의존성 기반 워크플로우 마이닝 기법

박민재, 김광훈
경기대학교 전자계산학과 워크플로우 기술 연구실
e-mail : mean222@kyonggi.ac.kr

Dependency based Workflow Mining Method

Min-Jae Park, Kwang-Hoon Kim
Dept. of Computer Science, Kyonggi University

요 약

워크플로우의 도입으로 인하여 다수의 기업은 업무의 효율성을 증대 시켰다. 하지만, 진정한 기업 업무의 효율성 증대는 워크플로우의 도입으로 인한 비즈니스 프로세스의 자동화뿐만 아니라 비즈니스 프로세스 설계의 최적화 및 개선이라고 판단된다. 이에 본 논문에서는 의존성 기반의 워크플로우 마이닝 기법에 관하여 제시 하며, 비즈니스 프로세스 설계의 최적화 및 개선을 위하여 어떠한 방법으로 접근해 나아가야 할지에 관하여 논한다.

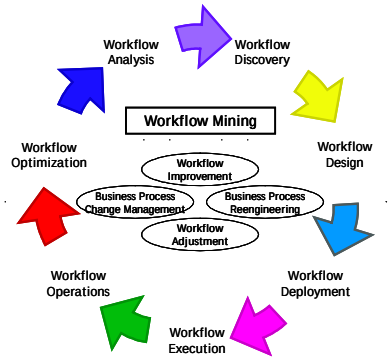
1. 서론

현대 정보기술 분야에 있어서 최근의 가장 두드러진 변화는 기존의 데이터 중심 정보기술에서 프로세스 중심 정보기술로 빠르게 전이되고 있다는 사실이다. 1960 년대의 파일 시스템을 기반으로 한 정보기술의 발전은 1980 년대 관계형 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 2000 년대에는 프로세스 또는 워크플로우 중심의 정보기술(BPM/WfM: Business Process/Workflow 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. 이러한 개선의 노력의 일환으로 본 연구를 통해, 워크플로우 절차 및 비즈니스 프로세스의 재설계 및 개선에 용이할 수 있는 워크플로우 마이닝 기법에 관하여 연구해 나아가고자 한다.

2. 관련연구

2.1 워크플로우 마이닝

워크플로우 절차 및 비즈니스 프로세스는 실 세계의 워크플로우 절차를 발견하고, 전산화하여 디자인하고 전개하며, 워크플로우 엔진을 통해 실행을 시키고 운용하고 최적화 시키고 분석하는 일련의 반복적인 행위를 통해 워크플로우 절차 및 비즈니스 프로세스를 개선시키고 향상시키고자 하는 것이 프로세스 및 워크플로우 마이닝의 목적이라고 할 수 있다.



(그림 1) 워크플로우 마이닝

본 논문에서는 이러한 워크플로우 마이닝의 한 도태로 의존성 기반의 워크플로우 마이닝 기법에 관하여 연구하고자 한다.

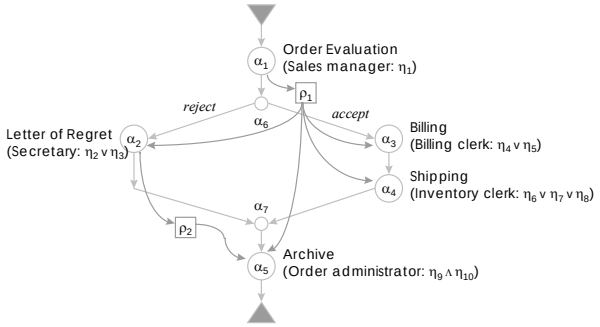
2.2 ICN(Information Control Net)

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

ICN 은 사무실(Office)의 개념을 일련의 관련된 프로시져(Procedures)의 집합으로 정의하며 이러한 프로시져는 선후관계가 존재하는 액티비티들의 집합으로 표현된다. ICN 은 그림형태로 프로시져, 액티비티, 저장소(Repositories), 선후관계를 나타내는 제어 흐름(Control Flow)과 데이터 흐름(Data Flow)을 표현한다. 이러한 방법으로 형성된 ICN 은 각 단계를 정형화한 방법으로 구성 및 분석 할 수 있다.

2.2.1 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 AND 노드, 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.



