데이터 의존성 그래프 : 비즈니스 프로세스 설계를 위한 데이터 요구사항의 표현

장 무 경*
*남서울대학교 산업경영공학과

Data Dependency Graph : A Representation of Data Requirements for Business Process Modeling

Moo-Kyung Jang*
*Dept. of Industrial & Management Engineering, Namseoul University

Abstract

Business processes are often of long duration, and include internal worker's decision making, which makes business processes to be exposed to many exceptional situations. These properties of business processes makes it difficult to guarantee successful termination of business processes at the design phase. The behavioral properties of business processes mainly depends on the data aspects of business processes. To formalize the data aspect of process modeling, this paper proposes a graph-based model, called Data Dependency Graph (DDG), constructed from dependency relationships specified between business data. The paper also defines a mechanism of describing a set of mapping rules that generates a process model semantically equivalent to a DDG, which is accomplished by allocating data dependencies to component activities.

Keywords : Business Process, Data Dependency, Data Dependency Graph, Process Modeling

1. 서 론

1.1 연구의 배경

비즈니스 프로세스란 어떤 특정한 목적을 달성하기 위하여 필요한 단위 활동(activity)들과 그 활동들 간에 존재하는 시간적 선후관계로 표현된다. 시간적 선후관계를 표현하기 위해서 일반적으로 프로그래밍 언어의 제어구조를 활용하고 있지만, 그것만으로는 비즈니스 프로세스의 실행 로직을 구체적으로 묘사하기가 어렵다는 문제점이 있다. 이는 비즈니스 프로세스가 일반적인 프로세스에 비해 상대적으로 긴 실행시간을 가지며, 내부적으로 사람의 의사결정이 포함되는 등의 특성을 가지고 있기 때문이다. 기업 내에 존재하는 다양한 요소들 간의 예측 불가능한 상호작용 등으로 인한 비즈니스 프로세스의 복잡성을 표현하고자, 특히 활동 간의 시간적 선후관계를 표현하기 위한 시맨틱의 확장을 중심으로, 많은 연구가 이루어져 왔다 [1] [5] [8] [9] [17] [18] [19].

그럼에도 불구하고, 활동 중심의 비즈니스 프로세스 표현에 있어서 문제가 되고 있는 것은 설계된 프로세스의 실행 가능성을 사전에 평가하기가 어렵다는 것이다. 이러한 문제점을 해결하기 프로세스의 실행 상태에 영향을 줄 수 있는 부분으로서 데이터에 대한 고려가 중

† 이 논문은 2009년도 남서울대학교 학술연구비 지원에 의해 연구되었음.
† 교신저자: 장무경, 충남 천안시 서북구 성환읍 매주리 21 남서울대학교 산업경영공학과
M · P: 017-390-9025, E-mail: mkjang@nsu.ac.kr
2011년 4월 17일 접수; 2011년 6월 13일 수정본 접수; 2011년 6월 14일 게재확정

심이 되고 있다. Sadiq의 연구에서 강조된 바와 같이, 프로세스 검증이란 프로세스 모델링 시에 기대했던 대로 프로세스가 진행하는 지를 고찰하는 과정이며, 이를 위해서 가장 중요한 측면은 프로세스 실행시의 데이터 플로우를 분석하는 것이라는 점이다 [14]. 즉, 현재 설계하고 있는 비즈니스 프로세스가 데이터 측면의 요구사항을 충분히 반영하지 못하면 실행 중에 다양한 예외상황과 마주치게 된다는 의미이다.

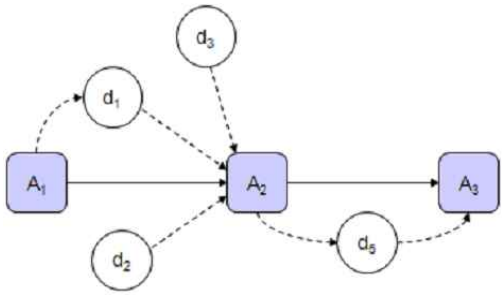
이러한 관점의 연장선 상에서, 본 논문에서는 프로세스의 실행 특성에 영향을 줄 수 있는 요인으로 데이터 측면을 반영할 수 있는 방안을 제안하고자 한다. 이를 위하여, 데이터 측면의 프로세스 설계 요구사항을 데이터 간에 존재하는 의존성으로 나타내고 이러한 요구사항이 어떻게 프로세스 설계에 반영될 수 있는 지를 활동 매핑 (activity mapping) 이라는 매커니즘을 소개함으로써 설명하고자 한다.

1.2 데이터:프로세스의 실행가능성 결정요인

기업 내 일반 사무실에서 수행되는 비즈니스 프로세스와 관련된 데이터 요소는 아주 다양한 형태로 나타난다. 고객 이름과 같이 하나의 값이거나, 고객의 신상명세와 같은 레코드 형태일 수도 있고, 구매요청서와 같이 구조화된 형태를 가지는 문서일 수도 있으며, 전자 메일을 통해 전달되는 간단한 메모, 메시지 등과 같이 비구조화된 문서일 수도 있다. 또한, 특정 애플리케이션을 통해서만 데이터에 대한 접근이 가능한 경우가 있는 반면에 일반적인 의미의 파일 저장소를 통해 공유되는 경우도 있다.

이렇게 다양한 형태로 분산되어 있는 데이터는 프로세스를 통해 생성, 참조되거나 변경된다. 데이터는 프로세스의 최종 산출물의 일부로서 다른 데이터와 구조적으로 통합되기도 하며, 비즈니스 프로세스에서 부과된 제약조건으로 말미암아 특정 범위 내의 값 만을 가지도록 강제된다거나, 또는 데이터 간의 무결정 제약조건으로 인해 다른 데이터의 변경에 연동되어 변경되는 형태로 재구성되기도 한다. 이러한 데이터의 생성과 변경에 있어 특히 데이터의 가용성, 신뢰성, 그리고 무결성 측면은 프로세스의 실행 가능성을 결정할 수 있는 요인으로 작용한다.

비즈니스 프로세스를 설계함에 있어 데이터 관점의 요구사항은 일반적으로 활동들 간의 콘트롤 플로우로 표현된 프로세스 위에 활동들 간에 존재하는 데이터 플로우를 추가하여 표현하는 것으로 나타내고 있다. 하지만, 이러한 접근은 앞서 설명한 데이터의 가용성, 신뢰성,



<그림 1> 비즈니스 프로세스의 예

무결성 등을 보장하기 어려운 측면이 있다. 이를 <그림 1>의 예를 통해 살펴보기로 한다. 그림에서 사각형 심볼은 활동을, 원은 데이터를 나타낸다. 활동 간의 화살표는 시간적 선후관계를, 활동과 데이터 간의 점선 화살표는 활동 간의 데이터 플로우를 표현하고 있다.

1.2.1 데이터의 가용성 측면

<그림 1>에서 활동 A_2를 수행하기 위해서 d_1, d_2, d_3 등의 3개 데이터가 필요하다. 이 경우 활동 A_1이 수행이 종료되어 활동 A_2를 시작하려면, 바로 그 시점에서 이들 3개 데이터가 모두 가용한 상태에 있어야 한다. 즉, 어떤 활동을 수행하기 위해서는 그 활동을 수행하는 데 필요한 데이터들이, 활동 수행의 사전조건으로서, 가용한 상태로 존재해야 한다. 그 데이터들은 앞서 수행된 활동에서 생산되어 작업지시와 함께 전달된 데이터일 수도 있고(d_1의 경우), 이와 반대로, 공유 저장소 또는 애플리케이션을 통해 전달받는 경우도 있다(d_2 또는 d_3의 경우). 만약 필요한 모든 데이터를 확보하지 못하는 경우에는 현재 존재하지 않는 데이터가 가용해질 때 까지 기다리던지, 아니면 그 데이터를 소유하고 있거나, 생산할 수 있는 다른 작업자에게 데이터를 요청해야 한다. 결국, A_2 활동은 시작이 보류되며 추가적인 데이터를 확보하기 위한 서브 프로세스를 실행해야 한다. 즉, 활동이 수행하는데 필요한 데이터가 가용하지 않는 경우에는, 이와 같이 불필요한 시간적 지연이 발생하거나 부가적인 활동의 실행을 위해 추가적인 비용이 소요된다.

