워크플로우 마이닝을 위한 워크플로우 최적 축소 모델☆

Minimal Workflow Model for Workflow Mining

박 민 재* 원 재 강** 김 창 민*** 김 광 훈****
Min Jae Park Jae Kang Won Chang Min Kim Kwang Hoon Kim

요 약

본 논문에서는 워크플로우 프로세스 재발견 문제를 해결하기 위한 적절한 해결책으로 워크플로우 최적 축소 모델을 제안한다. 워크플로우 최적 축소 모델은 워크플로우 최적 축소 넷으로 표현할 수 있다. ICN(Information Control Net) 모델링 기법으로 표현되는 프로세스 모델은 ICN을 구성하고 있는 액티비티 사이의 액티비티 의존성에 따라 적절한 알고리즘의 적용으로 액티비티 의존 넷을 구성할 수 있다. 워크플로우 최적 넷 또한 액티비티 의존 넷의 몇 가지 속성에 대한 알고리즘 적용으로 찾아낼 수 있으며, 찾아낸 워크플로우 최적 넷을 프로세스 재발견 문제를 해결하기 위한 방안으로 제안하며, 프로세스 개선에도 의미를 둘 수 있다.

Abstract

This paper proposes a minimal workflow model as a feasible solution to the workflow process rediscovery problem. The minimal workflow model can be represented by the minimal workflow net. The process model is represented by ICN(Information Control Net) Modeling method. ICN can configure activity dependent net applying proper algorithm according to activity dependency among activities which configure the ICN. The proposed model is possible to develop with the application of minimal workflow net and with the application of the algorithm related to activity dependent net properties. Hence, it can solve the process rediscovery problem and can also be helpful on process improvement.

☞ Keyword : Workflow/Process Mining, Process Rediscovery, ICN(Information Control Nets)

1. 서 론

정보기술 분야에 있어서 최근의 가장 두드러진 변화는 기존의 데이터 중심 정보기술에서 프로세스 중심 정보기술로 빠르게 전이되고 있다는 사실이다. 1960년대의 파일시스템을 기반으로 한 정보기술의 발전은 1980년대 관계 형 데이터베이스 관리 시스템의 개발과 더불어 더욱 발전되었고, 오늘날의 거의 모든 정보기술을 데이터베이스기술을 기반으로 한다고 해도 과언이 아닐 정도이다. 하지만 데이터베이스기술은 정보화의 핵심을 해당 도메인의 데이터와 그 데이터를 중심으로 한 업무처리 프로그램 중심의 생산성 향상에 초점을 두고 있다. 그러나 조직내의 업무처리의 생산성을 분석한 결과 업무처리의 전체 시간 중에 단지 10%만이 업무자체에 소요되고 나머지 90%의 시간은 업무간의 전이 또는 전달시간에 소요된다는 것을 알게 되면서, 업무처리 프로세스에 대한 생산성 향상 문제로 정보기술의 초점이 바뀌게 된다. 이러한 사실이 곧 비즈니스 프로세스 리엔지니어링과 자동화를 통한 업무생산성 향상에 초점을 두게 되었고, 최근 2000년대에는 프로세스 또는 워크플로우 중심의 정보기술(BPM/WfM: Business Process/Work-

* 준 회 원 : 경기대학교 대학원 전자계산학과 석사과정
mean222@kyonggi.ac.kr(제1저자)
** 준 회 원 : 경기대학교 대학원 전자계산학과 박사과정
jkwon@kyonggi.ac.kr
*** 종신회원 : 성결대학교 컴퓨터공학부 교수
kimcm@sungkyul.ac.kr
**** 종신회원 : 경기대학교 정보과학부 조교수
kwang@kyonggi.ac.kr
[2005/02/15 투고 - 2005/03/17 심사 - 2005/10/18 심사완료]
☆ 본 연구는 정보통신연구진흥원 정보통신 기초기술연구지원사업(04-기초-0005)의 지원으로 수행되었음.

flow Management)이 핵심으로 등장하고 있다. 이러한 정보기술 컴퓨팅환경의 변화는 웨어하우스 및 마이닝 기술에도 영향을 미치고 있다. 즉, 데이터를 중심으로 하는 마이닝 기술에서 프로세스를 중심으로 하는 마이닝 기술로의 천이와 함께 이에 대한 연구 및 개발의 필요성이 요구되고 있다. 기존의 데이터 마이닝 기술이 새로운 서비스 및 지식 발견(Knowledge Discovery)에 초점을 맞추고 있다면, 워크플로우 마이닝 기술은 프로세스 발견(Process Discovery)이나 프로세스 리엔지니어링 및 개선에 초점을 두고 있다. 근래에는 워크플로우 및 프로세스 마이닝 기법에 관한 연구가 대두 되면서, 이벤트 기반 데이터를 이용한 워크플로우 및 프로세스 마이닝 기법, 예외처리 마이닝을 이용한 워크플로우 예외처리 기법과 실행정보를 이용한 프로세스 마이닝 기법 등이 연구되고 있다. 본 논문에서는 프로세스 재발견[5] 문제를 풀어내기 위하여, 워크플로우 모델정보의 액티비티 의존성에 따른 액티비티 의존 넷과 액티비티 의존 넷의 몇 가지 특성을 고려한 워크플로우 최적 축소 넷 구성 알고리즘을 적용하여 워크플로우 최적 축소 넷을 도출하여 워크플로우 최적 축소 모델을 도출한다. 워크플로우 최적 축소 모델을 프로세스 재발견 문제의 해결방안으로 제안하며, 더 나아가 워크플로우 마이닝을 하기 위한 하나의 중요한 기법이 될 것이다.

2. 관련기술연구

수많은 기업들이 워크플로우 관리시스템을 도입하고, 그에 따른 효과적인 이윤을 창출해 내고 있는 실정이다. 하지만, 워크플로우 도입만으로는 기업의 효율을 극대화 할 수 없다. 실제 기업내의 비즈니스 프로세스는 프로세스 정의 시점에서 정의한 프로세스 정의와 실행시점에서 발생하는 프로세스 인스턴스와는 많은 다른 부분이 있음을 볼 수 있으며, 이는 실제 일어나는 업무에 대한 파악이 잘못되어 프로세스의 정의 및 생성이 비 효율적으로 이루어짐을 볼 수 있다. 이러한 불일치 또는 기업 업무의 효율성의 극대화를 위하여 기업 프로세스를 기반으로 프로세스 내에서 발생하는 정보에 대한 새로운 정보의 창출, 즉 워크플로우 마이닝 기술이 도입되어야 한다. 현재 프로세스 및 워크플로우 마이닝 기술은 몇 가지 기술을 중심으로 연구되고 있다.

