액티비티 의존성을 이용한 워크케이스 마이닝 메커니즘

A Workcase Mining Mechanism using Activity Dependency

김 상 배* 김 학 성** 백 수 기***
Sang-Bae Kim Hak-Seong Kim Su-Ki Paik

요 약

워크플로우 마이닝이란 워크플로우 관리시스템에서 생성되고 실행되어진 비즈니스 프로세스의 인스턴스의 실행경로를 분석하여 새로운 정보를 추출하는 과정을 의미한다. 본 논문에서는 빌드타임(Build Time)에 정의된 모델의 실행경로와 런타임(Run Time)에서 실행된 워크케이스의 실행경로를 비교/분석하여 새로운 워크플로우 모델의 제시 또는 작성된 비즈니스 프로세스의 개선을 위한 작업으로서 액티비티(activity) 의존성을 이용한 워크케이스 마이닝에 관하여 기술하겠다. 제안된 워크케이스 마이닝을 위하여 각 액티비티의 의존성을 분석하여 중요경로(Essential Path)를 결정하는 액티비티 의존 넷 알고리즘(Activity-Dependent Net Algorithm)과 각 워크케이스 실행 정보를 이용하여 중요 경로와 일치여부를 결정하는 실행계열 분석 알고리즘(E-Walk Series Analysis Algorithm)을 제안한다.

Abstract

Workflow mining is a newly emerging research issue for rediscovering and reengineering workflow models from workflow logs containing information about workflows being executed on the workflow engine. In this paper, we proposed workcase mining which was used dependency among activities. Main purpose of this paper is to minimize discrepancies between the modeled workflow process and the enacted workflow process as it is actually being executed. That is, we can get a complete set of activity firing sequences on buildtime. Besides, we can discover from workflow logs that which path out of all reachable paths a workcase has actually followed through on runtime. For this purpose we proposed two algorithm, the one is "Activity-Dependent Net Algorithm"and the other is "E-Walk Series Analysis Algorithm".

Keyword : Workcase Mining Mechanism, Activity Dependency, E-Walk Series Analysis Algorithm

1. 서 론

어느 시대 또는 어느 형태의 사회라 할지라도 그 사회가 갖는 사람과 사람들간의 상호 활동과 그들 간의 관계를 지원하기 위한 특징적인 수단과 방법을 제공하고 있다. 오늘날과 같이 사무실에서 뿐 만 아니라 가정에서도 컴퓨터를 광범위하게 이용하고 있는 정보화 사회에서는 사람들간의 상호 활동 및 관계에 있어서 10년 전이나 20년 전과 별 차이가 없겠지만, 그들을 지원하기 위한 수단이나 방법 면에서는 많은 차이가 있음에 틀림없다. 즉, 컴퓨터 기술과 전자통신 기술의 급진적인 발전과 이러한 기술들간의 수렴은 바로 전자적인 작업환경(Electronic Workspace)이라는 새롭고도 매우 효율적인 상호 작용 지원 수단 및 방법을 잉태하게 되었다. 전자적인 작업환경이란 전(全) 조직체적 통합 시스템으로 정보 처리 활동과 정보 통신 활동의 통합을 통해 조직내의 구성원들간의 상호 활동 및 관계를 정의하고 지원하는 개선된 형태의 조직 활동 수단 및 방법이다.

워크플로우 기술이란 바로 이러한 전자적 작업환경을 구현하기 위한 종합적인 연구 분야로서 컴퓨터 및 통신 분야 뿐 만 아니라 사회학 분야나 언어학 분야, 경영학 분야 등의 다각적인 협력관계를 통해서만 성공적으로 완성될 수 있는 매

* 정 회 원 : 경기대학교 전자계산학과 박사과정
skim@chollian.net(제1저자)
** 정 회 원 : 동남보건대학 웹컨텐츠개발과 교수
amang@dongnam.ac.kr(공동저자)
*** 정 회 원 : 경기대학교 전자계산학과 교수
skpaik@kyonggi.ac.kr(공동저자)

우 다중 적인 연구 분야라고 할 수 있다[1,2].

최근에는 조직체내에서 처리하고 있는 일련의 사무업무 처리 과정들을 비즈니스 프로세스(Business Process)를 통해 정의하고 이들의 효율적이면서 효과적인 관리를 위한 자동화된 작업 및 사무환경을 지원하는 사무운영체제(Business Operating System)라는 용어가 등장하였는데, 워크플로우 기술은 바로 이러한 사무운영체제의 근간이 되는 기술이라고 할 수 있다. 즉, 워크플로우 기술은 조직체내 또는 조직체들간에 처리되고 있는 일련의 사무업무 또는 그룹 작업 및 활동이 어떻게 이루어지는가를 분석하고 컴퓨터 및 통신을 비롯한 첨단 기술들을 이용하여 어떻게 그룹 활동들을 효과적으로 그리고 자동적으로 지원할 수 있는가를 연구하는 광범위한 분야이다.

워크플로우 기술이 전자상거래를 비롯한 고객관리기술(CRM : Customer Relationship Management), 공급망 및 가치사슬망 관리기술(SCM : Supply Chain Management), 데이터 및 응용 프로그램 통합기술(EAI : Enterprise Applications Integration), 전사적 자원관리(ERP : Enterprise Resource Planning) 등과 같은 최첨단 정보기술의 핵심 기반 기술로서 중요하게 인식되고 있는 이유는 현재와 같이 분산되고 복잡한 업무 프로세스를 지원할 수 있고, 고성능, 확장성, 신뢰성, 유연성, 상호 가용성 등과 같은 요구 사항들을 제공할 수 있는 기술이 워크플로우 기술이라고 기대하기 때문이다. 또한 B2C/B2B로 대변되는 전자상거래 및 전자시장(E-Market Place)의 활성과 함께 전자정부(E-Government) 실현을 위한 기반 구조가 급속하게 확장됨에 따라 워크플로우를 기반으로 한 조직내부 업무 흐름의 자동화를 의미하는 B2E의 구축을 더욱 가속화시키고 있으며, 워크플로우 기술 및 시스템은 결국 기업 또는 조직체내의 모든 업무 처리 절차들을 통합 관리하는 인프라구조(Infrastructure)로 인식되고 있다.

워크플로우는 크게 워크플로우 모델 분야와 워크플로우 관리 시스템 분야로 나눌 수 있고 워크플로우 모델 분야는 일련의 비즈니스 관련 업무들의 흐름(Control Flow)뿐만 아니라 관련 데이터(Data Flow)와 조직내의 참여자(Actor) 또는 참여자(Participant), 역할(Role) 등을 표현하고 정의하는 수단을 제공하고, 워크플로우 관리 시스템 분야는 워크플로우 모델 분야에서 정의된 업무 흐름을 실제 수행하고 그와 관련된 각종 자원들을 관리함으로서 업무 흐름 전반을 관리 및 제어하는 기능을 수행한다. 즉, 워크플로우 관리 시스템의 수행이 정상적으로 종료되었다는 의미는 워크플로우 모델에서 정의된 비즈니스 프로세스의 수행을 마쳤다는 것을 의미하고 있다.

