 순위가 높은 한 분기만을 선택하기 때문에 OR-분할과 동일하게 오직 한 액티비티만을 수행시킨다.

병합 제어 구성자는 분할 제어 구성자와는 반대로 여러 개의 노드들로부터 하나의 노드로 제어 흐름을 전달한다. AND-병합, 선택적인-AND-병합, OR-병합, 선호적인-OR-병합 등이 병합 제어 구성자에 속한다. AND-병합은 시작 집합 S에 속하는 둘 이상의 병렬로 수행되는 노드들을 하나의 노드로 동기화시킨다. 선택적인-AND-병합은 AND-병합과는 달리 S집합에 속하는 병렬로 수행되는 노드들 중에서 하나 이상의 노드들을 선택하여 D집합에 속하는 한 노드로 동기화 시킨다. 선택되지 않은 나머지 노드들은 취소되거나 무시된다. OR-병합에서는 둘 이상의 선택적인 분기들이 하나의 노드로 합쳐지며, 노드들이 병렬적으로 수행되지 않았기 때문에 동기화가 필요하지 않다. 선호적인-OR-병합에서는 D집합으로 향하는 각 분기가 우선 순위를 가지기 때문에 가장 우선 순위가 높은 한 분기가 선택되어 D집합의 한 노드에게 제어 흐름이 전달된다. 한 분기만을 선택한다는 점에서 OR-병합과 유사하고, 노드들 사이의 동기화가 필요하지 않다는 점에서 선택적인-AND-병합과는 다르다. 여기서 주목할 점은 모든 분할 제어 구성자들은 대응하는 병합 제어 구성자들과 연결되어야 만이 완전한 워크플로우 제어 경로를 표현할 수 있다는 것이다.

정의 2.1 분할 제어 구성자 α와 대응하여 올바른 워크플로우 스키마를 구성하는 병합 제어 구성자들의 집합 C_α가 존재할 때, C_α를 α의 **닫힘 집합(closure)**이라고 부른다.

각 α에 대해 C_α는 다음과 같다.

$C_{AND\ 분할}$={*AND*-병합, 선택적인-*AND*-병합, 선호적인-*OR*-병합}

$C_{선택적인\text{-}AND\ 분할}$={선택적인-*AND*-병합, 선호적인-*OR*-병합}

$C_{OR\ 분할}$={*OR*-병합, 선호적인-*OR*-병합}

$C_{선호적인\ OR\ 분할}$={*OR*-병합, 선호적인-*OR*-병합}

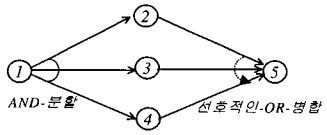


그림 3 AND-분할 & 선호적인-OR-병합

예를 들면, 동시에 세 항공사에서 예약을 하여 가장 빨리 예약이 가능한 곳을 선택하는 경우에 그림 3과 같이 (AND-분할, 선호적인-OR-병합)이 사용된다. 그리고, 선택되지 않은 두 항공사에서의 예약은 취소된다.

정의 2.2 워크플로우 스키마에서 아래 조건들 중 하나를 만족하는 네트워크를 **비순차 제어 블록(non-sequential control block)** 또는 간단히 **제어 블록**이라고 부른다.

- 분할 제어 구성자로 시작하여 대응되는 병합 제어 구성자로 끝나는 네트워크
- 반복 제어 구성자에 속하는 모든 노드와 에지들로 구성된 네트워크

정의 2.2에 따르면 순차 제어 구조는 제어 블록이 아님을 주목할 필요가 있다. 제어 블록 V가 다른 제어 블록 U의 일부라고 가정하자. V의 모든 노드와 에지가 U에 포함되면 U는 V를 완전히 포함한다고 하고, 그렇지 않은 경우에는 부분적으로 포함한다고 한다. 다른 제어 블록을 완전히 또는 부분적으로 포함하지 않는 제어 블록을 **원자 제어 블록(atomic control block)**이라고 부른다. 제어 블록 U가 V를 완전히 포함하고 다른 제어 블록을 부분적으로 포함하지 않을 때, U를 **중첩 제어 블록(nested control block)**이라고 부른다. 이때 U는 외부(outer) 제어 블록 또는 간단히 외부 블록이라 하고, V는 내부(inner) 제어 블록 또는 간단히 내부 블록이라 한다. V의 외부 블록이고 U의 내부 블록인 제어 블록이 존재하지 않을 때, U는 V의 인접 외부(immediate outer) 블록이고 V는 U의 인접 내부(immediate inner) 블록이 된다. 한 제어 블록에 대해 여러 인접 내부 블록들이 존재할 수 있지만, 인접 외부 블록은 하나만 존재할 수 있음에 주목할 필요가 있다. 그리고, 다른 제어 블록에 포함되지 않는 제어 블록을 **최상위 제어 블록(top-level control block)**이라고 부른다. 최상위 제어 블록은 원자 제어 블록이거나 중첩 제어 블록이다. 그림 4의 액티비티 17과 같이 제어 블록에 포함되지 않는 노드를 **단순 노드(simple node)**라고 부른다. 그러면, 워크플로우 스키마를 최상위 제어 블록과 단순 노드로 구성된 순차 제어 구조로 간주할 수 있다.

