생산구조의 분할과 데이터 모델링에 관한 연구

A Study on the Split and Data Modeling of Production Structure

손 승 희 (Seyung-Hee Sohn) 단국대 경영학부

목 차

Keywords: *Object Class Network, Production Graph, Production Structure(Alternative Process, Recurive Process), Split Function, Production Data Model*

Ⅰ. 서 론

<Model>이란 현실 세계의 문제를 해결하기 위해서 복잡한 현실 세계를 정형화시키며, 모형화하기 위한 일종의 표현 방법이다. 그러므로 어떠한 모델링 기법을 사용하느냐에 관계없이 현실 세계의 실상(semantic of modeled scope)을 묘사하는 모든 객체들과 이들 객체 사이의 논리적 연관 관계는 가능한 한 현실에 가깝게 그리고 논리적 모순없이 표시되어야만 한다. Entity-relationship-method (Chen, 1976) (이후에는 언제나 E/R-Method라고 칭함) 또는 객체 지향적 모델링 방법등은 현실 세계를 모형화하기 위해서 주로 사용되는 대표적인 모델링 방법들이다. 그러나 E/R-Method나 객체 지향적 모델링 방법에 있어서 entity type또는 object class는 구조적 유사성을 근거로 형성되므로, 어떠한 entities들을 동일 type에 귀속시킬 것인가? 또는 object class를 어떻게 정의할 것인가? 하는 문제가 발생하며, E/R-Method에서 entity type사이의 논리적 연관 관계는 relation을 통해서, 그리고 객체 지향적 모델링 방법에서는 object를 묘사하는 속성(attribut)의 유전 방식(inheritance type)에 의해 설명되는데, 속성의 유전 방식을 설명하는 mechanism의 부족으로 인해서 현실 세계의 실상을 묘사하고 모형화하는데 많은 문제점을 안고 있다(Rumbaugh, 1991, p. 6, p.84). 이러한 문제는 특히 생산 구조를 모형화하는데 있어서 두드러지게 나타나는데, 단순 생산 구조를 전제로 모델링 할 경우에는 기존의 모델링 방법을 이용 하더라도 문제가 없지만, 현실적으로 자주 등장하는 복합 생산 구조나 순환 생산 구조와 같은 특수한 생산 구조를 전제로 모델링할 경우에는 현실 세계에 해당하는 전체 생산 구조를 모형화 시킬 수 없다는 한계점에 도달하게 된다.

본 연구 논문의 근본 목적은, 기존의 모델링 방법들이 가지고 있는 문제점들을 보안하여 모델화 대상을 보다 현실에 가깝게 모형화시킬 수 있는 방법을 개발하여, 단순 생산 구조 뿐만아니라, 복합 생산 구

조와 순환 생산 구조와 같은 특수한 생산 구조를 현실에 가깝게 모델화 하려는데 있다. 본 연구 논문에서 소개하는 object class network을 이용한 모델링 방법에서는, object class network을 node와 line system만을 이용하여 표기함으로서 기존의 방법과 비교해서 모델의 복잡성을 해소시켰을 뿐만아니라, 기존의 E/R-Method의 entity type과 객체 지향적 모델링 방법에서 객체에 해당하는 object class는 구조적 유사성을 근거로 형성되는 것이 아니라, 무결점 조건(local integrity constraint)의 성립 여부에 따라서 결정되며, object class사이의 논리적 관계도 또한 시스템적 유전 방식에 의해서 정의되는데, object class로부터 파생된 관계형 객체(relational object class)를 묘사적 유전(descriptive inheritance)과 생성적 유전(generating inheritance)으로 분리함으로서 모델화 대상을 현실에 맞게 모형화시킬 수가 있다. 이처럼 현실 세계의 실상을 단계적인 무결점조건을 통해서 데이터 모델에 집약시킬 수가 있기 때문에, 기존의 모델링 방법보다 진보된 형태의 데이터 모델링 방법이라할 수가 있다. 이러한 모델링 방법을 이용하여 단순 생산 구조 뿐만 아니라, 특수한 생산 구조까지도 충분한 이론적 뒷받침하에 쉽게 모델화할 수 있는 장점이 있다.

