우선순위규칙을 이용한 워크플로우의 효율적 운영 방안*

이승현** · 유우식*** · 배혜림**** · 김영호** · 박용태**

An Efficient Method of Workflow Management using a Dispatching Rule*

Seung Hyun Rhee** · Woo Sik Yoo*** · Hyerim Bae****
Yeongho Kim** · Yong-Tae Park**

■ Abstract ■

A workflow management system is a software system to support accurate execution, control and management of business processes. It has been known that the system automatically executes complex processes and effectively manages them to improve the productivities. However, though existing systems assure the automatic execution of an individual process, there is a room for enhancement from the view of efficient execution of all the processes. In this paper, we propose a method of executing business processes efficiently by introducing the PERT/CPM techniques in the workflow management systems. We first consider the differences between workflow process models and PERT/CPM models, and then develop a method of calculating the critical path and slack time in workflow processes. This leads us to develop a dispatching rule that can guide task performers to prioritize their tasks to increase the efficiency of all the processes. We have carried out a set of simulation experiments and analyzed the results to demonstrate the effectiveness of the proposed method.

Keyword : Workflow Management System, Business process, Dispatching rule, Simulation

논문접수일 : 2002년 11월 5일 논문게재확정일 : 2003년 6월 2일

* 이 연구는 한국과학기술기획평가원(KISTEP)의 국가지정연구실사업(NRL)과 한국과학재단 지정 인천대학교 동북아 물류연구센터의 지원에 의한 것임.

** 서울대학교 산업공학과
*** 인천대학교 산업공학과
**** 동의대학교 인터넷비즈니스학과

1. 서 론

전자거래(electronic commerce)나 협력적 상거래(collaborative commerce) 등 최근의 컴퓨터 네트워크 응용은 전통적인 기업 관계에 변화를 초래하고 있다. 이들은 한편으로는 기업간 경쟁을 심화시키고, 동시에 다른 한편으로는 상호간의 협력을 촉진하기도 한다. 이러한 변화는 기업의 업무 절차와 프로세스를 복잡하게 만들었으며, 결국 복잡한 업무 프로세스의 효율적 수행과 관리가 경쟁력의 핵심으로 인식되고 있다.

워크플로우관리시스템(WFMS : WorkFlow Management System)은 복잡한 프로세스를 효과적으로 수행하고 관리하여 업무의 효율을 높인다[7]. 이 시스템은 프로세스를 정형화하여 모델링 할 수 있는 기능을 제공하고, 이 모델을 바탕으로 업무를 스케줄링하여 사용자에게 전달한다. 그리고, 업무 수행에 필요한 문서나 응용 프로그램 등의 자원을 자동으로 찾아서 사용자에게 제공한다[15]. 즉, WFMS는 비즈니스 프로세스의 수행을 자동화하여 관리해주는 소프트웨어 시스템인 것이다[12].

기존의 WFMS는 개별 프로세스 모델을 정확한 순서대로 수행하는 것은 보장한다. 그러나, 전체 프로세스가 모두 효율적으로 진행되는 것을 보장하지는 못한다[17]. 업무담당자는 보통 여러 종류의 프로세스에 참여하거나 또는 같은 종류이지만 복수의 프로세스에 참여한다. 따라서 어떤 한 시점에 처리해야 할 업무가 여러 개 있는 경우가 매우 흔하다. 이 때 모든 프로세스의 효율적 수행을 위해서 어떤 업무를 먼저 처리해야 하는가 하는 문제가 발생한다. 이러한 상황을 적절히 처리하여 전체 프로세스를 효율적으로 수행하는 방법이 기존의 WFMS에서는 미흡하였다.

본 연구에서는 전체 프로세스의 완료시간 및 이용되는 자원의 낭비시간 최소화를 위해 업무 처리 우선순위를 결정하는 방법을 제안한다. 이를 위하여 전통적 프로젝트 관리 기법인 PERT/CPM을 도입한다. 그런데, PERT/CPM 기법에서 사용하는 네트워크의 구조가 워크플로우 프로세스 모델의 구조와는 다르므로 이 기법을 WFMS에 그대로 적용할 수는 없다. 본 연구에서는 WFMS의 프로세스 구조를 고려하여 PERT/CPM 기법의 주경로(critical path)와 여유시간(slack time) 개념을 적용할 수 있는 변경된 방안을 제시한다. 그리고 제안한 방법의 효과를 검증하기 위해 시뮬레이션 실험을 실시하고, 그 결과를 분석한다.

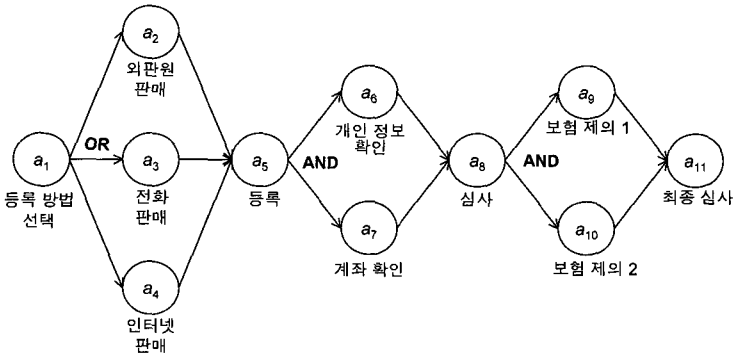
2. WFMS

워크플로우는 문서, 정보, 혹은 업무가 정해진 절차나 규칙에 따라 적절한 업무담당자에게 전달되는 전체 혹은 부분적으로 구현된 비즈니스 프로세스의 자동화를 의미한다[12]. 그리고, WFMS는 이러한 워크플로우를 자동으로 실행, 조정 및 관리하는 소프트웨어 시스템이다.

2.1 기본 개념

프로세스 자동 수행을 위해서는 컴퓨터가 이해할 수 있는 형태로 프로세스가 표현되어 있어야 한다. 이것을 프로세스 모델이라고 한다. 여기에는 프로세스가 추구하는 목적 달성에 필요한 업무와 이들간의 상호관계가 명세 되어 있다[5].

WFMS의 프로세스 모델은 일반적으로 정의시(build-time) 단계와 실행시(run-time) 단계로 나누어 생각한다[2]. 정의시 단계는 프로세스 수행을 준비하는 과정으로, 프로세스를 이루는 단위업무와 이들 간의 선후 관계 그리고 각 단위업무를 수행하는데 필요한 세부 속성들을 정의한다. 실행시 단계에서는 정의시 단계의 프로세스 모델을 해석하여 실제 프로세스를 실행, 관리, 통제한다[12, 19]. WFMS에서 실행시 단계를 담당하는 부분을 워크플로우 엔진(engine)이라 한다[10]. 한편, 실제로 처리해야 할 프로세스가 발생하면 워크플로우 엔진은 해당되는 정의시의 프로세스 모델을 바탕으로 실제 프로세스를 발생시키게 되는데, 이 때 생성되는 개별



〈그림 1〉 프로세스 모델의 예

프로세스를 프로세스 인스턴스(instance)라고 한다 [20].