(그림 2) 주문처리 관계를 나타내는 워크플로우

여기서 전체 워크플로우 절차의 이름은 '주문처리'이며, 각 과정을 담당하는 액티비티는 주문평가(Order Evaluation), 거절(Letter of Regret), 결재(Billing), 발송(Shipping), 문서집적(Archive)(α_1, α_2, α_3, α_4, α_5)이다. 그리고 α_6, α_7 은 OR 분기 및 결합을 나타내는 노드이며, ρ_1, ρ_2 는 데이터의 저장소를 나타낸다. 또한 직선형태로 된 실선은 제어의 흐름을 나타내고 곡선형태의 실선은 정보의 흐름을 나타낸다.

2.2.2 ICN 의 정형명세

A 를 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 출력 저장소들의 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 $\pi_p : C \rightarrow \wp(P)$ 조합된 수행자들의 집합을 역할에 단일 값으로 연결하는 것이고, $\pi_c : P \rightarrow \wp(C)$는 조합된 역할들을 수행자에 단일 값으로 연결하는 것이다.
- $\kappa = \kappa_i \cup \kappa_o$ 이때 κ_i는 $\alpha \in A$일 때, $(\delta_i(\alpha), \alpha)$ 사이에서, 제어흐름(T) 조건들의 집합이고, κ_o는 $\alpha \in A$일 때, $(\alpha, \delta_o(\alpha))$ 사이에서, 제어흐름(T) 조건들의 집합이다. 이때 집합 T = {*default, or*(*conditions*), *and*(*conditions*)}.
- I 는 초기 입력되는 자료 저장소들의 유한 집합이며, ICN의 실행 전에 어떠한 외부의 프로세스에 의해서 로드 되어야 한다.
- O 는 마지막으로 출력되는 자료저장소들의 유한 집합이며 ICN의 실행 후에 어떠한 외부의 프로세스에 의해서 이용되는 정보를 포함하고 있다고 가정한다.

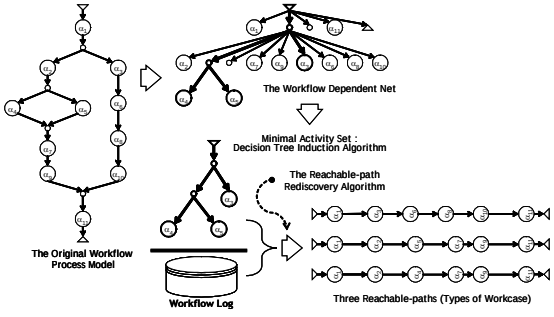
(표 1) 주문 처리 관계를 나타내는 워크플로우의 정형 명세

$\Gamma = (\delta, \gamma, \varepsilon, \pi, \kappa, I, O)$ over A, R, P, C, T // Order Processing in ICN
$A = \{ \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_I, \alpha_F \}$ // Activities
$R = \{ \rho_1, \rho_2 \}$ // Repositories
$P = \{ \varpi_1, \varpi_2, \varpi_3, \varpi_4, \varpi_5 \}$ // Roles
$C = \{ \eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7, \eta_8, \eta_9, \eta_{10} \}$ // Actors
$T = \{ d(default), or_1('Reject'), or_2('Accept') \}$ // Control-transition conditions
$I = \{ \varnothing \}$; $O = \{ \varnothing \}$