그러나 최근에 조직 내 또는 조직간에 발생되는 비즈니스 프로세스가 복잡해지고, 규모가 커지고, 오랜 수행 시간을 요구하고 있어 비즈니스 프로세스를 디자인하는 작업은 매우 복잡하고, 많은 시간을 요구하는 작업이다. 또한 작성된 비즈니스 프로세스가 워크플로우 관리 시스템에서 정상적으로 수행되리라고 보장 할 수 없다. 즉, 워크플로우 관리 시스템에서 수행되기 위한 완벽한 비즈니스 모델을 정의하는 것이 쉽지 않다는 것을 의미하고 있다.

잘못된 비즈니스 프로세스의 정의로 인하여 워크플로우 관리 시스템의 실행 중 오류 또는 정상적인 종료를 할 수 없다면 이는 조직 또는 기업의 관점에서 볼 때 치명적(Mission Critical)인 문제가 될 수 있다. 또한 최근의 비즈니스 프로세스 또는 워크플로우 프로시져와 같이 처리해야 할 규모가 매우 크고, 복잡하고 그리고 생명주기(Life-Cycle)와 짧아짐에 따라 워크플로우 마이닝의 중요성은 점차 증가하고 있다.

워크플로우 마이닝이란 "사용자에 의해 정의된 워크플로우 모델과 워크플로우 프로세스를 시스템을 통해 실행된 정보를 이용하여 현재 실행된 프로세스를 평가하고, 기존의 프로세스를 재구성하거나 워크플로우 프로세스를 생성할 때 도움을 줄 수 있는 데이터를 추출하는 것"과 같이 정의할 수 있다. 워크플로우 마이닝은 실행정보의 마

이닝 데이터 - 워크케이스의 실행 시간, 각 워크케이스에 참여한 액터(actor), 워크플로우 프로시져(workflow procedure)로부터 발생된 인스턴스(instance), 워크케이스에 할당된 역할(role) 등 - 에 따라 다양한 정보를 제공할 수 있다[2,3,4].

본 논문에서는 빌드타임(Build Time)에 정의된 모델의 실행경로와 런타임(Run Time)에서 실행된 워크케이스의 실행경로를 비교/분석하여 새로운 워크플로우 모델의 제시 또는 작성된 비즈니스 프로세스의 개선을 위한 작업으로서 액티비티(activity) 의존성을 이용한 워크케이스 마이닝에 관하여 기술하겠다. 제안된 워크케이스 마이닝을 위하여 각 액티비티의 의존성을 분석하여 중요경로(Essential Path)를 결정하는 액티비티 의존 넷 알고리즘(Activity-Dependent Net Algorithm)과 각 워크케이스 실행 정보를 이용하여 중요 경로와 일치여부를 결정하는 실행계열 분석 알고리즘(E-Walk Series Analysis Algorithm)을 제안한다.

논문의 구성은 다음과 같다. 2장에서는 이론적 배경으로 워크플로우 마이닝과 관련된 일반적인 사항에 대하여 기술하였고, 3장에서는 워크플로우 마이닝 프레임웍에 대하여 기술한다. 그리고 4장에서는 본 논문에서 제안된 알고리즘을 기술하고 이와 같은 알고리즘을 이용하여 처리되는 과정의 예를 설명한다. 마지막으로 5장에서 결론 및 향후 연구과제에 대하여 기술하겠다

2. 이론적 배경

2.1 워크플로우 마이닝 개념

지난 10여년 동안 많은 상용 워크플로우 관리 시스템 - IBM MQSeries, COSA, Stafware 등등 -은 다양한 기업 정보기술 분야에 적용되어 사용되어 왔다. 그러나 대부분의 워크플로우 관리 시스템 제품들이 처리해야 할 비즈니스 프로세스의 복잡화, 대규모화, 매우 긴 실행시간(very long execution time) 등과 같은 조건에 만족하기 위한 유연성(flexibility)과 동적 변경(Dynamic Change) 등을 처리하는데 있어서 많은 문제점을 나타내고 있다[3,4].

현재 대부분의 워크플로우 관리 시스템은 빌드-타임에 정의된 비즈니스 프로세스를 이용하여 런-타임에서 실행시키는 과정이다. 그러나 자의든 타의든 어떠한 이유로 두 단계의 정보의 모순 또는 불일치로 인하여 워크플로우의 실행이 더 이상 불가능할 때 이런 문제를 처리할 수 있는 유연성을 제공치 못하고 있다. 또한 워크플로우 프로세스에 대한 실행 정보 및 부하 정보를 활용하여 진행중인 처리 프로세스의 속성 - 종료예정일, 단위업무 담당자, 종료 단위업무 등 -과 해당 처리 프로세스에 포함된 단위업무를 변경할 수 있는 동적 변경 기능을 제공해야 한다. 즉, 하나의 프로세스가 워크플로우 시스템에서 수행되었을 경우, 이들의 프로세스 수행 내용이 실행 중에 변화될 필요성이 발생할 수 있다는 것을 의미하고 있다.

비즈니스 프로세스를 정의하기 위한 빌드-타임이 아닌 런-타임에서 정의된 프로세스가 변경되기 위해서는 프로세스의 흐름정의와 관련된 모든 정보들이 프로세스 모델과 처리 프로세스에서 별도로 관리되어야 한다. 이와 같은 동적 변경을 지원하는 워크플로우 관리시스템에는 예외적인 상황이 발생할 경우에 사용자나 관리자의 요청에 의해서 프로세스의 흐름을 변경하여 업무를 처리할 수 있는 기능을 제공하지 못하고 있다. 유연성과 관련된 문제, 즉 빌드-타임과 런-타임 정보의 불일치로 발생될 수 있는 문제를 해결하기 위하여 프로세스의 실행된 정보를 이용하여 빌드-타임의 프로세스 정의에 이용하는 방법(reverse the process)이나, 워크플로우 시스템의 런-타임 시점에 빌드-타임에서 정의된 프로세스가 변경되기 위해서는 프로세스의 흐름정의 및 실행된 모든 정보를 알기 위한 방법으로 워크플로우 마이닝이라는 개념이 필요하게 되었다[3,4,5,6].

워크플로우 마이닝 개념은 기업 또는 조직의 프로세스 개선이라는 문제와도 직결될 수 있는 사항이다. 프로세스 개선은 두 가지 관점에서 고

려할 수 있는데, 첫 번째는 처리가 완료된 프로세스의 통계자료를 분석하고, 처리 지연요인과 상황을 파악하는 통계적 관점에서의 프로세스 개선과 두 번째는 처리 진행중인 프로세스가 워크플로우 시스템에 의해서 자동으로 적절하게 변경되는 실시간 관점에서의 프로세스 개선이 존재한다.