Ⅱ. OCNet을 이용한 데이터 모델링

Object class network을 이용한 데이터 모델링 방법도 기존의 방법들과 마찬가지로 현실 세계의 실상을 직접 묘사하는 기본 실체(basic object class)와 이들 실체사이의 관계를 통해서 파생되는 관계형 실체(relational object class)를 통해 설명되는데(Booch, 1991, Koenig;Wolf, 1993, Scheer, 1994, p.54-57), 근본적인 차이는 기존의 방법들에 있어서 entity type이나 object class는 구조적인 유사성을 근거로 정의되는 반면, object class network을 이용한 모델링 방법에서는 기능적 의존 관계(functional dependence)와 같은 무결점 조건에 의해서 정의되기 때문에 모델의 실상을 현실에 가깝게 묘사할 수 있다는 점과, 기본 실체로부터 파생되는 관계형 실체가 철저한 mechanism (생성적 또는 묘사적 유전 방식)에 의해서 형성된다는 것이다(Steffens, 1994, Sohn, 1996, p.90-96).

2.1 Object class

영(0)을 제외한 A를 현실 세계의 실상을 묘사하는 object를 특징짓는 속성들의 유한 집합이라 할 때, 모든 속성들은 언제나 특정한 값을 갖기 때문에, Xa는 a⊆A가 갖는 특정값들의 집합이며, A와 Xa를 통해서 object A의 <object area>를 다음과 같이 정의 된다:

$$\mathbf{X^A := \{\ w{:}A \rightarrow \cup X_a \mid \forall a \subseteq A{:}\ w(a) \subseteq X_a\ \}}$$

X^A의 한 부분집합 $R \subseteq X^A$은 object quantities에 해당하며, 이중의 한 원소 $w \subseteq X^A$는 object A에 속하는 objcet라 칭한다. X^A를 object A의 object area라고 한다면, $p{:}_2X^A \rightarrow$ BOOL을 object area X^A에 대한 <local integrity constraint(무결점 조건)>라고 정의할 수가 있다. 이제 X^A를 object A의 object area라 하고, $p{:}_2X^A \rightarrow$ BOOL을 object area X^A에 대한 무결점 조건이라 한다면, 이 조건을 만족하는 object area상의 한 <object class>는 다음과 같이 정의된다:

$$\mathbf{X^{A,p} := \{\ R \subseteq X^A \mid p(R)\ \}}$$

무결점 조건의 대표적인 예로서 <functional dependence>를 들 수가 있는데, functional dependence를 통한 무결점 조건은 다음과 같이 작성될 수가 있을 것이다: $X^{A,p}$를 무결점 조건을 충족시키는 object area상의 한 object class라 하고, B와 C를 A의 한 부분 원소 (B, C⊆A)라 한다면, C는 다음 조건을 만족할 때에 B에 <funcational dependence>라고 하고, (B→C)로 표기한다.

$$\mathbf{\forall R \subseteq X^{A,p}{:}\ \forall w, w' \subseteq R{:}\ w|_B = w'|_B \Rightarrow w|_C = w'|_C.}$$

이와같은 funcational dependence를 이용하여 object area X^A에 대한 무결점 조건은 이제 다음과 같이 구체화할 수가 있다[Sohn, 1996, p.92].

$$(B \rightarrow C): {}_2X^A \rightarrow BOOL$$

$$(B \rightarrow C)\ (R) = THRU : \Leftrightarrow (\forall w, w' \subseteq R: w|_B = w'|_B \Rightarrow w|_C = w'|_C).$$

앞에서 살펴본 바와 같이 object class는 많은 속성들에 의해서 특징 지워지며, 이 속성들을 통해서 object의 종류가 정해진다. Object class를 특징짓는 많은 속성 중의 일부는 주요 키와 같은 역할을 담당하고, 나머지는 보조 키와 같은 역할을 담당한다. 특히 후자의 속성들이 모여서 object를 확인하는 역할을 담당하기도 한다. 모든 속성들은 또한 특정 범위의 값을 가지고 있으며, object는 이를 특징짓는 속성과 이 속성의 특정 범위의 값이 일대일 대응되는 모형으로 이해되어질 수 있다. 같은 종류에 속하는 object들이 동일 class에 소속되기 위해서는 일련의 조건들이 충족되어야 하는데, 이때에 이 조건들을 충족시키는 모든 object들만이 동일 class에 포함된다.

모든 object class는 데이터 모델에서 주 필드와 보조 필드로 이루어진 네모 상자로 표기한다. 주 필드에는 object class의 이름 (Yi)을, 보조 필드의 좌측에는 object class를 특징짓는 속성 (K(Yi))들이, 보조 필드의 우측에는 속성의 유전 방식이 기록된다.