$\delta_i(\alpha_1)=\{\alpha_I\}$,	$\gamma_i(\alpha_1)=\{\varnothing\}$,	$\varepsilon_P(\alpha_1)=\{\varpi_1\}$;	$\pi_c(\varpi_1)=\{\eta_1\}$;	$\kappa_i(\alpha_1)=\{d\}$,
$\delta_o(\alpha_1)=\{\alpha_6\}$;	$\gamma_o(\alpha_1)=\{\rho_1\}$;	$\varepsilon_P(\alpha_2)=\{\varpi_2\}$;	$\pi_c(\varpi_2)=\{\eta_2,\eta_3\}$;	$\kappa_o(\alpha_1)=\{d\}$;
$\delta_i(\alpha_2)=\{\alpha_6\}$,	$\gamma_i(\alpha_2)=\{\rho_1\}$,	$\varepsilon_P(\alpha_3)=\{\varpi_3\}$;	$\pi_c(\varpi_3)=\{\eta_4, \eta_5\}$;	$\kappa_i(\alpha_2)=\{or_1\}$,
$\delta_o(\alpha_2)=\{\alpha_7\}$;	$\gamma_o(\alpha_2)=\{\rho_2\}$;	$\varepsilon_P(\alpha_4)=\{\varpi_4\}$;	$\pi_c(\varpi_4)=\{\eta_6, \eta_7, \eta_8\}$;	$\kappa_o(\alpha_2)=\{d\}$;
$\delta_i(\alpha_3)=\{\alpha_6\}$,	$\gamma_i(\alpha_3)=\{\rho_1\}$,	$\varepsilon_P(\alpha_5)=\{\varpi_5\}$;	$\pi_c(\varpi_5)=\{ \eta_8, \eta_9\}$;	$\kappa_i(\alpha_3)=\{or_2\}$,
$\delta_o(\alpha_3 =\{\alpha_4\}$;	$\gamma_o(\alpha_3)=\{\varnothing\}$;	$\varepsilon_P(\alpha_6)=\{\varnothing\}$;		$\kappa_o(\alpha_3)=\{d\}$;
$\delta_i(\alpha_4)=\{\alpha_3\}$,	$\gamma_i(\alpha_4)=\{\rho_1\}$,	$\varepsilon_P(\alpha_7)=\{\varnothing\}$;		$\kappa_i(\alpha_4)=\{d\}$,
$\delta_o(\alpha_4)=\{\alpha_7\}$;	$\gamma_o(\alpha_4)=\{\varnothing\}$;	$\varepsilon_P(\alpha_I)=\{\varnothing\}$;		$\kappa_o(\alpha_4)=\{d\}$;
$\delta_i(\alpha_5)=\{\alpha_7\}$,	$\gamma_i(\alpha_5)=\{\rho_1, \rho_2\}$,	$\varepsilon_P(\alpha_F)=\{\varnothing\}$;		$\kappa_i(\alpha_5)=\{d\}$,
$\delta_o(\alpha_5)=\{\alpha_F\}$;	$\gamma_o(\alpha_5)=\{\varnothing\}$;			$\kappa_o(\alpha_5)=\{d\}$;
$\delta_i(\alpha_6)=\{\alpha_1\}$,	$\gamma_i(\alpha_6)=\{\varnothing\}$,			$\kappa_i(\alpha_6)=\{d\}$,
$\delta_o(\alpha_6)=\{\alpha_2, \alpha_3\}$;	$\gamma_o(\alpha_6)=\{\varnothing\}$;			$\kappa_o(\alpha_6)=\{d\}$;
$\delta_i(\alpha_7)=\{\alpha_2, \alpha_4\}$,	$\gamma_i(\alpha_7)=\{\varnothing\}$,			$\kappa_i(\alpha_7)=\{d\}$,
$\delta_o(\alpha_7)=\{(\alpha_5)\}$;	$\gamma_o(\alpha_7)=\{\varnothing\}$;			$\kappa_o(\alpha_7)=\{d\}$;
	$\gamma_i(\alpha_I)=\{\varnothing\}$,			$\kappa_i(\alpha_I)=\{\varnothing\}$,
	$\gamma_o(\alpha_I)=\{\varnothing\}$;			$\kappa_o(\alpha_I)=\{d\}$;
	$\gamma_i(\alpha_F)=\{\varnothing\}$,			$\kappa_i(\alpha_F)=\{d\}$,
	$\gamma_o(\alpha_F)=\{\varnothing\}$;			$\kappa_o(\alpha_F)=\{\varnothing\}$;

3. 의존성 기반 워크플로우 마이닝 기법

본 논문에서 제시하고자 하는 워크플로우 마이닝 기법은 다음 그림 3 에서 표현하고 있는 것과 같이, 워크플로우를 구성하고 있는 액티비티간의 의존성을 파악하여 프로세스의 실행경로의 빈도수를 워크플로우 로그로부터 파악하여, 워크플로우의 성향을 알아낸 후 개선하고자 하는 노력이다.



(그림 3) 의존성 기반 워크플로우 마이닝 기법

3.1 워크플로우 의존 넷(Workflow Dependent Net)

ICN 으로 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 워크플로

우 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] ICN 에서 일은 W. Γ = (δ, γ, ε, π, κ, I, O)로 표기하는 ICN 에서 일은 W 라고 하며, n>0 이고 i=1, 2, …, n-1 동안 $\alpha_{i+1} \in \delta_o(\alpha_{i+1})$ 일 때, $\alpha_1, \alpha_2, \ldots, \alpha_n$ 와 같은 일련의 액티비티들의 집합이다. 일 W = $\alpha_1 \alpha_2 \ldots \alpha_n$ 의 길이는 |W|로 표기하며 W 에서 발생한 액티비티 발생 개수다.

