업무 프로세스와 데이터 요구사항의 통합 모델링
A Modeling Approach to Integrate Business Processes and Data Requirements

장 무 경*

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 design processes to support uncertainties from internal or external environments. 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, Process Modeling

1. 서 론

비즈니스 프로세스란 그 프로세스를 실행하고자 하는 어떤 조직의 특정한 목적을 달성하기 위하여 필요한 단위 액티비티 (activity) 들을 식별하고, 프로세스의 목적을 효과적으로 달성할 수 있도록 그 액티비티들 간의 상호작용을 컨트롤할 수 있도록 묶어 주는 단위이다. 기업 내에 인적, 시간적, 지리적으로 분산되어 있는 단위 업무들을 하나의 프로세스로 묶는 가장 기본적인 개념은 단위 업무들 간에 존재하는 “flow of something” 을 찾아내는 것이다. 그것을 이벤트로 할 것인지 아니면 데이터, 또는 메시지로 할 것인가에 따라 모델링 방법론은 많은 차이를 보이게 된다. 대상 도메인의 특성에 따라 또는 프로세스의 실행 방법에 따라 다양한 형태의 프로세스 모델이 사용되고 있으며, 일반적으로 상태 기반, 액티비티 기반, 커뮤니케이션 기반으로 구분될 수 있다.

* 남서울대학교 산업경영공학과

조직의 업무 프로세스를 설계하고 표현하는데 가장 일반적으로 활용되고 있는 액티비티 중심의 접근 방법에서 가장 문제가 되는 것은 설계된 프로세스의 실행 가능성을 사전에 평가하기가 어렵다는 것이다. 이 문제가 단순한 문제가 아닌 것은, 비즈니스 프로세스는 소프트웨어 프로세스, 생산이나 통신 프로세스와는 다르게 다양한 불확실성을 가지고 있고 그에 따라 프로세스 실행 중에 작업자가 프로세스의 실행 루트를 변경하게끔 만드는 의사결정을 하게 된다는 점 때문이다. 즉, 설계된 프로세스란 하나의 정의적 관점이고, 실제 실행에 있어서는 다양한 시나리오로 나타날 수 있다는 점이다. 현재의 액티비티 중심의 접근 방법은 이러한 측면에 대한 고려가 어렵다. 이러한 측면에 대한 고려가 부족하면 실제의 프로세스와 시스템 상의 프로세스 간에 불일치가 발생하여 프로세스 실행결과에 일관성을 잃게 되는 중요한 요인이 될 수 있으며 [1, 5], 이와 같은 오류는 전체 시스템의 실행 오류 중 가장 많은 부분을 차지하고 있다 [4].

이러한 문제점을 해결하기 위해서는 결국 프로세스의 실행 상태를 표현할 수 있는 방안을 찾아야 하는데, 관련 연구들을 살펴보면 프로세스의 실행 상태에 가장 큰 영향을 줄 수 있는 부분으로서 데이터에 대한 고려가 중심이 되고 있다. Sadiq [9]의 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.

이러한 관점의 연장선상에서, 본 논문에서는 프로세스의 실행 특성에 영향을 줄 수 있는 요인으로 데이터 요구사항을 고려하고, 이로부터 프로세스와 데이터 요구사항 간의 일관성을 높일 수 있는 방안을 제시하고자 한다. 프로세스의 데이터 요구사항을 데이터 의존성 (data dependency)이라는 개념으로 정형화 한 이후에, 데이터 요구사항으로서의 데이터 의존성을 만족하는 프로세스 설계는 어떻게 이루어질 수 있는지를 살펴보고자 한다.