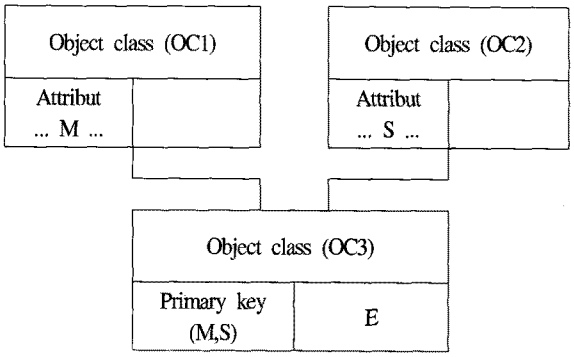
Object class name (Yi)	
attribute K((Yi))	유전방식 (inheritance)

2.2 Inheritance

Object class network을 구성하기 위해서는 일반적으로 많은 object class들이 필요한데, 그중의 일부분은 모델화 대상을 직접적으로 설명하기 위해서 기본적으로 주어질 수도 있지만 (이러한 경우의 object class를 가르켜서 <basic object class>라고 한다), 대부분의 object class는 basic object class사이의 논리적 연관 관계에 따라서 basic object class로부터 파생되는 object class들이다 (이러한 경우의 object class를 가르켜서 <relational object class>라고 부른다). Relational object class는 basic object class와는 달리 단독적으로 형성되는 것이 아니라, 존재 여부 자체가 다른 object class (주로 basic object class이지만, 경우에 따라서는 relational object class가 될 수도 있음)들에 종속되어 있기 때문에 <파생적 object class>라고 부르기도 한다. Relational object class의 존재 여부가 하나의 또는 여러개의 object class에 종속되어 있을 때 relational object class를 단 단위 또는 다 단위라 하며, 특히 단 단위의 relational object class를 <sub class>라고 부른다.

Relational object class는 자신의 존재를 결정짓는 object class의 일부 속성을 역할 분담시켜서 relational object class에 유전시키면, 이곳에서 다시 주요 키로서의 역할을 담당하는 방식으로 형성된다[Sohn, 1996, p.93]. Relational object class의 모든 주요 키로서의 역할을 담당하는 속성은 따라서 다른 object class들로부터 상속받은 것들로 이루어져 있다. 주요 키의 역할을 담당하는 속성이 이처럼 새로운 relational object class를 생성시킬 목적으로 전달되는 유전 방식을 가르켜 <생성적 유전방식 (generating inheritance)>이라고 한다. 속성을 유전시키는 방법중에 이같은 생성적 유전 방식만이 유일한 것이 아니며, 단지 다른 object class를 설명할 목적으로 전달되는 경우도 있는데, 이 같은 유전 방식을 <묘사적 유전방식 (descriptive inheritance)>이라 부른다(Steffens, 1994, p.12).

Relational object class는 이처럼 두가지의 유전 방식에 의해서 형성되는데, object class를 나타내는 네모 상자의 하단 우측에 어떤 유전 방식에 의해서 형성된 object class인지 표기함으로서 모델를 쉽게 이해할 수가 있다. 생성적 유전 방식의 경우에는 <E>를, 묘사적 유전 방식의 경우에는 <B>를 표기함으로서 구별된다 <그림2-1 참조>.

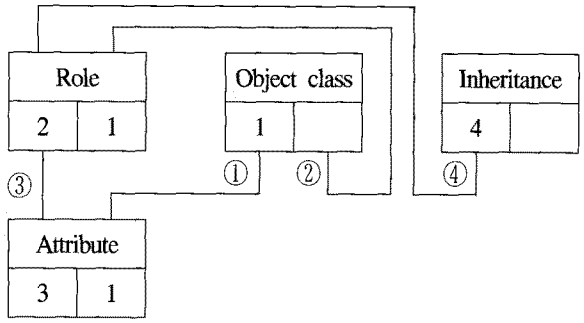


〈그림2-1〉 Relational object class의 형성 방법

2.3 Object class network

Object class network은 일종의 semi-hypergraph로 표시되는데, semi-hypergraph는 네모상자들과 이들이 서로 연결된 선들의 집합으로 구성되어 있다.

지금까지 설명한 object class, object class를 특징짓는 속성, 이 속성의 역할이 유전되는 방법 그리고 object class network의 형성을 설명해 주는 역할 분담들을 토대로 다음과 같은 기본 원리 (data model basic)를 이용하여 object class network을 작성할 수가 있다[Sohn;Steffens, 1992, p.4] <그림2-2 참조>.