[정의 2] ICN 에서의 *지배하는 성질*. ICN 에서, Γ은 각각의 다음 조건을 만족한다: (*a*) Γ 은 두가지의 구별되는 액티비티들은 포함한다: $\delta_I(\alpha_I)$에 대한 시작 α_I, 액비티비의 집합은 ∅이며, $\delta_o(\alpha_F)$에 대한 끝 액티비티 α_F 의 집합 ∅이다. (*b*) Γ의 모든 액티비티는 $\alpha_I - \alpha_F$ 사이에서 이루어진다.

- **Γ는 임의의 ICN이라 하자. 모든 단계인 Γ가 포함하는 *u*에서 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, *u*가 *v*를 앞으로 지배할 때, *u*는 *적절히 앞으로 지배하는 성질(properly forward dominates)을 가진다.***
- **Γ는 임의의 ICN이라 하자. 만약 *u*가 *v*를 앞으로 지배하는 성질을 가지며, *v*와 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*(α)로 표기한다.**
- **Γ는 임의의 ICN이라 하자. 만약 모든 단계 Γ, $v - v_I$ 에서 *u*를 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, *u*가 *v*를 뒤로 지배하는 성질을 가질 때, *u*는 *v*를 *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*(α)라고 표기한다.**

[정의 3] ICN 에서의 *워크플로우 의존 넷*. 워크플로우 의존 넷은 Ω = (φ, ξ, *S*, *E*)에 따라 정형적으로 정의한다. 이때, *A* 는 액티비티들의 집합이며, *T* 는 흐름제어 조건들이다.

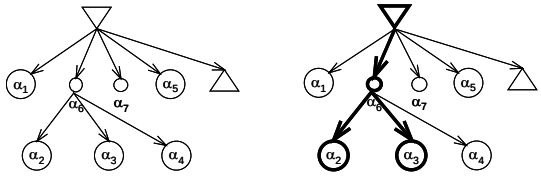
- **$\varphi = \varphi_i \cup \varphi_o$**
  이때, φ_o: $A \rightarrow \wp(A)$ 은 하나의 액티비티를 그것과 제어 의존적으로 후행하는 액티비티들의 집합에 연결하는 연결관계를 나타내며, φ_i: $A \rightarrow \wp(A)$ 은 하나의 액티비티를 그것과 제어 의존적으로 선행하는 액티비티들의 집합의 연결관계를 나타낸 것이다.
- **$\xi = \xi_i \cup \xi_o$**
  $\alpha \in A$ 인 범위에서, $\tau \in T$ 인 ξ_i 는 각각의 실선 ($\varphi_i(\alpha)$, α)의 제어흐름 조건들의 집합이며, $\tau \in T$인 ξ_o은 각각의 실선(α, $\varphi_o(\alpha)$)의 제어흐름 조건들의 집합이다,
- ***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 ICN 의 제어흐름 정보를 중심으로 구성하게 된다. 조건 분기(or-split 과 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 ICN 의 액티비티와 제어흐름 정보를 중심으로 워크플로우 의존 넷을 구성하는 알고리즘이다.