<그림 1>의 예제 프로세스를 생각해보자. 이는 보험가입처리 프로세스로서, 정의시 단계에서는 이 프로세스를 구성하는 여러 단위업무와 이들 간의 선후 관계가 그림과 같이 표현된다. 실행시 단계에서는 실제 고객의 보험가입 요구가 있을 때 <그림 1>의 모델을 바탕으로 프로세스 인스턴스를 생성하여 프로세스를 수행한다. 결국, 하나의 프로세스 모델을 바탕으로 고객의 보험 가입 요청이 있을 때마다 프로세스 인스턴스를 반복하여 생성하고 수행할 수 있게 되는 것이다. 이 프로세스에서 업무가 여러 갈래로 나누어지는 곳에 주목해보자. 여기에는 OR 또는 AND 표시가 있는데, OR는 여러 갈래 가운데 일부만을 수행해도 됨을, 그리고 AND는 모든 갈래의 업무를 수행해야 함을 나타낸다.

2.2 프로세스 구조

프로세스 구조란 프로세스를 이루는 단위업무와 이들간의 선후 관계를 말하는 것으로, 본 연구에서 고려하는 프로세스 구조는 다음과 같이 정의된다.

[정의 1] : 프로세스 구조

하나의 프로세스를 P라고 할 때, P의 구조는 단위업무의 집합 A와 단위업무 간의 선후관계를 표현하는 링크집합 L로 표현된다. 즉, $P=(A,L)$ 이며, 이 때,

- $A=\{a_i \mid i=1,2,\cdots,I\}$, I는 단위업무의 수
- $L=\{(a_i,a_j) \mid a_i \in A, a_j \in A,$ 그리고 $i \neq j\}$

이다.

위의 정의에서 L의 원소 (a_i,a_j)는 j-번째 단위업무를 수행하기 위해서는 i-번째 단위업무의 수행이 완료되어 있어야 함을 의미한다.

프로세스 구조를 위와 같이 정의할 때, 임의의 서로 다른 두 단위업무 사이에는 다음과 같은 경로가 존재할 수 있다.

[정의 2] : 경로

임의의 두 단위업무 $a_i, a_j (i \neq j)$가 하나 이상의 링크로 연결될 때, 연결된 단위업무들의 집합을 경로라고 하며, 두 단위업무 사이의 경로는 두 개 이상 존재할 수 있다. 따라서, 이들 경로 중 p-번째 경로를 r_p라고 하며, 다음과 같이 표현한다.

$$r_p=\{a_i, a_k, a_{k+1}, \cdots, a_{k+n}, a_j\}$$

단, $(a_i,a_k),(a_k,a_{k+1}),\cdots,(a_{k+n},a_j) \in L$

즉, 경로는 하나의 단위업무가 또 다른 단위업무로 하나 이상의 링크를 통해 연결되어 있을 때, 이 링크로 연결된 단위업무의 집합을 의미한다.

한편, 서로 다른 두 단위업무 사이에 다수의 경로가 존재할 수 있는데 이는 프로세스에 분기와 병합이 있을 때이다. 분기와 병합이 일어나는 단위업

무를 각각 분기 단위업무, 병합 단위업무라고 한다. 분기 단위업무는 <그림 1>에서 a_1, a_5, a_8과 같이 후행 단위업무가 둘 이상인 경우를 말한다. 반대로 병합 단위업무는 a_5, a_8, a_{11}과 같이 선행하는 단위업무의 수가 다수인 경우이다. 여기서 a_5, a_8은 분기 단위업무이면서 병합 단위업무이다.

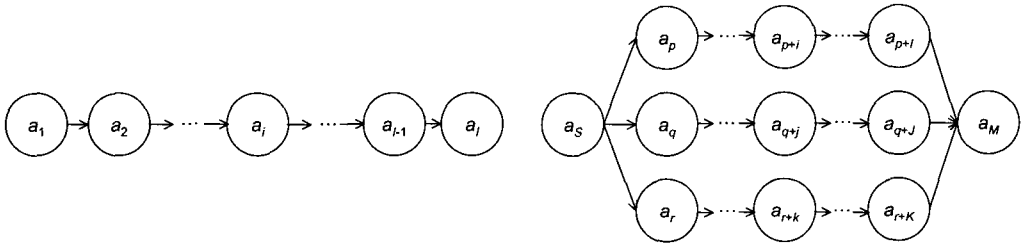
워크플로우 프로세스의 구조는 분기와 병합의 형태에 따라 몇 가지 유형으로 구분된다. 이 구분을 위해서 본 연구에서는 블록(block) 개념[6]을 도입한다. 블록은 크게 직렬(serial) 블록과 병렬(parallel) 블록으로 나누어진다. 직렬 블록은 <그림 2>(가)와 같이 분기와 병합이 발생하지 않는 하나의 경로를 가지는 구조이다. 병렬 블록은 <그림 2>(나)와 같이 분기 단위업무(a_S)로부터 병합 단위업무(a_M)까지 두 단위업무 사이에 다수의 경로가 존재하는 구조이다[20].

병렬 블록은 다시 AND 블록과 OR 블록으로 나눌 수 있다. AND 블록은 모든 경로를 동시에 진행하는 구조인 반면, OR 블록에서는 분기된 여러 경로 중 하나 이상만 수행하면 된다. 즉, AND 블록은 모든 경로를 완료해야 전체 구조를 완료했다고 하고, 하나의 경로라도 완료하지 못하면 전체 구조를 완료하지 못했다고 한다. 반면에 OR 블록은 하나의 경로라도 완료하면 전체 구조를 완료했다고 하며, 모든 경로를 다 완료할 수 없을 때 전체 구조를 완료할 수 없다고 말한다.

OR 블록은 분기의 의미에 따라 다시 몇 가지 종류로 나눌 수 있다. OR블록의 유형은 일반 OR (NOR : Normal OR), 우선 순위 OR (POR : Priority OR), 조건 OR(COR : Conditional OR), 배타적 OR(XOR : eXclusive OR) 등으로 분류될 수 있으며, 본 연구에서는 이들 가운데 다음 세 가지 OR 블록만 다루기로 한다.

- NOR : 이 유형은 가장 일반적인 OR 블록이다. 분기 단위업무에서 다수의 경로가 동시에 발생하지만 어떠한 경로든 가장 먼저 병합 단위업무 직전까지 수행을 완료하면 진행 중인 다른 경로를 무시하고 다음 단계의 단위업무를 수행하는 구조이다.
- POR : 경로에 우선순위를 주어 가장 높은 순위를 갖는 경로를 우선 수행하는 구조이다. 수행중인 경로가 도중에 실패하면 다음 순위의 경로를 시도하고, 반대로 성공적으로 처리되면 나머지 경로를 무시하고 병합 단위업무를 수행한다.
- COR : 각 경로에 조건이 설정되어 있고, 조건을 만족하는 경로를 진행시킨다. 즉, 조건에 따라 수행되는 경로가 달라진다. 이러한 조건은 분기 단위업무 또는 그 이전 업무의 처리 결과에 따라 결정되고 그에 따라 경로 역시 결정된다.

위의 OR 블록 중, 가장 일반적인 NOR에서 하나의 경로가 선택됨으로 인해 무시되는 경로들은 결과적으로 낭비를 초래하는 비효율성이 있어서, 프로세스 관리의 관점에서 POR나 COR를 이용하는 것이 유리하다. 그러나, 각 경로간의 경쟁적인 수



(가) 직렬 블록 (나) 병렬 블록

〈그림 2〉 프로세스 구조

행을 통하여 블록에 의한 수행작업의 질을 높일 경우, NOR를 도입하는 것이 필요하다.