전자의 경우에는 워크플로우 시스템이 제공하는 기능의 측면보다는 워크플로우 시스템의 부가적인 활용 방안에 가까우며, 개선을 위한 처리 내용도 업무 처리 프로세스의 재구성 및 업무 효율 증가에 필요한 리소스 충원 등의 시스템 외적인 요인이 대부분이다.

반면 후자의 경우는 워크플로우 시스템이 프로세스를 처리하는 도중에도 프로세스의 개선을 위해서 시스템이 적절한 조치를 취하는 것으로, 필요에 따라 프로세스 우선순위를 적절히 조절하며, 집중되는 업무를 여러 참여자에게 분배 시키는 기능들을 제공하는 시스템의 기능이다. 이와 같은 워크플로우 마이닝의 개념을 정리하여 보면 다음 표 1과 같다.

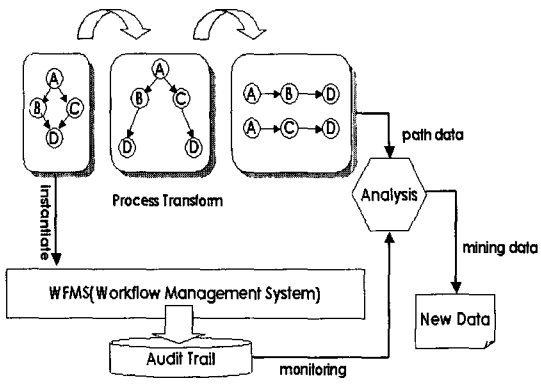
이와 같은 워크플로우 마이닝의 개념을 도식화하면 그림 1에 나타난 것과 같다. 비즈니스 프로세스의 실행경로를 마이닝하여 작성된 비즈니스 프로세스의 개선 또는 새로운 비즈니스 프로세스를 생성하기 위한 목적으로 진행되는 개략적인 마이닝 과정이다.

이러한 목적을 달성하기 위한 그림 1에서 보여지듯이 마이닝 과정은 정의된 비즈니스 프로세스와 실제 실행된 인스턴스의 실행 경로를 비교 분석하는 과정이다. 즉, 빌드타임에서 정의된 비즈니스 프로세스를 분석하여 모든 가능한 경로를 구한다. 한편 정의된 비즈니스 프로세스가 워크플로우 엔진에서 하나의 인스턴스로 발생되고 실행된 정보를 이용하여 구해진 실제 실행된 경로의 정보를 구한다. 이렇게 얻어진 두 정보를 비교, 분석하여 새로운 실행경로에 대한 유용한 정보를 구할 수 있다[10].

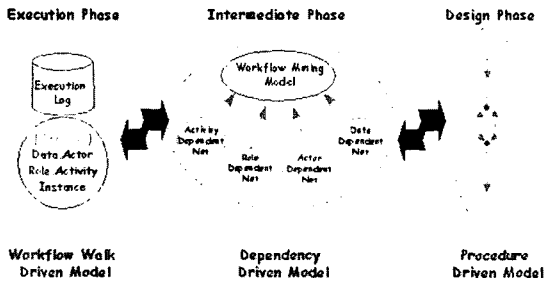
이와 같이 얻어진 정보를 이용하여 현재 실행된 비즈니스 프로세스를 개선 또는 새로운 비즈니스 프로세스를 작성하는 경우 유용하게 사용될 수 있다.

(표 1) 워크플로우 마이닝 개념

사용자에 의해 정의된 워크플로우 프로세스를 시스템을 통해 실행된 정보를 바탕으로 현재 실행된 프로세스를 평가하고, 기존의 프로세스를 재구성하거나 새로운 워크플로우 프로세스를 생성할 때 도움을 줄 수 있는 데이터를 추출하는 과정



(그림 1) 워크플로우 마이닝 개념도



(그림 2) Abstract Workflow Mining Model

2.2 추상적 워크플로우 마이닝 모델

워크플로우 관리시스템에서 실행정보를 이용하여 일련의 과정을 거쳐 유용한 의미의 정보 - 비즈니스 프로세스의 개선, 조직 또는 기업의 자원의 활용도 등 -를 추출해내는 과정인 워크플로우 마이닝 개념은 그림 2와 같이 3단계의 추상화된 모델 - *Workflow Walk Driven Model, Dependency Driven*

Model, Procedure Driven Model - 로 표현할 수 있다.

*Workflow Walk Driven Model*은 워크플로우 시스템의 엔진에서 발생된 인스턴스가 실행되면서 워크플로우 마이닝을 위한 데이터인 워크플로우 실행정보를 발생되는 단계이다. 이 때 발생되는 데이터는 뒤에서 설명할 것과 같이 워크플로우 마이닝의 목적에 따라 사용되어질 데이터가 상이할 것이다. 예를 들어 실행되고 있는 데이터의 사용빈도 또는 발생되고 실행되고 있는 인스턴스 내의 각 액티비티의 수행시간 또는 각 액티비티의 실행경로 등과 같은 정보가 실행정보(Execution Log)에 저장될 수 있을 것이다.

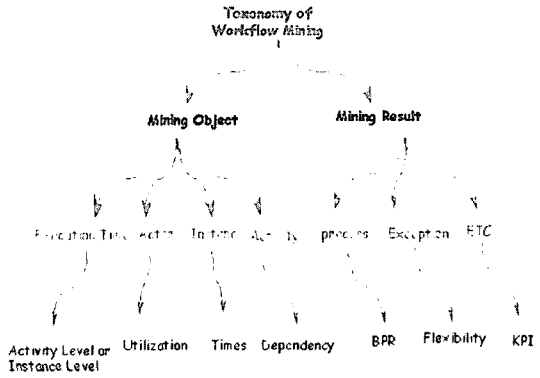
이와 같이 저장된 정보는 워크플로우 마이닝을 위한 원시데이터가 될 것이다. 현재 실행된 인스턴스의 모든 정보를 가장 정확하게 표현한 실행정보와 같은 원시데이터는 워크플로우 마이닝을 위한 가장 중요한 요소이므로 어떤 목적으로 마이닝을 하느냐에 따라 실행된 정보로부터 저장될 요소가 정의될 수 있다.

*Dependency Driven Model*은 워크플로우 마이닝을 위한 사전단계로서 *Workflow Walk Driven Model*에서 발생된 실행 정보인 원시데이터 워크플로우 마이닝 모델에 따라서 각 의존성을 분석하여 저장하는 단계이다

*Procedure Driven Model*은 *Dependency Driven Model*에서 저장된 데이터를 이용하여 워크플로우 마이닝 정보를 비즈니스 프로세스의 개선 또는 새로운 비즈니스 프로세스의 정의에 적용되는 단계이다. 즉, 워크플로우 마이닝을 통하여 얻어진 유용한 정보를 이용하여 자원의 재배치, 비즈니스 프로세스의 변경, CRM 구축 등의 작업을 통하여 워크플로우 마이닝 정보를 이용하는 단계이다.