1.2.2 데이터의 신뢰성 측면

어떤 활동을 수행하는데 필요한 데이터 중에 부정확한 데이터가 있는 경우에는, 그 활동의 수행 결과에 대한 신뢰도에 영향을 미치게 된다. 이러한 상황은 결국 전체 프로세스를 성공적으로 종료하지 못하고 중단 (abort) 해야 하는 상황이 발생하거나, 그 오류를 회복 (recovery) 하기위해 추가적인 비용 및 시간을 소모하는 상황이 나타나게 된다. <그림 1>의 예에서 본다면,

만약 d_3가 부정확한 데이터라면 A_2에 의해 생산된 d_5 데이터 또한 정확성을 신뢰할 수 없게 된다. 또한, 일반적인 경우에 어떤 데이터를 사용하는 활동이 그 데이터가 정확한지 또는 부정확한지에 대해 사전에 확인하기는 매우 어렵다.

1.2.3 데이터의 무결성 측면

데이터는 그 자체의 무결성을 위해 다른 데이터들과 일정한 관계를 가지게 된다. 어떤 데이터의 의미는 그 데이터가 다른 데이터와 가지는 관계 속에서 결정된다는 뜻이다. 따라서, 활동의 실행결과로서 어떤 데이터를 생성하거나 업데이트하려고 할 때에는 다른 데이터에 대한 업데이트가 동시에 발생되어야 하는 경우가 많다. 하지만, 데이터 간의 무결성을 위해 부수적으로 발생하는 이러한 업데이트 요구를 개별 활동에서 처리하기는 어렵다. 예를 들어서, <그림 1>에서 A_2에 의해 새로 생성된 데이터 d_5로 인해 다른 데이터에 대한 업데이트 필요성이 있다면, 이 요구를 하나의 단위 활동인 A_2가 처리하는 것은 어려울 뿐 아니라, 또 다른 문제가 발생될 소지가 있다. 즉, 전체 프로세스의 문맥에 맞도록 제어되어야 하는 부분이다.

2. 관련연구

비즈니스 프로세스를 구성하는 활동들 간의 시간적 선후관계는 그들 사이에서 공유되는 자원의 특성으로부터 나온다 [2] [4] [7]. 예를 들어, 한 사람이 수행하기로 되어 있는 2개의 업무는 동시에 수행될 수 없다 라던가 또는 어떤 데이터를 생산하는 활동은 그 데이터를 사용하는 활동보다 앞서 수행되어야 한다 등이 좋은 예가 된다. 이와 같이 활동들 간에 시간적 선후관계에 대한 조정이 필요한 이유를 활동들이 공유하는 자원에 대한 의존성으로 보고 의존성의 분류 형태를 제안한 Malone의 연구가 있었다 [12]. Malone은 이 연구에서, 의존성의 형태를 공유 자원 관계, 생산자/소비자 관계, 동시성 제약조건, 활동과 부활동 간의 관계 등의 4가지 형태로 나누었다. Crowston 등은 그들의 보고서에서 어떤 활동의 수행에 필요한 자원을 그 태스크의 사전 조건으로 정의하고, 만약 2개 이상의 활동들 간에 사전 조건들이 서로 겹치면 그 들 간의 의존성을 조정하기 위한 추가적인 컨트롤이 필요함을 지적하였다 [11]. 이들의 연구는 이후 MIT 프로세스 핸드북 의 프로세스 모델링 아키텍처를 제안하는데 활용되었다 [15].

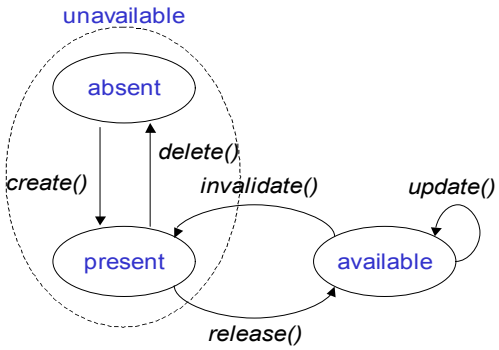
비즈니스 프로세스 설계 시에 데이터 요구사항을 고려하고자 했던 시도들은 문서의 흐름을 지원하는 워크플로우 또는 폼기반 워크플로우 연구 등을 중심으로 진행되어 왔으나, 이 연구들은 프로세스 지향의 워크플로우를 설계하기 위한 연구라기 보다는 데이터 중심의 애플리케이션 개발을 위한 연구로 분류될 수 있다 [3] [6] [16] [20] [21]. Aalst 등은 워크플로우 설계를 위한 새로운 패러다임으로서 케이스 핸들링의 개념을 제안하였다 [21]. 케이스 핸들링이란 지식 중심의 업무에서 지식 노동자의 판단에 의해 프로세스를 진행하고자 하는 것으로, 프로세스 진행 중에 데이터의 입력은 지식 노동자의 판단 하에 결정할 수 있도록 한 것이 특징이다. 그는 또한 이전 연구에서, 워크플로우 설계 시에, 워크플로우의 최종 산출물이 되는 데이터의 구조를 고려하여야 함을 주장하였다 [18]. 후속 연구인 Vanderfeesten의 연구는 프로세스가 성공적으로 수행이 완료되면 얻어질 수 있는 하나의 데이터 구조 (수직적 포함관계) 를 정의하면 그 구조로부터 프로세스를 설계할 수 있다는 점을 강조하였다 [10]. 데이터 간의 관계를 통해서 프로세스를 설계하고자 하는 점에서는 유사하나 최종 산출물에 포함되는 데이터 만을 고려하고 있다는 측면에서, 전체 프로세스의 수명주기 범위에 포함된 데이터를 고려하는 본 논문의 접근과는 차이를 보인다. Vanderfeesten의 연구와의 비교는 4장에서 소개하고자 한다.

3. 데이터 의존성

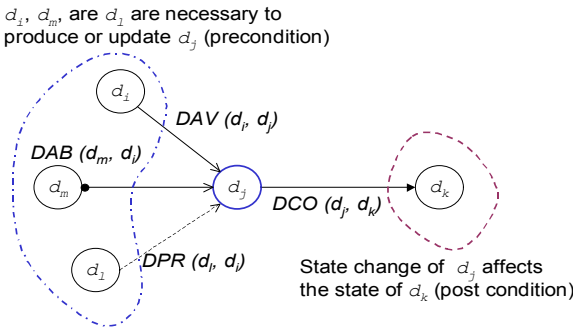
3.1 데이터 의존성의 분류

본 논문의 관점에서 데이터 간의 의존성이란, 데이터 d_j를 생성하기 위해서 다른 어떤 데이터 d_i가 필요한 경우에 데이터 d_j의 생성 여부는 데이터 d_i에 의존한다는 의미의 표현으로 사용한다. 물론 이러한 데이터 의존성은 불변량적인 규칙이 아니다. 이 2개 데이터가 연관되어지는 하나의 문맥으로서 어떠한 비즈니스 프로세스가 우선 존재해야 한다. 예를 들어, 데이터 d_i 가 데이터 d_j 에 포함된다고 하자. 만약 데이터 d_i 가 생성된 이후에야 d_j 가 만들어질 수 있다면 이들 간의 관계는 $d_i \rightarrow d_j$ 와 같이 나타낼 수 있다. 하지만 역으로, 하나의 컨테이너로서 d_j 가 먼저 만들어져 있어야 그 컨테이너에 포함되는 데이터 d_i 를 생성하게 되는 경우도 있다. 즉, d_i 가 생성되기 위한 사전조건으로서 d_j 의 생성이 필요한 경우이다. 이러한 경우를 앞의 경우와 비교한다면 $d_j \rightarrow d_i$ 와 같이 표현되어야 한다. 즉, 데이터 간의 의존성이란 프로세스가 포함하는 로직 또는 실행 규칙과 밀접한 관계를 가지게 된다.