워크플로우 설계자가 워크플로우 제어 구성자들에 대한 정확한 분석 없이 복잡한 업무 프로세스를 워크플로우로 정의할 때 각종 논리적 및 구조적 오류들이 발생하기 쉽다[3][4]. 예를 들면, 그림 4에서 만약 반복 제어 구성자의 피드백(feedback)이 액티비티 10 대신에 액티비티 7이면, 이 워크플로우 정의는 AND-병합 제어 구성자의 논리적 의미에 의해서 올바른 제어 흐름을 가지지 못한다.

정의 2.2 원자 제어 블록과 중첩 제어 블록을 **구조적 제어 블록(structural control blocks)**이라고 부른다. 구조적 제어 블록과 단순 노드로 구성된 워크플로우 스키마를 **구조적 워크플로우 스키마(structural workflow schema)**라고 부른다.

지금부터 다루어지는 워크플로우 스키마는 특별한 언급이 없는 한 본 논문에서 정의한 워크플로우 제어 구성자들에 의해서 정의된 구조적 워크플로우 스키마임을 가정한다.

2.2 워크플로우 대기 행렬 네트워크

워크플로우 요청이 포아송(Poisson) 프로세스로 도착하고 각 액티비티의 서비스 시간이 지수(exponential) 분포를 가진다고 가정하면, 구조적 워크플로우 스키마는 기존의 전화 혹은 컴퓨터 통신 네트워크와 같이 M/M/1 대기 행렬 네트워크로 모델링될 수 있다. M/M/1 워크플로우 대기 행렬 네트워크에서 각 액티비티는 하나의 독립된 M/M/1 대기 행렬 시스템임이 알려져 있다[13]. 그러므로, 워크플로우의 초기 도착률, 각 액티비티의 서비스율, 그리고 OR-분할, 선호적인-OR-분할, 반복 제어 구성자에서의 분기 확률 등의 정보를 활용하여 그림 4에서처럼 각 액티비티에서의 도착 및 출발률을 명시할

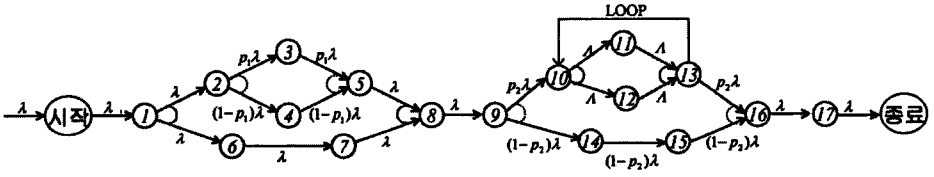


그림 4 워크플로우 대기 행렬 네트워크

수 있다. 이는 아래의 사실들에 바탕을 두고 있다.

시간 역행성(time reversibility)에 의해 OR-분할, OR-병합, 선호적인-OR-분할, 선호적인-OR-병합 제어 구성자에서 서로 독립적인 포아송 프로세스들의 분할과 병합 역시 포아송 프로세스임이 알려져 있다[13]. 그림 4의 액티비티 2, 5, 9, 16이 이에 속한다. 한편, 반복 제어 구성자는 피드백을 가지고 있기 때문에 내부 흐름이 실제적인 포아송 프로세스가 아니지만, 반복 제어 구성자에 속하는 각 액티비티들은 마치 독립적인 M/M/1 시스템들처럼 동작한다고 이미 알려져 있다[14]. 액티비티 1과 같은 AND-분할 제어 구성자와 선택적인-AND-분할 제어 구성자에서의 출발 프로세스는 명백히 포아송 프로세스이지만, 액티비티 8과 같은 AND-병합 제어 구성자와 선택적인-AND-병합 제어 구성자에서는 병렬적으로 수행되어온 각각의 요청들이 동기화가 되기 때문에 도착 프로세스는 실제로 포아송 프로세스로 모델링 되기가 어렵다. 이들 제어 구성자에서의 동기화는 여러 개의 분기들이 병렬적으로 수행되는 n개의 경로들 중에서 가장 긴 평균 수행 시간을 가지는 경로에 의해서 결정되기 때문에, 도착 프로세스는 이 경로를 통해 도착하는 요청들의 포아송 프로세스에 근사적으로 접근한다고 할 수 있다. 그러므로 결론적으로 구조적 워크플로우 스키마는 모든 액티비티들이 포아송 도착 및 출발 프로세스를 가지는 M/M/1 대기 행렬 네트워크로 모델링될 수 있다.