2. 관련 연구

비즈니스 프로세스를 구성하는 액티비티들 간의 선후관계(precedence relationship)는 그 들 사이에서 공유되는 자원의 특성으로부터 나온다 [2]. 예를 들어, 한 사람이 수행하기로 되어 있는 2개의 업무는 동시에 수행될 수 없다 라던가 또는 어떤 데이터를 생산하는 액티비티는 그 데이터를 사용하는 액티비티보다 앞서 수행되어야 한다 등이 좋은 예가 된다. 이와 같이 공유자원을 고려한 프로세스 모델링 접근 방법으로, 액티비티들 간에 coordination이 필요한 이유를 액티비티들이 공유하는 자원에 대한 의존성(dependency)으로 보고 의존성의 taxonomy를 제안한 Malone 의 연구[8]가 있었다. Malone은 이 연구에서, dependency의 타입을 shared resource, producer/consumer relationships, simultaneity constraints, task/subtask 등의 4가지로 나누었다. Crowston 등[7]은 그들의 보고서에서 어떤 태스크 (task)의 수행에 필요한 자원을 그 태스크의 사전조건 (preconditions) 으로 정의하고, 만약 2개 이상의 태스크들 간에 사전조건들이 서로 겹치면 그 들 간의 dependency를 coordination하기 위한 추가적인 컨트롤이 필요함을 지

적하였다. 이들의 연구는 이후 MIT process handbook [10] 의 프로세스 모델링 아키텍처를 제안하는데 활용되었다.

비즈니스 프로세스 설계 시에 데이터 요구사항을 고려하고자 했던 시도들은 document flow를 지원하는 워크플로우 [3] 또는 폼기반 워크플로우 [12] 연구 등을 중심으로 진행되어 왔으나, 이들 연구들은 프로세스 지향의 워크플로우를 설계하기 위한 연구라기보다는 데이터 중심의 애플리케이션 개발을 위한 연구였다. Aalst 등[12]은 워크플로우 설계를 위한 새로운 패러다임으로서 case handling 을 제안하였다. Case handling이란 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 연구[11]에서, 워크플로우 설계 시에, 워크플로우의 최종 산출물이 되는 product data를 고려하여야 함을 주장하였다. 그 연구의 후속으로 Vanderfeesten[6]의 연구가 있었는데, 이 연구의 핵심은 프로세스가 성공적으로 수행이 완료되면 얻어질 수 있는 하나의 데이터 구조 (수직적 포함관계) 를 정의하면 그 구조로부터 프로세스를 설계할 수 있다는 것이다.

3. 데이터 의존성 그래프 (Data Dependency Graph)

3.1 데이터 의존성

본 논문의 관점에서 데이터 간의 의존성이란, 간단하게 말해서, 데이터 dj를 생성하기 위해서 다른 어떤 데이터 di 가 필요한 경우에 데이터 dj의 생성 여부는 데이터 di에 의존한다는 의미이다. 본 논문에서는 2개 데이터 간의 데이터 의존성을 다음과 같이 4개의 유형으로 분류한다.

- DAV(di, dj) : 타겟 데이터 dj가 생성되기 위해서 소스 데이터 di가 available 상태에 있어야 한다는 제약조건을 표현함.
- DPR(di, dj) : 타겟 데이터 dj가 생성되기 위해서 소스 데이터 di가 present 상태에 있어야 한다는 제약조건을 표현함.
- DAB(di, dj) : 타겟 데이터 dj가 생성되기 위해서 소스 데이터 di가 absent 상태에 있어야 한다는 제약조건을 표현함.
- DCO(di, dj) : 소스 데이터 di의 상태나 값이 변경되면, 그에 따라 타겟 데이터 dj의 상태가 변경되어야 한다는 제약조건을 표현함.

3.2 데이터 의존성 그래프

비즈니스 프로세스의 범위 내에 포함되는 데이터 (즉, 그 프로세스의 실행에 영향을 주거나, 그 프로세스의 실행의 결과로 생산되는 모든 데이터)를 모두 식별하고, 그 들 간의 의존성 타입을 결정하게 되면, 1개 이상의 그래프를 얻게 된다. 본 논문에서는,

이를 데이터 의존성 그래프 (DDG: Data Dependency Graph)라고 부른다. DDG는 데이터를 나타내는 노드, 데이터 간의 의존성을 나타내는 방향성 아크로 구성된 싸이클(cycle)이 없는 방향성 그래프로서 다음과 같이 정의된다.