(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
```

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



(그림 4) 주문처리 관계를 나타내는 워크플로우의 워크플로우 의존 넷(Workflow Dependent Net)

다음 표 3 은 앞서 ICN 을 통해서 모델링 한 주문 프로세스 모델 정보를 워크플로우 의존 넷 구성 알고리즘을 통해 워크플로우 의존 넷의 정형명세 방법인 Ω = (φ, ξ, I, O)의 집합으로 구성한 주문 프로세스 모델 워크플로우 의존 넷의 정형명세이다.

(표 3) 주문처리 관계를 나타내는 워크플로우의 워크플로우 의존 넷(Workflow Dependent Net)의 정형명세

Ω = (φ, ξ, *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 = { ∅ }	
E = { ∅ }	
$\varphi_i(\alpha_I)$={∅}, $\varphi_o(\alpha_I)$={$\alpha_1, \alpha_5, \alpha_6, \alpha_7, \alpha_F$};	$\xi_i(\alpha_I)$={ ∅}, $\xi_o(\alpha_I)$={ *d* };
$\varphi_i(\alpha_1)$={α_I}, $\varphi_o(\alpha_1)$={ ∅};	$\xi_i(\alpha_1)$={ *d* }, $\xi_o(\alpha_1)$={ ∅};
$\varphi_i(\alpha_2)$={α_6}, $\varphi_o(\alpha_2)$={ ∅}; // *ifd* of the or-split	$\xi_i(\alpha_2)$={ *or*$_2$ },$\xi_o(\alpha_2)$={ ∅};
$\varphi_i(\alpha_3)$={α_6}, $\varphi_o(\alpha_3)$={∅}; // *ifd* of the or-split	$\xi_i(\alpha_3)$={ *or*$_1$ }, $\xi_o(\alpha_3)$={ ∅};
$\varphi_i(\alpha_4)$={α_6}, $\varphi_o(\alpha_4)$={∅};	$\xi_i(\alpha_4)$={ *or*$_1$ }, $\xi_o(\alpha_4)$={ ∅};
$\varphi_i(\alpha_5)$={α_I}, $\varphi_o(\alpha_5)$={∅};	$\xi_i(\alpha_5)$={ *d* }, $\xi_o(\alpha_5)$={ ∅};
$\varphi_i(\alpha_6)$={α_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*$_2$};
$\varphi_i(\alpha_7)$={α_I}, $\varphi_o(\alpha_7)$={∅};	$\xi_i(\alpha_7)$={ *d* }, $\xi_o(\alpha_7)$={ ∅};
$\varphi_i(\alpha_F)$={α_I}, $\varphi_o(\alpha_F)$={ ∅};	$\xi_i(\alpha_F)$={ *d* }, $\xi_o(\alpha_F)$={ ∅};

3.2 워크플로우 최적 축소 넷 (Minimal Workflow Net)

워크플로우 최적 축소 넷 은 ICN 을 기반 워크플로우 의존 넷을 구성한 결정 트리 중 실제 실행경로를 결정하는 최소

한의 액티비티 집합을 가지고 구성하고자 한다. 워크플로우 최적 축소 넷을 구성하기 위해서는 다음 몇 가지 정의를 필요로 한다.

[정의 4] Γ는 임의의 ICN 이라 하자. 임의의 액티비티 α ∈ (A – {α_F})에서 *액티비티 타입이 즉시 앞으로 지배하는 것(activity-type immediate forward dominator)*는 *aifd*(α)와 같이 표기한다.

[정의 5] Γ는 임의의 ICN 이라 하자. 임의의 액티비티 α ∈ (A – {α_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 ICN 이라 하자. 임의의 액티비티 α ∈ (A – {α_F })의 *분리 논리 타입이즉시 지배하는 것(disjunctive-logic-type immediate forward dominator)*는 *difd*(α)와 같이 표기한다.

[정의 7] ICN 의 Minimal-workflow Net. Minimal-workflow net 은 M = (χ, ξ, *S*, *E*)에 따라 정형적으로 표기한다. A 는 액티비티들의 집합이며, T 는 흐름제어 조건들이다.

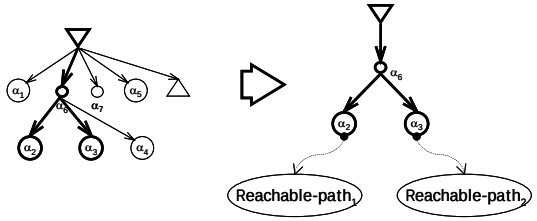
- **$\chi = \chi_i \cup \chi_o$ 이때 $\chi_o : A \rightarrow \wp(A)$은 하나의 액티비티를 'ibd-type', 'conjunctive-type', 'disjunctive-type' 중 하나 일 경우 후행 액티비티들의 집합에 연결하는 관계를 나타내며, $\varphi_i : A \rightarrow \wp(A)$은 하나의 액티비티를 '$\alpha_1$', 'or-split', 'and-split' 중에 속하는 선행하는 하나의 액티비티에 연결하는 것을 나타낸다.**
- **$\xi = \xi_i \cup \xi_o$ 이때 ξ_i은 $\tau \in T$일 때, (χ_i (α), α) 사이에서, 제어흐름 조건들의 집합이고, ξ_o은 $\tau \in T$일 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 4 는 워크플로우 의존 넷의 의존 관계의 종류를 통해 워크플로우 최적 축소 넷을 구성할 수 있는 알고리즘을 나타낸다.

(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 ← {0}; /* T is global */
PROCEDURE (In s ← {α1}) /* Recursive Procedure */
BEGIN
    ν ← s; χi(ν) ← φi(ν); χo(ν) ←{ν}; T ← T∪ {ν};
    O ← φo(ν);
    FOR (∀ ∈O) DO
        SWITCH (What type of dependency between ν and u is?) DO
        Case 'ibd-type dependency':
            χo(υ) ← u; χi(u) ← υ;
            ϑ o(υ) ← ξo(υ);        ϑ i(υ) ← ξi(υ);
            T ← T∪ {ν};
            break;
        Case 'Conjunctive-type dependency':
            χo(υ) ← u; χi(u) ← υ;
            ϑ o(υ) ← ξo(υ);        ϑ i(υ) ← ξi(υ);
            T ← T∪ {ν};
            Call PROCEDURE (In s ← u);
            break;
        Default:
            T ← T∪ {ν};
            break;
    END SWITCH
END FOR
IF ((x, y∈χo(υ)) ∧ (x≠y) ∧ ( ϑ i(x) = ϑ i(y)) ∧ (x=ibdtp(ν)) DO
    Then do
        Eliminate x (the ibd-type dependency) from minimal workflow net;
    End;
END IF
END PROCEDURE
```