3. 임계 경로

3.1 CPWS 알고리즘

본질적으로 워크플로우 스키마는 AND-분할과 AND-병합으로 이루어진 제어 구조에서와 같이 병렬 수행 경로들이 존재한다. 워크플로우 임계 경로는 구조적 워크플로우 스키마의 여러 수행 경로들 중에서 가장 긴 평균 수행 시간을 가지는 경로로 정의된다. 본 논문에서는 2.2절에서 설명한 워크플로우 대기 행렬 네트워크 모델을 이용하여 워크플로우 스키마에서 임계 경로를 결정하는 CPWS(Critical Path in a Workflow Schema) 알고리즘을 제안한다. CPWS 방법은 임의의 워크플로우 액티비티에서 평균 수행 시간을 고려하면서 본 논문에서 정의한 확장된 워크플로우 제어 구성자들의 특징을 반영하고 있다. 하나의 워크플로우 스키마에 대해 동시에 여러개의 워크플로우 인스턴스들이 수행될 수 있으므로, 임의의 워크플로우 액티비티에서 어떤 워크플로우 인스턴스들은 먼저 도착한 워크플로우 인스턴스들이 수행되는 동안 액티비티의 큐에서 대기해야 한다. 이러한 현상은 임의의 액티비티에서 한 워크플로우 인스턴스의 평균 수행 시간은 액티비티의 평균 서비스 시간과 워크플로우 인스턴스가 액티비티의 큐에서 기다리는 평균 대기 시간의 합으로 고려되어야 함을 의미한다.

워크플로우 스키마는 2.1절에서 언급했듯이 단순 노드들과 최상위 제어 블록들로 구성된 순차 제어 구성자이기 때문에, 모든 단순 노드들과 각각의 최상위 제어 블록에서의 가장 긴 수행 경로를 연결하면 워크플로우 스키마의 임계 경로가 된다. 그러므로, 본 논문에서 제안하는 CPWS 방법은 기본적으로 각각의 최상위 제어 블록에서 가장 긴 수행 경로를 갖는 부 임계 경로(sub-critical path)를 찾고 이들을 서로 연결하여 최종적으로 워크플로우 임계 경로를 구성한다.

```
1. 모든 단순 노드들은 임계 경로에 속한다.

2. FOR (각각의 최상위 제어 블록에 대해)
   {
     3. WHILE (최상위 제어 블록이 중첩 제어 블록이면)
        /* WHILE 루프를 마치면, 중첩 제어 블록은 원자 제어 블록이 된다. */
        {
          /* 가장 내부의 제어 블록이 여러 개가 존재할 때, 처리되는 순서는 무관하다. */
          4. IF (가장 내부의 제어 블록이 LOOP 원자 제어 블록이면)
             4.1 LOOP를 순차 제어 구성자로 변환.
          5. ELSE IF (가장 내부의 제어 블록이 분할/병합 제어 블록이면)
             5.1 분할/병합 제어 블록에서 가장 긴 수행 경로를 선택.
             5.2 이 경로를 한 액티비티로 변환
        }

     /* 여기서, 최상위 제어 블록은 원자 제어 블록이 된다. */
     6. 최상위 제어 블록에서 가장 긴 수행 시간을 가지는 부임계 경로를 결정.
   }

7. 단순 노드들과 부임계 경로들을 연결하여 워크플로우 임계 경로를 구성.
```

그림 5 CPWS 알고리즘

최상위 제어 블록에서 가장 긴 수행 시간을 가지는 부 임계 경로의 결정은 그림 5의 CPWS 알고리즘의 단계 2에서 설명하는 바와 같이 최상위 제어 블록의 가장 내부의 제어 블록에서부터 가장 외부의 제어 블록으로 점차적으로 찾아 나간다. 만약 가장 내부의 제어 블록이 LOOP이면, CPWS 알고리즘의 단계 4와 같이 이 LOOP 제어 블록을 대응하는 순차 제어 구성자로 변환한다. 만약 가장 내부의 제어 블록이 2.1절에서 설명한 분할과 병합 제어 구성자로 만들어진 제어 블록이면, 이 제어 블록에서 가장 긴 평균 수행 시간을 가지는 하나의 경로를 결정하고 이를 다시 하나의 워크플로우 액티비티로 변환한다. 이 과정은 CPWS 알고리즘의 단계 5에 해당되며, 특히 "순차 경로를 한 액티비티로 변환"하는 단계 5.2의 필요성은 아래 3.7절에서 상세히 설명한다. 위의 과정들을 반복적으로 적용하면 최종적으로 우리는 n개의 분기를 가지는 하나의 새로운 원자 제어 블록을 얻게 된다. 즉, 알고리즘의 단계 6에 이르면 모든 최상위 제어 블록들은 각각 원자 제어 구조로 변환된다. 각각의 원자 제어 구조에서 가장 긴 수행 시간을 요구