정의1 (데이터 의존성 그래프) DDG는 (D, DR, DT, RT) 의 4개 요소로 정의된다. D = {d1, d2, …, dN} 로서 데이터 노드의 집합을 나타낸다. DR ⊆ (D × D) 은 데이터 의존성 관계의 집합을 나타낸다. DT : D → {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 → { DAV, DPR, DAB, DCO } 는 DR 의 속성 (의존성 타입)을 결정하는 함수이다.

위의 정의에서 DT는 프로세스 간의 데이터 플로우 (data flow) 를 표현하기 위해 도입되었다. CDD는 현재의 설계 대상 프로세스의 범위 내에 있는 어떠한 액티비티에 의해서도 그 존재성을 영향받지 않는 데이터로서, 여러 프로세스 간에 공유되는 데이터 또는 다른 프로세스에 의해 생산된 데이터를 의미한다. 이와 반대로, CID는 프로세스 내부의 액티비티에 의해 생성 또는 업데이트 되는 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∪{S, T}, DR∪{DPR(S, k)| k∈DA } ∪{DPR(k, T)| k∈DB })와 같이 정의된다. 여기에서, 내부 데이터 중 DA 에 포함되는 데이터는 그 데이터를 생산하기 위해 필요한 데이터가 전혀 없다는 것을 의미한다. 이러한 데이터에 해당하는 노드를 새로 도입한 S 노드와 연결하게 된다. 반대로, 내부 데이터 중 DB 에 포함되는 데이터는 프로세스의 최종 산출물에 해당한다. 즉, 프로세스가 성공적으로 실행되었다는 것은 DB 에 포함되는 모든 데이터가 생산되어 존재하는 것과 같은 의미가 된다. 이러한 데이터에 해당하는 노드들은 새로 도입한 T 노드와 연결된다. DDG+ 의 예는 그림 4 (a)를 참조하기 바란다.