Object class:

번호	object class이름
1	Object class
2	Role
3	Attribute
4	Inheritance

Role :

번호	object class 이름	유전방식	유전자 object class
①	Object class	E	1
②	Role	B	1
③	Attribute	E	2
④	Inheritance	B	4

Attribte:

번호	상속자 object class 번호	object class 이름
1	3	Object class
2	2	Role
3	3	Attribute
4	2	Inheritance

Inheritance:

유전방식	역할전달방법
E	generating
B	descriptive

〈그림2-2〉 Data model basic

<그림2-2>에서 네 개의 네모상자 Role, Object class, Inheritance, Attribute 는 각각 object class 자체를 (따라서 meta object class로 볼 수 있음(Steffen;Sohn, 1992, p.3-4), 그리고 서로 연결된 선들은 논리적인 연관관계를 나나낸다. object class를 잇는 선들에 표기된 원문자 예를 들어, ①은 두 object class Object class와 Attribute 사이의 논리적인 연관관계를 의미하는 동시에, object class Object class(유전자)에서 Attribute(상속자)로 속성이 전달됨을 의미한다. 네 개의 속성은 각각 1, 1, 2, 3번 object class에서 출발하여 각각 3, 2, 3, 2 번 object class로 전달된다. 또한 네 개의 속성이 전달되는 방법은 다른 object class를 생성하기 위해서 또는 묘사하기 위해서 전달될 수 있는데, E generating 와 B descriptive 는 속성이 전달되는 두가지 방법을 나타낸다.

Ⅲ. 생산구조의 분할과 데이터모델

3.1. 생산모형

<생산(production)>이란 개념을 넓은 의미로 해석하여 모든 형태의 부가가치 창출 과정으로 보든, 생산요소의 조합 또는 변형 과정이 생산이라는 좁은 의미로 해석하든, 생산이란 개념속에는 언제나 input, output, throughput이라는 세가지 개념이 가지는 의미를 내포하고 있다. Throughput을 생산 요소를 조합하거나 변형시키기 위한 또는 부가가치를 창출하기 위한 변형 기술(transformation technology)이라 한다면, 이 technology에 투입되는 모든 대상을 input, 산출되는 모든 결과를 output으로 해석할 수가 있기 때문이다. 따라서 동일한 input으로부터 어떤 결과 즉, 얼마만큼의 output이 산출되느냐는 input과 output을 연결시켜주는 변형 기술에 의해서 결정되기 때문에, 이 변형 기술은 생산자가 선택할 수 있는 구체적인 생산공정(production process)이며, 생산 공정을 통해서 연결된 input과 output의 조합을 생산 행위(production activity)라고 할 수가 있다. 이런 의미에서 생산은 결국 input-(throughput)-output model로 나타낼 수가 있다 (Sohn, 1996, p.81-83).

3.2 생산구조 모형

3.2.1 생산구조를 모델화하기 위한 object class

단순 생산 구조 뿐만아니라, 복합 생산 구조와 순환 생산 구조와 같은 특수한 생산 구조라 할지라도 다음과 같은 basic object class와 이로부터 파생되는 relational object class를 이용하여 모델화할 수가 있으며 (object class name 후의 첫 괄호는 가상의 주요 키를 의미함), 이들 object class가 지니는 의미는 다음과 같다(Sohn, 1996, p.97-101).

〈Basic object class〉

ProdPro (P) (생산공정): 생산을 <개별 공정들의 집합>으로 이해할 때에, 생산 공정은 생산자가 선택하여 실현시킬 수 있는 최소한의 공정 단위 (개별 투입-산출모형)이다.

IObject (IO) (투입대상)와 OObject (OO) (산출대상): Input과 output은 생산 공정에 투입되거나, 생산 공정을 거친후 결과로 얻어지는 산출물을 의미하기 때문에, 생산 공정을 실행함에 있어서 필수적인 요소라할 수 있는데, 생산 공정에 투입된후 산출물의 일부분이 되어 원래의 모습으로 환원될 수 없는 모든 종류의 생산요소와 부분품 또는 중간제품등도 완제품을 생산하는데 필요한 중간공정에 투입된다면, 이 모든 투입물이 IObject에 속한다.

전체 생산 공정은 개별 공정들의 집합체로 해석되기 때문에, OObject는 완제품만을 의미하는 것이 아니라, 개별 공정의 실행후 산출되는 모든 대상을 의미한다. 이렇게 볼 때, 하위 공정 단계로부터 산출되는 OObject는 다시 상위 공정 단계로 투입되어야하기 때문에, 모든 생산 대상은 IObject와 OObject로서의 두가지 성격을 동시에 가지고 있다.

ProdMit (M) (생산수단): 생산 수단은 생산 공정을 실행하기 위해서 직접 또는 간접적으로 필요한 모든 기술적인 보조 수단을 의미한다.

ProdStel (S) (생산장소): 생산 장소는 IObject가 투입되거나, OObject가 산출되는 구체적인 장소 또는 생산 수단이 투입되는 장소를 의미한다.