2.3 워크플로우 마이닝 구분

워크플로우 마이닝의 분류를 위해서 본 논문에서는 두 가지의 항목을 이용하여 구분하였다. 첫 번째 항목은 마이닝 대상에 따른 분류로서 워크



(그림 3) 워크플로우 마이닝의 분류

플로우의 실행정보 중에서 어떠한 대상을 마이닝하느냐에 따른 구분이다. 두 번째 항목은 워크플로우 마이닝 목적에 따른 분류이다. 즉 워크플로우 마이닝의 목적이 프로세스의 개선을 위한 목적이냐 또는 예외상황을 처리하기 위한 워크플로우 마이닝인가 또는 성능 테스트를 위한 척도(KPI :Key Performance Indicator)인가에 따라 분류될 수 있다.

이와 같은 워크플로우 마이닝의 분류는 그림 3과 같다.

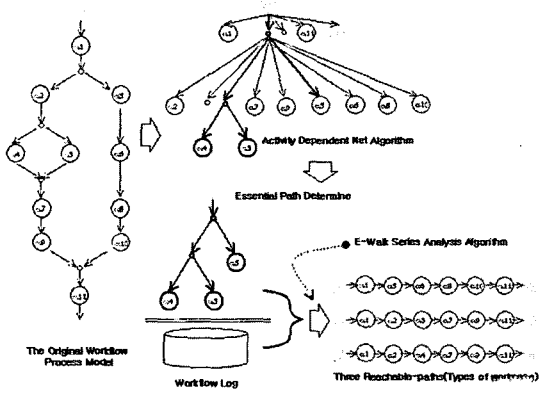
워크플로우 마이닝의 대상이란 실행정보로 발생 된 정보 중에서 어떤 정보를 이용하여 워크플로우 마이닝을 수행 할 것인가에 대한 구분이다. 각 의미는 다음과 같다.

- Execution Time : 수행시간의 분석은 워크플로우 자원관리 측면에서 매우 중요한 요소로서 수행시간의 분석 대상은 프로세스에 정의되어 있는 각각의 액티비티가 될 수도 있고, 발생된 하나의 인스턴스의 수행시간이 될 수도 있다.
- Actor : 프로세스 처리과정에 참여한 참여자의 수행 결과를 분석하고 업무 프로세스 내의 업무가 집중되는 부분과 그렇지 않은 부분들을 파악하여 병목현상의 원인을 분석할 수 있다.
- Instance : 프로세스의 수행과 관련되어 발생된 인스턴스의 수를 이용하여 수행과정의 문제점과 현상을 파악할 수 있다.

- Activity : 프로세스 처리를 위하여 발생된 각 인스턴스에서 실행된 액티비티 간의 의존성을 이용하여 실행되고 있는 프로세스의 재 설계 또는 새로운 프로세스의 설계의 목적으로 이용할 수 있다.

3. 워크플로우 마이닝 프레임웍

본 논문의 목적은 정의된 워크플로우 모델의 액티비티의 의존성을 분석하여 생성된 중요 경로와 실행경로와의 비교/분석을 통하여 수행 가능한 경로와 작성된 모델의 재 설계에 정보를 제공하고자 한다. 이와 같은 목적을 수행하기 위한 프레임웍은 아래 그림 4와 같이 3단계의 변환과정을 수행한다. 첫 번째 변환과정은 ICN으로부터 액티비티의 의존성을 고려하여 액티비티 의존넷으로 변환하는 것이고, 두 번째 변환과정은 액티비티 의존넷으로부터 중요경로만을 찾아내는 과정으로 변환하는 것이고 마지막 세 번째 변환과정은 실행정보를 이용하여 실행 가능한 경로 즉, 워크케이스 마이닝 과정을 수행하는 것이다.



(그림 4) 워크플로우 마이닝 프레임웍

4. 액티비티 의존성을 이용한 워크케이스 마이닝

액티비티 의존성을 이용한 워크케이스 마이닝이란 빌드타임에 정의된 비즈니스 프로세스 모델의 정보로부터 각 액티비티간의 의존성의 분석을 통하여 정적으로 정의된 중요 경로(Essential Path)와 워크플로우 엔진에서 실행된 워크케이스의 실행 정보를 이용하여 빌드타임에 정의된 모든 경로와 비교함으로써 도달 가능한 모든 경로 중에서 작성된 비즈니스 모델이 어떤 경로로 진행되는지를 살펴봄으로써 정의된 모델의 개선 여부와 새로운 모델의 작성 시 이용할 수 있는 정보를 작성하는 과정을 의미한다.

4.1 중요경로 결정을 위한 액티비티 의존넷 알고리즘

워크케이스 마이닝 예제의 첫 번째 단계로서 빌드타임에서 정의된 비즈니스 프로세스의 실행 가능한 모든 경로를 알아보기 위하여 비즈니스 프로세스에 정의된 액티비티의 의존성을 조사하여 각 액티비티간의 의존성 관계를 액티비티 의존 넷(Activity-Dependency Net)으로 표현한다.

이와 같이 구성된 액티비티 의존 넷은 정의된 비즈니스 프로세스의 제어경로 즉, 조건 분기수행 또는 병렬수행, 순차실행과 같은 정의된 워크플로우 모델의 실행경로를 찾을 수 있는 기능을 제공한다. 또한 액티비티 의존 넷을 통하여 각 액티비티의 전이 조건을 표현할 수 있고 이와 같은 전이 조건은 각 액티비티의 수행순서 뿐 만 아니라 실행 중 작성된 모델의 동적 변경을 위한 타당성 자료로도 될 수 있다. 또한 액티비티 의존 넷의 최종의 결과로 나타난 그래프를 이용하여 정의된 비즈니스 모델의 중요경로(Essential Path)를 찾아낼 수 있다.

정의된 비즈니스 프로세스가 수행될 수 있는 모든 각각의 경로 또는 워크케이스를 알아보기 위하여 각 워크케이스에서 수행되는 모든 액티비티를 진행과정으로 표현할 수 있다. 즉, 정의된 비즈니스 프로세스의 시작 노드부터 종료 노드까지 각각의 워크케이스에 참여하는 모든 액티비티

를 이용하여 표현하는 방법이다.

그러나 현재와 같이 표현되어야 할 비즈니스 프로세스가 복잡하고, 많은 액티비티를 필요로 하는 상황에서 모든 액티비티를 이용하여 각 워크케이스를 표현하는 것 보다 각 워크케이스를 구분할 수 있는 액티비티를 이용하여 표현하는 것이 효과적이다.

이와 같은 이유에서 본 논문에서는 정의된 비즈니스 프로세스 모델을 이용하여 액티비티 의존 넷(Activity Dependency Net)을 구성하고 그 결과로부터 중요경로(Essential Path)를 구하고자 한다.