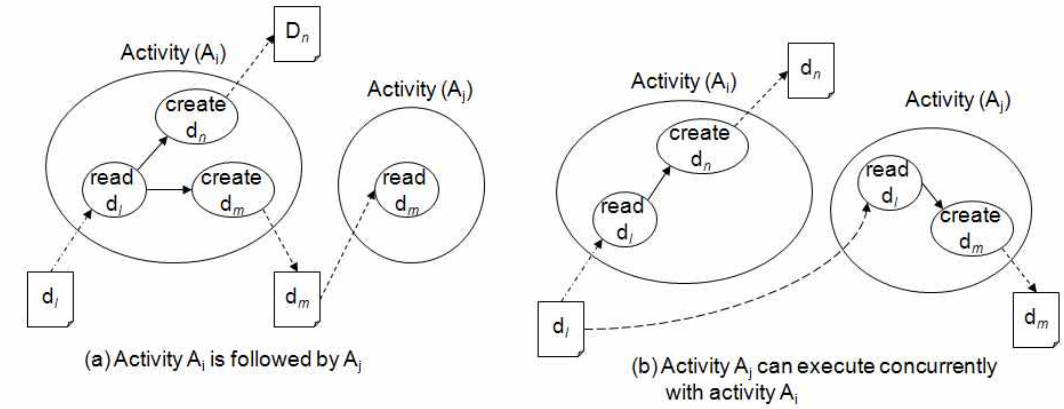
4. DDG 기반의 프로세스 모델링

4.1 기본 개념

비즈니스 프로세스를 구성하는 개별 액티비티의 내부 로직에는 다양한 오퍼레이션들, 특히 해당 액티비티를 수행하는데 필요한 데이터를 처리하는 오퍼레이션들을 포함한다. 그러한 오퍼레이션으로는 (1) 액티비티 실행을 위해 필요한 데이터를 읽어 들인다거나 (2) 액티비티 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 삭제하여 더 이상 available 하지 않도록 하거나 (5) 어떤 데이터를 archive함으로써 더 이상의 업데이트를 금지하거나 (6) 어떤 데이터의 유효기간을 조정한다거나 하는 다양한 오퍼레이션들이 포함될 수 있다. 그리고, 각 오퍼레이션들은 그 오퍼레이션의 실행을 위해 필요한 입력데이터와 오퍼레이션 실행의 결과로서 출력데이터를 가진다. 여기에서 입력데이터와 출력데이터 간의 관계는 데이터 의존성으로 분석할 수 있다.

그림 1은 dl→dm , dl→dn 등 2개의 데이터 의존성을 포함하고 있는 어떤 프로세스의 부분을 나타내고 있다. 그림 1(a)는 이 2개 데이터 의존성이 액티비티 Ai 에 할당됨으로 해서, Ai와 Aj 간에 순서 (시간적 실행순서) 가 성립되고 있는 경우를 보여 주고 있으며, 반대로 그림 1 (b)는 dl→dn 는 액티비티 Ai에서, dl→dm 은 액티비티 Aj 에서 구현되고 있음에 따라, 이 2개 액티비티들은 동시에 수행가능하다는 것을 나타낸다. 이 예에서 볼 수 있듯이, 개별 액티비티에 데이터 의존성을 할당한 후에는, DDG 내에서 정의된 각 데이터 의존성 아크 간의 관계에 따라 액티비티들 간의 시간적 선후관계를 파악할 수 있다. 본 논문에서 1개 이상의 데이터 의존성을 하나의 액티비티에 할당하는 것을 액티비티 매핑 (activity mapping)이라고 부른다.

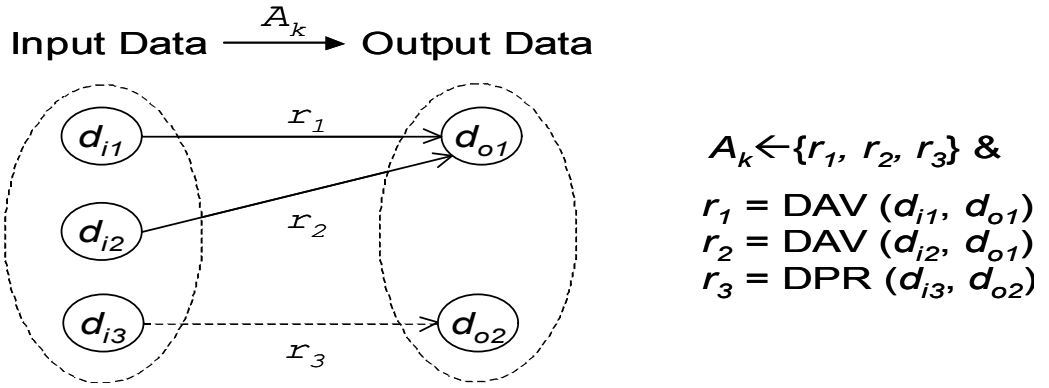


<그림 1> 데이터 의존성의 차이에 따른 프로세스 변경

다시 말해서, 비즈니스 프로세스를 구성하는 개별 액티비티는 1개 이상의 데이터 의존성을 성립시키는 과정이라고 볼 수 있으며 역으로, DDG에 기반한 비즈니스 프로세스 모델링이란, 개별 액티비티에 1개 이상의 데이터 의존성을 분배 (allocation) 하는 과정으로 볼 수 있다. 즉, 비즈니스 프로세스를 구성하는 개별 액티비티는 DDG에서 정의된 1개 이상의 데이터 의존성과 매핑되며, 그러한 매핑 관계로부터 액티비티 들 간의 시간적 선후관계를 결정할 수 있다. 실행 타이밍 관점에서 2개의 액티비티 간의 관계는 동시에 수행할 수 있거나 (concurrently executable), 동시에 수행할 수 없거나 (sequentially executable), 또는 하나의 액티비티가 수행되면 다른 하나의 액티비티가 수행될 수 없는 (mutually exclusive) 3가지 관계 중에 하나가 된다.