〈Relational object class〉

생산 공정에 투입되거나, 생산 공정으로부터 산출되는 모든 투입 대상과 산출 대상은 언제나 구체적인 생산 장소와 연결되어 어떤 생산 장소에서 투입되고, 산출되는지를 명확하게 구별할 수가 있으며, 모든 생산 수단도 마찬가지로 생산 장소와 연결됨으로서 구체적으로 투입 장소를 구별할 수가 있다. 환언하면, 모든 IObject와 OObject 그리고 ProdMit는 언제나 ProdStel과 언제나 연관 관계를 맺고 있다는 것이다. 이로부터 일차적으로 다음과 같은 relational object class가 파생된다.

SIObject (IOS) (장소-투입대상): 구체적인 장소와 연결된 투입 대상

SOObject (OOS) (장소-산출대상): 구체적인 장소와 연결된 산출 대상

SProdMit (MS) (장소-생산수단): 구체적인 장소와 연결된 생산 수단

상술한 세가지의 relational object class는 이제 다시 구체적인 생산 공정과 연결됨으로서 그 의미의 명확성을 더하게 된다. 예를들어, SOObject에 포함된 P1을 통해서는 구체적으로 어떤 생산 장소에서 산출되었는지는 알수 있으나, 어떤 생산 공정의 결과로 얻어진 산출물인지는 알 수가 없을 것이다. 그러나 만일 SOObject와 ProdPro를 연결시킨다면, P1은 어떤 생산 장소에서 그리고 어떤 생산 공정 결과 얻어진 산출물이라는 것까지 확인할 수가 있게 된다. SIObject와 SProdMit도 마찬가지로 ProdPro와의 연관 관계를 통해서 그 의미를 명확하게 식별할 수가 있다. 이로부터 다음과같은 object class가 파생된다.

PInput (PIOS) (투입구조): 생산 장소와 생산 공정별 투입 구조

POutput (POOS) (산출구조): 생산 장소와 생산 공정별 산출 구조

PMBeleg (PMS) (분배구조): 생산 장소와 생산 공정별 생산 수단의 분배 구조

3.2.2 생산구조의 정의

생산 구조는 전술한 object class를 이용하여 다음과 같이 정의될 수가 있다: P를 생산공정중의 부분집합, IOS를 장소-투입 대상 중의 부분 집합, OOS를 장소-산출 대상 중의 부분 집합, MS를 장소-생산 수단 중의 부분 집합이라 한다면, PInput(p), POutput(p), PMBeleg(p) $\forall p \subseteq P$ 의 의미가 함축된 correspondence triple $(P \rightarrow 2^{IOS} x 2^{OOS} x 2^{MS})$을 생산 구조라 칭한다.

PInput(p)과 POutput(p)는 생산 장소와 연결된 IObject와 OObject의 수량을 나타내며, PMBeleg(p)은 생산 장소와 연결된 생산 수단의 분배량을 나타낸다.

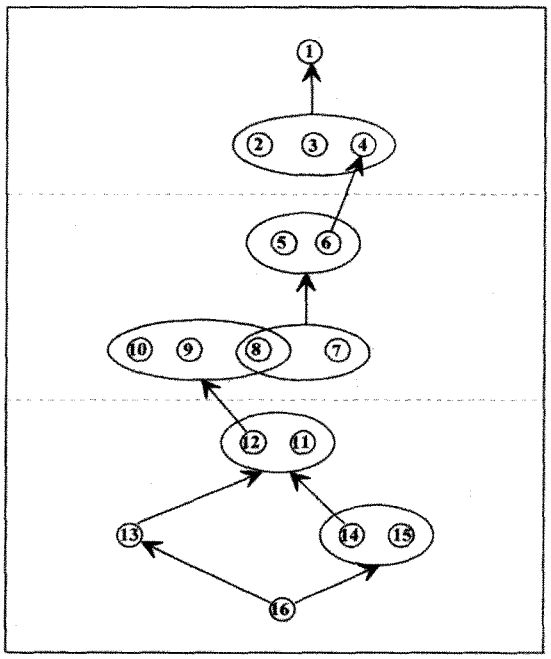
3.2.3 생산구조의 표기

PInput(p), POutput(p), PMBeleg(p)의 correspondence triple로 정의된 생산 구조는 다음의 세가지 구성 요소를 이용하여 <production graph>로 나타낼 수가 있다 <그림3-1>.