워크플로우에서 다루는 모든 프로세스 모델의 구조는 이러한 여러 블록 유형의 조합을 통해 구성될 수 있다고 가정한다. 본 논문에서 XOR 블록을 고려하지 않는 것은 COR를 이용하여 하나의 경로만을 실행하는 것으로도 충분히 XOR 블록의 의미를 반영하기 때문이다. 본 논문의 주요 목적은 프로세스 구조 자체를 논하는 것이 아니므로 블록에 대한 더 자세한 사항은 [2]을 참고하기 바란다.

2.3 업무 할당 및 처리 규칙

워크플로우 엔진은 실행시 단계에 프로세스 모델을 해석하여 업무를 사용자에게 할당하고, 사용자는 할당된 업무를 처리한다. 이 업무 할당과 처리는 보통 어떤 규칙에 따라 이루어진다.

업무 할당에서 중요한 것은 업무 처리 적임자 선택이다. 보통 이 적임자는 정의시 단계에서 정해둔 사람이나 해당 업무 실적이 우수한 사람을 선정하는 규칙을 이용하여 결정한다. 또 다른 방법으로 해당 업무를 처리할 수 있는 후보 가운데 업무부하가 가장 적은 후보를 선택하는 것도 자주 이용된다. 이 같은 규칙은 워크플로우 엔진의 관점에서는 효과적 또는 효율적 업무 할당 방법이라 생각할 수 있다.

한편, 할당된 업무를 수행하는 업무담당자의 관점을 생각해보자. 워크플로우 시스템 내에는 여러 프로세스 정의를 따르는 다수의 인스턴스가 동시에 진행되는 것이 보통이다. 따라서 개별 사용자에게 한 시점에 여러 업무가 할당되어 있는 것이 일반적일 것이다. 개별 사용자에 할당된 업무집합을 업무항목집합 또는 워크리스트(work list)라 한다. 사용자는 업무의 기대소요시간 및 업무기한을 고려하여 업무항목집합의 업무에 대하여 우선순위를 결정하고 업무를 처리할 것이다. 이를 업무처리 우선순위규칙이라고 하며, 다양한 규칙을 생각해 볼 수 있다.

우선순위규칙(dispatching rule)은 원래 제조 공정의 스케줄링 및 할당 문제의 해법으로 연구되어 왔다[3]. 대표적으로 선입선출(FIFO : First In First Out), 최단처리시간(SPT : Shortest Processing Time), 그리고 임의순서(Random Rule) 등 다양한 종류의 우선순위규칙이 있으며[8, 18], 그 적용 효과에 대한 연구 결과도 매우 많다[16].

그러나 이러한 우선순위규칙이 워크플로우 시스템에서는 아직 체계적으로 적용되지 않고 있고, 더욱이 이런 규칙이 업무 효율성에 미치는 영향에 대한 연구는 지금까지 없었다. 이는 기존의 상용 시스템이 아직은 업무의 자동화된 흐름을 지원하는데 중점을 두고 있기 때문이다. 일부 시스템의 경우, 정의시 단계 또는 프로세스 인스턴스 초기화 때 정해진 프로세스의 중요도에 따라 몇 가지 수준의 업무우선순위를 부여하여 담당자가 업무를 수행할 때 참조할 수 있도록 하지만, 이는 본 연구에서 제안하는 방식과는 전혀 다르며, 그 사용 목적 또한 다르다.

본 연구에서는 전체 프로세스의 완료시간을 단축하기 위해 최소여유시간(LST : Least Slack Time) 규칙의 사용을 제안한다. 이 규칙은 처리해야 할 여러 업무 가운데 여유시간이 가장 작은 업무부터 먼저 처리하는 것이다. LST 규칙은 제조공정의 스케줄링에서 병목현상을 줄이기 위한 대기행렬(queue) 관리 방법으로 활용되어 왔으며[9], 작업 지연의 분산을 최소화하는데 가장 적합한 방식으로 알려져 있다[14]. 사용자에게 할당된 업무의 집합을 하나의 대기행렬로 간주할 수 있다. 따라서, LST 규칙을 적용함으로써 프로세스 효율을 전체적으로 향상하는 것을 기대할 수 있다. 그러나, LST를 적용하기 위해서는 그 명칭에서 알 수 있는 것과 같이 업무의 여유시간을 계산할 수 있어야 한다. 본 연구에서는 이를 위하여 PERT/CPM을 이용한다.

3. PERT/CPM 적용을 위한 고려사항

전통적 프로젝트 관리 기법인 PERT/CPM[13]은

프로젝트를 다수의 활동으로 이루어진 네트워크로 모델링 하여, 이를 시간의 관점에서 계획하고 평가한다. 이 시간 관리는 주경로 및 여유시간 개념을 통해 이루어진다. 이 개념을 이용해서 어떤 경로가 집중 관리를 요하는지 또는 각 활동이 얼마나 시간적 여유가 있는지 분석할 수 있다. 이러한 점은 WFMS에서도 역시 중요하다. 즉, 업무담당자가 자신에게 할당된 업무의 수행이 얼마나 다급한지 알 수 있다면, 이를 전체 프로세스의 효율 향상을 위해 이용할 수 있을 것이다.

그러나, WFMS에 PERT/CPM 기법을 도입하는 데에는 몇 가지 제약이 있다. 이 제약은 두 기법이 사용하는 모델 간의 구조적, 개념적 차이에서 비롯된다. WFMS는 프로세스를 자동으로 실행하기 위하여 여러 대안(alternative) 경로를 모두 프로세스 모델에 반영한다. 반면에, PERT/CPM에서는 네트워크를 이루는 모든 활동을 수행하는 것을 전제로 하는데 이것은 반드시 수행할 경로만을 네트워크로 표현하기 때문에 그렇다.

두 모델의 구조적 차이는 대안 경로의 표현, 즉, 경로가 분기되는 방식에서 다음과 같이 발생한다. 만일, 워크플로우 프로세스 모델에 분기가 없거나 AND 블록만 있다면, 이 모델은 PERT/CPM 네트워크와 다른 점이 없다. 그러나, OR 블록을 포함하는 경우에는 그 블록의 어떤 경로가 실제로 실행되고 어떤 경로가 실행되지 않을지 결정이 되어야 하는 것이 PERT/CPM 네트워크와 다른 점이다. 이는 실행될 경로가 정해지기 이전에는 전체 프로세스의 기대소요시간을 구하는 것을 어렵게 만들고, 또 실제 소요시간도 어떤 경로가 실행되는가에 따라 달라진다. 결국 실제로 실행할 OR 블록의 경로가 정해지기 전에는 프로세스의 주경로 역시 결정할 수 없고, 따라서 각 업무의 여유시간도 구할 수 없는 문제가 발생한다.

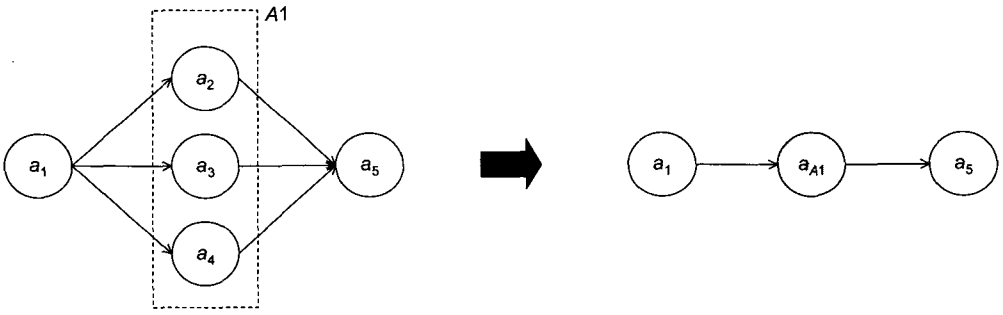
4. 주경로 및 여유시간

본 장에서는 워크플로우 프로세스 모델을 위한 주경로 및 여유시간 계산에서 OR 블록을 처리하는 방법을 설명하고, 이 주경로와 여유시간을 활용한 LST 우선순위규칙을 제안한다.