2.1 Event Based Data를 이용한 워크플로우 및 프로세스 마이닝 기법

Event-Based Data를 이용한 워크플로우 및 프로세스 마이닝 기법은 워크플로우의 실행정보를 이용하여 최종적으로 비즈니스 프로세스를 정확하게 표현하기 위해 페트리 넷(Petri Net) 모델링 방법을 이용하고 있다. 이렇게 표현함으로써 AND 병합과 분기, OR 병합과 분기, 그리고 반복에 대해서도 정확하게 표현할 수 있다. Event-Based Data를 이용한 프로세스 마이닝 기법은 AND 또는 OR 분기와 병합에 대하여 표현할 수 있는 장점이 있지만 현재의 연구결과는 제한된 수의 태스크를 이용하여 실험한 결과이기 때문에 완벽한 비즈니스 프로세스를 표현 할 수 있는지의 여부에 대하여 좀 더 연구가 필요하고, 마이닝 과정을 통하여 얼마나 정확하고 빠르게 비즈니스 프로세스를 표현할 수 있는 가와 같은 질의 향상에 좀 더 많은 연구가 필요하다고 판단된다.

2.2 예외처리 마이닝을 이용한 워크플로우 예외처리 기법

현재 워크플로우 시스템의 적용환경이 처리해야 할 비즈니스 프로세스의 대규모화, 복잡성, 매우 긴 수행시간, 사용자 요구의 증가 그리고 관련 데이터의 증가 등과 같이 변화함에 따라 워크플로우 시스템에서 예외처리에 대한 연구는 매우 중요한 분야로 인식되고 있다. 이와 같은

워크플로우 시스템에서 발생되는 예외상황 또는 오류를 처리하기 위하여 많은 방법들이 제공되고, 연구되어 지고 있다. 그러나 대부분의 방법은 빌드-타임에서 예측되는 오류(expected exception)에 대해서만 처리할 수 있는 방법을 제공하고 있다. 즉, 워크플로우 시스템이 실행 중에 발생될 수 있는 예측치 못한 오류 또는 예외상황(unexpected exception)에 대한 처리는 워크플로우 이외의 부가적인 시스템으로 처리하는 것이 대부분이다. 예외처리 마이닝을 이용한 워크플로우 예외처리 기법에서는 빌드-타임에서 정의될 수 있는 예외상황은 물론 런-타임에서 발생될 수 있는 예외상황, 예를 들어 이미 수행이 완료된 액티비티를 롤백(rollback)을 수행 여부를 결정한다든지, 워크플로우 시스템의 동적 변경에 가능여부의 결정하는 것과 같은 작업 그리고, 런-타임에 발생된 예외상황에 대한 해결책을 제시하기 위한 방법으로 이전에 발생된 예외상황과 처리결과와 현재 발생된 예외상황을 비교하여 최적의 해결책을 찾는 다음과 같은 3가지 알고리즘을 제안하고 있다.

- SEQ-E(SEQuEntial matching algorithm)
- SEQ-C(SEQuential matching algorithm on Concept level permutation)
- BIN-C(BINary search on the sequence of Concept level permutation)

2.3 실행정보(Execution Log)를 이용한 프로세스 마이닝

워크플로우 시스템의 실행은 프로세스의 정해진 진행순서를 바탕으로 한다. 즉, 업무 처리를 위한 프로세스에 정의된 액티비티들의 정해진 순서의 실행을 의미한다. 이와 같이 정해진 순서에 의해 실행되는 액티비티 사이에는 종속성(dependency)이 존재하게 되며, 이러한 액티비티의 종속성 관계를 방향 그래프(Directed Graph)로 표시하고 있다. 또한 액티비티 간의 전이(transition)을 나타내기 위하여 제어흐름의 부울린 값(Boolean Value)에 의해서 결정된다. 실행정보를 이용한 마이닝 기법에서는 워크플로우가 실행정보(Log)를 이용하여 새로운 프로세스 모델 즉, 액티비티의 종속성을 표현하는 그래프 모델을 제시하기 위한 알고리즘을 제시했으며, 액티비티 사이의 종속성을 나타내어 그 관계를 표현하는 그래프로 나타내는 것이 이 과제의 최종의 목표이다. 제시된 알고리즘에 따라 작성된 그래프 모델은 비즈니스 프로세스의 제어 흐름(Control Flow)를 의미한다.

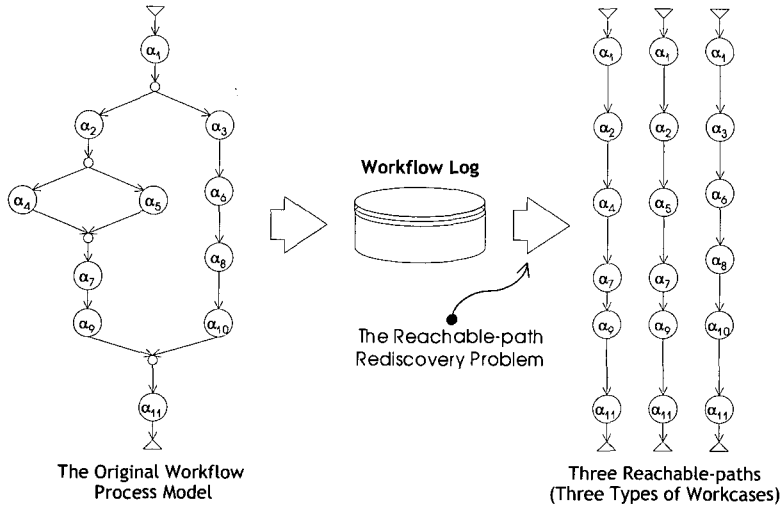
3. 실행경로 재발견 문제 및 해결방안

비즈니스 프로세스 정의 시점에서 정의한 비즈니스 프로세스의 정의 정보는 실제 프로세스를 실행시키는 프로세스 실행시점에서의 비즈니스 프로세스의 정보와 일치하지 않는다. 특히, 비즈니스 프로세스의 분기가 일어났을 경우 프로세스의 실행경로는 어느 한쪽 측면 또는 몇몇 측면에서만 집중적으로 일어날 수 있으며, 일어나지 않을 가능성도 가지고 있다. 다음 그림 1에서 보여지는 것과 같이 비즈니스 프로세스는 정의 시점의 정보와 실제 프로세스 인스턴스를 발생시킨 후에 나타난 프로세스 실행경로와 다른 양상을 가진다. 그리고, 워크플로우 실행 정보(Workflow Log)를 통해 얻을 수 있는 정보를 가공하여 프로세스를 다시 한번 재발견 할 수 있다.

3.1 ICN(Information Control Net)

본 논문에서는 워크플로우를 모델링하기 위한 기본적인 방법으로 표현 방법이 비교적 간단하고, 정형명세가 가증한 ICN 모델링 방법을 채택하였다.

