워크플로우 관리시스템을 위한 제약이론 적용방안에 대한 연구
TOC based Approach for Workflow Management System

김훈태*, 안동준**, 강석호***
*대진대학교 산업시스템공학과, **오토에버시스템즈, ***서울대학교 산업공학과

Abstract

본 논문은 제약이론의 프로세스 운영 개선기법에 기반하여 워크플로우에서 제약 자원을 찾고, 워크플로우의 운영 성과가 개선될 수 있도록 효율적인 업무 흐름 관리 방법을 개발하는 것을 목적으로 한다. 과거의 워크플로우 기록을 분석하여 제약업무담당자(CCP)를 결정하고, 제약 업무 담당자의 업무처리속도에 발생된 프로세스 인스턴스의 워크플로우 투입속도를 동기화하여 CCP의 업무부하가 일정수준으로 유지되도록 하며 비 제약 업무담당자의 여유능력이 향상되도록 한다. 이를 통해 CCP의 업무부하를 가능한 업무담당자들에 분산시킴으로써 프로세스 인스턴스의 처리가 원활히 이루어지도록 한다.

본 논문에서 개발한 알고리즘을 워크플로우 관리시스템에 적용함으로써 전체 워크플로우가 효율적으로 운영되는 효과를 기대할 수 있다.

1. 서론

워크플로우란 문서와 정보, 혹은 업무가 일련의 절차나 규칙에 따라 한 참여자에서 또 다른 참여자로 전달되는 전체 혹은 부분적으로 구현된 비즈니스 프로세스의 자동화를 말하며, 워크플로우 관리 시스템은 IT도구나 어플리케이션을 호출하면서 프로세스 정의를 해석하고, 워크플로우 참여자와 상호작용하며, 하나 이상의 워크플로우 엔진 상에서 실행 가능한 워크플로우를 정의하고 생성하고 관리해주는 소프트웨어 시스템이다[4].

워크플로우 관련 기존의 연구들은 주로 프로세스 모델 자체에 관한 것과 컴퓨터를 통해 이를 구현할 수 있는 방안에 대해 진행되어왔다. 이러한 연구들을 바탕으로 워크플로우 관리 시스템은 복잡한 개별 프로세스 모델을 정의하고 정확한 순서대로 수행하는 것을 보장하게 되었다. 그러나 전체 프로세스가 모두 효율적으로 진행되는 것을 보장하지는 못한다[2]. 그러한 인식을 바탕으로 최근에는 워크플로우 운영상의 문제를 집중적으로 다루는 연구들이 나타나게 되었다.

본 논문은 시스템 개선기법인 제약이론의 개념을 워크플로우에 도입하여 제약이 존재하는 워크플로우의 운영을 개선하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 워크플로우 효율적 관리와 제약이론의 연구 현황을 알아보고, 제약이론 도입의 필요성을 살펴본다. 다음으로, 3장에서는 제약이론에서 소개되는 집중개선 단계와 DBR 스케줄링 기법을 참고하여 개발한 워크플로우 운영개선 알고리즘을 자세히 설명한다. 4장에서는 기존의 워크플로우 운영방식과 본 논문에서 개발한 운영방식을 시뮬레이션 실험을 통해 비교 분석하며, 5장에서 제약이론의 워크플로우 도입 효과를 정리하고 결론을 맺는다.

2. 워크플로우의 효율적 관리

워크플로우의 운영상의 문제를 다루는 연구들에는, 프로젝트 관리기법을 도입하여 복잡한 워크플로우에서 핵심관리 대상이 되는 주공정경로(critical path)를 찾는 방안[1], 워크플로우 로그나 현장의 데이터를 수집, 분석하여 업무 처리 시간의 추정을 통한 워크플로우 라이프 사이클 시간관리[2], 워크플로우 관리 문제를 스케줄링 모델에 연관시키려는 시도[5] 등이 있다.

