「정보시스템연구」 제6권 제1호
한 국 정 보 시 스 템 학 회
1997. 6, pp. 5~26

분산 조직을 위한 객체지향 비즈니스 프로세스 엔지니어링*

박 광 호**

< 목 차 >

Ⅰ. 서 론

최근 환경변화에 대응하기 위한 기업의 변신 노력이 비즈니스 프로세스 리엔지니어링[Hammer and Champy, 1993], 벤치마킹[Camp, 1989], TQM[Hradesky, 1995] 등의 경영 혁신 방법론으로 추진되고 있다. 이러한 내부 변혁의 대상과 범위를 효과적으로 정하기 위해서 기업의 경영 활동에 대한 명확한 분석이 선행되어야 한다. 더욱이 기업이 경쟁력을 지속적으로 유지하려면 이제는 프로세스를 제품과 같은 개념으로 보고 이를 최적화 시키기 위한 노력을 경주해야 하는 것이다. 비즈니스 프로세스 엔지니어링(Business Process Engineering, 이하 BPE)은 이런 상황에서 새로운 조명을 받고 있다. BPE는 추상 활동으로서 현 기업 업무 상태를 정확히 반영하는 비즈니스 모델을 구축하게 된다. 여기서 추상이란 불필요한 것은 배제하고 관심 있는 핵심만 추출하는 과정이다. 이런 추상의 산출물인 비즈니스 모델은 공식적인 언어로 작성되고 표기

* 이 논문은 1997년도 한양대학교 교내연구비에 의하여 연구되었슴.
** 한양대학교 경영학부 조교수

법이나 다이어그램으로 가시화된다. 비즈니스 모델의 존재 여부는 문제 발견, 원인 분석, 해결책 도출 등 일련의 경영 혁신 과정의 성과를 크게 좌우한다. 또한, 이 모델은 정보시스템과 일대일로 대응되는 요소들로 구성되어 정보시스템 개발의 기반을 제공한다.

현대 기업이 대처해야 하는 문제로 대두되고 있는 변화 유형 중의 하나는 분권화이다[Taylor, 1995]. 그러나, 기업 규모의 확장, 세계화, 사업 다각화에 따른 필연적인 조직 형태인 분권화는 전사적 업무 추진과 통제 기능의 상실, 표준의 파괴, 조직간의 조정이 어려워지는 등 여러 문제점을 내포하고 있다. "어떻게 전체로서의 일관성을 타협하지 않고도 각 사업장에 자율성을 보장할 수 있을까"가 전체 문제를 대변한다고 볼 수 있다. 분산 조직에 있어 BPE는 업무의 복잡성과 이질성을 이해하고 각 사업장이 전사적 표준을 준수할 수 있는 방안을 강구하기 위해 필연적으로 추진될 수 밖에 없을 것이다. 반면에 분산 조직의 BPE는 지속적으로 합병, 설립되는 사업장에 의해 증가하는 복잡성과 이질성을 고려할 때 매우 어려운 작업임을 알 수 있다. 따라서, 이런 분산 조직에 대한 비즈니스 모델의 구축은 중앙 집중식 조직에 비해 평면적인 관점보다는 입체적인 관점에서 추진해야 하는 어려움이 존재하고 있다.

기업이 특정 목적에 따라 업무를 설계하는데 사용하는 기술들의 집합인 BPE는 기업의 구성요소를 자동차의 부품처럼 표준화, 정형화하고 이들 경영 부품을 조립하여 기업을 추상하는 것이라고 정의할 수 있다. 일단 기업이 BPE 프로젝트에 착수하게 되면 업무의 비효율성을 고치거나 근절시키기 의해 비즈니스 모델을 검증하거나 실행하기도 한다. 비즈니스 프로세스 리엔지니어링이 바로 이런 활동인 것이다. 또한, BPE는 경영 활동을 추상하는데 있어 프로세스 측면과 조직, 구성원 등의 인간적이며 유동적인 측면을 분리시켜 모델링하는 접근 방법이다.