소매듭: 개별 IObject또는 개별 OObject

대매듭: 단일 공정에 소요되는 투입 또는 단일 공정으로부터 산출되는 산출물의 집합

화살표: 개별 생산공정



〈그림3-1〉 Production graph

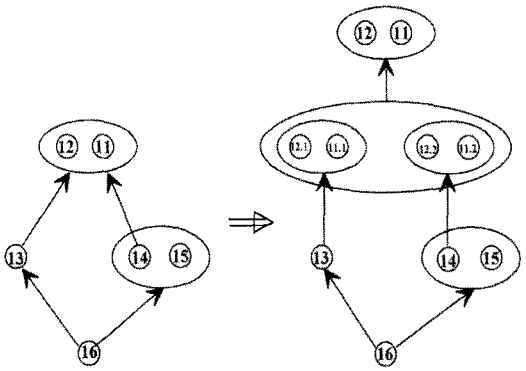
Production graph의 표기 방법은, 기존의 제품 구성도나 gozinto-graph에서와 마찬가지로 동일한 구성 요소 (매듭과 화살표)를 사용하기 때문에 외관상 거의 흡사한 형태를 지니고 있지만, 기존의 제품 구성도나 gozinto-graph에서와는 전혀 다르게 해석된다. 예를 들어 <그림3-1>의 하 단면은, 제품 구성도나 gozinto-graph에서는 "중간 제품 12는 부분품 13과 14로 구성

되어 있다"는 것을 의미하지만, production graph상에서는 12라는 OObject를 생산하기 위한 방법이 두가지가 있는데, 하나는 13이라는 IObject를 투입함으로, 다른 하나는 14라는 IObject를 투입함으로 생산할 수 있다는 <alternative output process>를 나타낸다. 즉, OObject 12를 생산하는데 두가지 production technology가 있다는 함축적인 의미를 내포하고 있다.

만일 모든 생산구조가 단순한 선형구조라면 production graph와 같은 새로운 방법이 필요하지 않을 수도 있다. 그러나 자주 등장하지는 않지만, 현실적으로 순환 생산구조 또는 복합 생산구조와 같은 alternative input 또는 output process의 등장 가능성을 배제할 수가 없기 때문에, 이러한 현실을 제대로 표현할 수 있는 새로운 방법이 요구된다. SAP사의 R/3에서도 복합 생산구조와 같은 특수한 생산구조가 등장할 수 있음을 설명하고 있다. 예를 들어, 복합 생산구조는 기존의 gozinto-graph를 이용하여 표현할 수가 없으나, <그림3-1>의 하단부에서 보는 바와 같이 production graph를 이용하여 표현할 수가 있다.

3.2.4 생산구조의 분할

Production graph상의 SIObject(ios)⊆(IOS) 또는 SOObject(oos)⊆(OOS)에 대해서 개별생산 공정 ProdPro(p) ∀p⊆P가 존재할 때, p를 (ios)의 <input process> 또는 <output process>라고 부르는데, 가장 단순한 생산 구조는 input process와 output process가 일렬로 접속된 형태가 될 것이다. 이러한 경우의 input process와 output process를 <단순 process>라고 한다. 단순 process의 경우에는 POutput(p)∩PInput(p')≠0이 됨을 의미하는데, 이 경우 process (p)를 <선행 process>, process (p')를 <후행 process>라고 부른다. 그러나 현실적으로 input process와 output process가 일렬로 접속된 형태로만 나타나는 것이 아니라, 같은 OObject라 하더라도 사용되는 기술적 방법 즉, 생산공정에 따라서 여러 가지 방법을 이용해서 산출될 수가 있으며, 반대로 동일 IObject라 하더라도 서로 다른 생산 공정에 투입될 수가 있기 때문에, 복잡한 형태를 취하는 것이 보통이다.



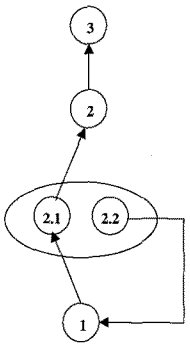
〈그림3-2a〉 alternative output process

예를 들어, 동일 IObject가 생산 공정을 달리하여 서로 다른 OObject를 산출하기 위해 투입될 수 있다면, 이들 개별 input process들은 생산자가 선택적으로 선별하여 사용할 수 있는 구체적인 production technology를 의미하기 때문에, 이러한 input process를 가르켜 <alternative input process>라고 부르며, 반대로 동일 OObject가 생산 공정을 달리하여 서로 다른 IObject를 이용하여 산출될 수 있는 방법들이 존재한다면, 이들 개별 output process들은 input process에서와 마찬가지로 서로 상이한 production technology를 의미하며, 이런 process를 가르켜 <alternative output process>라 부른다 <그림3-2a 참조>.