프로세스를 효율적으로 운영하고자하는 이러한 연구들을 통해 워크플로우 관리시스템은 프로세스를 정확히 운영하는 것뿐만 아니라 프로세스를 재설계하며 효율적으로 운영하는 경영혁신 도구로서 고려대상이 되게 되었다. 워크플로우 관리 시스템이 좀 더 지능적으로 작동하여 업무의 흐름을 최적화하기 위해서 철저히 업무 진행을 통제하여 가용한 자원을 효율적으로 관리해야 한다. 제약이론은 생산 프로세스에서 제약 자원을 찾고 그 자원의 집중 관리를 통해 작업 진행을 효율적으로 통제하여 시스템의 성과를 극대화 하고자 하는 방법론이다. 이러한 제약이론을 활용하면 자원의 효율적 관리와 업무 흐름 최적화에 많은 도움을 줄 수 있다.

제약이론의 목적은 시스템의 어느 한 부분에 대해 집중투자, 집중관리를 통해 전체 최적화를 이루고자 하는 것이다. Goldratt의 The Haystack Syndrome [3]을 통해 제약이론의 생산개선활동는 다음의 집중개선 5단계로 정리된다.

1. 제약조건을 찾아낸다.(Identifying the constraint)
2. 제약조건을 철저히 활용한다.(Exploiting the constraint)
3. 제약조건 이외의 것은 제약조건에 종속 시킨다.

(Subordinating the remaining resources)
4. 제약조건의 능력을 향상 시킨다.(Elevating the constraint)
5. 타성에 젖지 않도록 주의하면서 제1단계로 되돌아간다.(Repeating the process)

생산 프로세스에 위의 개선활동을 구체적으로 실현해 주는 도구가 DBR 스케줄링 기법이다. 드럼, 버퍼, 로프는 생산 공정에서 전형적으로 활용되며, 효율적인 시간관리와 자원관리를 목적으로 한다. 이러한 기본 개념은 업무담당자를 자원으로 비즈니스 프로세스를 자동화시키는 워크플로우 환경에서도 적용될 수 있다. 이를 위해서는 워크플로우에 적절한 드럼, 버퍼, 로프가 만들어져야 하며, 이들을 이용하여 효율적인 관리 기법이 고안되어야 한다.

3. 워크플로우 특성에 맞춘 제약이론의 적용

제약이론에서 논의하는 제약의 종류에는 크게 3가지로 시장 제약, 물리적 제약, 방침상의 제약이 있다. 시장이 제약이 되는 상황은 생산능력은 충분하나 시장의 수요가 그 보다 낮은 경우이고, 방침상의 제약은 조직의 전략적, 일상적 의사결정을 지배하는 잘못된 평가지표와 평가방법으로 인한 것으로 이 두 가지는 사고 프로세스를 통해 해결하게 된다. 물리적 제약에 대해서는 DBR 스케줄링 기법과 Critical Chain 프로젝트 관리기법을 도구로 사용하여 문제를 해결한다. 본 논문은 제약이론의 도구들이 잘 적용되는 구조화된 생산 프로세스처럼 주문의 발생이 일상적, 반복적이며 예견 가능하며 프로세스가 잘 구조화 될 수 있는 프로덕션 워크플로우(production workflow)나 관리적 워크플로우(administrative workflow)를 대상으로 한다. 이러한 워크플로우에서 운영 성과에 결정적으로 영향을 미치는 업무담당자가 제약이 된다.

3.1 생산 프로세스와 워크플로우 비교

워크플로우에서 제약을 찾고 그것을 최대한 활용하기 위해서 DBR 스케줄링 기법을 참고하여 집중개선 5단계의 절차를 적용한다. 이에 앞서 생산프로세스와 워크플로우 프로세스의 차이점을 먼저 고찰해야 한다. 아래의 <표 1>에 그 차이점을 정리하였다.