ICN은 사무실(Office)의 개념을 일련의 관련된 프로시져(Procedures)의 집합으로 정의하며 이러한 프로시져는 선후관계가 존재하는 액티비



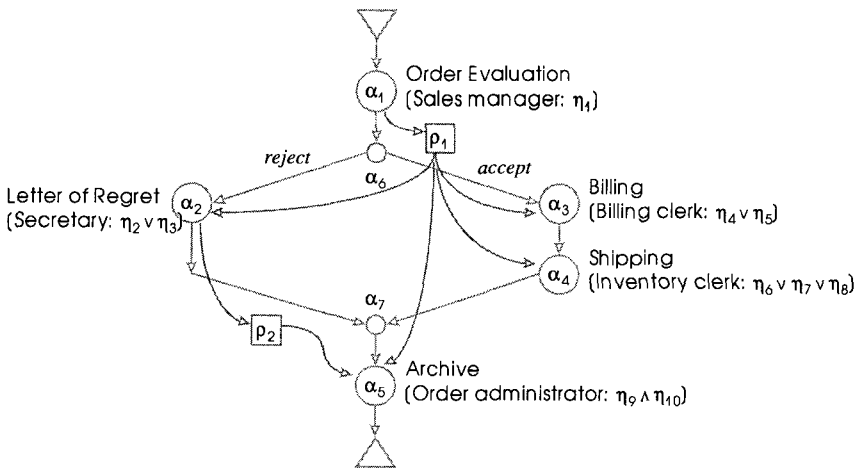
〈그림 1〉 실행경로 재발견 문제

티들의 집합으로 표현된다. ICN은 그림형태로 프로시져, 액티비티, 저장소(Repositories), 선후관계를 나타내는 제어 흐름(Control Flow)과 데이터 흐름(Data Flow)을 표현한다. 이러한 방법으로 형성된 ICN은 각 단계를 정형화한 방법으로 구성 및 분석 할 수 있다.

3.1.1 기본(Basic) ICN 그래프 형태의 구성 및 특성

ICN 제어흐름 그래프는 큰 원으로 표현되는 일련의 액티비티와 작고 빈 원으로 표현되는 OR 노드, 작고 채워진 원으로 표현되는 AND노드, 그리고 이러한 노드들을 연결하는 선(edge)로 구성된다. 화살표(Arc)는 실선(Solid)과 점선(Dashed)으로 표현되는데 이들은 노드들 사이의 선후관계 및 자료저장소와의 입출력을 표현한다.

여기서 전체 프로시져의 이름은 '주문처리[8]'이며, 각 과정을 담당하는 액티비티는 주문평가(Order Evaluation), 거절(Letter of Regret), 결



〈그림 2〉 주문처리 관계를 나타내는 워크플로우

재(Billing), 발송(Shipping), 문서집적(Archive) (α_1, α_2, α_3, α_4, α_5)이다. 그리고 α_6, α_7은 OR분기 및 결합을 나타내는 노드이며, ρ_1, ρ_2는 데이터의 저장소를 나타낸다. 또한 직선형태로 된 실선은 제어의 흐름을 나타내고 곡선형태의 실선은 정보의 흐름을 나타낸다.

3.1.2 ICN의 정형명세

A를 액티비티의 집합, R을 저장소의 집합, P를 역할, C를 수행자, 그리고 T를 제어흐름의 상태라고 하면 이전 장에서 표현한 그림 2의 주문 프로세스 모델의 다음과 같은 정형명세 방법에 따라 표 1과 같이 표현할 수 있다.

- $\delta = \delta_i \cup \delta_o$ 이때 $\delta_O : A \rightarrow \wp(A)$은 하나의 액티비티를 후행하는 액티비티들의 집합에 연결하는 관계를 나타내며 $\delta_i : A \rightarrow \wp(A)$은 하나의 액티비티를 선행하는 액티비들의 집합에 연결하는 관계를 나타내는 관계이다.
- $\gamma = \gamma_i \cup \gamma_o$ 이때 $\gamma_o : R \rightarrow \wp(A)$은 하나의 액티비티를 후속하는 액티비티 집합들을 출력저장소들의 집합과 연결하는 것 중 하나이며, $\gamma_i : R \rightarrow \wp(A)$은 하나의 액티비티를 선행하는 액티비티 집합들을 입력 자료저장소들의 집합과 연결하는 관계를 나타내는 것 중 하나이다.
- $\varepsilon = \varepsilon_a \cup \varepsilon_p$ 이때 $\varepsilon_a : P \rightarrow \wp(A)$는 하나의 액티비티를 조합된 역할들의 집합에 액티비티를 단일 값으로 연결하는 것이고, $\varepsilon_p : A \rightarrow \wp(P)$는 조합된 액티비티들의 집합을 역할에 단일 값으로 연결하는 것이다.
- $\pi = \pi_p \cup \pi_c$ 이때 π_p : C $\rightarrow \wp$(P) 조합된 수행자들의 집합을 역할에 단일 값으로 연결하는 것이고, π_c : P $\rightarrow \wp$(C)는 조합된 역할들을 수행자에 단일 값으로 연결하는 것이다.
- $\kappa = \kappa_i \cup \kappa_o$ 이때 κ_i는 $\alpha \in$A일 때, ($\delta_i(\alpha)$, α) 사이에서, 제어흐름(T) 조건들의 집합이고, κ_o는 $\alpha \in$ A일 때, (α , $\delta_o(\alpha)$) 사이에서, 제어흐름(T) 조건들의 집합이다. 이때 집합 T = {***default, or***(*conditions*), ***and***(*conditions*)}.
- I는 초기 입력되는 자료 저장소들의 유한 집합이며, ICN의 실행 전에 어떠한 외부의 프로세스에 의해서 로드 되어야 한다.
- O는 마지막으로 출력되는 자료저장소들의 유한 집합이며 ICN의 실행 후에 어떠한 외부의 프로세스에 의해서 이용되는 정보를

〈표 1〉 주문 처리 관계를 나타내는 워크플로우의 정형 명세

Γ = (δ, γ, ε, π, κ, *I*, *O*) over *A*, *R*, P, *C*, *T* // Order Processing in ICN
A = { α_1, α_2, α_3, α_4, α_5, α_6, α_{7_1}, α_{7_2} } // Activities
R = { ρ_1, ρ_2 } // Repositories
P = { ω_1, ω_2, ω_3, ω_4, ω_5 } // Roles
C = { η_1, η_2, η_3, η_4, η_5, η_6, η_7, η_8, η_9, η_{10} } // Actors
T = { *d*(*default*), *or*$_1$('*Reject*'), *or*$_2$('*Accept*') } // Control-transition conditions
I = { $\varnothing$ }; *O* = { $\varnothing$ }				

