Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 2, pp.61－70, June 2008.

기업 온톨로지를 활용한 작업흐름 변화 영향 분석

박지현 · 양재군 · 배재학†

울산대학교 컴퓨터 · 정보통신공학부

Workflow Changes Impact Analysis with Enterprise Ontology

Ji-Hyun Park · Jae-Gun Yang · Jae-Hak J. Bae†

Department of Computer Engineering and Information Technology, University of Ulsan

This paper describes case studies on workflow impact analyze with an enterprise ontology (EO). We build the EO with an ontology editor Protégé and integrate concepts of enterprise architecture and a process model into it to expand the EO. We use the expanded EO and Prolog query rules to analysis workflow dependency relations from the perspectives of routing, data and role. Through this, we analyze impact of changes in workflow process. Moreover, we have found the effectiveness of the EO and Protégé in modeling business management and, in particular, workflow representation and management.

Keywords : Enterprise Ontology, Workflow Impact Analysis, Dependency Analysis, Protégé

1. 서 론

1.1 연구 배경

현재 기업은 급변하는 외부환경과 고객의 다양한 요구에 보다 신속하게 대처할 수 있는 경영 시스템을 원하고 있다. 정보기술적인 관점에서 업무환경 변화를 탄력적이고 민첩하게 수용하는 효과적인 업무절차 관리기능의 필요성이 대두되었고 이러한 배경으로 작업흐름 관리 시스템(WfMS : Workflow Management System)의 개발이 시작되었다. 이러한 작업흐름(Workflow)[15]은 다양한 기업요소와 실체들로 구성되어 있고 작업흐름에 변화가 생길 경우 그 구성요소들은 서로 영향을 주게 된다. 현재 업무재설계(BPR : Business Process Reengineering) 등과 같은 사유로 업무절차를 수정할 필요가 있을 때 업무 전문가는 변경시킬 작업흐름 요소를 파악하고 WfMS가 제공하는 작업흐름 모델링 도구를 사용하여 반자동으로 개정한다. 이러한 기존 접근법에서는 작업흐름 변화의 영향을 받는 정보 시스템 요소들을 빠짐없이 확인하고 신속하게 변경하기 어렵다는 약점이 있다. 이러한 반자동식 개정의 약점과 작업흐름 경직성을 극복하기 위하여 적응형 또는 지식기반형 작업흐름 관리기법이 모색되고 있다[17].

적응형 작업흐름 관리기법에서는 업무처리 절차의 대안을 미리 마련하거나 또는 시스템에 재설계 기능을 부여하여 예상해 둔 예외상황을 처리하는 방식을 취한다. 그러나 이 기법은 모든 변화를 예견할 수 없음으로 인하여 발생하는 융통성의 제한과 재설계에 투입되는 시스템 자원의 부담을 고려할 때 아직은 성공적이라고 할 수 없다. 지식기반형 작업흐름 관리기법은 다수의 소프트웨어 대리자(Agent)를 활용하거나 미리 정해둔 작업흐름 개요(Schema)와 함께 규칙(Rule)을 사용하는 방법이다. 규칙은 특정 사건의 발생에 반응하여 정해진 작업의 실행을 규정한다. 이 방법 또한 대리자나 규칙의 다양성과 그 적용범위에 따라 작업흐름의 융통성이 제한된다.

논문접수일 : 2008년 02월 18일 논문수정일 : 2008년 04월 15일 게재확정일 2008년 04월 30일
† 교신저자 jhjbae@ulsan.ac.kr
※ 이 논문은 2005년 울산대학교의 연구비에 의하여 연구되었음.

이러한 연구방향은 업무절차의 실행에 초점을 맞추어 작업흐름의 융통성을 추구한 것이다. 기존연구의 한계를 극복하기 위해서는 WfMS를 업무절차의 실행이 아닌 재설계의 관점에서 볼 필요가 있다. 업무 재설계 관점에서 볼 때 업무절차의 변화에 따라 영향을 받는 업무요소를 예견하는 것은 중요한 문제이다[12, 14]. 작업흐름의 융통성 및 자동화 수준의 제고를 위해서는 변화에 영향을 받는 작업흐름 구성요소들을 확인해 내는 것이 선결과제이다. BPR의 실행과 WfMS의 구현에는 기업고유의 지식이 필요하다. 이에 본 논문에서는 기업의 인지적 모델인 기업 온톨로지의 기반을 구축하고 확장[2, 3]하였다. 또한 이를 활용하여 변화에 영향을 받는 작업흐름 구성요소를 확인[1, 2]하고 프롤로그 질의 규칙을 적용하여 작업 흐름 실체들 간의 의존관계를 파악함으로써 변화에 대한 작업 흐름의 영향을 분석하였다.

1.2 작업흐름

작업흐름 기술 교류 및 표준 제정을 위한 단체인 WfMC (Workflow Management Coalition)에서는 작업흐름(Workflow)을 "업무수행을 위한 일련의 처리절차 및 규칙에 따라 문서, 정보, 업무가 한 업무 참여자에서 다른 참여자로 자동적으로 전달되는 업무 절차의 전체 혹은 부분적인 자동화" 라고 정의한다[15]. 작업흐름은 크게 업무절차(Business Process), 인사조직(Human Organization), 경영정보(Information Structure)로 나눌 수 있고 활동(Activity), 이벤트(Event), 업무 참여자(Actor), 역할(Role), 자원(Resource), 컨트롤 데이터(Control Data), 응용 프로그램(Application) 등 여러 종류의 실체들로 구성되어 있다. 이들은 작업흐름 내에서 서로 다른 역할을 수행하면서 상호작용을 한다. 이러한 실체들 간의 관계 중 가장 일반적인 관계가 한 실체가 다른 실체에 의존하는 것을 의미하는 의존 관계이다[6].

1.3 작업흐름 영향분석

작업흐름 영향분석은 작업흐름의 상태를 분석하는 것으로 특정 프로세스를 수정하는 과정에서 다른 프로세스상에 발생할 수 있는 각종 문제 및 위험성을 사전에 미리 파악하고자 하는 것이다. 이를 통해 프로세스의 수정이나 유지 보수 시 문제점을 미리 예측하여 시스템의 신규 개발 또는 기존 시스템의 유지보수 시간을 절감시키고 업무 프로세스를 안정적으로 관리하고자 하는 것이다.

현재 작업흐름에 대한 연구는 작업흐름 모델링과 검증, 작업흐름 성과 분석, 프로세스 재설계와 작업흐름관리시스템 영역으로 분류된다. 작업흐름 내부 또는 작업흐름 프로세스 간의 의존성 분석이나 영향분석에 대한 연구는 아주 소수이다. 그나마 작업흐름의 모델링 혹은 표현에 초점을 맞추고 있고 라우팅 의존성과 같은 프로세스 구조에 대한 의존관계를 분석하는 것이 대부분이다[6]. Kim[9]은 다양한 관점에서 작업흐름의 의존관계를 분석하였지만 프로세스의 변화와 그들의 영향에 대한 분석을 다루지는 않는다. 또 다른 논문에서 Dai[6]는 소프트웨어 영향분석 기법을 작업흐름 영향분석에 적용 하였다. 이 논문에서는 작업흐름의 변화에 대한 영향분석 방법으로 라우팅과 자료에 중점을 둔 의존성 분석을 선택하고 있고 이를 위해 프롤로그 질의 구조를 사용한다. 작업흐름 내의 모든 실체들과 그들 간의 의존관계를 프롤로그 지식으로 표현하고 그 의존관계를 분석하는 질의규칙을 정의하고 있다. 그러나 일단 모든 실체들과 의존관계가 빠짐없이 정의되어 있어야만 하고 이후에 새로운 실체들이 추가될 경우 그들의 의존관계를 수작업으로 추가해야 한다는 단점이 있다.