4.1 기대경로와 기대소요시간

프로세스 모델이 AND 블록만을 포함하는 경우에는 PERT/CPM 방식을 그대로 적용할 수 있다. 그러므로 본 절에서는 OR 블록을 포함하는 경우만을 설명한다. 이 경우 주경로와 여유시간은 확률적 접근법으로 구할 수 있다.

<그림 3>에서 OR 블록은 분기 단위업무 a_1과 병합 단위업무 a_5 사이에 세 개의 경로 $r_1=\{a_1, a_2, a_5\}$, $r_2=\{a_1, a_3, a_5\}$, $r_3=\{a_1, a_4, a_5\}$를 가진다. 이 세 경로를 하나의 경로 $\{a_1, A_1, a_5\}$로 표현하여 '기대경로'라고 하고, 이 기대경로에 대하여 기대소요시간을 산출한다. 그런데 2.2절에서 설



〈그림 3〉 OR 블록의 기대시간

명한 OR 분기의 유형에 따라 계산 방법과 사용하게 될 확률 정보가 아래와 같이 달라진다.

4.1.1 NOR

NOR 블록의 기대소요시간은 각 경로가 성공적으로 수행될 확률을 이용하여 구한다. <그림 3>의 예제가 NOR 블록이라고 하자. 각 경로는 분기 단위업무 a_1과 병합 단위업무 a_5 사이에 하나의 단위업무만을 가진다. 이때, i-번째 경로의 실제 소요시간분포, X_i는 λ_i을 서비스율(service rate)로 갖는 지수분포(exponential distribution)를 따른다고 가정한다[4]. 즉,

$$X_i \sim \mathrm{EXP}(\lambda_i),(i=1,2,3)$$

가 된다. 만일, 경로가 두 개 이상의 단위업무를 포함하면, 각 단위업무에 대한 지수분포를 합하여 경로 전체의 소요시간 분포를 구할 수 있다.

이제 P_i를 i-번째 경로 r_i가 가장 먼저 완료될 확률이라고 하면,

$$P_1 = P(X_1 < X_2) \cdot P(X_1 < X_3)$$

가 된다. 각 X_i들이 독립적이며 동일한 분포를 따른다고 할 때,

$$\begin{aligned} P(X_1 < X_2) &= \int_0^\infty P(X_1 < x \mid X_2 = x) \\ &\quad \cdot P(X_2 = x)dx \\ &= \int_0^\infty (1-\exp(-\lambda_1 x)) \cdot \lambda_2 \\ &\quad \cdot \exp(-\lambda_2 x)dx \\ &= \frac{\lambda_1}{\lambda_1+\lambda_2} \end{aligned}$$

가 되고, 같은 방법으로

$$P(X_1 < X_3) = \frac{\lambda_1}{\lambda_1+\lambda_3}$$

가 되어,

$$P_1 = \frac{\lambda_1}{\lambda_1+\lambda_2} \cdot \frac{\lambda_1}{\lambda_1+\lambda_3}$$

가 된다. P_2, P_3도 같은 방법으로 구할 수 있다. ET_i를 r_i에 속해 있는 분기된 업무의 기대소요시간이라고 하면, NOR 블록의 기대소요시간 ET는 다음과 같이 구해진다.

$$\begin{aligned} ET &= ET_1P_1 + ET_2P_2 + ET_3P_3 \\ &= \left(\frac{\lambda_1}{\lambda_1+\lambda_2} \cdot \frac{\lambda_1}{\lambda_1+\lambda_3}\right)ET_1 \\ &+ \left(\frac{\lambda_2}{\lambda_1+\lambda_2} \cdot \frac{\lambda_2}{\lambda_2+\lambda_3}\right)ET_2 \\ &+ \left(\frac{\lambda_3}{\lambda_1+\lambda_3} \cdot \frac{\lambda_3}{\lambda_2+\lambda_3}\right)ET_3 \end{aligned}$$

이를 n개의 경로를 가지는 NOR 블록으로 일반화하면 다음과 같다.

$$ET = \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} \left(\frac{\lambda_i}{\lambda_i+\lambda_j}\right)ET_i \quad (1)$$

4.1.2 POR

POR 블록이 n개 경로 r_1, r_2, …, r_n을 가진다고 하자. 이 때, 일반성을 잃지 않고 r_i가 r_{i+1}보다 항상 우선순위가 높다고 가정할 수 있다. POR 블록에서 각 경로 r_i가 먼저 수행될 확률 P_i를 본 연구에서는 다음과 같이 POR 블록을 수행한 과거 데이터로부터 구하기로 한다.

$$P_i = \frac{\text{우선순위 } i\text{-번째 단위 업무가 성공적으로 처리된 인스턴스 수}}{\text{우선순위 } i\text{-번째 단위업무가 도달한 프로세스 인스턴스 수}}$$

이 때, POR 블록에서 r_i가 성공적으로 완료될 확률은 우선순위가 r_i보다 높은 경로들이 모두 실패한 후, r_i가 성공할 확률이 된다. 따라서 n개의 경로를 포함한 POR 블록의 기대경로 기대소요시간은 다음과 같다.

$$ET = \begin{cases} P_1 \cdot ET_1 & , n=1 \\ P_1 \cdot ET_1 + \sum_{i=1}^{n}[\{\prod_{j=1}^{i-1} & \\ (1-P_j)\}P_i \cdot \{\sum_{k=1}^{i} ET_k\}] & , n \geq 2 \end{cases} \quad (2)$$

4.1.3 COR

COR 블록은 분기 단위업무의 수행 결과에 따라 이후의 경로가 결정되는 구조이다. COR에서도 POR과 같이 각 경로를 선택할 확률은 과거 데이터를 토대로 산출한다. COR에서 i-번째 경로가 선택될 확률 P_i는 다음과 같이 계산된다.

$$P_i = \frac{i\text{-번째 경로를 선택한 인스턴스 수}}{\text{전체 프로세스 인스턴스 수}}$$

이 때, COR 블록의 기대소요시간은 각 경로가 선택될 확률과 각 경로의 기대소요시간의 곱을 모두 합한 것으로 다음과 같다.

$$ET = \sum_{i=1}^{n}(P_i \cdot ET_i) \quad (3)$$

4.2 여유시간 계산 및 업무처리 방식

본 절에서는 주경로와 여유시간 계산 방법을 예를 들어 설명하고, 이를 업무처리 규칙으로 활용하는 방안을 제안한다. 아래 <그림 4>의 예제를 생각해보자.