4.2 액티비티 매핑 : 액티비티 간의 선후 관계 분석

그림 2와 같이, 어떤 액티비티 Ak에 3개의 데이터 의존성, r1 = DAV(di1, do1), r2 = DAV(di2, do1), r3 = DPR(di3, do2) 가 할당되었다고 하자. 그렇다면 Ak는 3개의 데이터 di1, di2, di3를 입력 받아 2개의 데이터 do1, do2를 출력하는 액티비티라고 정의할 수 있다. 다시 말한다면, Ak를 수행할 수 있는 사전조건은 di1, di2 가 available 하고 di3가 present 하는 것이다. 그렇다면, 위의 액티비티 Ak를 수행할 수 있는 사전조건을 만족시켜 주는 액티비티들은 Ak에 앞서 실행되어야 한다. 그렇지 않으면 영원히 Ak를 실행할 수 있는 조건이 만족되지 않기 때문이다. 만약 di1, di2, 또는 di3를 생성하는 어떤 액티비티 Aj 가 있다면 Aj 는 반드시 Ak 에 선행하여 실행되어야 한다. 이러한 사실로부터 간단한 증명을 통해 다음의 두 가지 정리를 이끌어 낼 수 있다.



<그림 2> 액티비티 매핑

정리1 (precedence >) DDG에서 incoming arc가 없는 source node로부터 outgoing arc가 없는 sink node 에 이르는 연속된 data dependency arc의 집합을 path라고 하면, DDG는 최소한 하나 이상의 path를 포함하게 된다. 여기서, 2개의 액티비티 매핑 Ai←{rik, k =1,…,m}과 Aj←{rjl, l =1,…,n}이 있다고 하자. 만약 임의의 ria∈{rik, k =1,…,m}와 임의의 rjb ∈{rjl, l =1,…,n}가 DDG를 구성하는 여러 개의 path 중 임의의 하나의 path 위에 있게 되면 이 2개 액티비티 간에는 선후 관계가 존재한다.

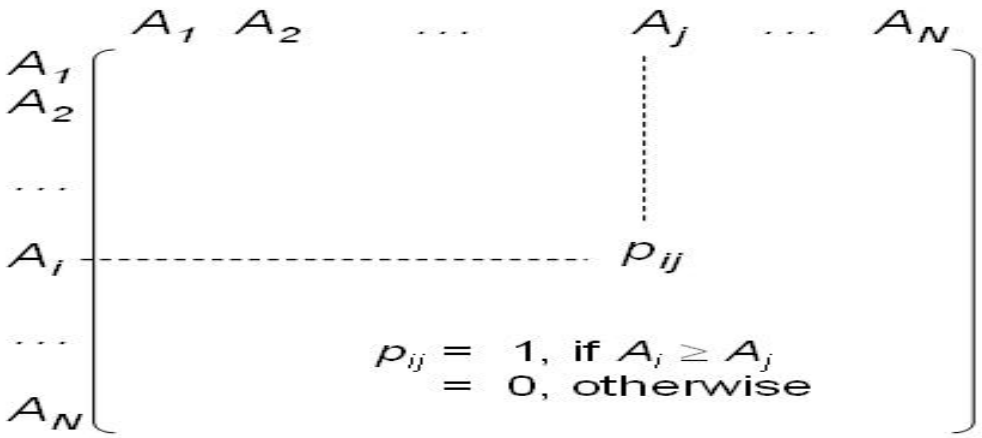
정리2 (transitivity closure on precedence >) Ai>Aj 이고 Aj>Ak 이면 Ai>Ak이다.