액티비티 의존 넷은 확장된 ICN Model에서 δ(제어 경로(Control Flow) 부분)로부터 생성된다[7,8,9].

- 정 의 (Definition)

액티비티 의존 넷은 일반적으로 액티비티의 집합 A와 전이조건의 집합 T에 대하여 $\Omega = (\varphi, \xi, S, E)$로 정의된다.

- $\varphi = \varphi_i \cup \varphi_o$

  $\varphi_o : A \rightarrow \psi(A)$는 각 액티비티에 후행 제어 의존성이 있는 액티비티 집합을 의미하며 $\varphi_i : A \rightarrow \psi(A)$는 하나의 액티비티에 선행 제어 의존성이 있는 액티비티 집합을 의미한다.

- $\xi = \xi_i \cup \xi_o$

  ξ_i는 하나의 액티비티에 대하여 선행 제어 전이조건의 집합을 의미하며, ξ_o은 하나의 액티비티에 대하여 선행 제어전이조건의 집합을 의미한다.

- S는 초기 제어전이조건의 유한 집합으로, 워크플로우 실행 전 외부 프로세스에 의해 정보가 적재된다고 가정한다.

- E는 최후의 제어전이조건의 유한 집합으로, 제어 정보는 외부의 프로세스에서 이용될 수 있다.

ICN으로 표현된 비즈니스 프로세스 모델에서 액티비티 의존 넷의 정형화된 의미를 위하여 액티비티 노드로 들어오는 실선으로 표현된 화살표는 φ_i , ξ_i로 표현되고, 액티비티 노드로부터 나가는 것과 같이 표현된 실선의 화살표는 φ_o, ξ_o 표현 할 수 있다.

- 액티비티 의존 넷 알고리즘 (Activity Dependency-Net Algorithm)

액티비티 의존 넷은 확장 ICN(EICN : Extended Information Control Net) 모델로부터 액티비티 의존 넷 알고리즘을 통하여 구할 수 있다.

액티비티 의존 넷 알고리즘을 정의하기 위하여 "제어 강의존성"(Strongly Control Dependent)과 개념을 정의하여야 한다. 즉, 각 액티비티의 의존성 중에서 어떤 액티비티가 강한 의존성 관계를 갖고 있는지를 구분해야 이후에 중요경로를 구할 수 있는 기능을 제공한다.

Γ는 확장 ICN이고 u, v가 액티비티 집합 A에 포함되었을 경우 즉, $u \in A$이고 $v \in A$과 같을 때, 다음과 같은 경우에 v-u 경로가 존재하고, vWu가 v의 전방지배(Forward Domination)가 존재하지 않는 경우에 액티비티 u는 액티비티 v에 제어 강의존성(Strongly Control-Dependent) 관계가 있다라고 할 수 있다

다음 (표 2)는 액티비티 의존 넷 알고리즘을 나타내고 있다

(표 2) 액티비티 의존 넷 알고리즘

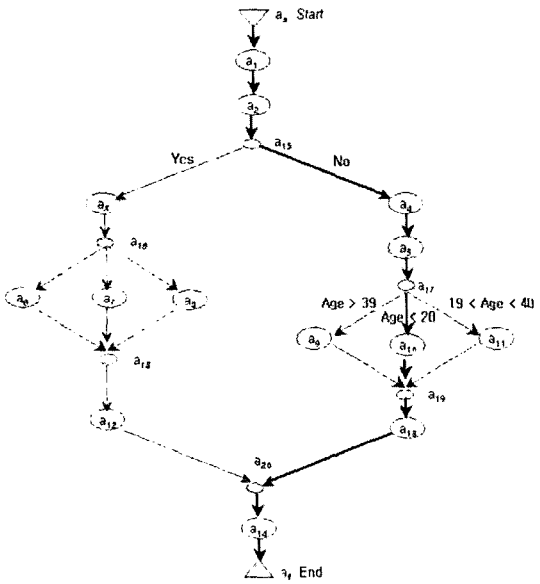
```
INPUT : An Extended ICN
OUTPUT : An Activity-Dependent Net

BEGIN
  FOR all x∈A and y∈A in an Extended ICN
      //OR or LOOP activity
      IF x is strongly control dependent on y
        //get a depedent flow
        ADD x TO φo(y) ;    ADD y TO φi(x);
        //get control transition conditions between y and x
        ADD ki(x) TO ξo(y);  ADD ki(x) TO ξi(x);
      ELSEIF fd(y) ∈ x AND (fd(y) ⊄ fd(ibd(y)) OR ibd(y) = Ø)
        //get a dependent flow
        ADD x TO φo(y);      ADD y TO φi(x);
        //get control transition conditions between y and x
        ADD ki(x) TO ξo(y);  ADD ki(x) TO ξi(x);
      ENDIF
  NEXT
END
```

위의 알고리즘에서 나타나듯이 각 액티비티의 중에서 분기된 조건 중에서 하나를 선택하는 OR 제어전이조건과 제어전이조건 자체로서 액티비티의 집합으로 이루어진 LOOP 제어전이조건 이후에 나타나는 액티비티는 제어전이조건에 대하여 강의존성 관계가 성립됨을 알 수 있다.

4.2 주문처리 워크플로우에서의 액티비티 의존 넷 예

표 2에서 표현한 알고리즘을 이용하여 액티비티 의존 넷을 나타내기 위하여 다음 그림 5와 같은 예제 비즈니스 프로세스가 있을 경우 그림 5에 대한 액티비티 의존 넷의 그래픽 표현은 그림 6과 같이 나타낼 수 있다. 그림 5의 예제 비즈니스 프로세스는 20개의 액티비티를 갖는 프로세스로서 a_{15}에서 OR 분기를 통하여 경로가 결정될 수 있다. 또한 a_{15} 분기에서 No라는 조건이 선택되어 분기를 하면 a_{17}의 분기 조건에 실행 경로가 결정될 수 있는 확장 ICN의 예를 나타내고 있다.



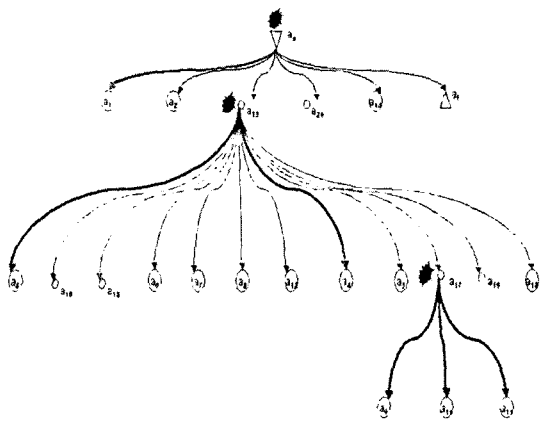
(그림 5) 주문처리 워크플로우 모델

그림 5에 표현된 주문처리 워크플로우 모델을 액티비티 의존넷으로 나타내면 다음 그림 6같다.