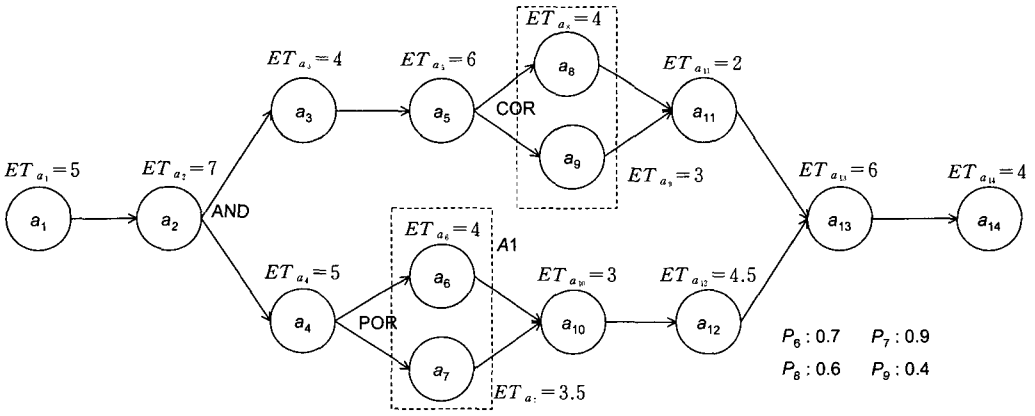
그림에서 보는 것처럼 이 예제는 AND, POR, COR 블록을 포함하고 있다. 주경로와 여유시간 계산을 위하여, 먼저 POR와 COR 블록 각각의 기대경로 $A1$, $A2$의 기대소요시간을 구한다. 그림에서 P_6, P_7는 POR 블록에서 분기된 두 경로가 성공적으로 완료될 확률이고, P_8, P_9는 COR 블록에서 a_5가 완료된 후 a_8로 진행할 확률과, a_9로 진행할 확률이다. $A1$, $A2$의 기대소요시간은 식 (2), 식 (3)에 의하여 다음과 같이 구해진다.

$$\begin{aligned} ET_{A1} &= P_6 \times 4 + (1-P_6) \times P_7 \times (4+3.5) \\ &= 0.7 \times 4 + 0.3 \times 0.9 \times 7.5 = 4.825 \\ ET_{A2} &= P_8 \times 4 + P_9 \times 3 \\ &= 0.6 \times 4 + 0.4 \times 3 = 3.6 \end{aligned}$$

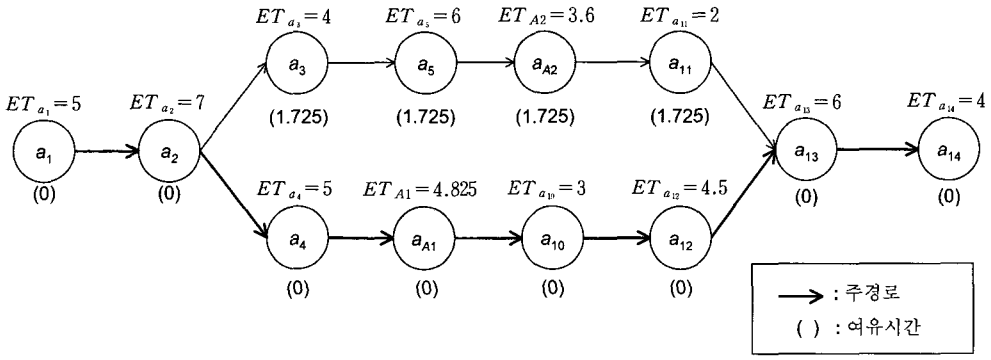
이제, 예제 프로세스에서 기대경로 $A1$, $A2$를 각각 하나의 단위업무로 간주하여 <그림 5>와 같이 a_{A1}, a_{A2}로 표현하였다. 이처럼 프로세스 모델에서 블록을 하나의 단위업무로 간주하여 모델을 점점 간단하게 줄여가는 것은 [11]을 참고하면 된다.

이제 <그림 5>의 프로세스는 다음의 두 경로 r_1, r_2를 가진다.

$$r_1 = \{a_1, a_2, a_3, a_5, a_{A2}, a_{11}, a_{13}, a_{14}\}$$
$$r_2 = \{a_1, a_2, a_4, a_{A1}, a_{10}, a_{12}, a_{13}, a_{14}\}$$



〈그림 4〉 여유시간 계산 예



〈그림 5〉 여유시간 계산 결과

각 단위업무의 기대소요시간을 토대로 두 경로의 기대소요시간을 계산하면

$$ET_1 = 5+7+4+6+3.6+2+6+4 = 37.6$$
$$ET_2 = 5+7+5+4.825+3+4.5+6+4 = 39.935$$

이므로, r_2가 주경로가 된다.

잘 알려져 있는 것처럼 주경로 상의 단위업무는 여유시간이 없고, 그렇지 않은 단위업무는 여유시간이 있다. 여유시간은 주경로의 소요시간과 해당 단위업무가 속한 경로의 소요시간의 차이로 구할 수 있다. 즉, 예제에서 주경로 r_2의 모든 단위업무의 여유시간은 0이고, 경로 r_1에만 속한 단위업무의 여유시간은 경로 r_2와 r_1의 소요시간 차이인 1.725가 된다.

한편, 프로세스를 실행하면서 주경로와 여유시간을 재계산할 필요가 있다. 이는 실제 단위업무 수행시간이 기대소요시간과 다를 수 있고, 또 OR 블록의 경우 실제 수행되는 경로가 프로세스 수행 과정에서 결정되기 때문이다. 이 같은 재계산의 필요성이 발생할 때마다 주경로와 여유시간을 갱신해야 한다. 재계산 알고리즘은 워크플로우 프로세스의 특징을 고려하여 간단히 만들 수 있는데, 구체적인 내용은 [1]을 참고하면 된다.

어떤 단위업무의 여유시간은 그 업무의 수행이 기대소요시간보다 늦어져도 전체 프로세스의 완료시간에 영향을 주지 않는 최대 시간이다. 반대로 여유시간이 0인 단위업무의 지연은 전체 프로세스의 완료시간을 늦춘다. 즉, 여유시간의 길이는 프로세스 완료시간을 단축하기 위한 업무의 긴급도를 나타낸다. 따라서, 업무담당자에게 할당된 여러 단위업무 중 여유시간이 작은 것부터 처리하게 하면, 전체 프로세스의 완료시간을 단축하는 데 효과적일 것이라 예상할 수 있다. 이것을 LST 규칙이라고 한다.

LST 규칙의 적용 방법을 사용자 관점에서 살펴보자. 업무담당자는 일반적으로 다수의 프로세스 인스턴스에 참여하며 그 종류도 다양할 수 있지만, 설명의 편의를 위하여 단일 프로세스에 대하여 여러 인스턴스가 발생한 경우를 고려한다. 특정 업무담당자가 <그림 4>의 프로세스 인스턴스에 대하여 a_5와 a_{10}의 업무 인스턴스를 할당 받았다면 두 업무 중 하나를 먼저 수행해야 한다. 만일, a_5와 a_{10}의 여유시간이 각각 0, 1.325라면, 이 업무담당자는 LST 규칙에 의해 a_5를 우선 처리한다. 처리된 a_5의 실제수행시간이 예상 수행시간과 차이가 있다면, 실제 수행시간을 기반으로 여유시간을 재계산한다. 재계산된 시간 정보는 각 업무목록에서 처리되지 않은 단위업무의 여유시간에 반영된다. 이처럼 여유시간은 프로세스가 진행하면서 지속적으로 변화하는데, 해당 업무담당자는 처리할 업무를 선택하는 시점에서 여유시간 값이 가장 작은 단

위업무를 우선 처리한다.