Alternative process의 개념을 도입함으로서, 생산을 투입-산출 모형으로 정의했던 첫 번째의 가정에 모순될 뿐만아니라, alternative process란 결국 production technology의 차이를 의미하기 때문에, 존재하는 production technology중에 어떤 방법을 택할 것인가?를 작업자가 언제나 결정해 주어야 한다는 문제점이 지적될 수 있으나, 이러한 문제점들은 alternative process를 split function (Steffens, 1993, Sohn, 1996, p.109-111)을 이용해 인위적으로 분리 표기함으로 해결될 수가 있다 <그림3-2a 참조>.

복합 생산 구조나 순환 생산 구조는 비 선형 생산 구조를 지닌 특수한 생산 구조의 대표적인 예가 될

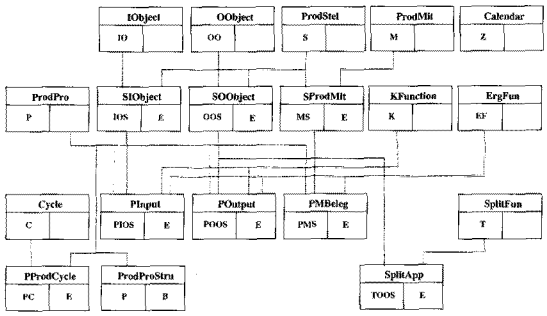
것이다. 생산 공정의 특정상 단일 공정으로부터 두 개 이상의 OObject가 산출되는 경우를 가르켜 <복합 생산 구조>라 부르며 <그림3-2a 참조 (alternative output process를 나타내는 <그림3-2a>에서 단일 공정에서 OObject 15라는 복합 생산물이 생산되는 복합 생산 구조를 동시에 표시하고 있음)>, 생산 공정이 계속 연결되어 있는 경우라면, OObject가 다음 공정에 투입되기 위해서 다시 IObject로 성격이 변환되어 재 투입되는 것은 당연한 것이지만, IObject로 성격이 변환되어 자신을 산출시켰던 IObject의 일부분을 이루기 위해서 또는 자신을 산출시켰던 output process가 계속 진행될 수 있도록 하기 우해서 재 투입되는 경우 (예를들어, chemical reactor)와 같은 생산 구조를 < 순환 생산 구조라고 한다. 복합 생산구조나 순환 생산구조를 이제 생산 공정에서 현실적으로 나타날 수 없는 불가능한 생산구조라고 일축할 것이 아니라, 현실적으로 가능한 (제조 공정에서는 특수한 경우이지만, 화학 공정에서는 일반적임) 생산구조이기 때문에, 이러한 생산 구조까지도 모델화할 수 있어야만 할 것이다. 그러나 이러한 특수한 생산구조는 제품 구성도나 gozinto-graph (화학 공정에서의 성분 배합도)로 표기하던 기존의 방법을 이용해서는 온전히 표시할 수가 없다는 것이다. production graph를 이용하여 이러한 특수한 생산구조까지도 표시할 수가 있으며, 여기에서도 alternative process에서와 마찬가지의 문제가 등장하지만, 인위적 분리 방법을 이용하여 문제를 해결할 수가 있다 <그림3-2b참조>.



〈그림3-2b〉 순환 생산구조의 split function

3.3 생산구조의 데이터 모델

생산 구조의 데이터 모델은, 현실적인 모든 생산 구조를 어떤 object class를 이용하여 모형화할 것인가? 그리고 이들 object class사이에는 어떤 논리적 연관 관계가 성립하는가를 보여 주는데, 앞 절에서 설명한 basic object class와 relational object class를 이용하여 다음과 같은 기본 데이터 모델로 나타낼 수가 있다 <그림3-3 참조>.



〈그림3-3〉 생산구조의 데이터 모델

<그림3-3>에서 표시하고 있는 object class ErgFun (EF)은 특별히 IObject와 OObject의 투입-산출 관계(수량/시간)를 모델화하기 위한 object class이다. 위에서 네모 상자로 표시된 것들이 2.1에서 설명한 object class를 나타내며, 서로 연결된 선들은 2.3에서 설명한 두 object class사이의 논리적인 연관관계를 나타낸다. 예를 들어, 두 개의 object class OObject와 ProdStel은 생산 프로세스를 통해서 산출된 Output과 생산장소를 나타내는데, 이들 object class로부터 SOObject라는 새로운 class가 생성된다. SOObject의 속성 중 OO는 object class OObject로부터, S는 ProdStel로부터 유전받은 것이다. 또한 OO와 S가 유전된 방식은 새로운 SOObject를 생성하기 위한 것으므로, object class SOObject의 유전방식은 생성적(E)가 된다. <그림3-3>의 기본 데이터 모델은, 생산구조의 형태에 따라서 필요한 경우 object class를 추가시킴으로 확장시켜 나갈 수가 있다.