그림 6에서 나타나듯이 7가지 - *default, OR(yes), OR(no), AND(default), OR(age<20), OR(19<age<40), OR(age>39)* -의 제어 전이 조건이 나타난다. *default*란 의미는 각 액티비티 사이에 시간적 순서에 따라 진행되는 일반적인 경우를 의미한다. 그러므로 액티비티 a_s, a_1, a_2, a_{15}, a_{20}, a_{14}, a_f의 실행순서는 시작노드인 a_s에 강 종속되어 있다.

*OR(yes), OR(no)*는 이후에 나타나는 액티비티에 대하여 실행경로를 결정할 수 있는 제어 전이 조건이다. 그러므로 a_3, a_4, a_5, a_6, a_7, a_8, a_{12}, a_{13}, a_{16}, a_{17}, a_{18}, a_{19}는 a_{15}에서 *OR(yes), OR(no)*의 조건에 강 종속되어 있다.

또한 *OR(age<20), OR(19<age<40), OR(age>39)*는 a_{15}에서 *OR(no)*가 선택된 경우에 발생될 수 있는 제어 전이 조건이다. 즉, a_{15}에서 *OR(no)*가 선택되었을 경우 a_{17}의 *OR(age<20), OR(19<age<40), OR(age>39)* 조건에 따라 a_9, a_{10}, a_{11}이 결정될 수 있으므로 a_9, a_{10}, a_{11}은 a_{17}에 강 종속되었음을 알 수 있다.



(그림 6) 주문처리 프로세스의 액티비티 의존 넷 그래프

- 액티비티 의존 넷의 정형적 표현 (Formal Specification for the Activity-Dependency Net) - 그림 6에서 표현된 액티비티 의존 넷의 정형적 명세는 아래의 표 3과 같다.

(표 3) 액티비티 의존 넷의 정형적 표현

$\Omega = (\varphi, \xi, S)$ *over* A, T //Activity Dependent Net
A={a_s,a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_{10},a_{11},a_{12},a_{13},a_{14},a_{15},a_{16},a_{17},a_{18},a_{19},a_{20},a_f} //activity
T={default, or1(예약='Yes'),or1(예약='No'),and(default),or_2(age<'20'), or_2('19'<age<'40'), or_2(age>'39')} //transition
S = { Ø }
E = { Ø }

$\varphi_i(a_s)$={Ø}, $\varphi_o(a_s)$={a_1,a_2,a_{14},a_{15},a_{20},a_f}	$\xi_i(a_s)$={Ø}, $\xi\ \varphi_o(a_s)$={d}
$\varphi_i(a_1)$={a_s}, $\varphi_o(a_1)$={Ø}	$\xi\ \varphi_i(a_1)$={d}, $\xi\ \varphi_o(a_1)$={Ø}
$\varphi_i(a_2)$={a_s}, $\varphi_o(a_2)$={Ø}	$\xi\ \varphi_i(a_2)$={d}, $\xi\ \varphi_o(a_2)$={Ø}
$\varphi_i(a_3)$={a_{15}}, $\varphi_o(a_3)$={Ø}	$\xi\ \varphi_i(a_3)$={or_1(예약='No')}, $\xi\ \varphi_o(a_3)$={Ø}
$\varphi_i(a_4)$={a_{15}}, $\varphi_o(a_4)$={Ø}	$\xi_i(a_4)$={or_1(예약='Yes')}, $\xi_o(a_4)$={Ø}
$\varphi_i(a_5)$={a_{15}}, $\varphi_o(a_5)$={Ø}	$\xi_i(a_5)$={or_1(예약='Yes')}, $\xi_o(a_5)$={Ø}
$\varphi_i(a_6)$={a_{15}}, $\varphi_o(a_6)$={Ø}	$\xi_i(a_6)$={and(default)}, $\xi_o(a_6)$={Ø}
$\varphi_i(a_7)$={a_{15}}, $\varphi_o(a_7)$={Ø}	$\xi_i(a_7)$={and(default)}, $\xi_o(a_7)$={Ø}
$\varphi_i(a_8)$={a_{15}}, $\varphi_o(a_8)$={Ø}	$\xi_i(a_8)$={and(default)}, $\xi_o(a_8)$={Ø}
$\varphi_i(a_9)$={a_{17}}, $\varphi_o(a_9)$={Ø}	$\xi_i(a_9)$={ or_2(age>'39'), $\xi_o(a_9)$={Ø}
$\varphi_i(a_{10})$={a_{17}}, $\varphi_o(a_{10})$={Ø}	$\xi_i(a_{10})$={or_2(age<'20')}, $\xi_o(a_{10})$={Ø}
$\varphi_i(a_{11})$={a_{17}}, $\varphi_o(a_{11})$={Ø}	$\xi_i(a_{11})$={or_2('19'<age<'40')}, ξo(a11)={Ø}
$\varphi_i(a_{12})$={a_{15}}, $\varphi_o(a_{12})$={Ø}	$\xi_i(a_{12})$={or_1(예약='No')}, $\xi_o(a_{12})$={Ø}
$\varphi_i(a_{13})$={a_{15}}, $\varphi_o(a_{13})$={Ø}	$\xi_i(a_{13})$={or_1(예약='Yes')}, $\xi_o(a_{13})$={Ø}
$\varphi_i(a_{14})$={a_s}, $\varphi_o(a_{14})$={Ø}	$\xi_i(a_{14})$={d}, $\xi_o(a_{14})$ = {Ø}
$\varphi_i(a_{15})$={a_s}, $\varphi_o(a_{15})$={a_3,a_4,a_5,a_6,a_7,a_8,a_{12},a_{13},a_{16},a_{17},a_{18},a_{19}}	$\xi_i(a_{15})$={d}, $\xi_o(a_{15})$={or_1(예약='Yes'),or_1(예약='No')}
$\varphi_i(a_{16})$={a_{15}}, $\varphi_o(a_{16})$= {Ø}	$\xi_i(a_{16})$={or_1(예약='No')}, $\xi_o(a_{16})$={and(default)}
$\varphi_i(a_{17})$={a_{15}}, $\varphi_o(a_{17})$={a9,a_{10},a_{11}}	$\xi_i(a_{17})$={or_1(예약='Yes')},
$\varphi_i(a_{18})$={a_{15}}, $\varphi_o(a_{18})$={Ø}	$\xi_o(a_{17})$={or_2(age<'20'),or_2('19'<age<'40'), or_2(age>'39')}
$\varphi_i(a_{19})$={a_{15}}, $\varphi_o(a_{19})$={Ø}	$\xi_i(a_{18})$={or_1(예약='No')}, $\xi_o(a_{18})$={Ø}
$\varphi_i(a_{20})$={a_s}, $\varphi_o(a_{20})$={Ø}	$\xi_i(a_{19})$={or_1(예약='Yes')}, $\xi_o(a_{19})$={Ø}
$\varphi_i(a_f)$={a_s}, $\varphi_o(af)$={Ø}	$\xi_i(a_{20})$={d}, $\xi_o(a_{20})$={Ø}
	$\xi_i(a_f)$={d}, $\xi_o(a_f)$={Ø}