본 논문에서는 이런 단점을 보완하고자 기업 고유의 지식을 정의하고 있는 기업 온톨로지를 활용하였다. 실체들 간의 의존관계를 기업 온톨로지의 지식을 탐색하여 추론할 수 있도록 프롤로그 규칙으로 표현함으로써 이후에 추가되는 실체들에 대한 의존관계를 자동으로 정의할 수 있다. 또한 질의를 통해 이를 분석함으로써 변화에 대한 작업흐름의 영향을 분석하고 상호작용하는 다른 구성요소의 변화를 예측한다.

2. 프로티지를 이용한 기업 온톨로지 구축과 확장

2.1 기업 온톨로지

온톨로지(Ontology)는 존재의 본질을 연구하는 형이상학으로, 인지체가 세계를 인식 및 분할하여 얻은 개체와 이들의 속성 및 관련성을 파악한 것의 총화이다[4]. 현재 온톨로지는 정보 검색, 의료 정보, 인공 지능, 전자상거래 등 다양한 기술 분야에서 적용되고 있다.

기업 온톨로지(EO : Enterprise Ontology)[16]는 기업 고유의 용어(Term)와 정의(Definition)의 모음으로 Edinburgh 대학에서 수행한 Enterprise Project를 통해 개발되었다. 크게 다섯 부류로 구성되며 자세한 내용은 다음과 같다 : (1) Meta-Ontology와 Time은 온톨로지 자체를 정의하기 위해 사용되는 용어와 개념을 정의한다. (2) Activity, Plan, Capability와 Resource는 업무절차와 계획에 관계된 것이다. (3) Organization은 조직이 어떻게 구성되었는가에 대한 것이다. (4) Strategy는 기업의 전략에 관련된 것이며 (5) Marketing은 제품과 서비스의 판매, 그리고 마케팅에 관련된 것이다. <표 1>은 Edinburgh 기업 온톨로지 체계이다.

<표 1> Edinburgh EO 체계

Activity	Activity Specification, Execute, Executed Activity Specification, T-Begin, T-End, Pre-Conditions, Effect, Doer, Sub-Activity, Authority, Activity Owner, Event, Plan, Sub-Plan, Planning, Process Specification, Capability, Skill, Resource, Resource Allocation, Resource Substitute
Organization	Person, Machine, Corporation, Partnership, Partner, Legal Entity, Organizational Unit, Manage, Delegate, Management Link, Legal Ownership, Non-Legal Ownership, Ownership, Owner, Asset, Stakeholder, Employment Contract, Share, Share Holder
Strategy	Purpose, Hold Purpose, Intended Purpose, Strategic Purpose, Objective, vision, Mission, Goal, Help Achieve, Strategy, Strategic Planning, Strategic Action, Decision, Assumption, Critical Assumption, Non-Critical Assumption, Influence Factor, Critical Influence Factor, Non-Critical Influence Factor, Critical Success Factor, Risk
Marketing	Sale, Potential Sale, For Sale, Sale Offer, Vendor, Actual Customer, Potential Customer, Customer, Reseller, Product, Asking Price, Sale Price, Market, Segmentation Variable, Market Segment, Market Research, Brand Image, Feature, Need, Market Need, Promotion, Competitor
Time	Time Line, Time Interval, Time Point

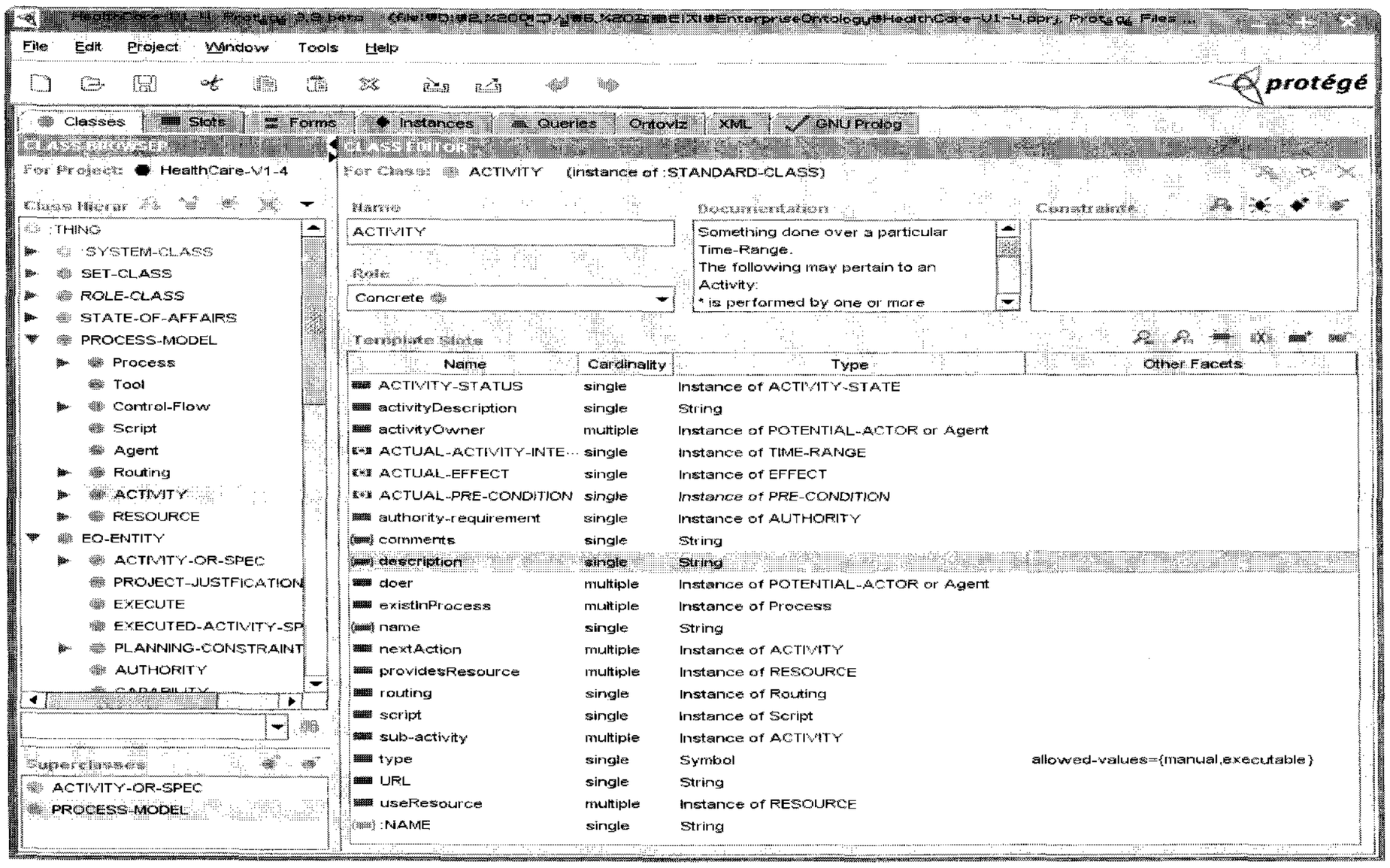
2.2 기업 온톨로지 구축

프로티지(Protégé)[7, 11]는 Stanford 대학의 의과대학에서 개발한 온톨로지 개발 및 관리도구이다. 온톨로지 모델링을 지원하기 위해 Protégé-Frames 편집기와 Protégé-OWL 편집기, 두 개의 플랫폼을 제공한다.