데이터 의존성과 고려하여 추가적으로 더 고려해 볼 것



<그림 2> 데이터 상태 다이어그램



<그림 3> 데이터 의존성의 형태

은 데이터의 상태이다. 활동들 간에 공유되는 데이터를 외부 개체들이 인식하는 상태는 (1) 그 데이터가 존재하는지 또는 존재하지 않는지 (2) 만약 존재한다면, 그 데이터를 사용할 수 있는지 또는 사용할 수 없는지 라는 관점이 중요하다. 이러한 관점에서 데이터의 상태 다이어그램을 나타내면 다음의 <그림 2>와 같다.

데이터가 생성되기 이전의 상태는 absent 상태이다. 어떤 활동에 의해 create()되면 present 상태가 된다. 어떤 데이터가 present 상태에 있다는 것이 곧 available 한 상태를 의미하지는 않는다. 어떤 데이터가 available 한 상태에 있다는 것은 그 데이터가 실제 존재하면서도 그 데이터를 사용하고자 하는 사용자 또는 그의 활동이 '비즈니스 프로세스에서 정의된 역할에 의거' 하여 그 데이터에 접근하여 사용할 수 있는 권한을 가졌다는 것을 의미한다.

본 논문에서 데이터 의존성이란 어떤 상태에 있는 데이터가 다른 데이터가 어떠한 상태로 변경되기 위한 사전조건이 되는 경우에 발생하는 것으로 정의한다. 이러한 의미에서, 2개의 데이터 d_i와 d_j 간에 데이터 의존성, $d_i \rightarrow d_j$,이 정의되어 있는 경우에는 2가지 해석이 가능하다. 첫째는 타겟 데이터(target data) d_j 를 중심으로 d_j 가 존재하기 위한 사전 조건으로서 소스 데이터(source data) d_i 가 특정 상태에 있기를 요구하는 경우이다. 이 경우 d_i 의 상태에 따라 DAB(dependency of absence, DPR(dependency of presence), DAV(dependency of availability)의 3가지 형태로 나누어 볼 수 있다. 둘째는 d_i를 중심으로 d_i가 존재하기 위한 사후 조건으로 d_j 데이터의 업데이트가 요구되는 경우이다. 이를 본 논문에서는 DCO(dependency of conformance) 라고 부른다 (<그림 3> 참조).

어떤 데이터 d_j를 생성하기 위해서 다른 어떤 데이터 d_i가 가용해야 하는 경우에, 이 조건을 DAV(d_i, d_j)로 나타낸다. DPR(d_i, d_j)는 데이터 d_j 의 존재가 데이터 d_i 의 존재 여부에 의존한다는 것을 나타낸다. DPR 관계는 특히 데이터들이 생성되는 시점 간에 동기화 조건이 필요한 경우, 그 로직을 표현하는데 유용하다. 이와는 반대로, DAB는 어떤 데이터 d_j 가 존재하기 위해서는 데이터 d_m 이 존재하지 않아야 한다는 것을 의미하며, DAB(d_m, d_j)로 나타낸다. DAB는 고객의 주문 취소와 같은 외부 이벤트에 대한 처리 로직을 표현하기 위해 도입되었다. 예를 들어, 어떤 시점에서 '고객의 주문취소가 없으면 계속 프로세스를 진행한다' 등과 같은 프로세스 규칙을 나타내고자 할 때 사용한다. 그 외에도, 어떤 데이터가 필요한 시점에서 그 데이터가 가용하지 않은 경우 사용할 수 있는 대체 데이터를 표현하기 위해 사용할 수 있다.

이상의 의존성 타입들은 타겟 데이터의 존재를 위한 사전 조건으로서 소스 데이터의 상태를 규정하기 위한 역방향 의존성의 형태들이다. 반대로, 소스 데이터의 상태 변화가 다른 데이터들의 상태 변화를 강제하는 경우도 존재한다. 임직원 데이터와 그의 부양가족 리스트 데이터의 예를 본다면, 만약 임직원 데이터 (d_i)가 삭제되면, 즉 available 또는 present 상태에서 absent 상태로 변하면, 2개 데이터 간에 정의된 일관성 제약조건을 위반하지 않기 위하여 부양가족 리스트 데이터 (d_j)도 삭제되어야 한다. 이러한 경우, 이 2개 데이터 간의 의존성 관계는, DCO(d_i, d_j)라고 나타낸다. DCO는 순방향 의존성이 된다.

3.2 데이터 의존성 그래프

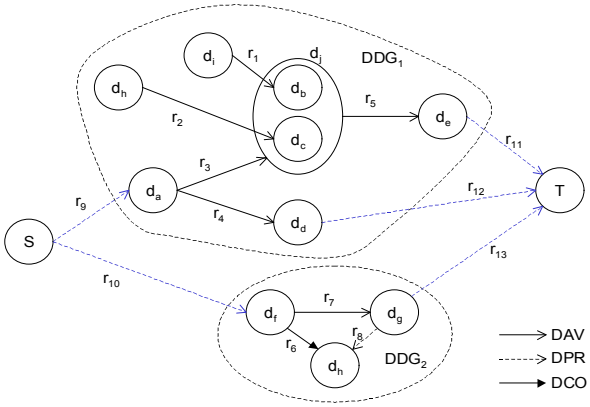
프로세스의 실행에 영향을 주거나, 프로세스의 실행의 결과로 생산되는 데이터와 같이 비즈니스 프로세스의 실행에 영향을 주는 데이터를 모두 식별하고, 그 들 간의 의존성 타입을 결정하게 되면, 1개 이상의 그래프를 얻게 된다. 본 논문에서는 이를 데이터 의존성 그래프 (Data Dependency Graph: DDG)라고 부른다. DDG는 데이터를 나타내는 노드, 데이터 간의 의존성을 나타내는 방향성 아크로 구성된 사이클이 없는 방향성 그래프로서 다음과 같이 정의된다.

정의 1 (DDG) DDG는 (D, DR, DT, RT) 의 4개 요소로 정의된다. $D = \{d_1, d_2, \cdots, d_N\}$ 로서 데이터 노드의 집합을 나타낸다. $DR \subseteq (D \times D)$ 은 데이터 의존성 관계의 집합을 나타낸다. $DT : D \rightarrow \{CID, CDD\}$ 는 각 데이터 노드의 속성을 결정하는 함수이다. *CDD* (Case Depenedent Data)는 프로세스 내부에서 생산된 데이터임을 나타내며, *CID* (Case Independent Data) 는 프로세스 외부에서 생산되어 전달된 데이터임을 나타낸다. $RT : R \rightarrow \{ DAV, DPR, DAB, DCO \}$ 는 DR 의 속성 (의존성 타입)을 결정하는 함수이다.

위의 정의에서 DT는 프로세스 간의 데이터 플로우를 표현하기 위해 도입되었다. CDD 는 현재의 설계 대상 프로세스의 범위 내에 있는 어떠한 활동에 의해서도 그 존재성에 대해 영향을 받지 않는 데이터로서, 여러 프로세스 간에 공유되는 데이터 또는 다른 프로세스에 의해 생산된 외부 데이터를 의미한다. 이와 반대로, CID 는 프로세스 내부의 활동에 의해 생성 또는 업데이트 되는, 일종의 내부 데이터를 의미한다.

어떤 비즈니스 프로세스의 데이터 요구사항으로서 위와 같은 정의에 의해 DDG를 만든 이후에, 새로운 2개의 더미 노드 S와 T를 도입하여 일종의 확장된 그래프를 얻게 된다. 이를 DDG+라고 한다면, DDG+ = $(D \cup \{S, T\}, DR \cup \{DPR(S, k) | k \in D_A\} \cup \{DPR(k, T) | k \in D_B\})$와 같이 정의된다. 여기에서, 내부 데이터 중 D_A 에 포함되는 데이터는 그 데이터와 역방향 의존성을 갖는 데이터가 전혀 없다는 것을 의미한다. 이러한 데이터에 해당하는 노드들은 S 노드와 연결된다. 반대로, 내부 데이터 중 D_B 에 포함되는 데이터는 프로세스의 최종 산출물에 해당한다. 즉, 프로세스가 성공적으로 실행되었다는 것은 D_B 에 포함되는 모든 데이터가 생산되어 존재해야 한다는 것으로 프로세스의 성공적 실행을 위한 사후조건에 해당된다. 이러한 데이터에 해당하는 노드들은 T 노드와 연결된다.