4.3 중요경로의 결정

작성된 액티비티 의존 넷을 이용하여 수행될 수 있는 모든 경로를 구할 수 있다. 이때 수행에 참여하는 모든 액티비티를 고려하여 기술하지 않고 가장 중요한 즉, 실행경로를 판단할 수 있는 액티비티만을 이용하여 수행 경로를 나타내는 것을 중요경로(Essential Path)라고 한다. 이와 같이 중요경로를 찾아내는 이유는 처리해야 할 비즈니스 프로세스가 복잡해지고 규모가 커짐에 따라 모든 액티비티를 이용하여 런타임에서 발생된 실행경로와 비교하는 것은 많은 시간을 필요로 할 수 있다. 이와 같은 이유로 실행에 참여하는 모든 액티비티를 고려하는 것이 아니라 수행 경로를 표현하는 모든 액티비티 중에서 영향력 있는 액티비티(Very Efficient Activity)를 추출하는 것이 중요하다.

일반적으로 표현된 비즈니스 프로세스의 수행 경로에 영향을 주는 OR 노드와 LOOP 노드인 경우이다. 순차적인 경우는 시간적 순서에 따라 진행되고 AND 노드인 경우에는 모든 액티비티를 수행해야 함으로 수행 경로를 결정하는 요소가 될 수 없다. 이러한 이유로 OR 노드 또는 LOOP 노드 뒤에 수행되는 액티비티는 실행 경로를 결정할 수 있는 영향력 있는 액티비티이다.

그림 6에서 굵은 실선으로 연결된 액티비티를 영향력 있는 액티비티라 할 수 있고 이러한 영향력 있는 액티비티를 이용하여 중요경로(Essential Path)를 찾을 수 있다. 즉, 그림 6을 영향력 있는 액티비티를 이용하여 간략하게 표현하면 다음과

같이 4가지의 경로로 나타낼 수 있다.

- Path 1 : {a_s, a_1, a_{15}, a_3, a_f}
- Path 2 : {a_s, a_1, a_{15}, a_4, a_{17}, a_9, a_f}
- Path 3 : {a_s, a_1, a_{15}, a_4, a_{17}, a_{10}, a_f}
- Path 4 : {a_s, a_1, a_{15}, a_4, a_{17}, a_{11}, a_f}

4.4 실행계열 분석 알고리즘

워크케이스 마이닝을 위한 마지막 단계로서 하나의 워크케이스가 워크플로우 엔진에서 수행되고 그 결과와 빌드타임에서 정의된 수행 가능한 모든 경로와 비교하여 어떠한 경로를 통하여 수행 되었는지를 알아보기 위하여 새로운 알고리즘이 필요하다.

즉, 빌드타임에서 정의된 수행 가능한 경로들 중에서 각각의 워크케이스가 어떤 실행경로를 통하여 수행되었는지는 비즈니스 프로세스에 참여하고 있는 자원들을 효율적으로 이용하기 위한 필요한 정보를 제공한다. 이러한 기능을 수행하는 알고리즘을 실행계열분석 알고리즘이라 하고 다음 표 4와 같이 나타낼 수 있다.

"Execution Path Log"는 각 워크케이스가 워크플로우 엔진에서 실제 실행된 경로를 의미하고 "Essential Path"는 빌드타임에 정의된 워크플로우 모델의 액티비티의 의존성을 이용하여 표현된 수행가능한 모든 경로의 중요경로를 나타낸다.

4.5 워크케이스 마이닝 예제

표 4에서 제시된 알고리즘을 이용한 워크케이스 마이닝 실행과정의 예를 살펴보기 위해서는 워크플로우 엔진에서 실제 실행된 워크케이스의 경로가 필요한데 그림 5와 같이 정의된 워크플로우 모델을 실행하였을 경우 한 워크케이스에서 {a_1, a_2, a_{15}, a_4, a_5, a_{17}, a_{10}, a_{19}, a_{13}, a_{20}, a_{14}}와 같이 실행되었다고 가정하고 4.2에서 기술된 4개의 중요경로 - {a_1, a_{15}, a_3}, {a_1, a_{15}, a_4, a_{17}, a_9}, {a_1, a_{15}, a_4, a_{17}, a_{10}}, {a_1, a_{15}, a_4, a_{17}, a_{11}} - 와 비교하는 과정을 설명하겠다.

(표 4) 실행계열분석 알고리즘

```
#DEFINE 중요경로의 수  Number_of_Path
#DEFINE 액티비티의 수 Number_of_Activity
INPUT : Execution Path Log(EPL[Number_of_Activity]) ,
        Essential Path(EP[Number_of_Path ][Number_of_Activity])
OUTPUT : Boolean
BEGIN
   FOR a_cnt = 1 TO Number_of_Activity
        IF  EPL[a_cnt] = Execution Activity THEN
                masking 1
        ELSE
                masking 0
        ENDIF
   NEXT // Execution Path Log Assign into EPL[]
END
BEGIN
   FOR cnt = 1 TO Number_of_Path
        FOR a_cnt = 1 TO Number_of_Activity
          IF (EP[cnt][a_cnt] XOR (EPL[a_cnt] AND EP[cnt][a_cnt]))
            == 0 THEN
              //Match
              Call Return_Process();
          ENDIF
        NEXT // Essential Path Assign into EP [ ][ ]
   NEXT // Number of Workcase
END
```

워크케이스의 실제 수행경로와 빌드타임에 정의된 모델을 이용하여 중요경로로 얻어진 것 중 하나인 {a_1, a_{15}, a_3}의 마이닝 과정은 다음과 같다. 다음의 그림 7은 워크플로우 엔진에서 실제 수행된 실행정보를 나타내고 있고 즉, 20개의 액티비티에서 수행된 액티비티는 1로, 그렇지 않은 액티비티는 0으로 표현한 것이다. 그림 8은 빌드타임에 정의된 모델을 이용하여 중요경로로 얻어진 것 중 하나인 {a_1, a_{15}, a_3}을 표현하고 있다.