하는 경로를 결정하여 워크플로우 단순 노드들과 서로 연결하면, 최종적으로 워크플로우 임계 경로를 얻게 된다. 그림 6은 이들 과정을 상세히 묘사하고 있다.

다음 절에서는 위의 CPWS 알고리즘에서 언급된 각각의 단계(즉, 단계 4, 5, 6)에 대해서 상세히 설명한다. 그리고 지금부터 그림 2에서 언급한 분할 및 병합 제어 구성자들로 형성될 수 있는 모든 제어 블록들을 간단히 (분할 제어 구성자, 병합 제어 구성자)로 표현한다. 예를 들면, AND-분할과 AND-병합 제어 구성자로 만들어지는 제어 블록을 간단히 (AND-분할, AND-병합) 제어 블록이라고 부른다.

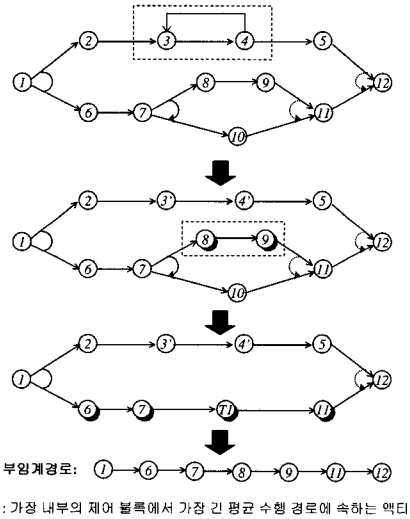


그림 6 부임계 경로의 결정

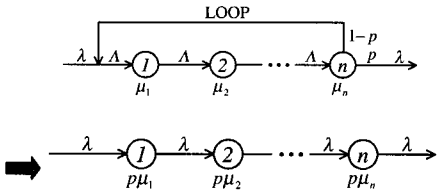


그림 7 LOOP 원자 제어 블록을 순차 제어 구성자로 변환

3.2 LOOP 원자 제어 블록

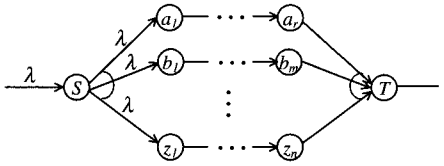
2.2절에서 언급했듯이 LOOP 원자 제어 블록에 속하는 각 액티비티는 하나의 독립된 M/M/1 대기 행렬 시스템으로 간주될 수 있기 때문에, LOOP 제어 블록은 그림 7에서와 같이 대응하는 순차 제어 구성자로 변환될 수 있다. 이는 CPWS 알고리즘의 단계 4.1에 해당한다. LOOP의 피드백으로 인해서 LOOP 제어 블록의 도착률 Λ는 다음과 같다:

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

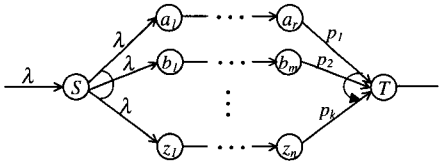
여기서 $1-p$는 노드 n에서 피드백이 일어날 확률이다. 이때, $\frac{\rho_i}{1-\rho_i}$이 그림 7-(a)의 액티비티 i에서 기다리는 서비스 요청들의 평균 개수이고, ρ_i는 $\frac{\Lambda}{\mu_i}\left(=\frac{\Lambda}{p\mu_i}, i=1,\cdots,n\right)$이다. 그러므로, LOOP의 평균 수행 시간은 리틀 정리(Little's Formula)에 의해 다음과 같다.

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

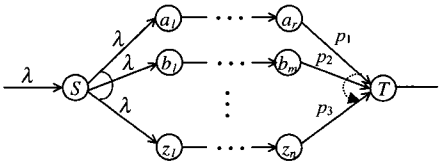
이는 그림 7-(b)와 같이 도착률 λ와 서비스율 $p\mu_i$를 가진 n개의 액티비티들로 구성된 순차 제어 구성자의 총 평균 수행 시간과 동일하다. 그러므로 LOOP 원자 제어 블록은 하나의 순차 제어 구성자로 변환될 수 있다.