본 논문에서는 프로티지 3.2버전의 Protégé-Frames 편집기를 활용하여 EO를 구축[3]하였다. 이것의 근간을 이루는 Edinburgh EO는 Stanford 대학의 지식시스템 저장소(Knowledge Systems Laboratory : KSL)[10]에 탑재되어 있는 EO(이하 KSL EO)를 채용하였다. KSL EO에는 Edinburgh EO에서 규정한 내용이 많이 누락되어 있어 이를 클래스, 속성, 관계의 형태로 EO에 추가하였다.

2.3 기업 온톨로지 확장

이와 같은 과정으로 구축된 EO는 Edinburgh EO 체계표에 나타난 기본적인 개념들로 한정되어 있다. 따라서 기업모형 구성에 필요한 다양한 개념이 추가되어야할 필요가 있다. 또한 기본 골격으로 삼은 KSL EO에는 프로세스에 대한 개념이 명시적으로 나타나 있지 않다. 작업흐름의 표현과 관리를 위해서는 프로세스에 대한 개



<그림 1> 처리절차 모형이 추가된 최종적인 EO

넘이 반드시 고려되어야 한다. 이러한 이유로 기업구조(Enterprise Architecture) 모형인 AdCom(Administrative Computing Services, UC Irvine)[5] 사례와 프로세스 관련개념을 추가하여 EO를 확장하였다.

AdCom 사례에서 통합된 새로운 개념들은 Edinburgh EO 체계표의 상위구조 및 개별 개념들과의 관련성을 고려하여 클래스 구조를 결정하였다.

KSL EO에서는 프로세스에 대한 개념을 대신하여 프로세스 명세(Process Specification) 개념을 정의하여 다양한 형태로 재사용이 가능하도록 하고 있다. 본 논문에서는 이러한 간접적인 접근방식 대신 EO 안에서 프로세스를 직접적으로 표현하기 위해 업무절차 모형(Business Process Model)[13]을 추가하였다. 업무절차 모형은 Process Model, Agent, Resource, Tool, Action, Control-Flow, Script로 구성되어 있다. 여기에서 Agent, Resource, Action은 각각 기존 EO의 개념인 Actor, Resource, Activity를 그대로 사용하고 새로운 개념은 클래스로 추가하였다. Activity를 중심으로 클래스의 속성 값(예 : next-action)으로 프로세스의 흐름을 표현하였다. 이러한 흐름을 도식화하는 데 프로티지의 플러그인(Plugin)으로 제공되는 Ontoviz Tab을 활용하였다.

<그림 1>은 기업구조 사례와 업무절차 모형이 추가된 최종적인 EO의 모습이다.

2.4 프롤로그 추론기

온톨로지 추론을 위해 별개의 추론 엔진을 활용하지 않고 프로티지에 프롤로그 탭[8]을 플러그인으로 추가하여 구축한 EO 내에서 자체적으로 추론이 가능하도록 하였다. TBox 추론, ABox 추론, SWRL 추론을 모두 사용할 수 있도록 하였고 추론 규칙 정의에는 프롤로그 프로그램을 사용하였다. 프롤로그 추론기[3]를 통해 업무절차 변경에 따른 작업흐름 구성요소의 변화 정보를 예측하고 그 결과를 검증할 수 있다.

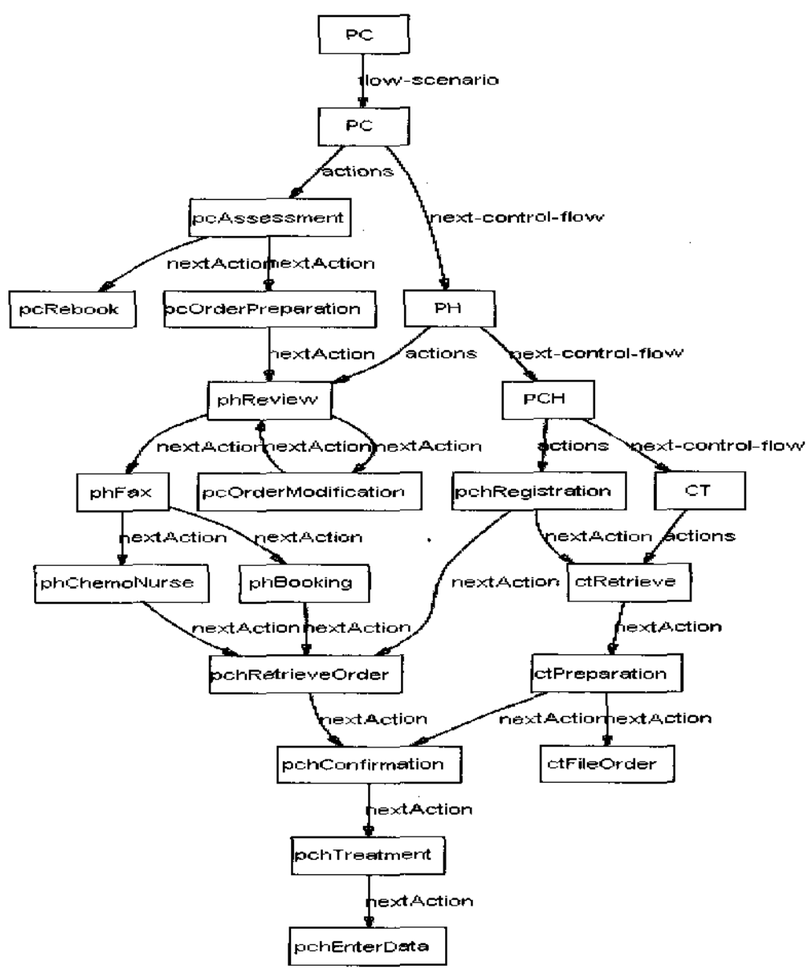
3. 기업 온톨로지 기반의 작업흐름 의존성 분석

3.1 기업 온톨로지 기반의 작업흐름 및 의존성 표현

작업흐름의 다양한 구성요소를 EO를 활용하여 표현하였다. 프로세스, 활동은 기존 EO의 개념인 Process, Activity를 그대로 사용하였고 자료, 역할은 각각 Resource, Actor의 개념을 활용하였다. 라우팅은 새로운 클래스로 생성하고 5가지 기본 라우팅을 그 하위 클래스로 정의하였다. 라우팅, 자료, 역할 관점에서의 의존관계는 EO 내에 정의하지 않고 EO의 지식을 탐색하여 추론할 수 있도록 프롤로그 규칙으로 정의하였다.