<그림 4>는 이러한 과정을 통해 만들어진 DDG의 예를 보이고 있다. 2개의 연결 그래프 DDG_1과 DDG_2로 구성되어 있던 그래프에 새로운 소스 노드 S를 도입하고, 역방향 의존성이 없는 노드들, 즉, d_a, d_f, d_h, d_i 중에서 내부 데이터인 d_a와 d_f를 S 노드와 DPR로 연결하고 있다. 또한, 새로 싱크노드(sink node) T를 도입하고 전체 프로세스의 완료를 위해 필요한 산출물인 데이터 노드들인 d_d, d_e, d_g를 DPR로 연결하고 있다. 여기서 데이터 노드 d_j는 다른 데이터 노드를 포함할 수 있는 노드 그룹 (node group)이다. 노드 그룹에 대해서는 다음 장에서 구체적으로 설명하도록 한다. 이후부터 DDG는 확장된 DDG+를 의미한다.



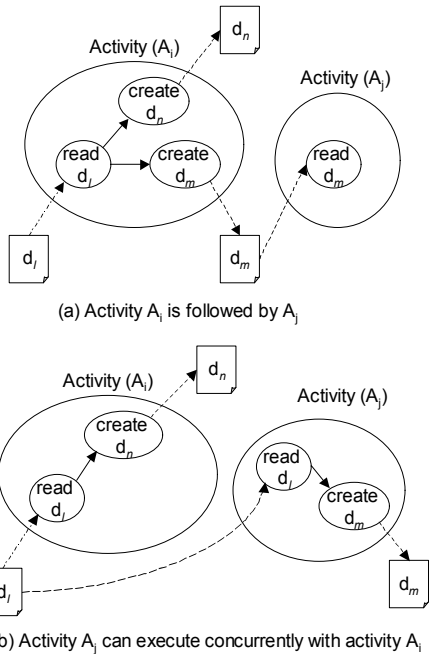
<그림 4> DDG의 예

4. DDG 기반의 프로세스 모델링

4.1 기본 개념

비즈니스 프로세스를 구성하는 개별 활동의 내부 로직에는 다양한 오퍼레이션들, 특히 해당 활동을 수행하는데 필요한 데이터를 처리하는 오퍼레이션들을 포함한다. 그러한 오퍼레이션으로는 (1) 활동 실행을 위해 필요한 데이터를 읽어 들인다거나 (2) 활동 실행의 결과로서 외부에 전달해야 하는 데이터를 생성한다거나 (3) 현재 존재하고 있는 어떤 데이터의 컨텐츠 또는 값을 업데이트 한다거나 (4) 어떤 필요성에 의해서 현재 존재하고 있는 데이터를 삭제한다거나 (5) 어떤 데이터에 대해 더 이상의 업데이트를 금지한다거나, 또는 (6) 어떤 데이터의 유효기간을 조정한다거나 하는 다양한 오퍼레이션들이 포함될 수 있다. 그리고, 각 오퍼레이션들은 그 오퍼레이션의 실행을 위해 필요한 입력 데이터와 오퍼레이션 실행의 결과로서 출력 데이터를 가진다. 여기에서 입력 데이터와 출력 데이터 간의 관계는 데이터 의존성으로 분석할 수 있다.

<그림 5>는 $d_f \rightarrow d_m$, $d_f \rightarrow d_n$ 등 2개의 데이터 의존성을 포함하고 있는 어떤 프로세스의 부분을 나타내고 있다. <그림 5>(a)는 이 2개 데이터 의존성이 활동 A_i에 할당됨으로 해서, A_i와 A_j 간에 시간적 선후관계가 성립되고 있는 경우를 보여 주고 있다. 반대로 <그림 5>(b)에서는 $d_f \rightarrow d_n$ 는 활동 A_i에서, $d_f \rightarrow d_m$ 은 활동 A_j 에서 구현되고 있음에 따라, 이 2개 활동들은 동시에 수행 가능하다는 것을 나타낸다. 이 예에서 볼 수 있듯이, 개별 활동에 데이터 의존성을 할당한 후에는, DDG 내에서 정의된 각 데이터 의존성 아크 간의 관계에 따라 활동들 간의 시간적 선후관계를 파악할 수 있다. 본 논문에서 1개 이상의 데이터 의존성을 하나의 활동에 할당하는 것을 활동 매핑이라고 부른다.

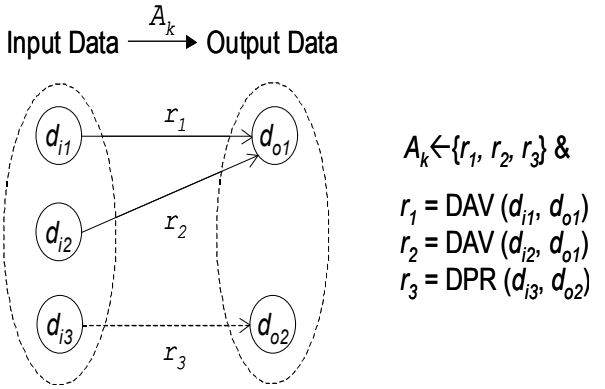


<그림 5> 프로세스 모델링에서 데이터 의존성의 의미

다시 말해서, 비즈니스 프로세스를 구성하는 개별 활동은 1개 이상의 데이터 의존성을 성립시키는 과정이라고 볼 수 있다. 이런 측면에서, DDG 에 기반한 비즈니스 프로세스 모델링이란, 개별 활동에 1개 이상의 데이터 의존성을 분배하는 과정으로 볼 수 있다. 즉, 비즈니스 프로세스를 구성하는 개별 활동은 DDG에서 정의된 1개 이상의 데이터 의존성과 매핑되며, 그러한 매핑 관계로부터 활동 들 간의 시간적 선후관계를 결정할 수 있다. 실행 타이밍 관점에서 2개의 활동 간의 관계는 동시에 수행할 수 있거나, 동시에 수행할 수 없거나, 또는 하나의 활동이 수행되면 다른 하나의 활동이 수행될 수 없는 3가지 관계 중에 하나가 된다.

4.2 활동 매핑: 활동 간의 선후 관계 분석

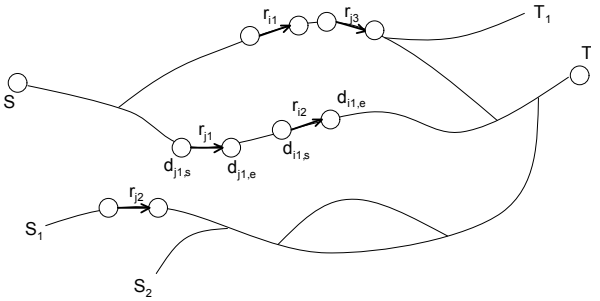
<그림 6>과 같이, 어떤 활동 A_k에 3개의 데이터 의존성, r_1 = DAV(d_{i1}, d_{o1}), r_2 = DAV(d_{i2}, d_{o1}), r_3 = DPR(d_{i3}, d_{o2}) 이 할당되었다고 하자. 이 경우 A_k는 3개의 데이터 d_{i1}, d_{i2}, d_{i3}를 입력 받아 2개의 데이터 d_{o1}, d_{o2}를 출력하는 활동라고 정의할 수 있다. 즉, A_k를 수행할 수 있는 사전조건은 d_{i1}, d_{i2} 가 available한 상태에 있고 d_{i3} 는 present 상태에 있는 것이다. 이러한 사전조건을 만족시켜 주는 활동들은 A_k에 앞서 실행되어야 한다. 그렇지 않으면 영원히 A_k를 실행할 수 있는 조건이 성립되지 않기 때문이다. 만약 d_{i1}, d_{i2}, 또는 d_{i3}를 생성하는 어떤 활동 A_j 가 있다면 A_j 는 반드시 A_k 에 선행하여 실행되어야 한다.