<표 1> 생산 프로세스와 워크플로우의 비교

항목	생산 프로세스	워크플로우
로트 크기	하나 이상의 부품	하나의 업무단위
경로	고정적 경로 혹은 대체 경로	조건, 확률적 경로 포함
자원	생산설비, 참여 작업자	업무담당자, Agent
주문발생	예측, 계획 가능	알 수 없음
계획생산	가능	불가능
일정관리	계획적, 체계적 관리	무계획적, 긴급 업무 우선 처리
재고유형	유형의 재공품	무형의 대기 업무
결과물	유형의 재화	무형의 서비스

3.2 워크플로우 운영개선 알고리즘

생산 프로세스와 구별되는 워크플로우의 특성을 감안하여 워크플로우의 효율적 운영을 위해 집중 개선 5단계에 맞추어 DBR 스케줄링을 적용한다. 제약이론에 기반하여 워크플로우 운영개선 알고리즘을 개발하기 위한 이슈는 다음과 같은 것들이 있다.

워크플로우의 제약인 CCP(제약업무담당자)를 찾는 방법, 그러한 제약업무담당자를 찾기 위해 각 업무담당자에 부과되는 업무부하의 계산 방법, 프로세스를 운영성과를 비교하는 측정 기준의 선정, 제약탐색 기간, 워크플로우에 드럼, 버퍼, 로프를 설정하는 방법, 드럼 버퍼 로프가 설정된 환경에서 발생되는 업무들의 업무담당자 할당 방법, 제약업무담당자를 갱신하는 방법 등이 그것이다.

워크플로우 운영개선알고리즘을 정리하면 다음의 5 단계로 정리할 수 있다.

Step I. CCP 확인 - 제약업무담당자 탐색

현재 시각을 T_{now}라하고, CCP를 분석하는 기간 T_{ccp}를 설정한다. 업무담당자의 수를 n이라하고, T_{ccp} 동안에 특정 시점 x에서 업무담당자 i의 워크리스트에 적체된 워크아이템의 시간량을 $W_i(x)$라고하면 업무담당자 i의 시간평균 업무부하 U_i는 다음과 같이 계산된다.

$$U_i = \frac{\int_{T_{now}-T_{CCP}}^{T_{now}} W_i(x)}{T_{CCP}} \qquad (1)$$

식 (1)으로부터 P_{MAX}= MAX{U_1, U_2,..., U_n}를 계산한다.

$$a \times P_{MAX} \geqq \frac{\sum_{i=1}^{n} U_n}{n} \qquad (2)$$

식 (2)에서 a는 $0<a<1$를 만족하는 값으로 CCP 구별 요소이다. 이 값은 사전에 관리자에 의해 설정되며, CCP로 설정되기 위한 업무부하량의 차이를 나타낸다. a가 작을수록 다른 업무담당자보다 시간평균 업무부하가 큰 업무담당자가 CCP로 정해진다. 식 (2)를 만족하는 업무담당자가 발견되면 해당 업무담당자는 CCP로 설정된다.

워크플로우에서 선, 후행 관계에 따라 각 단위업무들이 순차적으로 수행된다. 워크플로우 매니저는 워크리스트 핸들러를 통해 각 단위업무들에 대한 워크 아이템을 해당 업무담당자에 할당한다. 시간평균 업무부하 분석을 통해 제약업무담당자를 찾아내고, 프로세스 인스턴스의 진행이 원활히 일어 날 수 있도록 주문에 의해 발생되는 프로세스 인스턴스의 투입을 통제하게 된다.

Step II. Rope의 설정 - 프로세스 인스턴스 투입의 통제

CCP에 적정한 보호시간버퍼, B_{ccp}를 설정하고, 현재 CCP의 워크리스트 대기업무부하 시간량이 B_{ccp} 이하이면 프로세스 인스턴스가 발생하는 즉시 시작 단위업무에 투입하고, 그렇지 않으면 발생된 프로세스 인스턴스의 시작 단위업무 투입을 막고 투입대기행렬에 쌓아두면서, CCP의 워크아이템 처리종결신호가 발생하면, 투입대기행렬의 대기 프로세스 인스턴스 하나씩 시작 단위업무에 투입한다.