3.1.1 암 센터의 화학치료 프로세스 표현

의존성 분석용 사례로 암센터의 화학치료 프로세스[6]를 선택하였다. 화학치료 프로세스의 활동을 취합하고 각 활동들 사이의 라우팅을 5가지 기본 유형으로 분류하였다. 이렇게 분석한 사례를 EO 내에 인스턴스로 표현하였다. <그림 2>는 프로티지의 플러그인 Ontoviz를 이용하여 화학치료 프로세스의 활동 정보를 도식적으로 나타낸 것이다. <그림 3>은 활동과 라우팅 정보를 함께 표현하였다.



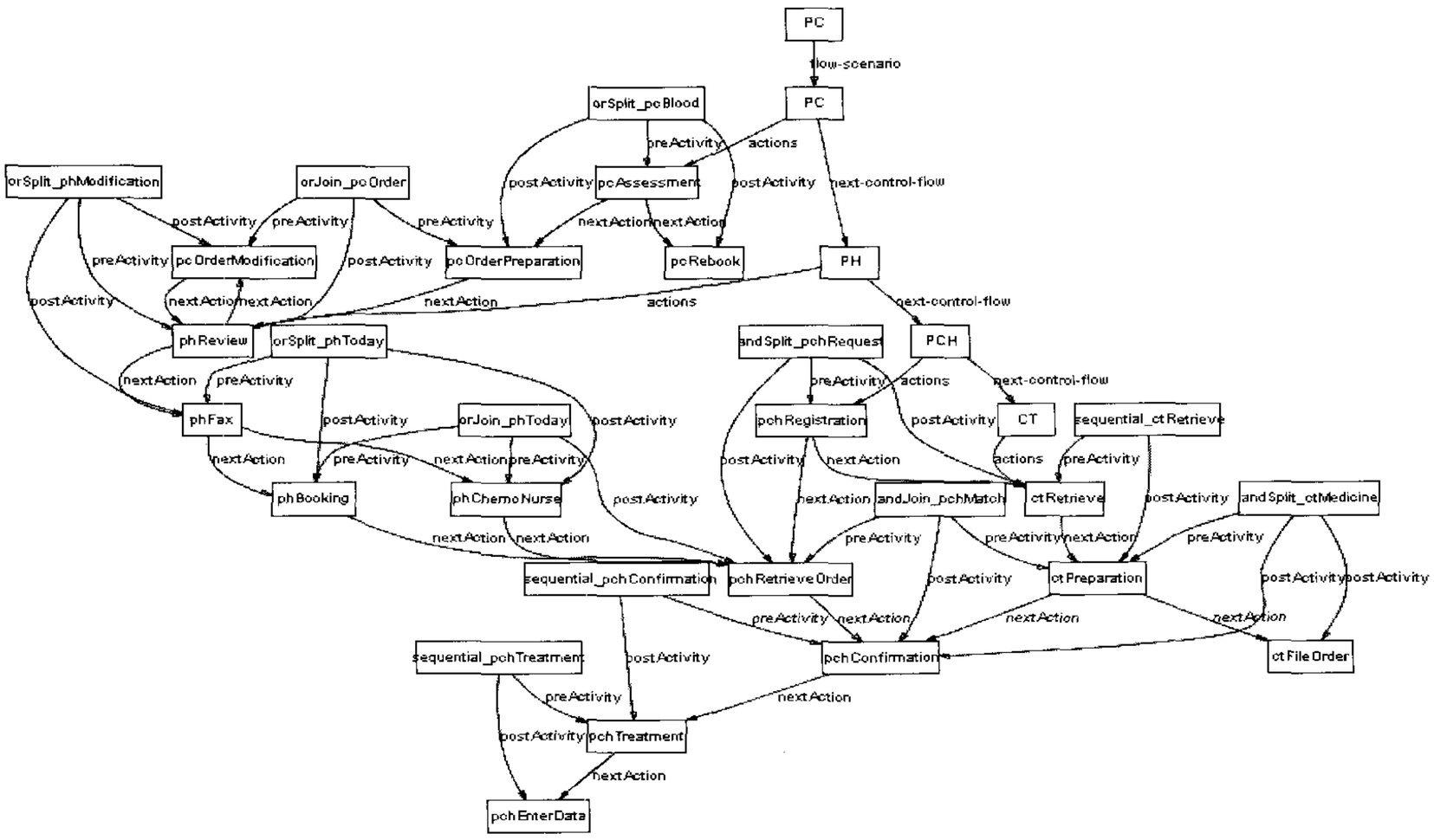
<그림 2> 암센터 화학치료 절차(활동)

3.1.2 라우팅 의존성(Routing Dependency)

라우팅(Routing)은 활동들 사이의 트랜잭션 형태 또는 제어 흐름을 나타내는 것으로 주로 활동과 프로세스의 실행 순서를 나타낸다. 순차(Sequential), AND 분할(AND-Split), AND 결합(AND-Join), OR 분할(OR-Split), OR 결합(OR-Join)과 같은 5가지 기본적인 형태가 있고 두 가지 이상의 형태가 복합적으로 나타나는 라우팅도 존재한다.

라우팅 의존성은 특정 프로세스 내 활동들의 실행 순서를 설명하고 정의한다. 라우팅 의존관계는 라우팅의 형태에 의해 결정되고 라우팅 의존성은 서로 이웃하고 있는 활동 사이의 의존 관계만을 나타낸다[6].

본 논문에서는 EO 내에 존재하는 프로세스, 활동, 라우팅과 관련된 지식 및 서로 간의 관계를 탐색하여 라우팅 의존성을 추론할 수 있도록 프롤로그 규칙으로 표현하였



<그림 3> 암센터 화학치료 절차(라우팅 정보 포함)

다. 이러한 라우팅 의존성을 표현하기 위하여 라우팅 클래스에 이전과 이후의 활동 정보를 나타내는 'preActivity'와 'postActivity' 속성을 추가하였다. <그림 4>는 프롤로그 탭에 라우팅 의존성을 정의한 내용이다.

```
routingDep(PreActivity, PreProcess, Routing, PostActivity,
PostProcess) : -
        allRoutings(Routing, PreActivity, PreProcess,
PostActivity, PostProcess).
allRoutings(Routing, PreActivity, PreProcess, PostActivity,
PostProcess) : - instanceof(Routing, frame('Routing'),
        preActivity(Routing, PreActivityL),
        postActivity(Routing, PostActivityL),
        member(PreActivity, PreActivityL),
        member(PostActivity, PostActivityL),
        existInProcess(PreActivity, PreProcessL),
        member(PreProcess, PreProcessL),
        existInProcess(PostActivity, PostProcessL),
        member(PostProcess, PostProcessL).
```

<그림 4> 라우팅 의존성 정의

- 라우팅 의존관계 술어
  routingDep(PreActivity, PreProcess, [Routing], PostActivity, PostProcess).
  서로 이웃하는 활동 PreActivity와 PostActivity간의 라우팅 의존 관계를 정의한다. 여기서 PreProcess와 PostProcess는 각각 PreActivity와 PostActivity가 속한 프로세스이다.