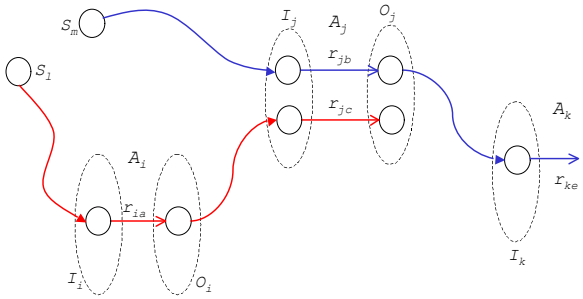
<그림 6> 활동 매핑

정리 1 DDG는 S 노드에서 T 노드에 이르는 하나 이상의 경로를 포함한다. 여기서, 2개의 활동 매핑 A_i←{r_{ik}, k =1,⋯,m}과 A_j←{r_{jl}, l =1,⋯,n}이 있다고 하자. 만약 임의의 r_{ia}∈{r_{ik}, k =1,⋯,m}와 임의의 r_{jb} ∈{r_{jl}, l =1,⋯,n}가 DDG를 구성하는 여러 개의 경로 중 임의의 동일한 경로 위에 있게 되면 이 2개 활동 간에는 선후관계가 존재한다.

(증명) 예를 들어, 다음의 <그림 7>에서 A_i←{r_{i1}, r_{i2}}, A_j←{r_{j1}, r_{j2}}라고 할 경우, A_i와 A_j 간에 선행/후속 관계가 있는지를 분석하기 위해서는 (r_{i1}, r_{j1}), (r_{i1}, r_{j2}), (r_{i2}, r_{j1}), (r_{i2}, r_{j2}) 등 4개의 데이터 의존성 페어(pair)에 대한 선후 분석이 이루어진다. 여기에서 선후분석이란 각 페어가 하나의 경로 상에 존재하는지를 찾아내는 것이다. <그림 7>의 예에서 보면 (r_{i2}, r_{j1}) 페어가 동일한 경로 상에 존재하며, r_{j1}이 r_{i2} 보다 S 노드에 가까이 있음을 알 수 있다 (이를 r_{j1}>r_{i2} 이라고 표시한다). 이는 r_{j1} 아크의 타겟노드인 $d_{j1,e}$ 이 present하거나 available한 상태가 되지 않으면, $d_{j1,e}$ 으로부터 T에 이르는 경로 상에 존재하는 어떠한 데이터 노드도 생성될 수 없다는 것을 의미한다. 따라서, A_i←{r_{ik}, k =1,⋯,m}과 A_j←{r_{jl}, l =1,⋯,n}이란 2개의 활동 매핑이 있다고 했을 때, 임의의 r_{ia}∈{r_{ik}, k =1,⋯,m}, 임의의 r_{jb} ∈{r_{jl}, l=1,⋯,n}가 동일한 경로 상에 존재하며 r_{jb}>r_{ja} 관계가 성립하면 활동 A_j는 활동 A_i에 선행된다. 즉, A_j>A_i의 관계가 성립된다. 단, 여기서 A_j←{r_{jl}, l =1,⋯,n}에 속하는 모든 의존성 아크는 A_i←{r_{ik}, k =1,⋯,m}에 대해 r_{jl}>r_{ik} 관계가 성립하던지 아니면, r_{jl} ‖ r_{ik} (‖는 선후관계가 없음을 나타낸다) 관계가 성립해야 한다. 만약 r_{jl}<r_{ik} 를 만족하는 r_{jl} 과 r_{ik}가 존재하면 2개 활동 간의 선후행 관계를 정의할 수 없다.



<그림 7> 활동 매핑 간의 선후행 관계

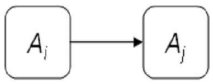


<그림 8> 활동 간의 선후관계

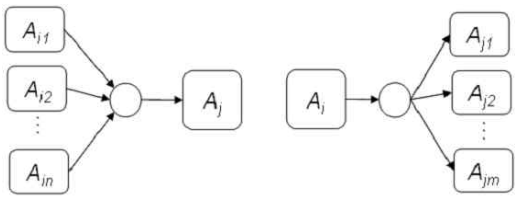
여기서 만약 r_{jb}의 타겟노드와 r_{ia}의 시작노드가 일치하면, A_i는 A_j에 즉시 후속한다 (immediately follow) 라고 하고 이를 $A_j \geq A_i$ 라고 표현한다. 또한, 2개의 활동 매핑, $A_i \leftarrow \{r_{ik},\ k=1,\cdots,m\}$와 $A_j \leftarrow \{r_{jl},\ l=1,\cdots,n\}$에 있어서 A_i를 구성하는 임의의 $r_{ia} \in \{r_{ik},\ k=1,\cdots,m\}$와 A_j를 구성하는 임의의 $r_{jb} \in \{r_{jl},\ l=1,\cdots,n\}$를 하나의 페어로 했을 때, 어떠한 $(r_{ia},\ r_{jb})$ 페어도 동일한 경로 상에 존재하지 않는다면, 이 2개 활동은 서로 간에 실행시작 조건에 영향을 주지 않는다. 즉, A_i와 A_j는 서로 독립적이다. <그림 7>에서 만약 $A_i \leftarrow \{r_{i1},\ r_{i2}\}$, $A_j \leftarrow \{r_{j2}\}$ 라고 한다면 이 2개 활동은 서로 독립적이다. 따라서, 동시에 수행 가능하다.

정리 2 $A_i > A_j$ 이고 $A_j > A_k$ 이면 $A_i > A_k$이다.

(증명) $A_i \leftarrow \{r_{ik},\ k=1,\cdots,n_i\}$, $A_j \leftarrow \{r_{jl},\ l=1,\cdots,n_j\}$, 그리고 $A_k \in \{r_{km},\ m=1,\cdots,n_k\}$이라고 하자. $A_i > A_j$ 조건으로부터, 정리 1에 따라, 임의의 소스노드 S_l로부터 시작하여 A_i를 구성하는 임의의 데이터 의존성 아크 r_{ia}를 경유, A_j를 구성하는 임의의 데이터 의존성 아크 r_{jc}에 이르는 경로가 존재함을 알 수 있다. 또한, $A_j > A_k$ 조건으로부터, 임의의 소스노드 S_m으로부터 시작하여 A_j를 구성하는 임의의 데이터 의존성 아크 r_{jb}를 경유, A_k를 구성하는 임의의 데이터 의존성 아크 r_{ke} 에 이르는 경로가 존재함을 알 수 있다. 여기서, I_j, O_j 를 각각 $r_{jb} \in \{r_{jk},\ k=1,\cdots,n_1\}$의 소스노드에 해당하는 데이터 노드들의 집합, 타겟노드에 해당하는 데이터 노드들의 집합으로 정의하자 (<그림 8> 참조). 여기에서 $A_i > A_j$ 조건은 I_j에 포함되는 데이터 중에 A_i의 실행이 종료되어야 가용해 지는 데이터가 1개 이상 존재한다는 것을 의미한다. 또한, $A_j > A_k$ 조건은 A_j의 실행이 완료되어 O_j에 속한 데이터들이 가용해야만 A_k가 실행될 수 있음을 의미한다. 결국 A_i가 실행되지 않으면, I_j에 포함되는 일부 데이터가 존재하지 않게 되므로 활동 A_j가 실행될 수 없다. 또한 A_j가 실행되지 않으면, O_j에 속한 데이터들이 존재하지 않게 되므로 결국 A_k를 실행할 수 없게 된다. 따라서, A_k가 실행되기 위해서는 A_i가 이미 수행되었어야 한다. 즉, $A_i > A_k$ 관계가 성립된다. □



<그림 9> 활동 간의 관계 : 시퀀스



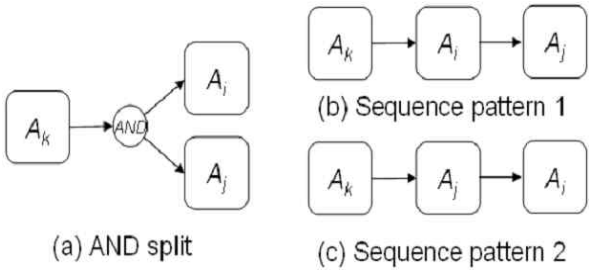
<그림 10> 활동 간의 관계 : 결합과 분기

4.3 프로세스 패턴

어떤 활동과 즉시후속 관계를 가지는 활동이 오직 1개인 경우에 그 2개 활동 간에는 <그림 9>의 시퀀스 패턴으로 나타나게 된다. 반면에, 어떤 활동이 2개 이상의 다른 활동들과 즉시후속 관계를 가지는 경우에는 그 들 사이에 결합(join) 관계 (<그림 10>(a)) 또는 분기(split) 관계가 (<그림 10>(b)) 성립된다. 결합과 분기관계는 내부 로직에 따라 AND 또는 OR 타입으로 나누어 볼 수 있다.