첫 번째 과정으로 두 실행경로에 대하여 AND 연산을 수행한다. 그 결과는 그림 9와 같고 다음으로 두 경로의 일치여부를 확인하여 위하여 AND 연산 결과 -그림 9 -와 액티비티 의존넷을 이용하여 빌드타임에서 얻어진 첫 번째 중요경로 -그림 8-와 Exclusive OR 연산을 수행하여 그 결

(그림 7) 엔진에서 수행된 워크케이스의 실제 수행경로

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0	1	1	0	0	0	0	1	0	0	1	1	1	0	1	0	1	1

(그림 8) 액티비티 의존넷을 이용하여 빌드타임에서 얻어진 첫 번째 중요경로

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

(그림 9) AND 연산 수행결과

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

(그림 10) Exclusive OR 연산 수행결과

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

과가 0 값을 갖으면 두 수행경로는 일치한다는 것을 알 수 있고 현재 워크플로우 엔진에서 수행된 워크케이스는 $\{a_1, a_{15}, a_3\}$의 경로로 수행되었음을 알 수 있다. Exclusive OR 연산의 수행결과는 그림 10과 같다.

그림 10과 같이 최종 결과를 살펴보면 3번째 bit에서 1이 마스킹 되었으므로 엔진에서 현재 수행된 워크케이스의 실행경로와 빌드타임에서 얻어진 중요경로 중 $\{a_1, a_{15}, a_3\}$과는 다르게 수행되었음을 알 수 있다.

이와 같은 작업을 모둔 중요경로에 각각 적용하여 수행하여 보면 최종적으로 현재 엔진에서 수행된 워크케이스는 $\{a_1, a_{15}, a_4, a_{17}, a_{10}\}$과 같은 경로로 수행 되었음을 알 수 있다.

5. 결 론

워크플로우 마이닝이 워크플로우 및 비즈니스 프로세스 연구 분야에서 중요한 연구 주제로 인식됨에 따라 본 논문에서는 워크플로우 마이닝의 개념과 마이닝 대상에 따른 워크플로우 마이닝의 구분 하였다. 또한 워크플로우 엔진에서 실행된 워크케이스의 실행경로를 마이닝을 위하여 빌드타임에 정의된 모델로부터 발생할 수 있는 모든 실행경로와 현재 워크플로우 엔진에서 수행되어진 워크케이스의 수행경로를 비교하기 위하여 두 개의 알고리즘을 제안하였다.

이와 같은 워크케이스 마이닝 결과로 얻어진 실행경로는 워크플로우 시스템이 갖고 있는 한계점인 유연성 및 예외상황 처리, 동적변경 등에 대한 해결책을 제시 할 것이라 생각되며, 본 논문에서 제시하였듯이 워크플로우 마이닝 개념은 현재 수행되고 있는 비즈니스 프로세스의 객관적 분석을 통하여 더욱 개선된 비즈니스 프로세스를 작성할 수 있으며, 조직 및 기업의 자원 활용도를 높임으로써 기업 또는 조직의 내부적인 질적 향상과 고객과의 관계개선을 위한 정보(CRM)를 이용하여 외부적인 발전을 할 수 있는 초석을 마련할 수 있을 것이다.

향후 연구과제로는 현재 제안된 알고리즘을 이용하여 시스템을 구현하는 것이다. 구현된 워크플로우 마이닝 시스템을 통하여 비즈니스 프로세스의 개선이 어떻게 이루어지는지 또는 새로운 비즈니스 프로세스에 어떻게 적용되는 지를 나타내어야 한다.

또한 현재 마이닝의 정보를 실시간으로 모니터링 하는 기능인 워크플로우 마이닝 모니터링의

기능에 대해서도 설계를 하여야 할 것이다. 워크플로우 마이닝 모니터링이란 워크플로우 마이닝 시스템에서 생성된 정보를 사용자 - 워크플로우 마이닝을 이용하는 시스템 또는 사람 - 모두에게 어떻게 제공할 것인지와 같은 단순 모니터링 기능 뿐 만 아니라 관리자(Administrator)의 기능도 수행할 수 있도록 기능 정의 및 아키텍처에 관한 부분도 변경이 필요할 것이다.

참고문헌

[1] Frank Leymann and Dieter Roller, "Production Workflow : Concepts and Techniques", 2000 published by Prentice-Hall, Inc

[2] Kim, K. "Architectures for Very Large Scale Workflow Management Systems. PhD Thesis. Computer Science Department, University of Colorado. May 1998

[3] A.J.M.M. Weijters, W.M.P van der Aalst, "Process Mining Discovering Workflow Model from Event-Based Data"

[4] A.J.M.M. Weijters, W.M.P van der Aalst, "Rediscovering Workflow Model from Event-Based Data"

[5] 김학성, "워크플로우 마이닝 프레임웍과 아키텍처에 관한 연구", 박사학위논문, 2003

[6] 김학성, 문기동, 백수기, "워크플로우 마이닝을 위한 실행계열분석 알고리즘", 한국인터넷정보 학회 춘계학술발표논문집, 4권1호, pp. 419~422, 2003. 05

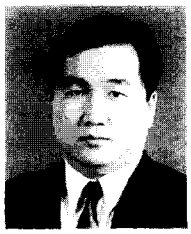
[7] Clarence A.Ellis and Gary J.Nutt, "Modeling and Enactment of Workflow Systems." In Application and Theory of Petri Nets 1993, Proceedings 14th International Conference, Chicago, Illinois, USA, pp. 1~16. Springer-Verlag, 1993.

[8] Clarence A. Ellis and Gary J.Nutt "Adding Actors and Roles to the Basic ICN Model"

[9] Kwang-Hoon Kim and Su-Ki Paik, Actor-Oriented Workflow Model, The Second Cooperative Database Systems for Advanced Applications, Wollongong Australia, March 1999.

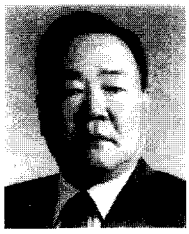
[10] R.Agrawal, D.Gunopulos, and F.Leymann. "Mining Process Models from Workflow Logs", in Proc. of the sixth International Conference on Extending Database Technology(EDBT), 1998.

◐ 저 자 소 개 ◑



김 상 배
1980년 청주대학교 경상대학 경영학과 입학
2001년 경기대학교 일반대학원 전자계산학과 졸업(석사)
2001년~현재 : 경기대학교 일반대학원 전자계산학과 박사과정
관심분야 : Workflow, Computer Network
E-mail : skim@chollian.net

김 학 성
1993년 경기대학교 전자계산학과 졸업(학사)
1995년 경기대학교 대학원 전자계산학과 졸업(석사)
2003년 경기대학교 대학원 전자계산학과 졸업(박사)
1998년~현재 : 동남보건대학 웹컨텐츠개발과 교수
관심분야 : Workflow Mining, CSCW, Workflow Exception Handling, Grid Computing
E-mail : amang@dongnam.ac.kr



백 수 기
1972년 연세대학교 토목공학과 졸업(학사)
1979년 동국대학교 대학원 계산통계학과 졸업(석사)
1992년 동국대학교 대학원 계산통계학과 졸업(박사)
1980년~현재 : 경기대학교 전자계산학과 교수
관심분야 : Computer Network, Workflow, Algorithm
E-mail : skpaik@kyonggi.ac.kr