3.1.3 자료 의존성(Data Dependency)

자료 의존성이란 한 활동의 입력 자료는 다른 활동들의 출력 자료에 의존한다는 것이다. 라우팅 의존성과는 달리 한 활동의 출력 자료는 서로 이웃하지 않는 프로세스의 활동에 대한 입력 자료가 될 수도 있다[6]. 자료 의존성 또한 EO 내의 프로세스, 활동, 자원과 관련된 지식 및 서로 간의 관계를 탐색하여 자료 의존성을 추론해 낼 수 있도록 하였다. EO 내의 지식을 탐색하기 위해 입력 자료는 Resource 클래스의 'requiredBy' 속성을, 출력자료는 Activity 클래스의 'providesResource' 속성을 이용하였다. 아래 <그림 5>는 프롤로그 탭에 자료 의존성을 정의한 내용이다.

```
data(D, A, P) : - activity(A), providesResource(A, DL),
        member(D, DL), existInProcess(A, PL),
        member(P, PL).
dataDep(PostA, PostP, input(Data, PreA, PreP)) : -
        data(Data, PreA, PreP),
        requiredBy(Data, PostAL),
        member(PostA, PostAL),
        existInProcess(PostA, PostPL),
        member(PostP, PostPL).
```

<그림 5> 자료 의존성 정의

- 자료 의존관계 술어
  dataDep(PostA, PostP, input(Data, PreA, PreP)).
  PostP 프로세스에 있는 활동 PostA는 PreP 프로세스

에 있는 활동 PreA의 출력 Data에 의존한다.

3.1.4 역할 의존성(Role Dependency)

어떤 역할 A가 다른 역할 B에 의해 수행되고 있는 동일한 활동에 할당될 수 있을 때, B는 A에 대해 의존 관계를 가진다고 말할 수 있다. 역할 의존성은 활동을 기반으로 하는 라우팅 의존성 및 자료 의존성과는 달리 조직과 기능의 계층적인 구조에 기반하여 구성된다. 또한 특정 프로세스의 특정 활동에 대해서 의존 관계를 가진다[6]. 이러한 역할 의존성을 표현하기 위해 Activity 클래스의 'doer' 속성을 활용하여 주어진 역할의 대체 가능성을 프롤로그 규칙으로 정의하였다. 아래 <그림 6>은 프롤로그 탭에 역할 의존성을 정의한 내용이다.

```
role(Role, A, P) : - doer(A, RoleL), activity(A),
                     member(Role, RoleL),
                     existInProcess(A, PL), member(P, PL).
roleDep(Role, A, P, ReplaceRole) : -
                     role(Role, A, P),
                     role(ReplaceRole, A, P),
                     Role \== ReplaceRole.
```

<그림 6> 역할 의존성 정의

- 역할 의존관계 술어
  roleDep(Role, A, P, ReplaceRole).

프로세스 P의 활동 A에 할당된 Role은 ReplaceRole에 의해 대체될 수 있다.

3.2 프롤로그 질의 규칙을 이용한 의존성 분석

작업흐름에 변화가 발생할 경우 상호작용하는 다른 실체들에 미치는 영향을 확인하기 위해 라우팅, 데이터, 역할 관점에서 작업흐름 실체들 간의 의존관계를 분석하였다. 이를 자동적으로 수행하기 위해 프롤로그 탭에 EO의 지식을 탐색하여 의존관계를 추론해 낼 수 있는 질의 규칙[6]을 정의하였다.

3.2.1 라우팅 의존성 분석

작업흐름의 변화가 업무절차의 라우팅에 어떠한 영향을 미치는지를 확인하기 위해 아래와 같은 질의 규칙을 정의하고 이를 통해 라우팅 의존관계를 파악하였다. <그림 7>은 라우팅 의존성 분석을 위한 질의 결과이다.

- postActivityRouting(PreActivity, PreProcess, Results)
  이후의 활동정보와 주어진 활동과 이후 활동 사이에 관련된 라우팅 관계를 모두 반환한다(<그림 7>의 ①).
- preActivityRouting(PostActivity, PostProcess, Results)
  이전의 활동정보와 주어진 활동과 이전 활동 사이에 관련된 라우팅 관계를 모두 반환한다(<그림 7>의 ②).
- postProcessRouting(PreProcess, Results)
  주어진 프로세스에 의존적인 이전 프로세스 정보(이전 프로세스에 있는 활동과 관련 라우팅 정보)를 반환한다(<그림 7>의 ③).
- preProcessRouting(PostProcess, Results)
  주어진 프로세스에 의존적인 이후 프로세스 정보(이후 프로세스에 있는 활동과 관련 라우팅 정보)를 반

```
?-postActivityRouting(frame('phReview'), frame('PH'), Results). ①
time = 376ms
Results =
[activity(frame(pcOrderModification),frame(PC),frame(orsplit_phModification)),activity(frame(phFax),frame(PH),frame(orsplit_phModification))]
SUCCESS LAST
?-preActivityRouting(frame('phReview'), frame('PH'), Results). ②
time = 376ms
Results =
[activity(frame(pcOrderPreparation),frame(PC),frame(orJoin_pcOrder)),activity(frame(pcOrderModification),frame(PC),frame(orJoin_pcOrder))]
SUCCESS LAST
?-postProcessRouting(frame('PH'), Results). ③
time = 392ms
Results =
[activity(frame(pcOrderModification),frame(PC),frame(orsplit_phModification)),activity(frame(pchRetrieveOrder),frame(PCH),frame(orJoin_phToday)),a
ctivity(frame(pchRetrieveOrder),frame(PCH),frame(orJoin_phToday))]
SUCCESS LAST
?-preProcessRouting(frame('CT'), Results). ④
time = 392ms
Results = [activity(frame(pchRegistration),frame(PCH),frame(andSplit_pchRequest))]
SUCCESS LAST
?-activityReachPaths(frame('phReview'), frame('PH'), frame('pchRegistration'), frame('PCH')). ⑤

time = 0ms

SUCCESS LAST
?-activityReachPaths(frame('phFax'), frame('PH'), frame('pchTreatment'), frame('PCH')). ⑥
[activity(phBooking, PH),activity(pchRetrieveOrder, PCH),activity(pchConfirmation, PCH),activity(pchTreatment, PCH)]
[activity(phChemoNurse, PH),activity(pchRetrieveOrder, PCH),activity(pchConfirmation, PCH),activity(pchTreatment, PCH)]
time = 0ms
```

<그림 7> 라우팅 의존성 분석 질의 결과

환한다(<그림 7>의 ④).

- activityReachPaths(Preactivity, PostActivity, L)
  주어진 두 활동 사이에 서로 연결되는 경로가 있는지를 확인한다. 경로가 존재할 경우에는 그 경로를 구성하는 활동들이 리스트로 반환되고(<그림 7>의 ⑥) 그렇지 않을 경우에는 빈 응답을 한다(<그림 7>의 ⑤).

3.2.2 자료 의존성 분석

작업흐름의 변화로 자료 흐름에 어떠한 변화가 나타나는지를 확인하기 위해 아래와 같은 질의 규칙을 정의하고 이를 통해 자료 의존관계를 파악하였다. <그림 8>은 자료 의존성 분석을 위한 질의 결과이다.