4.3 DDG 기반의 프로세스 설계 방법

위와 같은 분석 과정을 모든 액티비티 매핑 페어(pair)에 대해 수행하고 나면 각 페어에 대해, 그 페어를 구성하는 2개 액티비티 간에 immediate precedence 관계 (≥ 기

호로 표시한다) 가 존재하는지, 아니면 서로 독립적인지를 확인할 수 있다. 이 과정을 통해 얻어진 2개 액티비티들 간의 partial order들 간에는, 정리 2에서 살펴본 것처럼, transitivity가 존재한다. 그로부터 모든 partial order를 만족하는 full order를 구하게 되면, 그렇게 구해진 full order는 하나의 프로세스를 나타내게 된다.

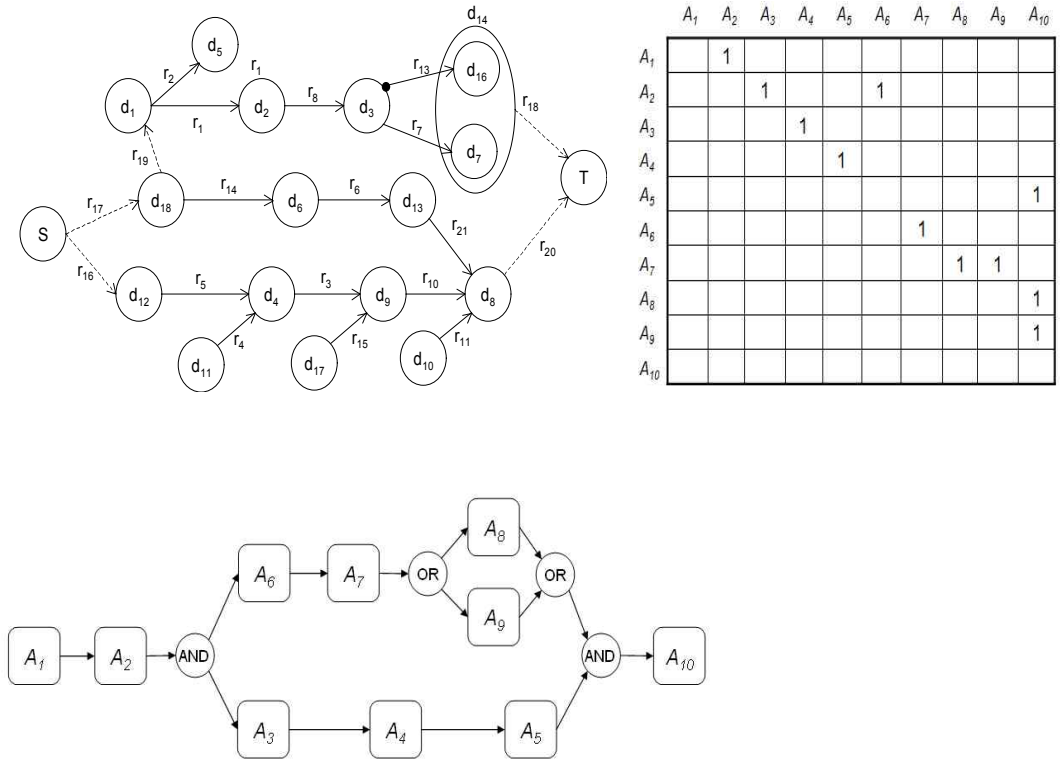
액티비티들 간의 partial order가 결정되면, 이로부터 그림 3과 같은 precedence matrix, PM = (pij)N×N 를 정의할 수 있다. PM의 각 행과 열은 하나의 액티비티와 대응된다. PM을 구성하는 각 셀 pij의 값은, 만약 Ai≥Aj 이면 1이 되고 그렇지 않으면 0이 된다. PM의 패턴과 프로세스 모델링 요소를 비교해 보면 다음과 같다. 첫째, 어떤 행 Ai에 대해, pij=1인 셀이 오직 하나 존재하면 액티비티 Ai와 액티비티 Aj는 sequence 관계에 있다. 둘째, 어떤 행 Ai에 대해, pij=1인 셀이 2개 이상 존재하면, 액티비티 Ai와 pij=1인 셀의 열에 해당하는 액티비티들 간에 AND 또는 OR split 관계에 있다. 셋째, 어떤 열 Aj에 대해, pij=1인 셀이 2개 이상 존재하면, 액티비티 Aj와 pij=1인 셀의 행에 해당하는 액티비티들 간에 AND 또는 OR join 관계에 있다.