$\delta_i(\alpha_1)=\{\alpha_7\}$, $\delta_o(\alpha_1)=\{\alpha_6\}$; $\delta_i(\alpha_2)=\{\alpha_6\}$, $\delta_o(\alpha_2)=\{\alpha_7\}$; $\delta_i(\alpha_3)=\{\alpha_6\}$, $\delta_o(\alpha_3)=\{\alpha_4\}$; $\delta_i(\alpha_4)=\{\alpha_3\}$, $\delta_o(\alpha_4)=\{\alpha_7\}$; $\delta_i(\alpha_5)=\{\alpha_7\}$, $\delta_o(\alpha_5)=\{\alpha_{7_1}\}$; $\delta_i(\alpha_6)=\{\alpha_1\}$, $\delta_o(\alpha_6)=\{\alpha_2$, $\alpha_3\}$; $\delta_i(\alpha_7)=\{\alpha_2$, $\alpha_4\}$, $\delta_o(\alpha_7)=\{(\alpha_6)\}$;	$\gamma_i(\alpha_1)=\{\varnothing\}$, $\gamma_o(\alpha_1)=\{\rho_1\}$; $\gamma_i(\alpha_2)=\{\rho_1\}$, $\gamma_o(\alpha_2)=\{\rho_2\}$; $\gamma_i(\alpha_3)=\{\rho_1\}$, $\gamma_o(\alpha_3)=\{\varnothing\}$; $\gamma_i(\alpha_4)=\{\rho_1\}$, $\gamma_o(\alpha_4)=\{\varnothing\}$; $\gamma_i(\alpha_5)=\{\rho_1, \rho_2\}$, $\gamma_o(\alpha_5)=\{\varnothing\}$; $\gamma_i(\alpha_6)=\{\varnothing\}$, $\gamma_o(\alpha_6)=\{\varnothing\}$; $\gamma_i(\alpha_7)=\{\varnothing\}$, $\gamma_o(\alpha_7)=\{\varnothing\}$; $\gamma_i(\alpha_7)=\{\varnothing\}$, $\gamma_o(\alpha_7)=\{\varnothing\}$; $\gamma_i(\alpha_7)=\{\varnothing\}$, $\gamma_o(\alpha_7)=\{\varnothing\}$;	$\varepsilon_a(\alpha_1)=\{\omega_1\}$; $\varepsilon_a(\alpha_2)=\{\omega_2\}$; $\varepsilon_a(\alpha_3)=\{\omega_3\}$; $\varepsilon_a(\alpha_4)=\{\omega_4\}$; $\varepsilon_a(\alpha_5)=\{\omega_5\}$; $\varepsilon_a(\alpha_6)=\{\varnothing\}$; $\varepsilon_a(\alpha_7)=\{\varnothing\}$; $\varepsilon_a(\alpha_{7_2})=\{\varnothing\}$;	$\pi_c(\omega_1)=\{\eta_1\}$; $\pi_c(\omega_2)=\{\eta_8, \eta_9\}$; $\pi_c(\omega_3)=\{\eta_4, \eta_5\}$; $\pi_c(\omega_4)=\{\eta_6, \eta_7, \eta_8\}$; $\pi_c(\omega_5)=\{\eta_8, \eta_9\}$;	$\kappa_i(\alpha_1)=\{d\}$, $\kappa_o(\alpha_1)=\{d\}$; $\kappa_i(\alpha_2)=\{or_1\}$, $\kappa_o(\alpha_2)=\{d\}$; $\kappa_i(\alpha_3)=\{or_2\}$, $\kappa_o(\alpha_3)=\{d\}$; $\kappa_i(\alpha_4)=\{d\}$, $\kappa_o(\alpha_4)=\{d\}$; $\kappa_i(\alpha_5)=\{d\}$, $\kappa_o(\alpha_5)=\{d\}$; $\kappa_i(\alpha_6)=\{d\}$, $\kappa_o(\alpha_6)=\{d\}$; $\kappa_i(\alpha_7)=\{d\}$, $\kappa_o(\alpha_7)=\{d\}$; $\kappa_i(\alpha_7)=\{\varnothing\}$, $\kappa_o(\alpha_7)=\{d\}$; $\kappa_i(\alpha_{7_2})=\{d\}$, $\kappa_o(\alpha_{7_2})=\{\varnothing\}$;

포함하고 있다고 가정한다.

3.2 워크플로우 마이닝 프레임 워크

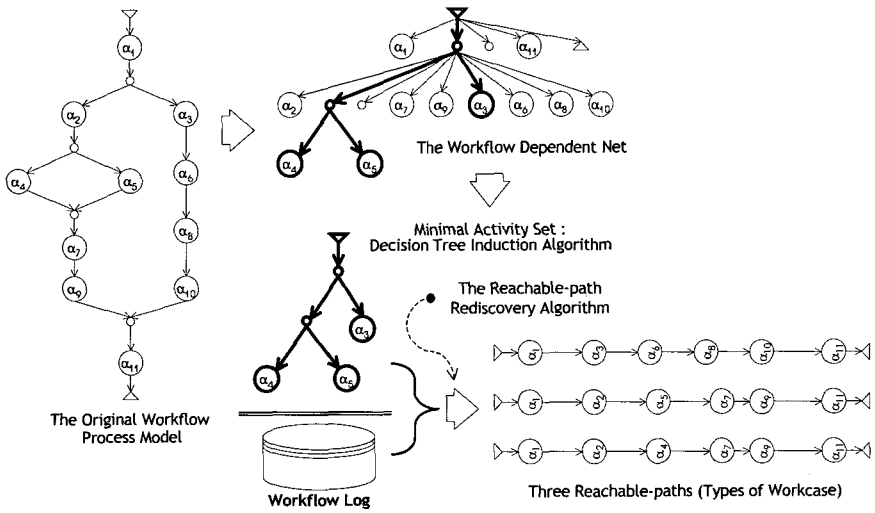
효율적인 비즈니스 프로세스를 정의하기 위해서는 이미 정의된 비즈니스 프로세스 정보의 의존성 분석 및 액티비티 최소 집합의 구성과 함께, 워크플로우가 실행되어 남겨진 워크플로우 로그[6]의 정보를 통해 도달가능경로를 구성할 수 있다. 구성된 도달가능경로와 워크플로우 로그를 통한 프로세스 인스턴스의 빈도수를 파악하고 분석하여 프로세스 발견(Process Discovery)이나 프로세스 리엔지니어링 및 개선하여 효율적인 비즈니스 프로세스를 만들어 낼 수 있다. 다음 그림 3는 효율적인 비즈니스 프로세스를 구성하기 위하여 프로세스를 재발견하기 위한 워크플로우 마이닝 프레임 워크를 나타낸 것이다.

비즈니스 프로세스 정의[9] 정보는 액티비티의 의존성에 따라 결정 트리를 만들어 워크플로우 의존 넷(Workflow Dependent Net)을 구성하고, 실제 마이닝에 필요한 최소 액티비티들의 집합을 최적 축소 워크플로우 넷(Minimal Workflow Net) 구성 알고리즘에 따라 결정트리를 구성하여, 워크플로우 축소 모델을 도출 한다. 만들어진 워크플로우 최적 축소 모델과 워크플로우 로그 정보에 따라, 워크플로우의 구체적인 실행경로와 빈도수를 얻어 낼 수 있다.