4.3.1 AND 결합/분기

어떤 활동이 A_i가 활동 A_j와 동시에 수행 가능한 경우에는 또 다른 어떤 활동 A_k와 AND 결합 또는 분기 관



<그림 11> 프로세스의 설계 대안

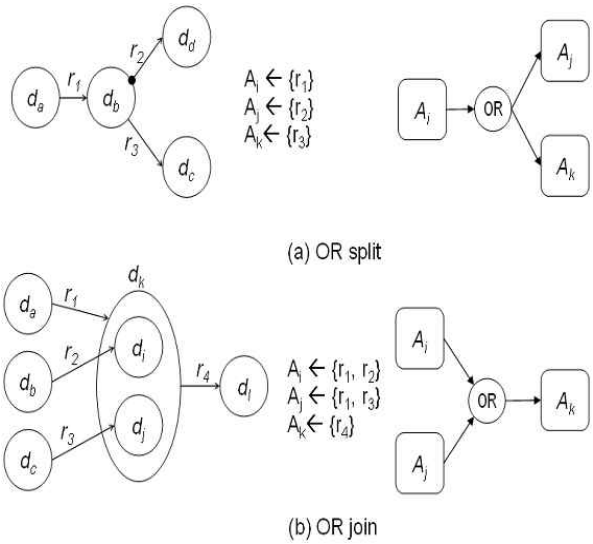
계로 나타난다. 여기에서 A_k는 A_i와 A_j의 시작 타이밍(AND 결합의 경우) 또는 종료 타이밍(AND 분기의 경우)을 결정하는 역할을 할 뿐이다. 예를 들어, $A_k > A_i$, $A_k > A_j$, $A_i \| A_j$ 등의 관계가 성립하는 경우에는 다음의 <그림 11>에서 보는 것처럼 다양한 프로세스 대안이 존재할 수 있다. 즉, AND 결합이나 분기는 프로세스 설계상 옵션으로 활용되는 것이지, 그것이 반드시 프로세스 실행상의 규칙이 되지는 않는다는 뜻이다. 물론, $A_k \geq A_i$, $A_k \geq A_j$, $A_i \| A_j$ 과 같이 즉시 후속의 관계로 한정되는 경우에는 <그림 11>(a) 형태만 가능한 대안이 된다.

4.3.2 OR 결합/분기

OR 결합과 분기가 있는 경우에는 프로세스가 2개 이상의 시나리오를 포함하게 되며, AND의 경우와는 다르게, 실제적인 프로세스 규칙으로 작용한다. OR 관계는 2개 이상의 데이터 간에 상호 배타적 관계 또는 대체 관계가 성립할 때 발생한다. 상호 배타적 관계로 인한 OR 관계는 어떤 데이터의 존재 또는 가용성 여부에 따라 후속 활동이 결정되는 경우이다. 이러한 경우를 DDG에서는 <그림 12>(a)와 같이 나타낼 수 있다. <그림 12>(a)는 A_i가 정상적으로 실행 완료되어 d_b가 가용해진 경우에는 A_k를 실행하고, 그렇지 않은 경우에는 A_j를 실행하여 d_c (예를 들어, 에러 메시지) 를 생성하는 로직을 보이고 있다. 즉, 선행 활동의 실행 결과에 따라 프로세스의 진행 방향이 달라지는 것, 즉 후속 활동이 달라지는 것으로 후속 활동들 간의 동기화에 OR 분기 규칙이 사용된다. DDG에서는 이러한 상황을 DAB 를 이용하여 모델링할 수 있다.

반면에, OR 결합을 위한 모델링 요소로서 고려할 수 있는 것은 '우선순위 없는 대체'이다. 여기서, 대체라는 것은 어떤 데이터가 존재하지 않을 경우, 다른 데이터를 대신 사용할 수 있다는 것을 의미하며, 우선순위가 없다는 것은 대체되는 데이터들 간에 선호되는 순위가 없다는 것을 말한다. 이러한 개념을 나타내기 위하여 DDG에 노드 그룹 이라는 모델링 요소를 포함하도록 시맨틱을 확장하였다. <그림 12>(b)는 데이터 d_f을 생성하기 위해서 데이터 d_i 또는 d_j 중의 하나가 필요한 경우를 나타내고 있다. 여기에서 d_i와 d_j는 d_f을 생성하기 위한 대안적 관계에 있다고 할 수 있다. 이러한 관계를 표현하는 것이 노드그룹 d_k와 아크 $d_k \rightarrow d_f$이다.

여기서, 노드그룹 d_k로 표현되는 데이터는 객체지향에서의 일반화된 데이터 타입과 동일한 개념으로 생각할 수 있으며, 실제 실행 시에 d_k는 d_i또는 d_j로 동적 바인딩될 수 있다. 즉, d_i가 가용하면 d_k는 d_i 로 바인딩되며, d_j가 가용하면 d_k는 d_j 로 바인딩될 수 있음을 나타낸다.

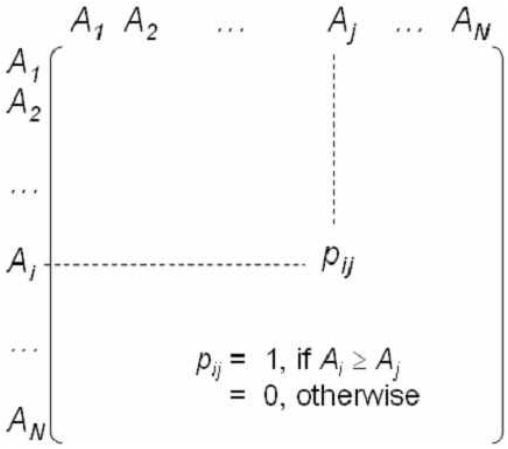


<그림 12> OR 결합과 OR 분기

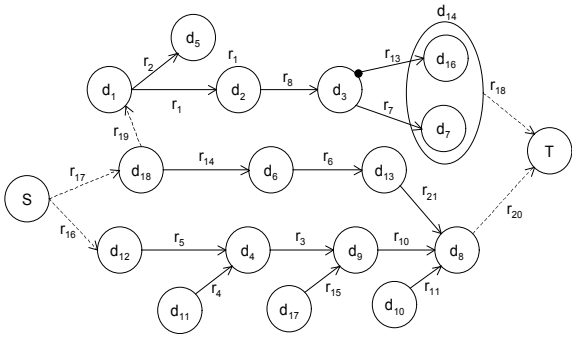
4.4 비즈니스 프로세스 설계

모든 활동 매핑 페어에 대해 위와 같은 분석과정을 수행하고 나면 각 페어에 대해, 그 페어를 구성하는 2개 활동 간에 즉시후속 관계가 존재하는지, 아니면 서로 독립적인지를 확인할 수 있다. 이 과정을 통해 얻어진 2개 활동들 간의 선행/후속 관계들 간에는, 정리 2에서 살펴본 것처럼, 이행성이 성립된다. 이와 같이 2개 활동 간의 선행/후속 관계와 그 관계 간에 성립하는 이행성을 고려하면 전체 활동들 간의 전체 선행/후속 관계를 결정할 수 있게 되는데, 이는 <그림 13>과 같은 순위 매트릭스, PM = $(p_{ij})_{N \times N}$ 으로 표현할 수 있다.