제약 업무담당자를 찾아낸 이후에 이를 중심으로 워크플로우의 운영을 통제하게 된다. 프로세스 인스

턴스의 투입을 제약 업무담당자의 업무처리속도에 동기화 시키면 운영성과에 도움이 되지 않으며 불필요하게 낭비되는 비 제약업무담당자의 능력을 줄여 줄 수 있다. 남은 잉여능력은 기존에 제약 업무담당자에 할당되어 있던 워크아이템의 처리를 위해 사용될 수 있다. 제약 업무담당자의 업무를 비교적 여유능력이 있는 비 제약업무담당자에 분산시킨다. 또한 새로 발생되는 워크아이템에 대해서도 비 제약업무담당자에 주로 할당되도록 제어하여 제약 업무담당자의 업무부하 수준이 일정하게 유지되도록 한다.

Step III. CCP의 워크리스트 대기 업무량 수준 B_{ccp}로 유지 - 워크아이템의 할당 통제

발생된 워크 아이템이 CCP가 포함된 역할 그룹에 할당되었을 때, CCP의 워크리스트 대기 업무량 수준이 Bccp이하이면 해당 워크아이템을 CCP에 할당하고, 그렇지 않으면 역할 그룹의 멤버 중에 대기 업무량의 수준이 가장 낮은 업무담당자에 워크아이템을 할당한다.

Step IV. CCP 및 비 CCP의 dispatching - CCP의 워크아이템 처리순서에 비CCP 종속

CCP의 워크리스트의 워크아이템 처리순서를 결정한 후, 비 CCP의 워크리스트에 CCP와 같은 프로세스 인스턴스 워크아이템이 있으면, 최우선적으로 처리되도록 순서를 조정한다. 해당 워크아이템이 다수일 때는 CCP 워크아이템의 처리순서를 따른다.

워크아이템들은 일정한 우선순위를 가지고 있다. 프로세스 인스턴스를 우선순위에 맞추어 원활하게 소통하기 위해서 제약 업무담당자의 워크아이템 처리순서에 비 제약업무담당자의 워크아이템 처리 순서를 동기화 한다. 이러한 Dispatching을 통해 제약업무담당자를 거쳐 가는 프로세스 인스턴스가 비 제약업무담당자로 인해 납기가 지연되는 것을 방지한다.

Step V. CCP의 갱신 - CCP에 대한 집중적 모니터링

T_{ccp}동안 II, III, IV의 단계를 반복하고, 다시 I 단계로 돌아간다.

<그림 1>은 워크플로우 운영 개선 알고리즘을 순서도로 나타낸 것이다.

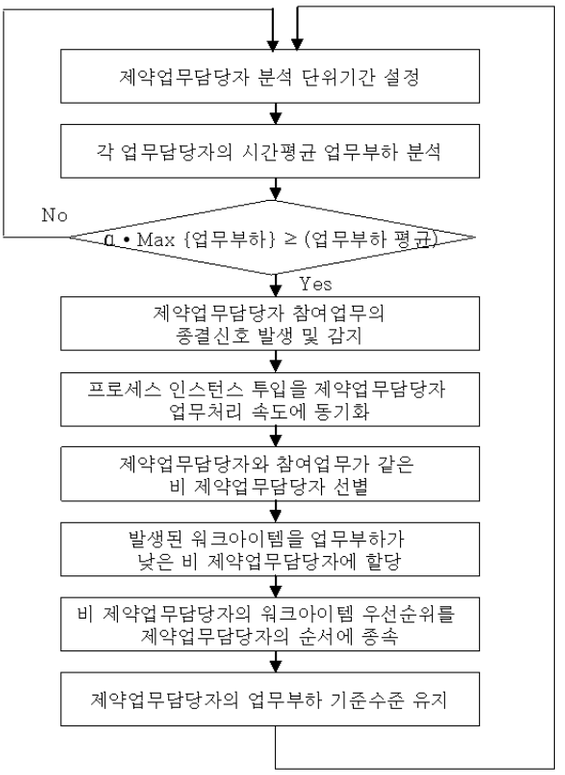
4. 실험 및 분석

워크플로우 운영개선 알고리즘의 효용성을 검증하기 위해서 시뮬레이션 실험을 수행하였다. 먼저 실험에 사용될 워크플로우 운영환경을 설정하고, 프로세스 모델을 정의한다. 시뮬레이션 결과를 프로세스 효율성 측면에서 분석하였다.