4. 워크플로우 최적 축소 모델(Minimal Workflow Net)

워크플로우 최적 축소 모델이란 워크플로우 마이닝을 위하여, 프로세스내 액티비티 사이의 의존관계를 통해 워크플로우 의존 넷을 구성하고, 비즈니스 프로세스의 재발견에 불필요한 요소를 제거하여 워크플로우 최적 축소 넷을 구성한 모델을 말한다.

본 장에서는 워크플로우 축소 모델을 구성하기 위한 단계로서, ICN을 통해 모델링 된 워크플로우 프로세스 모델을 워크플로우 의존 넷으로 구성하는 알고리즘 및 정형명세를 나타낸다. 또한, 워크플로우 의존 넷 구성 알고리즘으로 구성된 워크플로우 의존 넷을 통해 워크플로우 최적 축소 넷을 구성하는 알고리즘 및 정형명세 나타내어, 워크플로우 축소 모델을 도출한다.



〈그림 3〉 워크플로우 마이닝 프레임워크

4.1 워크플로우 의존 넷(Workflow Dependent Net)

ICN으로 정의된 정의 된 비즈니스 프로세스를 각 액티비티 사이의 의존성을 분석하여, 워크플로우 의존 넷으로 구성한다. 액티비티 사이의 의존성 분석을 통하여, 정적으로 정의된 중요 경로와 워크플로우 엔진에서 실행된 워크플로우 인스턴스의 실행정보를 이용하여 정의 시점에 정의된 경로와 비교함으로써 도달 가능한 모든 경로 중에서 비즈니스 모델이 어떤 경로로 진행되는지를 살펴봄으로써 정의된 모델의 개선 여부와 새로운 모델의 작성 시 이용할 수 있는 정보 알 수 있다.

워크플로우 의존 넷을 구성하기 위해 ICN을 이루고 있는 액티비티간의 의존성에 관련한 몇 가지 정의가 있다. 정의는 다음과 같다.

[정의 1] ICN에서 일은 W. $\Gamma = (\delta, \gamma, \varepsilon, \pi, \kappa, I, O)$로 표기하는 ICN에서 일은 W라고 하며, n>0 이고 i=1, 2, ···, n-1 동안 $\alpha_{i+1} \in \delta_o(\alpha_{i+1})$ 일 때, $\alpha_1, \alpha_2, \cdots, \alpha_n$와 같은 일련의 액티비티들의 집합이다. 일 $W = \alpha_1 \alpha_2 \cdots \alpha_n$ 의 길이는 |W|로 표기하며 W에서 발생한 액티비티 발생 개수다.

[정의 2] ICN에서의 *지배하는 성질*. ICN에서, Γ은 각각의 다음 조건을 만족한다: (*a*) Γ은 두가지의 구별되는 액티비티들은 포함한다: $\delta_i(\alpha_I)$에 대한 시작 α_I,액티비티의 집합은 ∅이며, $\delta_o(\alpha_F)$에 대한 끝 액티비티 α_F의 집합 ∅이다. (*b*)Γ의 모든 액티비티는 $\alpha_I - \alpha_F$사이에서 이루어진다.

- Γ는 임의의 ICN이라 하자. 모든 단계인 Γ가 포함하는 u에서 일어나는 모든 단계가 $v - v_F$라 고했을 때, 임의의 액티비티 $v \in A$ 이면, 임의의 액티비티 $u \in A$ *앞으로 지배하는 성질(forward dominates)을 가진다*. 만약 $u \neq v$이고, u가 v를 앞으로 지배할 때, u는 *적절히 앞으로 지배하는 성질(properly forward dominates)을 가진다.*
- Γ는 임의의 ICN이라 하자. 만약 u가 v를 앞으로 지배하는 성질을 가지며, v와 함께 시작하는 Γ에서 정수 $k \geq 1$ 모든 단계가 길이가 k 보다 큰 u 를 포함한다면, 하나의 액티비티 $u \in A$ 는 임의의 액티비티 $v \in A$를 *강하게 앞으로 지배하는 성질(strongly forward dominates)을 가진다,*
- Γ는 임의의 ICN이라 하자. $\alpha \in (A - \{\alpha_F\})$에서 *바로 다음 것을 지배하는 것(immediate forward dominator)은*, $ifd(\alpha)$로 표기한다.
- Γ는 임의의 ICN이라 하자. 만약 모든 단계 Γ, $v - v_I$ 에서 u를 포함한다면, 임의의 액티비티 $u \in A$는 임의의 액티비티 $v \in A$를 *뒤로 지배하는 성질(backward dominates)을 가진다*; 만약 $u \neq v$이며, u가 v를 뒤로 지배하는 성질을 가질 때, u는 v를 *적절히 뒤로 지배하는 성질(properly backward dominates)을 가진다,*
- Γ이 임의의 ICN이라 하자. 액티비티 $\alpha \in (A - \{\alpha_I\})$에서 *바로 전 것을 지배하는 것(immediate backward dominator)*은 $ibd(\alpha)$라고 표기한다.

[정의 3] ICN에서의 *워크플로우 의존 넷*. 워크플로우 의존 넷은 $\Omega = (\varphi, \xi, \boldsymbol{S}, \boldsymbol{E})$에 따라 정형적으로 정의한다. 이때, $\boldsymbol{A}$ 는 액티비티들의 집합이며, $\boldsymbol{T}$ 는 흐름제어 조건들이다.

- $\varphi = \varphi_i \cup \varphi_o$
  이때, φ_o: $A \rightarrow \wp(A)$은 하나의 액티비티를 그것과 제어 의존적으로 후행하는 액티비티들의 집합에 연결하는 연결관계를 나타내며, φ_i: $A \rightarrow \wp(A)$은 하나의 액티비티를 그것과 제어 의존적으로 선행하는 액티비티들의 집합의 연결관계를 나타낸 것이다.
- $\xi = \xi_i \cup \xi_o$
  $\alpha \in A$인 범위에서, $\tau \in T$ 인 ξ_i는 각각의 실선 $(\varphi_i(\alpha), \alpha)$의 제어흐름 조건들의 집합이며, $\tau \in T$인 ξ_o은 각각의 실선$(\alpha, \varphi_o(\alpha))$의 제어흐름 조건들의 집합이다,