PM의 각 행과 열은 하나의 활동과 대응된다. PM을 구성하는 각 셀 p_{ij}의 값은, 만약 $A_i \geq A_j$ 이면 1이 되고 그렇지 않으면 0이 된다. PM의 패턴과 프로세스 모델링 요소를 비교해 보면 다음과 같다. 첫째, 어떤 행 A_i에



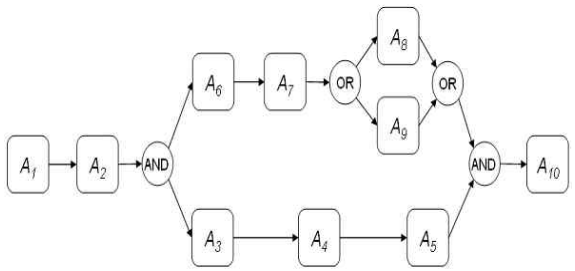
<그림 13> 순위 매트릭스



(a) DDG의 예

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
A_1		1								
A_2			1			1				
A_3				1						
A_4					1					
A_5										1
A_6							1			
A_7								1	1	
A_8										1
A_9										1
A_{10}										

(b) 순위 매트릭스



(c) 프로세스 모델

<그림 14> 프로세스 모델 설계의 예

대해, p_{ij}=1인 셀이 오직 하나 존재하면 활동 A_i와 활동 A_j는 시퀀스 관계에 있다. 둘째, 어떤 행 A_i에 대해, p_{ij}=1인 셀이 2개 이상 존재하면, 활동 A_i와 p_{ij}=1인 셀의 열에 해당하는 활동들 간에 AND 또는 OR 분기관계에 있다. 셋째, 어떤 열 A_j에 대해, p_{ij}=1인 셀이 2개 이상 존재하면, 활동 A_j와 p_{ij}=1인 셀의 행에 해당하는 활동들 간에 AND 또는 OR 결합 관계에 있다. 결합이나 분기 관계가 있는 경우, 그 패턴이 AND가 되는지 OR가 되는지는 앞의 4.3절에서 설명한 바에 따라 결정된다.

4.5 프로세스 설계의 예

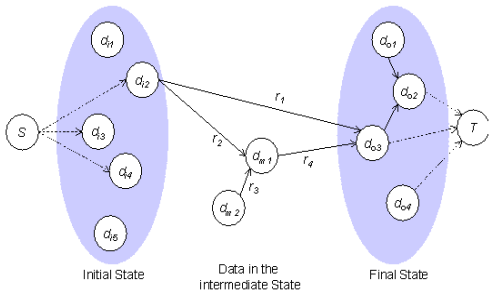
다음의 <그림 14>(a)는 하나의 DDG 예를 보여 주고 있다. 이 DDG에 대해 만약 활동 매핑을 $A_1 \leftarrow \{r_{16}\}$, $A_2 \leftarrow \{r_4, r_5, r_{17}\}$, $A_3 \leftarrow \{r_3, r_{14}\}$, $A_4 \leftarrow \{r_6, r_{15}\}$, $A_5 \leftarrow \{r_{10}, r_{11}, r_{21}\}$, $A_6 \leftarrow \{r_1, r_2, r_{19}\}$, $A_7 \leftarrow \{r_8\}$, $A_8 \leftarrow \{r_7\}$, $A_9 \leftarrow \{r_{13}\}$, $A_{10} \leftarrow \{r_{18}, r_{20}\}$ 와 같이 하게 되면, 그로부터 만들어 지는 PM은 <그림 14>(b)와 같이 결정된다. 그리고 그로부터 유일하게 결정되어지는 프로세스 모델은 <그림 14>(c)와 같다.

4.6 DDG 기반의 프로세스 설계

본 논문에서 제안하고 있는 데이터 의존성 그래프는 설계 대상이 되는 비즈니스 프로세스의 실행 결과물로 정의되는 데이터들과 그 프로세스에 입력되는 데이터들을 분석하는 과정을 통해 만들어진다. IDEF0 또는 DFD의 문맥 다이어그램을 도출하는 과정과 동일하다.

설계 대상 비즈니스 프로세스의 초기 입력 데이터와 최종 출력 데이터를 결정하게 되면, <그림 15> 에서 보는 바와 같이 프로세스의 초기 상태와 최종 상태를 결정할 수 있게 된다. 여기에서 프로세스의 초기 상태란, 설계대상 비즈니스 프로세스의 입력 데이터 중에서 외부 데이터와 내부 데이터를 구분하고, 내부 데이터인 경우에 소스 노드 S에 대해 DPR 아크를 연결하고, 만약 초기 입력 데이터 들 간에 데이터 의존성이 존재하면 그를 표시하는 것으로 설정된다. 마찬가지로, 프로세스의 최종상태란 프로세스의 최종 출력 데이터 중에서, 프로세스를 완료하기 위해 필요한 데이터들을 식별하고 이를 싱크 노드 T와 DPR 아크로 연결하는 것으로 설정된다. 물론, 최종 출력 데이터 간에 성립되는 데이터 의존성은 표현되어야 한다. 그럼 결국, DDG를 설계한다는 것은 프로세스의 초기 상태를 최종 상태로 이행하기 위한 중간 데이터와 그들 간의 데이터 의존성을 밝히는 과정으로 볼 수 있다.

만약 d_{o3}의 생산이 d_{i2}에만 전적으로 달려 있다면 r_1:$d_{i2} \rightarrow d_{o3}$를 정의하는 것으로 데이터 요구사항 식별이 완료될 수 있겠지만, 이러한 변환 과정 중에 존재할 수 있는 중간 데이터 d_{m1}, d_{m2}를 식별하고 이들 간에 존재하는 추가적인 데이터 의존성(r_2:$d_{i2} \rightarrow d_{m1}$, r_3:$d_{m2} \rightarrow d_{m1}$,그리고 r_4:$d_{m1} \rightarrow d_{o3}$ 등)을 식별한다는 것은 프로세스 컨트롤의 상세화 수준을 높이는 과정이라고 볼 수 있다.



<그림 15> 데이터 요구사항으로서의 DDG

관련연구에서 소개했던 Vanderfeesten([10])의 연구결과와 비교해 보면, 그 연구에서는 최종상태에 있는 데이터와 그 데이터 들 간의 수직적 포함관계 만을 고려한 반면, DDG는 프로세스 문맥 내에서 데이터의 상태 변화를 함께 고려할 수 있다는 장점을 가지고 있다. 여기에서 수직적 포함관계란 어떤 데이터가 다른 데이터의 일부분으로 포함되는 제약조건을 의미한다.

이로부터 분석되어지는 활동은 그러한 수직적 포함관계를 실현시키는 활동으로서 그 관계에 의해 정의되는 데이터 요소들을 합하거나 나누는 활동 만으로 제한되어진다.

1.2절에서 논의되었던 데이터 플로우 중심의 접근방법과 비교하였을 때 DDG 기반 접근방법의 장점을 정리하면 다음과 같다. 첫째, 데이터의 가용성 측면에서의 문제점은 나타나지 않는다. DDG 로부터 설계된 프로세스는 개별 활동의 실행을 위한 사전 조건을 완전하게 만족시킬 수 있도록 설계되기 때문이다. 따라서, 어떤 활동을 시작해야 하는 시점에서 사전조건이 완비되지 않아 프로세스 실행 상에 지연이 발생하는 경우를 사전에 방지할 수 있다. 이는 DDG 기반 프로세스 모델링의 핵심적인 장점이 된다.

둘째, 무결성 측면의 경우에는 데이터의 의존성 타입 중 DCO 관계의 정의에 의해 일부 해결가능한 부분이 있다. 또한, 본 논문에서 분명하게 제시하고 있지는 않지만, DDG 자체가 프로세스 실행 상태를 표현할 수 있기 때문에 일종의 프로세스 실행 문맥으로 작용할 수 있다는 점에서 해결방안을 찾을 수 있다. 여기에서 프로세스 실행 문맥이란, 현재 실행 중인 프로세스의 상태는 그 프로세스에 의해 현재까지 생성된 데이터의 집합으로 표현될 수 있다는 의미이다. 만약 어느 특정 시점에서, 예를 들어, 프로세스 진행 상 데이터 d_i 가 가용하지 않은 상태에서 데이터 d_j 는 결코 가용한 상태가 되어서는 안된다 라는 규칙이 필요하다면 이러한 규칙을 추가적으로 정의할 수 있도록 확장 가능하다.