Cycle(C)과 KFunction(K)는, 이미 언급했던 것처럼

기존의 모델링 방법을 통해서 모델화할 수 없는 복합 생산구조, 순환 생산구조와 같은 비 선형 생산구조를 모델화 하기 위한 object class로 이해할 수 있으며, SplitFun(T)은 split function을 이용하여 alternative process를 인위적으로 분리한 내용을 모델에 반영하기 위한 object class이다.

Ⅳ. 맺음말

생산 기업의 경쟁력을 높이기 위해서 지난 수년간 다져온 manufacuturing technology와 manufacturing re-organization을 통해서 생산 기업의 생산 구조는 지난 수년간에 걸쳐서 전혀 다른 모습으로 바뀌게 되었다. 과거에 화학 공정에만 나타난다고 생각했던 복합 생산 구조나 순환 생산 구조도 이제는 일반 생산 공정에서 더 이상 특수한 생산 구조라고 볼 수 없는 일반적인 구조가 되었다. 이처럼 생산 구조가 변화됨에 따라서 생산 구조를 모델화하는 작업은 기업에 산재해있는 고질적인 문제 (작업 시간지연, 납품 시간지연, 생산 요소의 비 효율적인 이용, 생산량 증가...)들을 해결하기 위한 방법이 될 것이다.

그러나 현재까지도 복합 생산 구조나 순환 생산 구조와 같은 특수한 생산 구조까지도 포괄적으로 모델화할 수 있는 모델링 기법들이 미비한 상태이며, 더욱 아쉬운 것은 특수한 생산 구조 자체를 심각하게 생각하지 않고 있다는 것이다. 예를들어 SAP사의 Real Time System과 같은 세계적인 시스템에서 조차도 순환 생산 구조는 현실적으로 나타날 수 없는 불가능한 생산 구조라고 일축하고 있으며, 복합 생산 구조도 BOM 상에 마이너스로 처리하는 등 비 현실적인 방법으로 해결하고 있다.

본 논문에서 소개한 object class network을 이용한 모델링 기법은 단순 생산 구조 뿐만아니라, 생산 분야에서 현실적으로 나타날 수 있는 모든 생산 구조를 모델화할 수 있는 모델링 기법이다. 기존의 방법들과 비교할때에, local integrity constraint를 통해서 object class를 정의하고, 생성적 유전 방식과 묘사적 유전 방식을 이용하여 relational object class의 형성을 체계화하고 구체화시킴으로 생산 구조의 실상을 데이터 모델로 집약시켰다는 점과, 간단한 표기 방법을 이용하여 모델의 복잡성을 해소시켰다는 점에서 높이 평가될 수가 있을 것이다.

참 고 문 헌

Abeln, O.: Ein Handbuch der computergestützten Ingenieurmethoden, München Wien 1990.

Booch, G.: Object-Oriented Design with Applications, Redwood City 1991.

Chen, P.P: The entity-relationship model, toward a unifield view of data, ACM transactions on database systems, vol. 1 (1976), No. 1, p. 9-36

König, W., Wolf, S.: Objectorientierte Software Entwicklung - Anforderungen an das Informations-management, in: Scheer, A.-W.: Handbuch Informations-management, Wiesbaden 1993, S. 868-898.

Rumbaugh, J.: Object-Oriented Modeling and Design, Englewood Cliffs 1991.

Scheer, A.-W.: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftspro- zesse, 5. Aufl., Berln u.a. 1994.

Sohn, S.-H., Steffens, F.: OrgIS - An Organization Information System. Funda-mentals and Conceptions, 1992.

Sohn, S.-H.: Konzepte moderner Produktionsplanungs-und-steuerungssysteme, Halle-Wittenberg, 1996.

Steffens, F.: Relationale Input-Output-Modelle, Mannheim 1993.

Steffens, F.: Grundlagen der Datenmodellierung, Mannheim 1994.

◐ 저 자 소 개 ◑

손 승 희 (shsohn@anseo.dankook.ac.kr)

독일 Mannheim 대학을 졸업하고(경영학사), 동 대학에서 경영학석사를 취득하였으며, 독일 Halle-Wittenberg 대학에서 경영정보학박사를 취득하였다. 독일 IDS Prof. Scheer GmbH와의 합작 컨설팅업체인 IDS Korea사에서 재직하면서 국내 유수 기업들의 BPR, ERP 프로젝트에 참가하였다. 부산외대를 거쳐 현재 단국대학교 경상학부에 재직 중이며, (사)한국전자상거래 연구소 이사로 재직하고 있다. 주요 관심분야는 시스템분석 및 설계, 데이터베이스 설계, 프로그래밍, 전자상거래, ERP 등이다.