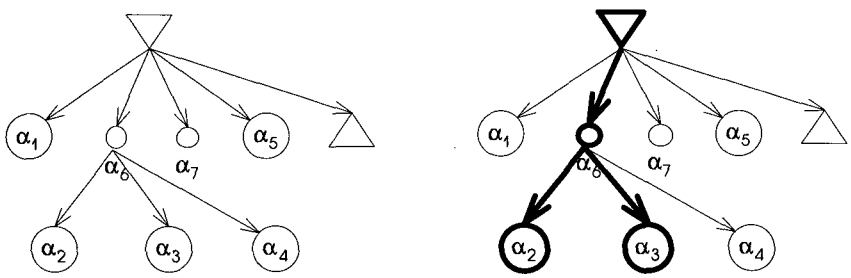
<표 2> 워크플로우 의존 넷 구성 알고리즘

```
Input An ICN, Γ = (δ, γ, ε, π, κ, I, O);
Output A Workflow Dependent Net, Ω = (φ, ξ, I, O);
Initialize T ← {α_I}; /*u, T are global*/
PROCEDURE(In s ← δ_0 (α_I), In  f← {α_F}) /*Recursive procedure*/
BEGINE
v ← s; φ_i(v)←δ_i(v); φ_o(δ_i(v)) ← v; T←T∪{v};
O ← δ_o(v);
WHILE (∃u ∈ O is not equal to f) DO
  FOR (∀u ∈ O and u ∉ T) DO
      IF (u is a strongly forward-dominator of v?)
           Then do
              IF (u is a multiply forward-dominator of v?);
                  Then do
               Call PROCEDURE (In s ← u, In f ← {'and-join'});
               φ_o(δ_i(v)) ← u; φ_i(u) ← δ_i(v); T←{u}; end
           Else do
                   φ_o(δ_i(v)) ← u; φ_i(u) ← v; T←{u}; end
             END IF
           Else do
            Call PROCEDURE (In s ← u, In f ← {'or-join});
             φ_o(δ_i(v)) ← u; φ_i(u) ← δ_i(v); T←{u}; end
      END IF
    END FOR
   Replace O To δ_o(u); /* O ← δ_o(u);*/
EDN WHILE
END PROCEDURE
```

- ***S*** 는 워크플로우 의존 넷을 구성하고 있는 액티비티 집합의 시작을 의미한다.
- ***E*** 는 워크플로우 의존 넷을 구성하고 있는 액티비티들 집합의 끝을 의미한다.

워크플로우 의존 넷은 ICN의 제어흐름 정보를 중심으로 구성하게 된다. 조건 분기(or-split과 or-join)의 시점을 중심으로 각 분기점에서 노드의 깊이를 늘여가는 방법으로 구성한다. 각 액티비티와 제어흐름은 트리의 형태로 구성되게 되며, 분기된 노드와 노드의 깊이를 통해, 워크플로우 마이닝에 필요한 정보를 얻어내고자 한다. 다음은 ICN의 액티비티와 제어흐름 정보를 중심으로 워크플로우 의존 넷을 구성하는 알고리즘이다.

ICN 모델링 방법으로 표현된 '주문처리 관계를 나타내는 워크플로우'에 위 표 2에서 제시한 워크플로우 의존 넷 구성 알고리즘을 통해 구성



〈그림 4〉 주문처리 관계를 나타내는 워크플로우의 Workflow Dependent Net

〈표 3〉 주문 프로세스 모델의 Workflow Dependent Net의 정형명세

$\Omega = (\varphi, \xi, S)$ over A, T // Workflow Dependent Net	
$A = \{ \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_I, \alpha_F \}$ // Activities	
$T = \{ d(default), or_1('accept'), or_2('reject') \}$ // transition conditions	
$S = \{ \emptyset \}$	
$E = \{ \emptyset \}$	
$\varphi_i(\alpha_I) = \{\emptyset\}$, $\varphi_o(\alpha_I) = \{\alpha_1, \alpha_5, \alpha_6, \alpha_7, \alpha_F\}$;	$\xi_i(\alpha_I) = \{ \emptyset \}$, $\xi_o(\alpha_I) = \{ d \}$;
$\varphi_i(\alpha_1) = \{\alpha_I\}$, $\varphi_o(\alpha_1) = \{ \emptyset \}$;	$\xi_i(\alpha_1) = \{ d \}$, $\xi_o(\alpha_1) = \{ \emptyset \}$;
$\varphi_i(\alpha_2) = \{\alpha_6\}$, $\varphi_o(\alpha_2) = \{ \emptyset \}$; // ifd of the or-split	$\xi_i(\alpha_2) = \{ or_2 \}$, $\xi_o(\alpha_2) = \{ \emptyset \}$;
$\varphi_i(\alpha_3) = \{\alpha_6\}$, $\varphi_o(\alpha_3) = \{\emptyset\}$; // ifd of the or-split	$\xi_i(\alpha_3) = \{ or_1 \}$, $\xi_o(\alpha_3) = \{ \emptyset \}$;
$\varphi_i(\alpha_4) = \{\alpha_6\}$, $\varphi_o(\alpha_4) = \{\emptyset\}$;	$\xi_i(\alpha_4) = \{ or_1 \}$, $\xi_o(\alpha_4) = \{ \emptyset \}$;
$\varphi_i(\alpha_5) = \{\alpha_I\}$, $\varphi_o(\alpha_5) = \{\emptyset\}$;	$\xi_i(\alpha_5) = \{ d \}$, $\xi_o(\alpha_5) = \{ \emptyset \}$;
$\varphi_i(\alpha_6) = \{\alpha_I\}$, $\varphi_o(\alpha_6) = \{\alpha_2, \alpha_3, \alpha_4\}$; // ibd of α_2, α_3	$\xi_i(\alpha_6) = \{ d \}$, $\xi_o(\alpha_6) = \{ or_1, or_2 \}$;
$\varphi_i(\alpha_7) = \{\alpha_I\}$, $\varphi_o(\alpha_7) = \{\emptyset\}$;	$\xi_i(\alpha_7) = \{ d \}$, $\xi_o(\alpha_7) = \{ \emptyset \}$;
$\varphi_i(\alpha_F) = \{\alpha_I\}$, $\varphi_o(\alpha_F) = \{ \emptyset \}$;	$\xi_i(\alpha_F) = \{ d \}$, $\xi_o(\alpha_F) = \{ \emptyset \}$;

된 워크플로우 모델을 이루고 있는 각각의 액티비티 의존성에 의해 다음의 결정 트리로 나타낼 수 있으며, 분기시점에서 바로 전후의 액티비티의 의존성은 또 다른 의미로 확장해 나아갈 수 있다.

다음 표 3은 앞서 ICN을 통해서 모델링 한 주문 프로세스 모델 정보를 워크플로우 의존 넷 구성 알고리즘을 통해 워크플로우 의존 넷의 정형명세 방법인 $\Omega = (\phi, \xi, I, O)$의 집합으로 구성한 주문 프로세스 모델 워크플로우 의존 넷의 정형명세이다.

4.2 워크플로우 최적 축소 넷 (Minimal Workflow Net)

워크플로우 최적 축소 넷 은 ICN을 기반 워크플로우 의존 넷을 구성한 결정 트리 중 실제 실행경로를 결정하는 최소한의 액티비티 집합을 가지고 구성하고자 한다. 워크플로우 최적 축소 넷을 구성하기 위해서는 다음 몇 가지 정의를 필요로 한다.