4.1 실험 모델

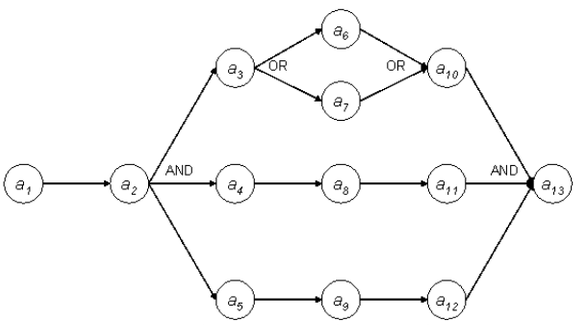
본 논문에서는 다음의 3가지 방식에 대하여 실험을 수행하였다.

- Type 1 : 해당 역할그룹의 담당자에게 균등하게 할당하는 방식
- Type 2 : 워크리스트의 워크아이템의 수가 가장 적은 업무담당자에 할당하는 방식
- Type 3 : 본 논문에서 제안한 알고리즘



<그림 1> 워크플로우 운영개선 알고리즘

본 논문의 Type 3에 대한 실험에서는 Type 2의 실험에서 나타난 로그를 분석하여 제약업무담당자를 선정하였다.



<그림 2> 프로세스 모델

각 프로세스 인스턴스에는 일정한 납기가 정해져 있다. 실험에서 사용되는 프로세스 정의는 <그림 2>와 같다. 1OR블록의 OR분기는 확률적으로 분기되는 상황을 가정한다. 업무담당자는 총 11명이며 이들은 각각 6개의 역할그룹에 할당되어 있다. 업무담당자의 구성과 참여 단위업무가 <표 2>과 <표 3>에 나타나 있다.

<표 2> 업무담당자의 참여 단위업무

Task	Task Performer	Task	Task Performer	Task	Task Performer
a1	Team 1	*a6*	Team2	*a11*	Team1
a2	Team 6	*a7*	Team3	*a12*	Team6
a3	Team 4	*a8*	Team3	*a13*	Team2
a4	Team 5	*a9*	Team4		
a5	Team5	*a10*	Team4		

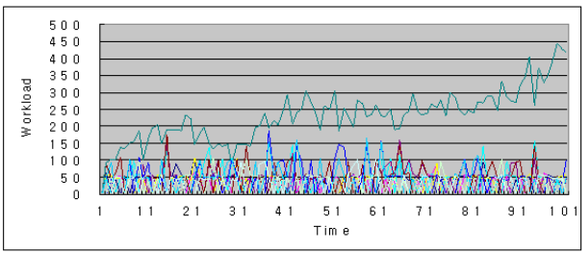
<표 3> 역할그룹의 업무담당자 구성

Team	Task performer	Team	Task performer
1	*1, 2, 3, 7*	*4*	*4, 9, 10, 11*
2	*4, 7, 9*	*5*	*5, 7*
3	*5, 6, 7, 8*	*6*	*7*

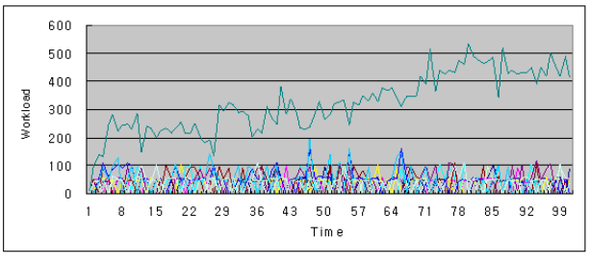
업무담당자의 참여 단위업무의 처리시간을 모두 N(50, 25)로 하였으며, 프로세스 인스턴스의 발생간격은 Expo(105)를 따르도록 하였다. 100,000분의 기간에 대하여 실험하였다. 이러한 실험을 30회 반복수행하여 그 평균을 사용하여 결과를 분석한다. ARENA 6.0을 사용하였다.