(a) AND-분할 & AND-병합



(b) AND-분할 & 선택적인-AND-병합



(c) AND-분할 & 선호적인-OR-병합

λ : 도착율 P_i : 선택 확률

그림 8 AND-분할의 닫힘 집합

아래의 3.3절에서부터 3.6절까지는 CPWS 알고리즘의

단계 5.1에 해당되는 것으로 워크플로우 임계 경로를 결정하는 과정에서 다양한 분할/병합 제어 블록들을 다루는 방법에 대해서 설명한다. 그리고 CPWS의 단계 5.2는 3.7절에서 설명한다.

3.3 AND-분할 제어 구성자의 닫힘 집합

AND-분할의 닫힘 집합은 {AND-병합, 선택적인-AND-병합, 선호적인-OR-병합}이고, AND-분할의 도착 집합(D)에 속하는 각 노드, 즉 그림 8의 $a_1, b_1, \cdots, z_1$는 액티비티 S와 동일한 서비스 도착률인 λ를 가진다.

그림 8-(a)의 (AND-분할, AND-병합) 원자 제어 블록에서 가장 긴 수행 경로는 총 평균 수행 시간이 $MAX\left(\sum_i \frac{1}{\mu_i - \lambda}\right)$인 하나의 경로이다. 여기서 λ는 도착률이고 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다.

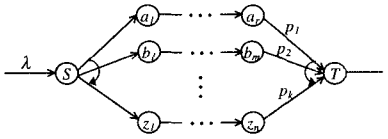
그림 8-(b)의 (AND-분할, 선택적인-AND-병합) 원자 제어 블록을 고려하자. 이 그림에서 p_j는 선택적인-AND-병합의 도착 노드로 분기 j가 선택될 확률이다. 이 분기 j를 통하는 경로는 유일하고 이를 경로$_j$라 명명하면, 경로$_j$에 속하는 모든 노드(즉, 액티비티)들은 결국 도착률 $p_j\lambda$라고 볼 수 있다. 그러므로, 이 제어 블록에서 가장 긴 수행 경로는 총 평균 수행 시간이 $MAX\left(\sum_i \frac{1}{\mu_i - p_j\lambda}\right)$인 경로이다. 여기서, $p_j\lambda$는 경로$_j$에 있는 모든 노드들의 도착률이고 μ_i는 이 경로에 있는 액티비티 i의 서비스율이다.

다음으로 그림 8-(c)와 같은 (AND-분할, 선호적인-OR-병합) 원자 제어 블록을 고려하자. 만약 선호적인-OR-병합이 여러 개의 분기 중에서 가장 먼저 도착하는 분기를 선택하는 우선순위가 있다고 가정하면, 이 제어 블록에서 가장 긴 수행 경로는 총 평균 수행 시간이 $MAX\left(\sum_i \frac{1}{\mu_i - p_j\lambda}\right)$인 한 경로이다. 여기서 $p_j\lambda$는 경로$_j$에 속하는 모든 노드들의 도착률이고 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다. 이 식은 선호적인-OR-병합의 정책에 따라서 변경될 수 있다.

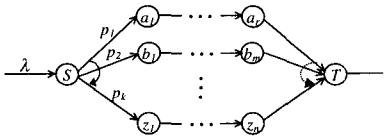
3.4 선택적인-AND-분할 제어 구성자의 닫힘 집합

선택적인-AND-분할 제어 구성자의 닫힘 집합은 {선택적인-AND-병합, 선호적인-OR-병합}이다.

그림 9-(a)와 같은 (선택적인-AND-분할, 선택적인-AND-병합) 원자 제어 블록에서, 경로 선택은 결국 선택적인-AND-병합에 의해서 이루어 지기 때문에, 이 제어 블록에서 가장 긴 수행 경로는 (AND-분할, 선택적인-AND-병합) 제어 블록에서와 동일한 방법에 의해서 결정된다.



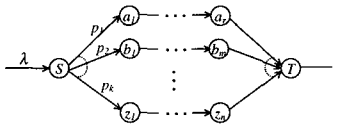
(a) 선택적인-AND-분할 & 선택적인-AND-병합



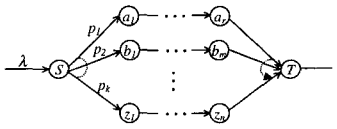
(b) 선택적인-AND-분할 & 선호적인-OR-병합

그림 9 선택적인-AND-분할의 닫힘 집합

그림 9-(b)와 같이 (선택적인-AND-분할, 선호적인-OR-병합) 원자 제어 블록에서, 선택적인-AND-분할 제어 구성자에 의한 분기 선택 확률 p_i를 가진다. 만약 선호적인-OR-병합은 가장 먼저 도착한 수행 경로를 선택한다고 가정하면, 이 제어 블록의 가장 긴 수행 경로는 총 평균 수행 시간이 $MIN\left(\sum_i \frac{1}{\mu_i - p_j\lambda}\right)$인 경로가 된다. 여기서 $p_j\lambda$는 선택적인-AND-분할 제어 구성자에서 분할되는 분기 j의 도착률이며 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다.