위와 같은 LST 규칙의 사용은 다음과 같은 효과가 있다. AND 블록의 병합 지점에서 다음 업무로 진행하기 위해서는 모든 분기 경로의 수행이 완료되어야 한다. 따라서 먼저 완료된 경로는 가장 마지막 경로가 완료될 때까지 대기해야 한다. LST 규칙은 이 대기시간을 줄여 프로세스 효율을 증가시킬 것이다. 이러한 효과는 업무담당자를 포함하여 업무 수행에 필요한 자원의 이용률(utilization) 증가로 이어질 것이다. 마지막으로, 전술한 LST 규칙의 적용 예를 발전시켜보면 각 업무목록에 쌓인 단위업무의 수가 많을 때, 그 효과는 더욱 커질 것으로 기대할 수 있다. 이러한 예상되는 효과를 다음 장에서 시뮬레이션 실험을 이용하여 검증한다.

5. 시뮬레이션 실험

제안한 LST 규칙의 유용성을 검증하기 위해 시뮬레이션 실험을 수행하였다. 앞에서 설명한 것처럼 기존의 워크플로우 시스템은 이 같은 규칙을 적용하는 데 필요한 정보를 제공하지 않는다. 따라서 업무담당자는 일반적으로 먼저 할당된 업무를 먼저 처리하는 선입선출 규칙을 사용한다고 보아야 할 것이다. 본 실험에서는 LST 규칙을 선입선출과 비교하였다.

실험에 사용한 세 개의 프로세스 모델은 부록에 수록하였다. 이 프로세스를 6명의 업무담당자가 2명씩 3개의 역할그룹(role group)으로 나뉘어 업무를 수행하는 것을 가정하였다. 역할그룹은 같은 일을 수행할 수 있는 직원의 그룹을 말한다. 정의시 단계에서 각 단위업무의 책임자를 역할그룹 단위로 지정하고, 실행시 단계에서는 특정 단위업무를 책임지는 역할그룹 내 업무담당자 가운데 업무 부하가 가장 적은 담당자에게 단위업무를 할당하였다.

시뮬레이션 소프트웨어로는 Arena 6.0을 사용하였다. 단위업무의 수행시간 및 프로세스 인스턴스 발생 간격은 지수분포를 따른다고 가정하였으며, 이 분포의 평균값(parameter)을 여러 가지로 변화시키면서 실험하였다. 또한, 1회당 수행시간(replication time)을 10,000으로 하였으며, 안정적인 결과를 얻기 위하여 초기준비기간(warming up period)을 2,000으로 하였다.

5.1 LST 규칙의 효과 분석

LST 규칙의 효과를 예제 프로세스에 대해서 프로세스 효율과 업무담당자 효율성을 이용하여 분석하였다. 먼저 프로세스 효율은 프로세스 평균 완료시간과 경로유휴시간으로 측정하였다. 여기서, 경로유휴시간이란 AND 분기 이후의 경로 가운데 가장 빨리 완료된 경로가 병합업무에 도달한 순간부터 가장 느린 경로가 병합업무에 도달한 시각까지의 차이이다. 아래 <표 1>은 프로세스 인스턴스

〈표 1〉 프로세스 평균완료시간과 경로유휴시간

업무할당규칙	프로세스 1		프로세스 2		프로세스 3	
	프로세스 완료시간	경로유휴 시간	프로세스 완료시간	경로유휴 시간	프로세스 완료시간	경로유휴 시간
선입선출	3,357.70	1,342.55	3,401.10	346.05	4,069.66	1,631.10
LST Rule	2,444.99	1,025.63	2,494.08	215.78	3,266.50	1,227.27
비율(%)	27.2	23.6	26.7	37.6	19.7	24.8
t 통계량*	3.39	2.28	4.81	5.52	2.49	1.90
t 기각치*	1.68	1.68	1.67	1.69	1.69	1.71

주) * t-검정의 가설은 'H_0 : (LST하의 프로세스 완료(유휴)시간) - (FIFO하의 프로세스 완료(유휴)시간) = 0
H_1 : (LST하의 프로세스 완료(유휴)시간) - (FIFO하의 프로세스 완료(유휴)시간) > 0'와 같다.

발생 간격을 260으로 하였을 때 각 프로세스 모델에 대하여 30회 실험한 결과를 요약한 것이다.

<표 1>에서 보는 바와 같이 실험 결과 프로세스 평균완료시간과 경로유휴시간에 대해서 LST 규칙이 선입선출에 비해 20% 이상 효율적임을 알 수 있다. 그리고, t-검정을 통하여 LST가 선입선출에 비해 효율적이라는 것을 유의수준 5%에서 확인하였다.

한편, LST 규칙이 구체적으로 얼마나 효율적인지를 t-검정을 통해 알아보았다. <표 2>는 프로세스 완료시간과 경로유휴시간 각각에 대해 유의수준 5%에서 비교하는 두 방식의 유의한 차이의 최대값과 선입선출 대비 개선 비율을 정리한 것이다.

예를 들어 프로세스 1의 경우 LST가 프로세스를 평균 460(13.6%) 빨리 처리할 수 있으며, 프로세스 유휴시간은 85(6.3%) 적다고 할 수 있다.

위 실험 분석은 LST 규칙이 경로유휴시간 감소와 프로세스 완료시간 단축에 효과적임을 입증하고 있다. LST 규칙은 주경로 업무 처리에 자원을 우선적으로 할당하는 것으로 생각할 수도 있다. 따라서 LST 규칙은 AND 병합 단위업무에서 주경로 업무가 완료될 때까지 불필요하게 기다려야 하는 시간을 단축시킨다. 이는 결국 특정 프로세스가 지나치게 지연됨이 없이 여러 프로세스가 균형적으로 진행되도록 하고, 또 전체적인 프로세스 진행 시간도 줄일 수 있다.

또, LST 규칙의 사용은 업무담당자의 효율성에도 영향을 미칠 것이다. 이러한 예상을 확인하기 위해, <표 3>과 같이 각 업무담당자의 업무목록에 들어있는 단위업무의 평균체류시간을 구하였다. 이것은 어떤 단위업무가 업무목록에 도착한 후 해당 업무담당자가 그 단위업무를 시작하기까지 업무목록 내에서 머무는 시간의 평균이다. 최대 차이는 <표 2>에서 사용한 방법과 동일하게 구하였다.

표에서 볼 수 있듯이, 업무 처리 효율성에 있어서도 LST 규칙을 도입한 경우가 상대적으로 좋은 결과를 가져왔다. 그룹 3의 경우 유의한 최대 차이가 0이지만, 여전히 "선입선출 > LST Rule"을 의미하므로 LST 규칙이 더 효율적이라 할 수 있다. 한편, 역할그룹에 따라 개선 효과에 차이가 나타난

〈표 2〉 유의한 프로세스 완료시간 및 경로유휴시간 차이의 최대값

	프로세스 1		프로세스 2		프로세스 3	
	프로세스 완료시간	경로유휴시간	프로세스 완료시간	경로유휴시간	프로세스 완료시간	경로유휴시간
유의한 최대 개선 효과(Max {c}*)	460	85	590	90	255	35
선입선출 대비 개선 비율(%)	13.6	6.3	17.3	26.0	6.3	2.1

주) * H_0 : (LST하의 프로세스 완료(경로유휴)시간) - (FIFO하의 프로세스 완료(경로유휴)시간) = c,
H_1 : (LST하의 프로세스 완료(경로유휴)시간) - (FIFO하의 프로세스 완료(경로유휴)시간) > c,
c를 간격 5만큼씩 증가시키면서 H_0를 기각하는 최대의 c를 찾음.