셋째, 데이터의 신뢰성 측면의 해결 방안은 분명하지 않다. 왜냐하면, 데이터의 신뢰성 자체가 그 데이터를 생산한 활동이 얼마나 정확하게 수행되었느냐에 의해 결정되는 부분이기 때문이다. 하지만, DDG에 포함된 데이터 요소에 대해 신뢰성을 위한 일종의 가드(guard) 조건을 설정할 수 있도록 확장한다면 일정 부분 해결할 수 있을 것으로 보인다.

5. 결 론

본 논문에서는 프로세스의 설계에 데이터 요구사항을 반영할 수 있는 하나의 방법론으로서 데이터 의존성 개념을 제안하고, 그로부터 활동 매핑이라는 메카니즘을 통해 프로세스 모델을 설계하는 과정을 살펴보았다. 비즈니스 프로세스는 다른 분야의 프로세스와는 다르게 하나의 활동이 수행되는 시간이 길고, 그 내부에 작업자의 의사결정 과정이 포함되어 있으며, 다양한 예외상황을 만날 수 있다는 차이점을 가지고 있다. 따라서, 효과적인 비즈니스 프로세스 모델링을 위해서는 프로세스의 실행 상태에 따라 발생할 수 있는 다양한 시나리오를 분석할 수 있는 방안이 필요하다. 본 논문에서는 데이터 요구사항으로 표현된 데이터 의존성 그래프에 대해 액티비티 매핑을 설정함으로써 프로세스 모델을 일의적으로 설계할 수 있음을 보였다.

본 논문에서 제안하고 있는 접근 방법은 설계자의 의도 (데이터 의존성과 데이터 매핑으로 반영됨)에 따라 다양한 형태의 시나리오를 도출해 낼 수 있는 구조를 가지고 있다. 프로세스 설계자의 입장에서 보면 데이터 의존성이란 하나의 설계 제약조건으로 작용한다.

제약조건을 줄일수록 대안적 시나리오를 많이 가지는 프로세스 모델을 설계할 수 있다. 심지어, 프로세스 실행 중에 프로세스의 상태에 맞추어 데이터 의존성을 변경함으로써 전체 프로세스를 보다 효율적으로 실행할 수 있는 방안을 찾을 수도 있다.

본 논문의 관점에서 비즈니스 프로세스 설계란 활동 간에 내재적으로 정의되는 데이터 의존성을 위배하지 않으면서 프로세스의 목적을 달성할 수 있는 컨트롤 플로우를 찾아내는 과정이 된다. 다시 말하면, 잘 설계된 비즈니스 프로세스란 그 목적을 달성하는데 필요한 데이터 또는 그 들 간의 관계가 프로세스 실행 과정에서 모순 또는 충돌이 발생하지 않도록 유연하게 설계된 프로세스를 말하는 것이다. DDG는 데이터 자원의 효율적 관리가 가능하도록 프로세스를 설계하고자 한다는 점에서 최선의 프로세스를 설계하기 위한 설계 메커니즘을 제공할 수 있다.

또한, DDG는 단순하게 설계 대상 프로세스와 관련된 데이터 만을 식별하는 것이 아니라, 그 데이터 들이 프로세스와 어떻게 동기화되어 사용되는지에 대한 정보를 함께 포함한다. 따라서, 실제로 프로세스를 데이터 측면으로 투사한 형태의 결과물로 볼 수 있다. DDG의 이러한 특성 때문에, DDG는 프로세스 설계 시에는 설계 대상 워크플로우의 데이터 요구사항을 표현하는 도구로, 프로세스 실행 시에는 프로세스의 실행 상태를 표현하는 도구로, 그리고 프로세스 변경 시에는 변경 대상 프로세스와 데이터 요구사항 간의 일관성을 보장하는 도구로 사용될 수 있을 것으로 기대하고 있으며 이를 위한 후속연구를 진행 중에 있다.

6. 참 고 문 헌

[1] 김학수, 박찬희, 설주영, 손진현, 컨트롤 흐름 경로 기반의 비즈니스 프로세스 타당성 검증기법, 정보처리학회논문지 D, Vol.14, No.5, p.531-544, 2007.

[2] 문미경, 가변성 결정기반 BPM 생성을 위한 가변성 의존관계 분석, 정보처리학회지D, Vol.16, No.5, p.791-800, 2008.

[3] A. Abecker, A. Bernardi, H. Maus, M. Sintek, C. Wenzel, Information supply for business processes: coupling workflow with document analysis and information retrieval, Knowledge- Based Systems13, p. 271-284, 2000

[4] A. Orso, S. Sinha, and M.J. Harrold, Classifying Data Dependencies in the Presence of Pointers for Program Comprehension, Testing, and Debugging, ACM Transactions on Software Engineering and Methodology, Vol.13, No.2, p.199-239, 2004.

[5] C. Hagen and G. Alonso, Exception Handling in Workflow Management Systems, IEEE Transactions on Software Engineering, Vol.26, No.10, p.943-958, 2000.

[6] C. Petrie and S. Sarin, Beyond Documents : Sharing work, IEEE Internet Computing, Vol.4, No.3, p.34-36, 2000.

[7] C. Marcelo, M.Audris, R. Jefferey, H. James, Software dependencies, Work dependencies, and Their Impact on Failures, IEEE Transactions on Software Engineering, Vol. 35 Issue 6, p.864-878, 2009.

[8] Fabio Casati, Stefano Ceri, Stefano Paraboschi, Giuseppe Pozzi, Specification and Implementation of Exceptions in Workflow Management Systems, ACM Trans. On Database Systems, Vol.24, No.3, p.405-451, 1999.

[9] G. Cugola, Inconsistencies and Deviation in Process Support Systems, Ph.D Thesis, Politecnico Di Milano, 1997.

[10] I. Vanderfeesten, H. Reijers, W. van der Aalst, Product Based Workflow Support : Dynamic Workflow Execution, CAiSE 2008:571-574.

[11] K. Crowston and Charles Osborn, A coordination theory approaches to process description and redesign, MIT Sloan School of Management, July 1998.

[12] K. Malone, The Interdisciplinary Study of Coordination, ACM Computing Survey, Vol.26, No.1, p.87-119, 1994.

[13] M. Michael, I. Marta, Modeling languages for business processes and business rules: A representative analysis, Information Systems, Vol. 35 Issue 4, p379-390, 2010.

[14] S. Sadiq, M. Orlowska, W. Sadiq, and C. Foulger, Data Flow and Validation in Workflow Modelling, The Fifteenth Australasian Database Conference Dunedin, NewZealand (ADC'04), p.18-22, 2004.

[15] T.W.Malone, K.Crowston, G.Herman, Organizing Business Knowledge: The MIT Process Handbook, MIT Press, 2003.

[16] W.M. Johnston, J.R. Paul hanna, and R.J. Millar, Advances in Dataflow Programming Languages, ACM Computing Surveys, Vol.36, No.1, March p.1-34, 2004.

[17] W.M.P. van der Aalst, On the automatic generation of workflow processes based on product structures, Computers in Industry 39, p.97-111, 1999.

[18] W.M.P. van der Aalst, S. Jablonski, Dealing with workflow change : Identification of issues and solutions, I.J. of Computer Systems Science and Engineering, 2000.

[19] W.M.P. van der Aalst and S.Jablonski, Dealing with workflow change: identification of issues and solutions, International Journal of Computer Systems Science & Engineering, Vol.5, p267-276, 2000.

[20] W.M.P. van der Aalst, A.H.M. ter Hofstede, YAWL: yet another workflow language, Information Systems, vol. 30, p.245-275, 2005.

[21] W.M.P. van der Aalst, Mathias Weske, Dolf Grünbauer, Case handling : a new paradigm for business process support, Data & Knowledge Engineering 53, p.129-162, 2005

저 자 소 개

장 무 경

서울대학교 산업공학과에서 학사, POSTECH 산업공학과에서 석사, POSTECH 산업경영공학과에서 박사를 취득하였으며 현재 남서울대학교 산업경영공학과 전임강사로 재직중.

전공은 경영정보시스템(MIS)이며 관심분야는 BPMS, 워크플로우, 소프트웨어 품질, 임베디드소프트웨어 등이다.

주소: 천안시 서북구 성환읍 매주리 21 남서울대학교 산업경영공학과