(a) OR-분할 & OR-병합



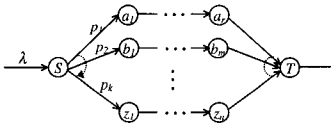
(b) OR-분할 & 선호적인-OR-병합

그림 10 OR-분할의 닫힘 집합

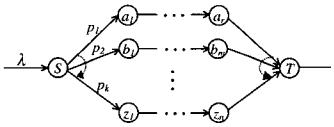
3.5 OR-분할 제어 구성자의 닫힘 집합

OR-분할 제어 구성자의 닫힘 집합은 {OR-병합, 선호적인-OR-병합}이다. OR-분할의 제어 특성상 OR-분할 제어 구성자에서 나가는 각 분기는 서로 배타적인 선택 확률 p_i를 가지며 이들 확률의 합 $\sum p_i$는 항상 1이 된다. OR-분할은 항상 하나의 분기만 선택하므로, 그림 10에서처럼 (OR-분할, OR-병합) 제어 블록과

(OR-분할, 선호적인-OR-병합) 제어 블록에서 가장 긴 수행 경로를 찾는 방법은 동일하다. 즉, 이들 제어 블록들에서 가장 긴 수행 경로는 총 평균 수행 시간이 $MAX\left(\sum_i \frac{1}{\mu_i - p_j\lambda}\right)$인 한 경로이다. 여기서 $p_j\lambda$는 OR-분할 제어 구성자에 의한 분기 j의 도착률이고 μ_i는 이 경로에 속하는 액티비티 i의 서비스율이다.



(a) 선호적인-OR-분할 & OR-병합

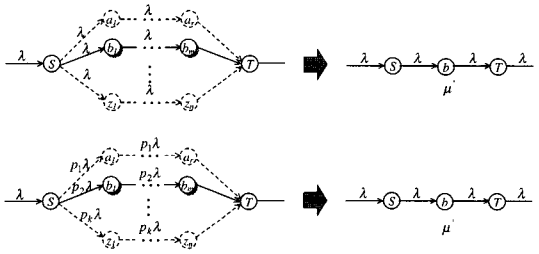


(b) 선호적인-OR-분할 & 선호적인-OR-병합

그림 11 선호적인-OR-분할의 닫힘 집합

3.6 선호적인-OR-분할 제어 구성자의 닫힘 집합

선호적인-OR-분할의 닫힘 집합은 그림 11과 같이 {OR-병합, 선호적인-OR-병합}이다. 선호적인-OR-분할은 오직 하나의 분기만을 선택한다는 면에서 OR-분할과 유사한 제어 특성을 가진다. 그러므로, 이 선호적인-OR-분할에 의해서 구성되는 제어 블록들에서의 가장 긴 수행 경로는 OR-분할의 경우와 동일하게 얻어질 수 있다.



: 제어블록에서 가장 긴 수행 경로에 속하는 액티비티

그림 12 순차 제어 구성자의 변환

3.7 순차 제어 구성자

CPWS 알고리즘의 단계 5.1에 의해서 임의의 분할/병합 제어 블록에서 가장 긴 수행 경로를 결정한 후, 그림 12와 같이 이 경로를 하나의 액티비티 b로 변환한다. 이는 CPWS 알고리즘의 단계 5.2에 해당된다. 이 변환 과정은 단계 5.1에서 선택된 경로와 동일한 평균 수행 시간을 가지는 액티비티 b의 서비스율 μ'을 결정하는 것과 동일하다.