- postActivityData(A, P, Results)
  특정 자료가 주어지지 않을 경우, 주어진 활동의 모든 출력자료에 의존적인 모든 활동정보를 반환한다(<그림 8>의 ①).
- postActivityData(A, P, D, Results)
  특정 자료가 주어지는 경우, 주어진 활동의 특정 출력자료에 의존적인 모든 활동정보를 반환한다(<그림 8>의 ②, ③).
- preActivityData(A, P, Results)
  특정 자료가 주어지지 않을 경우, 주어진 활동의 입력자료를 출력 자료로 생산하는 모든 활동정보를 반환한다(<그림 8>의 ④).
- preActivityData(A, P, D, Results)
  특정 자료가 주어지는 경우, 주어진 활동의 특정 입력자료를 출력자료로 생산하는 모든 활동정보를 반환한다(<그림 8>의 ⑤).
- postProcessData(P, Results))
  프로세스 P의 출력자료에 의존하는 다른 프로세스의 활동들을 반환한다(<그림 8>의 ⑥).
- preProcessData(P, Results)
  프로세스 P의 입력자료를 출력자료로 생산하는 다른 프로세스의 활동들을 반환한다(<그림 8>의 ⑦).

3.2.3 역할 의존성 분석

작업흐름의 변화가 특정 활동을 수행하는 역할에 미치는 영향을 확인하기 위해 아래와 같은 질의 규칙을 정의하고 이를 통해 역할 의존관계를 파악하였다. <그림 9>는 역할 의존성 분석을 위한 질의 결과이다.

- allRoleAssigned(A, P, Results)
  프로세스 P의 활동 A에 할당된 모든 역할을 반환한다(<그림 9>의 ①).
- allActivityAssigned(Role, Results)
  지정된 역할이 할당되어 있는 모든 활동 A를 반환한다(<그림 9>의 ②).
- roleReplaceable(ReplaceRole, Role, A, P)
  프로세스 P에 속해있는 활동 A에 할당된 역할 R2이 주어질 때, 역할 R1이 R2의 역할을 대체할 수

```
postActivityData(frame('phReview'), frame('PH'), Results). ①
time = 16ms
Results =
[activity(frame(ctFileOrder),frame(CT)),activity(frame(ctPreparation),frame(CT)),activity(frame(pcOrderModification),frame(PC)),activity(frame(pchConfi
rmation),frame(PCH)),activity(frame(phBooking),frame(PH)),activity(frame(phChemoNurse),frame(PH)),activity(frame(phFax),frame(PH))];
SUCCESS LAST
?-postActivityData(frame('pcAssessment'), frame('PC'), frame('treatment confirmation'), Results). ②
time = 0ms
Results = [activity(frame(pcOrderPreparation),frame(PC))];
SUCCESS LAST
?-postActivityData(frame('phReview'), frame('PH'), frame('confirmed order'), Results). ③
time = 0ms
Results =
[activity(frame(phFax),frame(PH)),activity(frame(phBooking),frame(PH)),activity(frame(phChemoNurse),frame(PH)),activity(frame(pchConfirmation),fra
me(PCH)),activity(frame(ctPreparation),frame(CT)),activity(frame(ctFileOrder),frame(CT))];
SUCCESS LAST
?-preActivityData(frame('phReview'), frame('PH'), Results). ④
time = 391ms
Results = [activity(frame(pcOrderModification),frame(PC))];
SUCCESS LAST
?-preActivityData(frame('pchConfirmation'), frame('PCH'), frame('medicine'), Results). ⑤
time = 376ms
Results = [activity(frame(ctPreparation),frame(CT))];
SUCCESS LAST
?-postProcessData(frame('CT'), Results). ⑥
time = 392ms
Results =
[activity(frame(pchConfirmation),frame(PCH)),activity(frame(pchTreatment),frame(PCH)),activity(frame(phBooking),frame(PH)),activity(frame(phChem
oNurse),frame(PH)),activity(frame(phFax),frame(PH))];
SUCCESS LAST
?-preProcessData(frame('CT'), Results). ⑦
time = 486ms
Results =
[activity(frame(pchRegistration),frame(PCH)),activity(frame(pchRetrieveOrder),frame(PCH)),activity(frame(pchRetrieveOrder),frame(PCH)),activity(fra
me(phBooking),frame(PH)),activity(frame(phBooking),frame(PH)),activity(frame(phChemoNurse),frame(PH)),activity(frame(phChemoNurse),frame(PH
)),activity(frame(phFax),frame(PH)),activity(frame(phFax),frame(PH)),activity(frame(phReview),frame(PH)),activity(frame(phReview),frame(PH))];
SUCCESS LAST
```

<그림 8> 자료 의존성 분석 질의 결과

```
allRoleAssigned(frame('phReview'), frame('PH'), Results). ①
time = 94ms
Results = [frame(Pharmacist)];
SUCCESS LAST
?-allActivityAssigned(frame('Nurse'), Results). ②
time = 47ms
Results =
[activity(frame(pchEnterData),frame(PCH)),activity(frame(pchConfirmation),frame(PCH)),activity(frame(pchTreatment),frame(PCH)),
activity(frame(pchRetrieveOrder),frame(PCH))];
SUCCESS LAST
?-roleReplaceable(frame('Physician'), frame('Pharmacist'), frame('phReview'), frame('PH')). ③
time = 15ms
FAIL
?-roleActivityReplace(frame('Pharmacist'), Results). ④
time = 63ms
Results = [role(frame(Nurse),frame(pchConfirmation),frame(PCH))];
SUCCESS LAST
?-roleReplace(frame('Nurse'), frame('pchConfirmation'), frame('PCH'), Results). ⑤
time = 0ms
Results = [frame(Pharmacist)];
SUCCESS LAST
```

<그림 9> 역할 의존성 분석 질의 결과

있는지를 논리값으로 반환한다(<그림 9>의 ③).

- roleActivityReplace(ReplaceRole, Results)
  역할 R에 의해 대체될 수 있는 역할의 집합을 그 관련된 활동과 프로세스의 정보를 포함하여 반환한다(<그림 9>의 ④).
- roleReplace(Role, A, P, Results)
  프로세스 P의 활동 A에 할당된 주어진 역할 R을 대체할 수 있는 역할들을 반환한다(<그림 9>의 ⑤).

4. 작업흐름 영향 분석 사례

기업의 업무 환경 변화가 작업흐름에 미치는 영향을 두 가지 각도에서 분석하고 이를 잘 보여줄 수 있는 사례를 각각 제시하였다. 하나는 작업흐름 구성요소 간의 의존관계를 분석함으로써 한 구성요소의 변화가 다른 요소에 미치는 영향을 분석한 사례이다. 또 다른 하나는 BPR 등과 같은 사유로 업무절차가 변경되었을 때 그에 따른 작업흐름 구성요소의 변화를 확인하고 검증하는 사례이다. 이처럼 서로 다른 사례 연구를 통해서 다양한 응용분야에 기업 온톨로지와 작업흐름 영향분석을 적용할 수 있음을 확인하였다.

4.1 암센터 화학치료 사례

앞 장에서 제시한 암센터 화학치료 사례에서, 의약분업 시행과 같은 제도 변화로 인해 'phReview' 활동이 삭제될 경우 작업흐름에 어떠한 영향을 주는지를 확인하기 위해 'phReview'의 라우팅, 자료, 역할 의존성을 분석하였다.