〈표 3〉 업무담당자별 업무목록 내 단위업무 평균체류시간

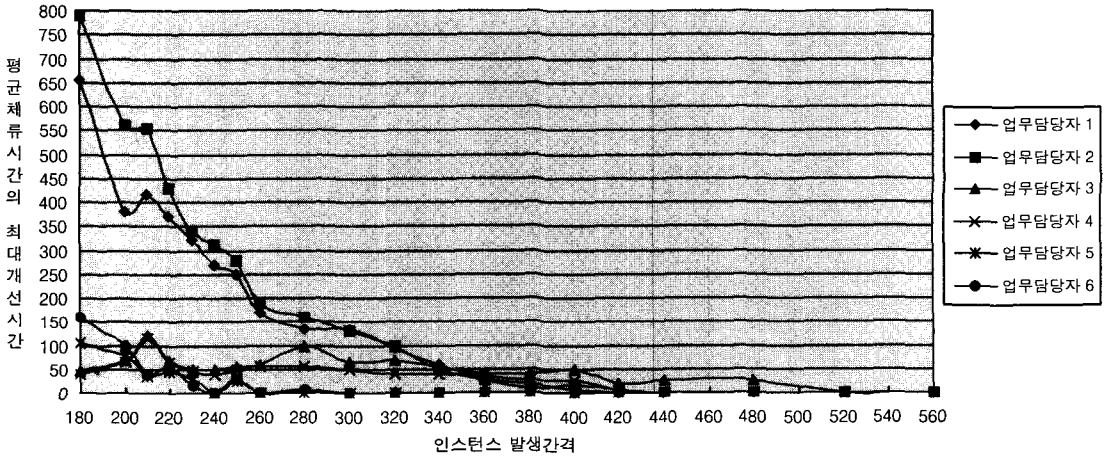
	역할그룹 1		역할그룹 2		역할그룹 3	
	업무담당자 1	업무담당자 2	업무담당자 3	업무담당자 4	업무담당자 5	업무담당자 6
선입선출	539.94	523.25	610.88	596.49	497.47	539.34
LST 규칙	301.58	274.27	446.12	438.22	400.50	423.30
최대 차이	170	190	60	55	0	0

다. 시간이 적게 걸리는 단위업무나 지체도가 덜한 프로세스에 위치한 단위업무를 상대적으로 많이 책임지는 그룹은 평균체류시간이 작을 것이라고 예상할 수 있으나, 이에 대한 설득력 있는 설명을 위해서는 추가 연구가 필요하다.

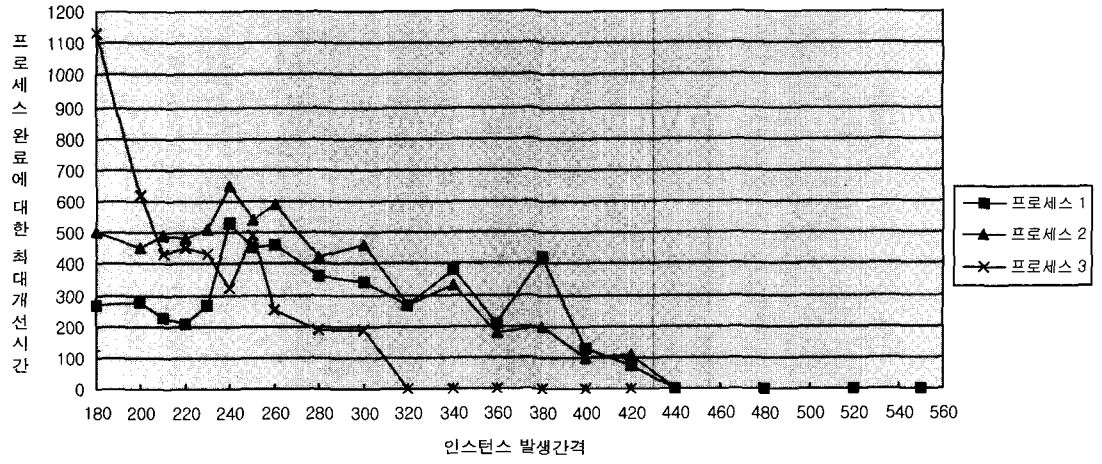
5.2 업무부하의 영향 분석

지금까지의 분석은 동일한 인스턴스 발생 간격 하에서의 결과이다. 이 결과는 업무 부하가 변함에 따라 달라질 수 있을 것이다. 따라서, 프로세스 인스턴스 발생 간격을 변화해가며 동일한 실험을 반복하여 결과를 비교하였다. <그림 6>과 <그림 7>은 각각 인스턴스 발생 간격이 증가함에 따라 업무담당자 및 프로세스 효율이 변화하는 것을 보여주고 있다.

위 그래프에서 공통적으로 인스턴스 발생 간격이 커짐에 따라 LST 규칙의 최대개선효과가 점점 작아지는 것을 볼 수 있다. 인스턴스 발생 간격이 커지면 업무량이 줄어들게 된다. 따라서 위의 결과는 업무량이 줄어들수록 LST 규칙에 의한 업무담당자의 이용률과 프로세스 완료시간 단축에 있어



〈그림 6〉 프로세스 인스턴스 발생 간격에 따른 이용률 개선 추이



〈그림 7〉 프로세스 인스턴스 발생 간격에 따른 시간 개선 추이

서의 개선효과가 줄어든다는 것을 의미한다. 이것은 본 논문에서 제안하는 우선순위규칙의 활용이 업무 부하가 많은 상황에서 더욱 필요한 전략임을 말해준다.

6. 결론 및 추후 연구과제

본 논문에서는 워크플로우 프로세스를 효율적으로 처리할 수 있는 우선순위규칙을 제안하고, 시뮬레이션으로 그 유용성을 실험하였다. 기존의 WFMS는 프로세스의 자동화된 수행, 조정, 제어에 중점을 두었기 때문에, 업무담당자에게 업무 수행과 관련된 시간 정보를 제공하거나, 프로세스 전체의 관점에서 효율적으로 업무를 처리하는 방향을 제시하지는 못하였다. 이런 단점을 해결하기 위하여 본 연구에서는 PERT/CPM 기법에서 제공하는 시간 정보를 이용할 수 있는 방안을 고안하였다. 여유시간을 이용하는 LST 규칙은 상대적으로 시급한 업무를 먼저 처리하는 규칙으로 프로세스의 효율을 높일 수 있다. 시뮬레이션 실험 결과는 제안하는 방법의 효과를 사용한 예제 환경에서 입증할 수 있었다. 또, 업무부하에 따른 규칙의 상대적 효과도 분석하였으며, 이를 통해 제안한 방법이 프로세스가 복잡하고, 업무부하가 많은 환경일수록 효과가 높음을 알 수 있었다.

본 연구의 결과는 추후 여러 관점에서 확장 보완할 필요가 있다. 먼저, 여유시간을 이용하는 우선순위규칙의 영향을 여러 유형의 프로세스에 대해 살펴볼 필요가 있다. 또, 업무 참여자의 능력이나 숙련도, 참여자의 수, 자동화된 업무의 개입, 프로세스 인스턴스 발생 패턴의 시간에 따른 변화 등과 같은 다양한 요소를 고려하여 실험하는 것도 필요하다. 이러한 요인들을 고려한 실험 결과는 워크플로우 시스템에 입력되어 효과적으로 프로세스를 진행하는 데 활용될 수 있을 것이다. 또, PERT/CPM 외에 스케줄링 기법, 제약이론(Theory of constraint) 등을 활용하여 프로세스를 효율화 하는 방법도 연구할 필요가 있다

참 고 문 헌

[1] 김영호, 이승현, 「실시간 여유시간 계산을 위한 알고리즘」, SNU IS Lab. 기술문서, SNUIS-TC-1017, 2002.