그림 12에서 보듯이 단계 5.1에서 선택되는 순차 경로는 두 가지 종류가 있다. 하나는 선택된 순차 경로가

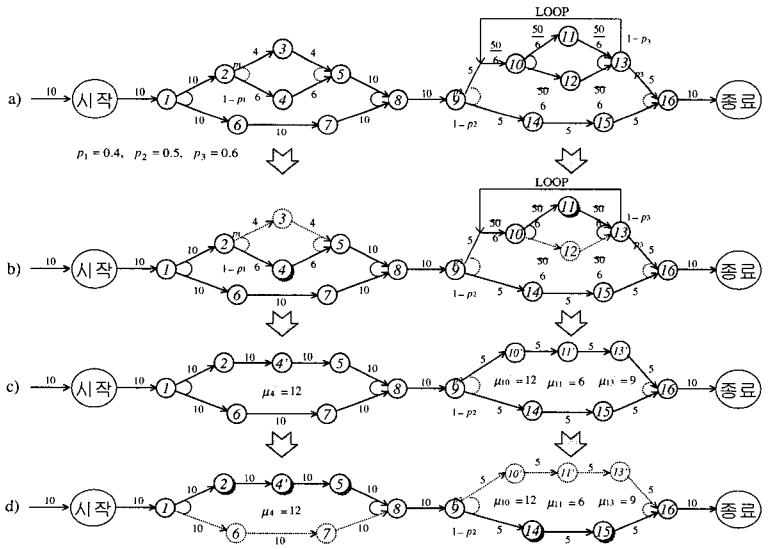


그림 13 CPWS 방법에 대한 전체 예제

분할 액티비티 S와 동일한 도착률을 가지는 경우이고, 다른 하나는 선택된 순차 경로가 분할 액티비티 S의 도착률 λ와는 달리 $p_i\lambda$의 도착률을 가지는 경우이다. 여기서 p_i는 분기 i가 선택될 확률이다. 선택된 순차 경로에 대응하는 액티비티 b의 서비스율을 μ'이라고 할 때,

$$\text{전자의 경우} : \mu' = \frac{1}{\sum_{i=1}^{n} \frac{1}{\mu_i - \lambda}} + \lambda$$

이고

$$\text{후자의 경우} : \mu' = \frac{1}{\sum_{i=1}^{n} \frac{1}{\mu_i - p_j\lambda}} + \lambda$$

이 식들에서 μ_i는 가장 긴 수행 경로에 속하는 액티비티 i의 서비스율이고 p_j는 이 경로가 선택될 확률이다.

표 1 액티비티의 서비스율

액티비티	서비스율	액티비티	서비스율	액티비티	서비스율
1	13	2	15	3	7
4	8	5	20	6	18
7	20	8	14	9	16
10	20	11	10	12	15
13	15	14	6	15	7
16	25				

4. CPWS 방법에 대한 전체 예제

본 절에서는 그림 13에서 묘사된 워크플로우 스키마를 사용하여 워크플로우 임계 경로를 결정하기 위해 본 논문에서 제안한 CPWS 방법에 대한 전체적인 예제를 보이고자 한다. 먼저 이 워크플로우 스키마에 대한 서비스 요청들의 도착률은 10이고 OR-분할과 LOOP 제어 구성자들에서의 분기 확률들 P_1, P_2, P_3는 각각 0.4, 0.5, 0.6이라고 가정한다. 이를 바탕으로 2.2절에서 언급한 것처럼 우리는 그림 13-(a)와 같이 각 액티비티에서의 도착률과 출발율을 명시할 수 있다. 그리고, 각 액티비티에서의 서비스율은 표 1에 주어져 있다.

그림 13-(a)의 워크플로우 스키마는 두개의 최상위 제어 블록들을 가지고 있으며 워크플로우 임계 경로를 결정하기 위해 이들 제어 블록에서의 부 임계 경로를 결정해야 한다. CPWS 알고리즘에서 언급했듯이, 임의의 최상위 제어 블록에서 부 임계 경로를 결정할 때 이 제어 블록에 속하는 가장 내부의 제어 블록에서부터 가장 외부의 제어 블록의 순서로 분석해야 한다. 먼저 (OR-분할, OR-병합)과 (AND-분할, AND-병합) 제어 블록들이 가장 내부 제어 블록들이기 때문에, 각 제어 블록에서 가장 긴 수행 경로를 선택하면 그림 13-(b)에서 보듯이 액티비티 4와 11이 선택된다. 이후 그림 13-(b)에 3.2절에서부터 3.7절까지 언급한 내용을 적용하면, 그림 13-(c)와 같이 새로이 변환된 액티비티 4′, 10′, 11′, 13′을 얻게 된다. 그리고 그림 13-(c)의 제어 블록들에서 가장 긴 수행 경로들을 결정하면 최종적으로 그림 13-(d)와 같이 최상위 제어 블록들에서 부 임계 경로들을 얻을 수 있다. 이들 부 임계 경로들을 연결하여 워크플로우 임계 경로 {1, 2, 4, 5, 8, 9, 14, 15, 16}이 결정된다.