'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(Minimal Workflow Net)

이는 또한 정형 명세로써, 각 노드등의 정보를 표현할 수 있는데, 다음 표 5 는 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.

(표 5) 주문처리 관계를 나타내는 워크플로우의 워크플로우 최적 축소 넷(Minimal Workflow Net)의 정형명세

M = (φ, ξ, *S*) over A, T	// Minimal-workflow Net	
A = { α_2, α_3, α_6, α_1 }	// Activities	
T = { *d*(*default*), *or*$_1$('*accept*'), *or*$_2$('*reject*')}	// Transition Conditions	
S = { ∅ }		
E = { ∅ }		
$\chi_i(\alpha_1)=\{\emptyset\}$, $\chi_o(\alpha_1)=\{\alpha_6\}$;		$\lambda_i(\alpha_1)=\{\emptyset\}$, $\lambda_o(\alpha_1)=\{d\}$;
$\chi_i(\alpha_2)=\{\alpha_6\}$, $\varphi_o(\alpha_2)=\{\emptyset\}$;	// *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	$\lambda_i(\alpha_3)=\{or_1\}$, $\lambda_o(\alpha_3)=\{\emptyset\}$;
$\varphi_i(\alpha_6)=\{\alpha_1\}$, $\varphi_o(\alpha_6)=\{\alpha_2, \alpha_3\}$;	// *ibd* of α_2, α_3	$\lambda_i(\alpha_6)=\{d\}$, $\lambda_o(\alpha_6)=\{or_1, or_2\}$;

4. 결론 및 향후 연구방안

본 논문에서는 의존성 기반의 워크플로우 마이닝 기법에 관하여 제시하였고, 의존성 기반 워크플로우 마이닝을 하기 위한 2 가지 알고리즘으로 워크플로우 의존 넷 구성 알고리즘과 워크플로우 최적 축소 넷 구성 알고리즘을 제시하였으며, 그에 따른 정형 명세 또한 제시하였다.

향후 연구로써, 본 연구를 통해 얻어진 워크플로우 마이닝 알고리즘을 실제 프로세스 모델에 적용해 보고자 하며, 적용한 모델을 바탕으로 프로세스 개선을 할 수 있는지에 관하여 연구하고자 한다.

Acknowledgement

본 연구는 정보통신연구진흥원 정보통신 기초기술연구지원사업(04-기초-0005)의 지원으로 수행되었음.

참고문헌

[1] Kwang-Hoon Kim, "Workflow Control Dependency Analysis and Its Implication on Distributed Workflow Systems", Journal of Applied Systems Studies, 2003.12

[2] Dustdar, S., Hoffmann, T. and van der Aalst, W.M.P., Mining of ad-hoc business processes with TeamLog, Technical Report TUV-1841-2004-07, Vienna University of Technology , 2004.

[3] B.F. van Dongen and , W.M.P. van der Aalst, EMiT: A Process Mining Tool, 25th International Conference on Applications and Theory of Petri Nets (ICATPN 2004), J. Cortadelle and W. Reisig, LNCS 3099, pages 454-463, 2004.