[2] J.S. Bae, *Automatic Enactment of Workflow Process using Active Databases*, Ph. D. dissertation, Seoul Nat'l Univ., Seoul, Korea, 2000.

[3] K.R. Baker, *Introduction to sequencing and scheduling*, Wiley, 1974.

[4] D.-H. Chang, J.H. Son and M.H. Kim, "Critical path identification in the context of a workflow," *Information and Software Engineering*, Vol.44, No.7(2002), pp. 405-417.

[5] Y.-L. Chang, S. Chen, C.-C. Chen and I. Chen, "Workflow process definition and their applications in e-commerce," *Proceedings of International Symposium on Multimedia Software Engineering*, (2000), pp.193-200.

[6] A. Dogac, E. Gokkoca, S. Arpinar, P. Koksal, I. Cingil, B. Arpinar, N. Tatbul, P. Karagoz, U. Halici, and M. Altinel, "Design and implementation of a distributed workflow management system : METUFlow," *Proc. NATO Advanced Study Institute (ASI) Workshop on Workflow Management Systems and Interoperability*, (Aug. 1997), pp.61-66.

[7] D. Hollingsworth, Workflow Management Coalition Specification : The Workflow Reference Model, WfMC specification, 1994, http://www.wfmc.org.

[8] O. Holthaus and C. Rajendran, "Efficient dispatching rules for scheduling in a job shop," *International Journal of Production Economics*, Vol.48, No.1(1997), pp.87-105.

[9] Y. Hu, R. Minciardi, M. Paolucci and R. Pesenti, "Techniques for dynamic scheduling in a manufacturing environment," *Proceedings of the 31st IEEE Conference on Decision and Control*, Vol.1(1992), pp.404-408.

[10] Y. Kim, S. Kang, D. Kim, J. Bae and K. Ju, "WW-Flow : Web-based workflow management with runtime encapsulation," *IEEE Internet Computing*, Vol.4, No.3(2000), pp. 55-64.

[11] Y. Kim, J. Bae, H. Bae and S. Kang, "Automatic Control of Workflow Processes using ECA Rules," Submitted to IEEE Transactions on Knowledge and Data Engineering, 2002.

[12] P. Lawrence, *Workflow Handbook*, Wiley, New York, 1997.

[13] R.I. Levin and C.A. Kirkpatrick, *Planning and control with PERT/CPM*, McGraw-Hill, New York, 1966.

[14] S.H. Lu and P.R. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Transactions on Automatic Control*, Vol.36, No.12(1991), pp.1406-1416.

[15] J.A. Miller, A.P. Sheth, K.J. Kochut and X. Wang, "CORBA-based run-time architectures for workflow management systems," *Journal of Database Management*, Vol.7, No.1(1996), pp.16-27.

[16] S.C. Park, N. Raman and M.J. Shaw, "Adaptive scheduling in dynamic flexible manufacturing systems : a dynamic rule selection approach," *IEEE Transactions on Robotics and Automation*, Vol.13, No.4 (1997), pp.486-502.

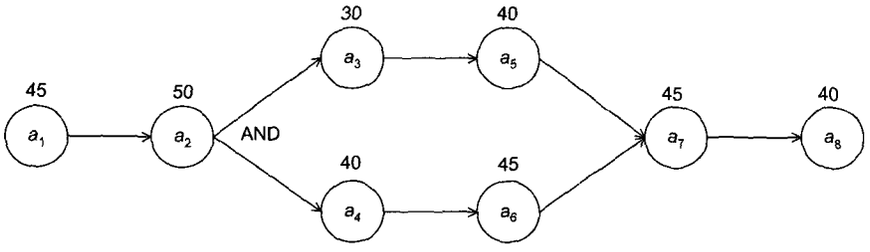
[17] H. Pozewaunig, J. Eder and W. Liebhart, "ePERT : Extending PERT for workflow management systems," *The First European Symposium in ADBIS*, (1997), pp.217-224.

[18] C. Rajendran and H. Ziegler, "A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs," *European Journal of Operational Research*, Vol.131, No.3(2001), pp.622-634.

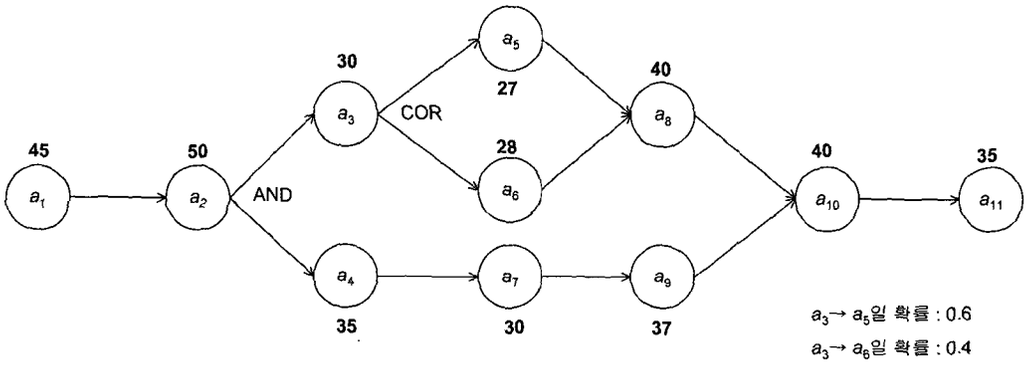
[19] Workflow Management Coalition : Process Definition Interchange, Document No. WfMC-TC-1016-P, 1999, http://www.wfmc.org.

[20] Workflow Management Coalition : Terminology & Glossary, Document No. WFMC-TC-1011, 1999, http://www.wfmc.org.

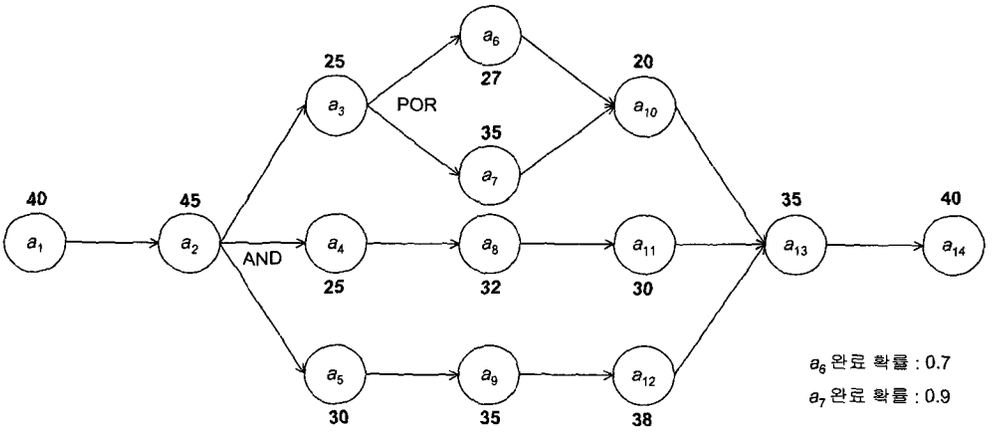
〈부록〉 시뮬레이션에 사용된 프로세스 모델



(가) 프로세스 1



(나) 프로세스 2



(다) 프로세스 3