5. 결론

본 논문에서 우리는 구조적인 워크플로우 스키마라는 개념을 도입함으로써 워크플로우의 구조적인 오류들을 쉽게 발견하고 교정할 수 있을 뿐만 아니라 어느 정도의 논리적인 에러들도 피할 수 있다. 또한, 확장된 워크플로우 제어 구성자들을 정의함으로써 다양하고 복잡한 업무 흐름을 워크플로우 스키마로 표현할 수 있다. 특히, 중첩 제어 블록과 최상위 제어 블록의 개념은 복잡한 워크플로우 스키마를 구조적으로 분석할 수 있게 하여 본 논문의 임계 경로 알고리즘에 활용되었다.

임계 경로의 분석은 워크플로우 스키마에 대한 중요한 정보를 제공한다. 주목할 만한 것은 임계 경로가 많은 워크플로우 병목 지점들을 포함하고 있다는 것이다. 그러므로, 현재 중요한 워크플로우 문제들 중의 하나인 고성능 워크플로우 시스템은 임계 경로의 정보를 이용함으로써 효과적으로 달성될 수 있다. 시간 제약성 워크플로우 영역 역시 임계 경로를 활용할 수 있는 중요한 예제이다. 일반적으로 업무 프로세스들은 마감시간과 같은 시간 제약성을 가지고 있기 때문에, 워크플로우 영역에서 효율적인 시간 관리는 워크플로우 성능과 연관하여 반드시 이루어져야 한다. 만약 워크플로우 인스턴스가 워크플로우 수행 중에 액티비티의 마감시간을 만족시키지 못했다면, 워크플로우 관리 시스템은 이 인스턴스에 대해 에스컬레이션(escalation)이라는 부가적인 조치를 취할 것이다. 에스컬레이션은 액티비티의 특성에 따라서 다르지만 부가적인 액티비티의 수행, 이미 수행된 액티비티에 대한 보상 수행, 사용자의 개입 등과 같은 과정을 요구한다. 이는 해당 업무 프로세스의 효율을 저하시키므로, 에스컬레이션 현상을 가능한 한 줄이거나 에스컬레이션 비용을 최소화시켜야 한다. 만약 이러한 워크플로우 응용 영역에서 임계 경로의 개념을 사용한다면 워크플로우 성능을 향상시킬 수 있는 효율적인 방

법들이 개발될 수 있다.

본 논문에서는 먼저 현재의 복잡한 업무 프로세스를 지원하기에 충분한 워크플로우 제어 구성자를 명시하였고, 이를 기반으로 정의된 구조적인 워크플로우 스키마에서 임계 경로를 결정하는 알고리즘을 제안하였다. 이러한 임계 경로 정보는 워크플로우 자원 및 시간 관리 등과 같은 다양한 워크플로우 응용 영역들에서 사용될 수 있기 때문에 아주 유용하다고 본다.

지금까지 매우 크고 복잡한 업무 프로세스를 워크플로우로 정의할 때, 워크플로우 실행 이전에 정의된 워크플로우 스키마에 대한 검증 기능에 대한 많은 고려가 있지 않았다. 그러므로 앞으로 워크플로우 스키마에 대한 구조적 뿐만 아니라 논리적 검증에 대한 연구가 필요하다.

참 고 문 헌

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

[2] Eder, J., Panagos, E., and Rabinovich, M., "Time Constraints in Workflow Systems," Conference on Advanced Information Systems Engineering, 1999.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] Pozewaunig, H., Eder, J., and Liebhart, W., "ePERT: Extending PERT for workflow management systems," The 1st European Symposium in ADBIS, 1997.

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

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

[12] 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.

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

[14] Disney, R. L., "Queueing Networks," American Mathematical Society Proceedings of Symposium in Applied Mathematics, 1981.

손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 2001년 한국과학기술원 전자전산학과 박사. 2001년 ~ 현재 한국과학기술원 전자전산학과 박사후연구원. 관심 분야는 분산 데이타베이스, 미들웨어, 데이터웨어하우징, 워크플로우, 분산시스템, CORBA, 객체지향모델링

김 명 호

1982년 서울대학교 컴퓨터 공학과 학사. 1984년 서울대학교 컴퓨터 공학과 석사. 1989년 MICHIGAN 주립대 전산학과 박사. 1989년 MICHIGAN 주립대 연구원. 1989년 ~ 1993년 한국과학기술원 조교수. 1993년 ~ 1999 년 한국과학기술원 부교수. 1999년 ~ 현재 한국과학기술원 정교수. 1992년 ~ 1993년 개방형 컴퓨터 통신 연구회(OSIA) 분산 트랜잭션처리 분과위(TG-TP) 의장. 1993년 ~ 1994년 한국통신기술협회(TTA) 분산 트랜잭션처리 실무 위원회 의장. 관심 분야는 데이타베이스, 분산트랜잭션, 분산시스템, 워크플로우