<그림 3> Precedence Matrix

4.4 프로세스 모델링의 예

다음의 그림 4 (a)는 DDG 예를 보여 주고 있다. 이에 대해 만약 액티비티 매핑을 A1←{r16}, A2←{r4, r5, r17}, A3←{r3, r14}, A4←{r6, r15}, A5←{r10, r11, r21}, A6←{r1, r2, r19}, A7←{r8}, A8←{r7}, A9←{r13}, A10←{r18, r20} 와 같이 하게 되면, 그로부터 만들어 지는 PM은 그림 4 (b)와 같이 결정된다. 그리고 그로부터 유일하게 결정되어지는 프로세스 모델은 그림 4 (c)와 같다.



<그림 4> 프로세스 모델 설계의 예

5. 결 론

본 논문에서는 프로세스의 설계에 데이터 요구사항을 반영할 수 있는 하나의 방법론으로서 데이터 의존성 개념을 제안하고, 그로부터 액티비티 매핑이라는 메카니즘을 통해 프로세스 모델을 설계하는 과정을 살펴보았다. 또한, 이렇게 구해진 프로세스 모델이 그 설계를 위해 분석되어진 데이터 요구사항 (데이터 의존성으로 표현된)을 만족할 수 있음을 검증하였다.

비즈니스 프로세스는 다른 분야의 프로세스와는 다르게 하나의 액티비티가 수행되는 시간이 길고, 그 내부에 인간의 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 논문에서 제안하고 있는 접근 방법은 설계자의 의도 (데이터 의존성과 데이터 매핑으로 반영됨)에 따라 다양한 형태의 시나리오를 도출해 낼 수 있는 구조를 가지고 있어, 프로세스 설계단계에서 뿐 만 아니라 프로세스 실행 단계에서도 활용할 수 있다는 것이 장점이다. 프로세스 설계자의 입장에서 보면 데이터 의존성이란 하나의 설계 제약조건으로 작용한다. 제약조건을 줄일수록 대안적 시

나리오를 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 액티비티 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(conflict)이 발생하지 않도록 유연하게 설계된 프로세스를 말하는 것이다. DDG는 데이터 자원의 효율적 관리가 가능하도록 프로세스를 설계하고자 한다는 점에서 최선의 프로세스를 설계하기 위한 검증 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 투사(projection)한 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 있다. 즉, 프로세스의 수명 주기 전반을 지원하는 통합된 도구로 사용할 수 있을 것으로 기대하고 있다.

6. 참 고 문 헌

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

[2] 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, April 2004, p.199-239.

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

[4] 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.

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

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

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

[8] K. Malone, The Interdisciplinary Study of Coordination, ACM Computing Survey, Vol.26, No.1, 1994, p.87-119.
[9] 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), January2004, p.18-22.
[10] T.W.Malone, K.Crowston, G.Herman, Organizing Business Knowledge: The MIT Process Handbook, MIT Press, 2003.
[11] W.M.P. van der Aalst, S. Jablonski, Dealing with workflow change : Identification of issues and solutions, I.J. of Computer Systems Scienceand Engineering, 2000.
[12] 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