4.2 실험결과 및 분석

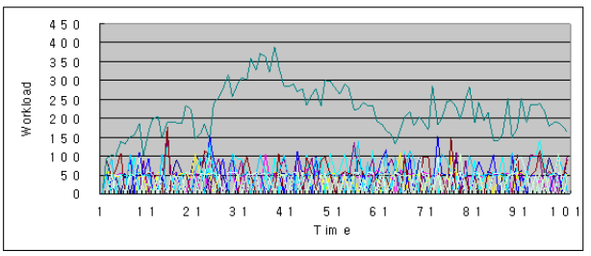
<그림 3>, <그림 4>, <그림 5>는 3가지 Type의 업무부하의 변화를 보여주고 있다. 이 때, Type 3의 경우, 7번 담당자가 제약업무담당자로 선정되었다.



<그림 3> Type 1의 업무부하 분석



<그림 4> Type 2의 업무부하 분석



<그림 5> Type 3의 업무부하 분석

<그림 5>는 운영개선 알고리즘이 적용된 모델의 업무담당자 별 업무부하의 변화를 보여준다. <그림 3>, <그림 4>에 비해 제약업무담당자 업무부하의 수준이 현저히 떨어져서 일정한 수준을 유지하는 것을 보여 준다. 이는 프로세스 인스턴스 투입을 제약업무담당자의 업무처리속도에 동기화 시키며, 제약업무담당자의 업무부하를 일정한 수준으로 유지하며 비 제약업무담당자에 업무부하를 분산시킨 결과이다.

프로세스의 효율성은 워크플로우 관리 시스템이 얼마나 시간을 효율적으로 활용하여 프로세스 인스턴스들을 처리하였는가에 대한 측정 양이다. 이의 측정 기준으로는 일정기간에 대해 완료된 프로세스 주문수, 납기를 어긴 프로세스 주문수, 평균적인 프로세스 리드타임 등을 사용하였다. <표 4>는 이러한 프로세스 효율성을 비교한 것이다.

<표 4> 프로세스 효율성 비교

워크플로우 운영방식	프로세스주문 평균완료 개수	프로세스주문 평균완료 시간	납기준수 주문수
Type 1	755.37	1166.07	299.07
Type 2	761.97	1333.65	257.50
Type 3	760.47	1090.26	375.30

각 워크플로우 운영방식에 대해 평균완료 개수에는 거의 차이가 없으나, 납기준수의 개선효과는 25 ~ 46%, 프로세스 인스턴스의 리드타임 개선효과는 7 ~ 22%이다.

5. 결론

본 논문은 제약이론이 처리결과를 증대시키는데 가장 약한 부분이 어디인지를 찾아내고, 그 약한 부분을 보호함과 동시에 능력을 최대한 활용한다는 사실에 착안하여, 제약이 존재하는 워크플로우에 적용하였다.

본 논문에서 개발한 알고리즘을 워크플로우 관리 시스템에 적용함으로써 자동화된 프로세스가 자원의 활용상황을 항상 감시하여, 프로세스의 시작을 적절히 통제하고 제약업무담당자를 최대한 활용하도록 발생되는 업무부하를 분산시킴으로써 워크플로우의 지속적인 운영개선이 달성되어 전체 워크플로우를 정확한 수행을 기반으로 효율적으로 운영할 수 있게 되었다.

참고문헌

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

[2] Eder, Johann., Panagos, Euthimios., Managing Time in Workflow Systems, WfMC, Workflow Handbook 2001, pp. 109-132. 2001.

[3] Goldratt, Eliyahu M., The Haystack Syndrome: Sifting Information Out of the Data Ocean January, 1991

[4] Hollingsworth, D., Workflow management coalition specification: the workflow reference model, WfMC specification, 1994.

[5] Smith, Stephen F., Hildrum, David W., and Becker, Marcel A., Workflow Management from a Scheduling Perspective, Technical Report WS-99-02, The AAAI Press, pp. 68-72, 1999.