[정의 4] Γ는 임의의 ICN이라 하자. 임의의 액티비티 $\alpha \in (A - \{\alpha_F\})$에서 *액티비티 타입이 즉시 앞으로 지배하는 것(activity type immediate forward dominator)*는 *aifd*(α)와 같이 표기한다.

[정의 5] Γ는 임의의 ICN이라 하자. 임의의 액티비티 $\alpha \in (A - \{\alpha_F\})$의 *연결 논리 타입이 즉시 앞으로 지배하는 것(conjunctive-logic type immediate forward dominator)*는 *cifd*(α)와 같이 표기한다.

[정의 6] Γ는 임의의 ICN이라 하자. 임의의 액티비티 $\alpha \in (A - \{\alpha_F\})$의 *분리 논리 타입이즉시 지배하는 것(disjunctive-logic type immediate forward dominator)*는 *difd*(α)와 같이 표기한다.

[정의 7] ICN의 Minimal-workflow Net. Minimal-workflow net 은 $M = (\chi, \xi, S, E)$에 따라 정형적으로 표기한다. A 는 액티비티들의 집합이며, T 는 흐름제어 조건들이다.

- $\chi = \chi_i \cup \chi_o$ 이때 $\chi_o : A \rightarrow \wp(A)$은 하나의 액티비티를 'ibd-type', 'conjunctive-type', 'disjunctive-type' 중 하나 일 경우 후행 액티비티들의 집합에 연결하는 관계를 나타내며, $\varphi_i : A \rightarrow \wp(A)$은 하나의 액티비티를 '$\alpha_1$,', 'or-split', 'and-split' 중에 속하는 선행하는 하나의 액티비티에 연결하는 것을 나타낸다.
- $\xi = \xi_i \cup \xi_o$ 이때 ξ_i은 $\tau \in T$일 때, $(\chi_i(\alpha), \alpha)$ 사이에서, 제어흐름 조건들의 집합이고, ξ_o

〈표 4〉 워크플로우 최적 축소 넷 구성 알고리즘

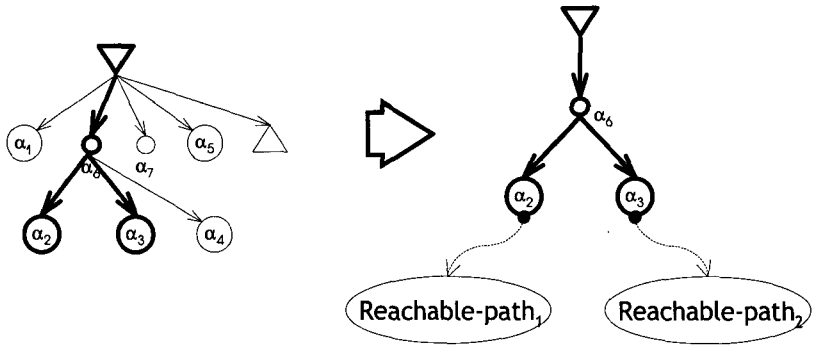
```
Input A Workflow Dependent Net, Ω = (Φ, Ξ, I, O);
Output A Minimal Workflow Net, M = (χ, ϑ , I, O);
Initialize T ← {O}; /* T is global */
PROCEDURE (In s ← {α_I}) /* Recursive Procedure */
BEGIN
    V ← s; χ_i(V) ← Φ_i(V); χ_o(V) ←{V}; T ← T∪{V};
    O ← Φ_o(V);
    FOR (∀∈O) DO
        SWITCH (What type of dependency between V and u is?) DO
          Case 'ibd-type dependency':
             χ_o(U) ← u; χ_i(u) ← U;
             ϑ_o(U) ← Ξ_o(U);      ϑ_i(U) ← Ξ_i(U);
             T ← T∪{V};
             break;
          Case 'Conjunctive-type dependency':
             χ_o(U) ← u; χ_i(u) ← U;
             ϑ_o(U) ← Ξ_o(U);      ϑ_i(U) ← Ξ_i(U);
             T ← T∪{V};
             Call PROCEDURE (In s ← u);
             break;
          Default:
              T ← T∪{V};
              break;
       END SWITCH
    END FOR
    IF ((x, y∈χ_o(U)) ∧ (x≠y) ∧ (ϑ_i(x) =ϑ_i(y)) ∧ (x=ibdtp(V)) DO
       Then do
          Eliminate x (the ibd-type dependency) from minimal workflow net;
       End;
    END IF
    END PROCEDURE
```

은 τ∈T일 때, (α , χ_o(α)) 사이에서, 제어흐름 조건들의 집합이다.

- ***S***는 워크플로우 최적 축소 넷을 구성 하고 있는 액티비티 집합 중 시작을 의미한다.
- ***E***는 워크플로우 최적 축소 넷을 구성 하고 있는 액티비티 집합 중 끝을 의미한다.

다음 표 4는 워크플로우 의존 넷의 의존 관계의 종류를 통해 워크플로우 최적 축소 넷을 구성할 수 있는 알고리즘을 나타낸다.

'주문처리 관계를 나타내는 워크플로우의 워크플로우 의존 넷'은 워크플로우 최적 축소 넷 구성 알고리즘을 통해 다음과 같은 결정 트리를 구성한다. 구성된 결정 트리에서 각 말단노드의 액티비티 집합은 실제 프로세스가 실행되는 실행경로들의 중요 액티비티, 즉 반드시 각 실행경로에서 포함하고 있는 액티비티를 나타내며, 이로써 실행경로를 알아낼 수 있다.



〈그림 5〉 주문처리 관계를 나타내는 워크플로우의 워크플로우 최적 축소 넷

〈표 5〉 주문 프로세스 모델의 Minimal Workflow Net의 정형명세

M = (φ, ξ, S) over A, T // Minimal-workflow Net
A = { α_2, α_3, α_6, α_I } // Activities
T = { d(default), or_1('accept'), or_2('reject')} // Transition Conditions
S = { Ø }
E = { Ø }

$\chi_i(\alpha_I)=\{\emptyset\}$, $\chi_o(\alpha_I)=\{\alpha_6\}$;	$\lambda_i(\alpha_I)=\{\emptyset\}$, $\lambda_o(\alpha_I)=\{$ d $\}$;
$\chi_i(\alpha_2)=\{\alpha_6\}$, $\varphi_o(\alpha_2)=\{\emptyset\}$; // ifd of the or-split	$\lambda_i(\alpha_2)=\{$ or_2 $\}$, $\lambda_o(\alpha_2)=\{\emptyset\}$;
$\chi_i(\alpha_3)=\{\alpha_6\}$, $\varphi_o(\alpha_3)=\{\emptyset\}$; // ifd of the or-split	$\lambda_i(\alpha_3)=\{$ or_1 $\}$, $\lambda_o(\alpha_3)=\{\emptyset\}$;
$\varphi_i(\alpha_6)=\{\alpha_I\}$, $\varphi_o(\alpha_6)=\{\alpha_2, \alpha_3\}$; // ibd of α_2, α_3	$\lambda_i(\alpha_6)=\{$ d $\}$, $\lambda_o(\alpha_6)=\{$ or_1, or_2 $\}$;

이는 또한 정형 명세로써, 각 노드등의 정보를 표현할 수 있는데, 다음 표 5는 주문처리 관계를 나타내는 워크플로우 모델의 워크플로우 의존 넷을 입력 값으로, 이전 장에서 표현한 워크플로우 최적 축소 넷 구성 알고리즘을 적용하여 나온 주문 처리 관계를 나타내는 워크플로우 모델의 워크플로우 최적 축소 넷을 정형명세 한 것이다.