앞 장의 <그림 7>, <그림 8>, <그림 9>에서 질의 결과를 보여주고 있다. <그림 7>의 ①과 ②는 'phReview'의 라우팅을 분석한 것이다. ①은 'phReview'의 이후 활동인 'pcOrderModification' 및 이후 활동과 'phReview'와의 라우팅 정보를 반환한다. ②는 'phReview'의 이전 활동인 'pcOrderModification'과 'pcOrderPreparation' 및 'phReview'와 이전 활동들과의 라우팅 정보를 보여준다. <그림 8>의 ①, ③, ④는 자료 의존성을 분석한 것이다. ①은 'phReview'의 모든 출력자료에 의존적인 활동정보를 반환한다. ③은 'phReview'의 출력자료인 'confirmed order'를 입력자료로 사용하는 활동정보를 모두 보여준다. ④는 'phReview'의 입력자료를 생산하는 활동 'pcOrderModification'을 반환한다. <그림 9>의 ①과 ④는 역할 의존성을 분석한 것이다. ①에서는 'phReview' 활동에 할당된 역할이 'Phamacist' 하나임을 보여준다. ④는 역할 'Phamacist'가 대체할 수 있는 역할이 있는가? 라는 질의로, 활동 'pchConfirmation'에 할당된 역할 'Nurse'를 'Phamacist'가 대신할 수 있음을 말한다.

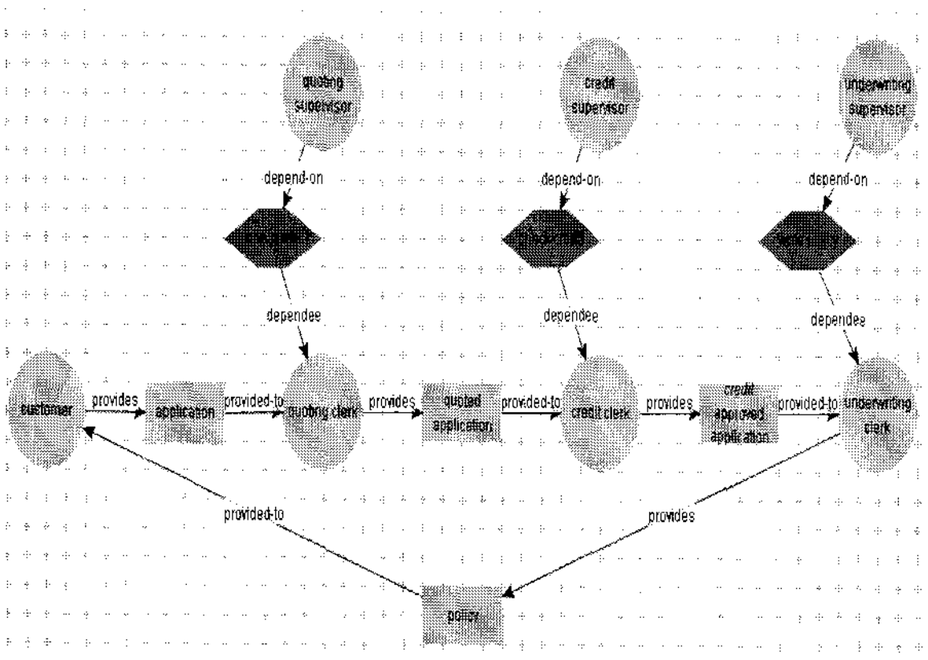
<표 2> 의존성 분석 결과 : phReview 삭제의 경우

	분석결과
라우팅	◦ 이전 활동인 pcOrderModification, pcOrder-Preparation에 영향을 줌 ◦ 이후 활동인 pcOrderModification에 영향을 줌
자료	◦ phFax, phChemoNurse, phBooking, pchCon-firmation, ctPreparation, ctFileOrder, pcOrder-Modification의 입력자료에 영향을 줌 ◦ pcOrderModification의 출력자료를 입력으로 받지 않아도 됨
역할	◦ 'Phamacist'에 할당된 역할이 감소

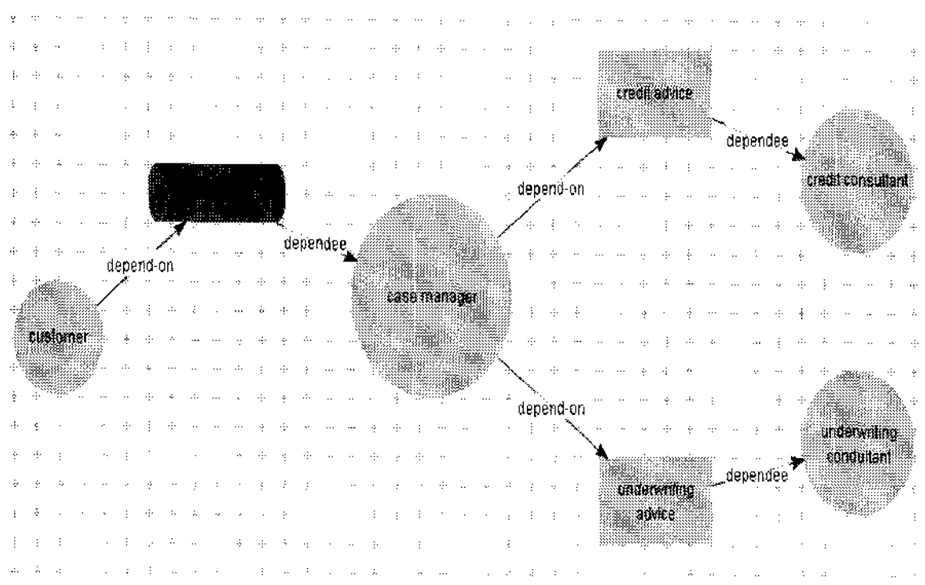
<표 2>는 'phReview' 활동을 삭제한 경우 각각의 관점에서 의존성을 분석한 결과이다. 이와 같이 의존성 분석을 통해 'phReview' 활동의 삭제가 다른 프로세스 및 활동, 입력 및 출력 자료 등 작업흐름 내의 다른 실체들에 미치는 영향을 확인할 수 있다. 또한 그 결과를 활용하여 발생할 수 있는 문제를 예측하고 대처방안도 알 수 있다. 위의 예에서 'phReview'를 삭제한다면 결과적으로는 'pcOrderModification'을 함께 삭제하는 한편 'phReview'의 출력자료를 이전 활동인 'pcOrderPreparation'에서 생산해야 함을 알 수 있다.

4.2 보험 계약 사례

또 하나의 사례로 보험계약 절차가 BPR로 인해 변경되었을 경우 이에 대한 작업흐름의 영향을 분석하였다. BPR 전후의 보험계약 업무절차를 EO에 표현하고 구성요소별 작업흐름의 변화를 예측하였다.



<그림 10> BPR 전 보험계약업무



<그림 11> BPR 후 보험계약업무

<그림 10>과 <그림 11>은 프로티지의 플러그인 Graph Widget을 이용하여 보험계약 업무를 행위자 의존 모형(Actor Dependency Model)[18]으로 표현한 것이다. <그림 10>은 보험 가입을 원하는 고객에게 견적(Quote), 신용평가(Credit), 승인(Underwrite) 단계를 거쳐 최종적으로 보험증서를 발급하는 전형적인 업무절차이다.

<그림 11>은 BPR을 수행한 후의 업무절차이다. 사례관리자(Case Manager)가 고객의 보험계약 요구에 대한 모든 업무를 담당하고 신용평가 및 승인에 관련된 업무는 고문(Consultant)의 조언(Advice)을 받는다.