5. 결 론

본 논문에서는 워크플로우 최적 축소 넷 모델과 그것과 관련된 알고리즘 및 기술들에 관하여 기술하였다. 특히, 본 논문에서는 실행경로 재발견 문제를 새롭게 제시하고, 그에 대한 적절한 해결방안을 제시하였다. 결국 해결방안은 워크플로우 실행정보로부터 실행경로를 재발견하기 위한 워크플로우 마이닝 프레임워크로 제시하였다. 앞에서는 ICN으로부터 워크플로우 최적 축소 모델을 구성하기 위하여, 워크플로우 의존 넷을 구성하고, 워크플로우 최적 축소 넷 구성하는 알고리즘을 적용하였다. 그리고 실행경로 구성하는 단계에 까지 이르게 되었다. 최근에는 다양하고, 진보적인, 그리고, 차별화된 워크플로우 마이닝 기술과 아키텍쳐가 필요한 실정이다. 이러한 요구로써 본 논문에서는 워크플로우 축소 모델을 구성에 대하여 논하였으며, 워크플로우 마이닝 프레임워크를 제시하였다. 본 모델은 대규모의 비즈니스 프로세스 모델을 분석하고, 프로세스를 재설계 하는데 적합할 수 있으며, 기업 업무의 효율성을 극대하기 위한 프로세스 재설계에도 적용할 수 있다. 앞으로, 초대형 워크플로우 관리 시스템(e-Chatauqua WfMS)[11] 개발 프로젝트와 함께 통합된 프로젝트를 진행해 나가 더 낳은 결과를 얻어내고자 한다.

참 고 문 헌

[1] Clarence A. Ellis and Gary J. Nutt, "The Modeling and Analysis of Coordination Systems", University of Colorado/Dept. of Computer Science Technical Report, CU-Cs-639-93, Jan 1993

[2] Frank Leymann, Dieter Roller "Production Workflow Concepts and Techniques"

[3] Kwang-Hoon Kim, Clarence A. Ellis, "Workflow Reduction for Reachable-path Rediscovery", IEEE Proceeding of the ICDM Workshop 2003, 2003.11

[4] Rakesh Agrawal, Dimitrios Gunoplusos Mining Process Models From Workflow Logs, IBM Almaden Research Center, San Joes, CA, USA

[5] W.M.P. van der Aslst, et el, "Work°ow Mining: Which processes can be redis-covered?", Technical Report, Department of Technology Management, Eindhoven University of Tech-nology, (2002)

[6] Workflow Management Coalition Specification Document, "Workflow Management Coalition Audit Data Specification", Version 1.1, Document Number: WFMC-TC-1015, September 1998.

[7] Workflow Management Coalition Specification Document, "Workflow management Coalition Terminology & Glossary", Version 2.0, Document Number: WFMC-TC-1011, June 1996.

[8] Workflow Management Coalition Specification, "Workflow Process Definition Interface - XML Process Definition Language" October 25, 2002

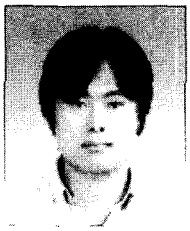
[9] 류재광, 김광훈, "실시간 협업지원 그룹 ICN 에디터의 설계 및 구현 ", 한국인터넷정보학회 논문지, 2권 5호, pp.1 - 7, 2001.1

[10] 박민재, 원재강, 김광훈, "워크플로우 도달 가능경로 분석도구", 한국정보과학회 추계 학술발표논문집(II), 31권 2호. pp.127-129

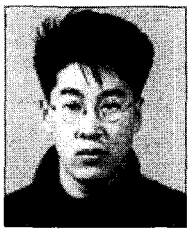
[11] 안형진, 김광훈, "초대형 워크플로우 시스템에서의 워크케이스 생성 기법에 대한 구현 분석", 한국정보과학회 춘계학술발표논문집(II), 31권 1호, pp.67-69, 2004.04

[12] 홍형석, "워크플로우 마이닝을 위한 확장형 트랜잭션 워크플로우 모니터링 시스템", 경기대학교 대학원 석사학위논문, 2000

◐ 저 자 소 개 ◑

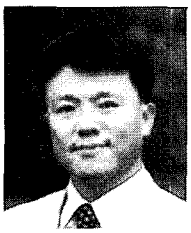


박 민 재 (Min Jae Park)
2000년 경기대학교 전자계산학과 졸업(학사)
2004년~현재 경기대학교 대학원 전자계산학과 석사과정
관심분야 : Workflow/Process Mining, BPM, RBAC
E-mail : mean222@kyonggi.ac.kr



원 재 강 (Jae Kang Won)
1999년 강릉대학교 생물학과 졸업(학사)
2002년 경기대학교 대학원 전자계산학과 졸업(석사)
2002년~현재 경기대학교 대학원 전자계산학과 박사과정
관심분야 : RBAC, Workflow, BPM
E-mail : jkwon@kyonggi.ac.kr

김 창 민 (Chang Min Kim)
1990년 중앙대학교 대학원 컴퓨터 공학과 졸업(박사)
1989년~1994년 관동대학교 전자계산학과 교수
1994년~현재 성결대학교 컴퓨터공학부 교수
관심분야 : Web Service, System Software, Workflow
E-mail : kimcm@sungkyul.ac.kr



김 광 훈 (Kwang Hoon Kim)
1984년 경기대학교 전자계산학과 졸업(학사)
1986년 중앙대학교 대학원 전자계산학과 졸업(석사)
1994년 University of Colorado at Boulder, Computer Science, MS
1998년 University of Colorado at Boulder, Computer Science, Ph.D
1986년 2월~1991년 8월 한국전자통신연구원
1993년 5월~1994년 8월 American Educational Products, Inc., Processional DB Consultant
1994년 9월~1995년 8월 Colorado Advanced Software Institute, Research Assistant
1995년 9월~1997년 2월 Aztek Engineering, Inc., Software Engineer
1998년~현재 경기대학교 정보과학부 조교수
관심분야 : Workflow, BPM, Groupware, Database
E-mail : kwang@kyonggi.ac.kr