위의 <그림 10>과 <그림 11>을 통해 전통적인 업무절차에서 *관리자(Supervisor)-실무자(Clerk)* 구조로 처리되는 업무가 변경 후에는 단일 업무참여자에 의해 처리되는 것을 알 수 있다. 즉, 작업흐름의 구성요소인 인사조직에서 변화가 나타났다.

EO의 정보는 업무절차 변경 시에 영향을 주고받게 되는 다른 실체들에 대한 정보를 포함하고 있다. 아래 <표 3>은 EO의 지식을 탐색하여 BPR 전후의 업무절차 변경에 따른 작업흐름 구성요소의 영향 정보를 예측한 것이다[1, 2, 3].

<표 3> EO를 활용하여 예측한 작업흐름 영향 정보

업무 절차	• Activity 구성과 순서의 변화
인사 조직	• 업무 참여자(Actor)의 역할 변화 ◦ Clerk(Actual-Doer, Managed-By)과 Supervisor(Activity-Owner, Manages)의 역할을 Case Manager(Actual-Doer, Activity-Owner)를 도입하여 대신함 ◦ Consultant(Actual-Doer)를 도입함 • 관리 구조의 변화 ◦ Supervisor(관리자) - Clerk(실무자) 구조로 처리되는 업무가 단일 Actor로 처리
경영 정보	• 자원의 변화 ◦ Application, Quoted Application, Credit Approved Application 등이 Application-To-Be로 통합됨 ◦ Credit Advice와 Underwriting Advice가 도입됨

선행 연구[3]에서는 프롤로그 추론기를 통하여 <표 3>의 예측 결과를 검증하였다. 이는 작업흐름 실체들 간의 의존성 분석을 통해서도 동일한 결과를 얻을 수 있을 것이다.

5. 결론 및 향후 연구

오늘날 기업은 기업 내외의 다양한 변화를 탄력적이고 민첩하게 수용하기 위해 지속적인 업무 절차 관리를 필요로 한다. 또한 이러한 변화들이 작업 흐름에 미치는 영향을 분석하는 일은 매우 중요한 문제이다.

현재 작업흐름의 영향분석은 주로 라우팅의 관점에서 이루어지는 점에 비해서 본 논문에서는 기업의 인지적 모델인 기업 온톨로지를 기반으로 자료, 역할 등 다양한 관점에서 영향분석을 하였다. 이를 위해 온톨로지 편집 도구인 프로티지를 이용하여 기업 온톨로지를 기반구축하고 이를 확장하였다. 이 기업 온톨로지를 활용하여 라우팅, 자료, 역할 의존성을 표현하고 각각을 분석할 수 있는 프롤로그 질의 규칙을 정의하였다. 이를 암센터 화학치료 사례에 활용하여 작업흐름 구성요소 간의 의존관계를 파악하고 그 결과를 토대로 변화에 대한 작업흐름의 영향을 분석할 수 있었다. 또한 보험계약 업무 사례에서 BPR 등으로 인해 업무 절차에 변화가 발생했을 경우 EO를 활용하여 작업흐름 구성요소의 변화를 확인하고 예측할 수 있었다. 본 연구의 결과를 활용하여 기업은 업무 환경 변화에 영향을 받는 작업흐름 구성요소를 정확히 예측하고 이런 변화를 민첩하게 반영하는 정보 시스템을 구축할 수 있을 것이다. 또한 업무 재설계에 투입되는 자원과 시간을 최소화함으로써 기업의 생산성을 높이는 방안이 될 수 있다. 이는 본 연구가 업무 절차 개정에 대한 반자동식 기존 접근법의 약점을 보완하고 적응형 또는 지식기반형 작업흐름 관리기법의 융통성을 제고시키는 방안이 될 수 있음을 시사한다. 또한 이러한 사례 연구를 통하여 기업경영 및 작업흐름 관리에 EO와 프로티지의 유용성을 확인할 수 있었다.

향후 연구로 작업흐름에 나타날 수 있는 변화를 활동, 자료, 역할 등의 관점에서 구체적으로 분류하고 각 경우에서 작업흐름의 영향분석 결과를 정형적으로 정의할 수 있는 방안을 모색하고자 한다.

참고문헌

[1] 김민찬, 남철기, 임정민, 강인수, 배재학, 이종혁; "Enterprise 온톨러지를 활용한 Workflow 변화 예측", 한국정보처리학회 춘계학술발표대회 논문집, 제11권 제1호, 중앙대학교 서울, 405-408, 2004.

[2] 박지현; "기업 온톨로지 기반의 작업흐름 변화 예상", 2007 한국산업경영시스템학회 추계학술대회 발표논문집, 국립금오공과대학교 구미, 88-91, 2007.

[3] 박지현; "기업 온톨로지 기반의 워크플로우 변화 예측 및 검증", 석사학위논문, 울산대학교, 2007.

[4] 양재군, 배재학, 이종혁; "온톨로지 재사용을 위한 범주 재분류", 정보처리학회논문지 B 12-B(1) : 69-80, 2005.

[5] Administrative Computing Services, UC Irvine, Enterprise Architecture, http://apps.adcom.uci.edu/EnterpriseArch/.

[6] Dai, Weizhen and Covvey, H. Dominic; "Query-Based Approach to Workflow Process Dependency Analysis," 2005 Technical Reports, David R. Cheriton School of Computer Science, University of Waterloo, 2005.

[7] Gennari, J. H., Musen, M. A., Fergerson, R. W. et al.; "The evolution of Protégé : an environment for knowledge-based systems development," *Int. Journal of Human-Computer Interaction,* 58(1) : 89-123, 2003.

[8] http://gnuprologjava.sourceforge.net/.

[9] Kim, K. H.; "Workflow Dependency Analysis and Its Implications on Distributed Workflow Systems," *Proceedings of the 17th International Conference on Advanced Information Networking and Applications,* 677-682, 2003.

[10] Knowledge Systems Laboratory Stanford University, http://www-ksl-svc.stanford.edu : 5915.

[11] Protégé, http://protege.stanford.edu/.

[12] Reijersa, H. A., van der Aalstb, W. M. P.; "The effectiveness of workflow management systems : Predictions and lessons learned," *Int. Journal of Information Management,* 25(5) : 458-472, 2005.

[13] Software Process Modeling with Protege, http://www.ics.uci.edu/.

[14] Stohr, E. A. and Zhao, J. L.; "Workflow Automation : Overview and Research Issues," *Information Systems Frontiers*, 3(3) : 281-296, 2001.

[15] The Workflow Management Coalition, http://www.wfmc.org.

[16] Uschold, M., King, M., Moralee, S., Zorgios, Y.; "The Enterprise Ontology," *The Knowledge Engineering Review*, 13(1) : 31-89, 1998.

[17] Wang, M. and Wang, H.; "From Process Logic to Business Logic--A Cognitive Approach to Business Process Management," *Information and Management,* 43(2) : 179-193, 2006.

[18] Yu, Eric S. K. and Mylopoulos, John; "An Actor Dependency Model of Organizational Work-With Application to Business Process Reengineering," *Proceedings of the Conference on Organizational Computing Systems(COOCS 93)*, 258-268, 1993.