비즈니스 모델의 기본 구성요소인 프로세스는 고객에게 가치를 제공하기 위한 내부 활동들의 집합이다[Jacobson, 1995]. 프로세스는 시간과 공간을 거쳐 시작과 끝, 입력과 출력을 가진 작업 활동의 순서, 즉 작업 구조를 의미한다. 분산 조직에 있어 프로세스의 추상은 효과적으로 이질성을 처리하기 위해 보다 엄격하고 풍부한 패러다임을 요구한다. 즉, 프로세스의 상세 내용을 추상하기에 앞서 프로세스 타입에 대한 정의가 선행되어야 하는 것이다. 이런 의미에서 분산 조직의 BPE의 대부분은 프로세스 타입의 정의라고 할 수 있을 것이다.

그러나, 최근까지 BPE는 만족할 만한 성과를 이룩하지 못했다. 오히려 정보공학[Martin, 1989]과 데이타 모델링의 상대적인 부상으로 프로세스 모델링은 일시적으로 무시되었다고 볼 수 있다. 그러나, 최근 비즈니스 프로세스 리엔지니어링의 활성화로 프로세스 모델링에 대한 연구가 활발히 진행되고 있는데, 김성근 외[1996]는 정보지향적 접근방법으로 프로세스 모델링 도구를 개발하고 적용하였으며 고일상[1996]은 객체지향 패러다임을 프로세스 모델링에 적용하였으나 프로세스 관리 시스템에 구현에 중점을 두었다. 권태형 외[1995]는 통합된 비즈니스 프로세스 모델링 관점을 제시하였는데, 첫째, 기존의 데이타 흐름도를 도구로 하는 기능적 관점(Functional View), 둘째, 시스템 흐름도(Flowchart)나 행위도(Action Diagram)를 도구로 하는 행위적 관점

(Behavioral View), 세째, 프로세스 정의 차트(Process Definition Chart)[Scherr, 1993]와 역할 상호 작용망(Role Interaction Net)[Cutis et al, 1992]을 도구로 하는 조직적 관점(Organizational View)에 새로이 프로세스의 수행 목적을 표현하는 정보적 관점(Informational View)를 추가하여 5W1H(What, Why, Who, Where, When, How)의 구성요소를 모두 추상하는 통합 프로세스 모델링 프레임웍을 제시하였다. Karagiannis[1995]는 비즈니스 프로세스 모델링의 복잡성을 정보 수집과 모델링 단계로 나누어 전통적인 분석에 치중하는 분석 접근법(Analysis Approach)과 디자인에 중점을 두는 조립 접근법(Synthesis)으로 비교 분석하고 있다. Baja와 Ram[1996]은 비즈니스 프로세스의 중요한 면들을 표현하는 개념적 모델로서 비즈니스 프로세스 모델을 제시하였는데 프로세스의 구조적인 측면과 동적인 측면을 표현하기 위해 객체들의 세부 내용들을 정의하는 도구를 제시하고 있다. 마지막으로, Tayor[1995], Jacobson[1995]는 객체지향 패러다임으로 비즈니스 프로세스를 정밀하게 추상하려는 엔지니어링 접근 방법을 제시하고 있다.

본 논문은 이런 일련의 최근 연구 방향의 연속 선상에서 분산 조직의 프로세스를 보다 완전히 추상하기 위한 방법론을 제시하고자 한다. 본 논문에서 제시하는 방법은 지금까지 객체지향 프로세스 모델링을 보다 구체적으로 객체지향 패러다임을 적용하였으며 특히 프로세스의 행동적 측면인 연산은 세가지 수직적 모델로 추상 되고 있다. 비즈니스 모델의 표기를 위한 기본 방법으로 OMT(Object Modeling Technique)[Rumbaugh et al., 1991]를 사용하였다. 2장에서는 본 논문의 기본 목표와 접근 방법을 설명하고 있으며, 3장에서는 모델링의 기본 구조, 프로세스 클래스, 그리고 세가지 수직적 모델의 정의를 제시하고 있다. 4장은 일반적 프로세스 클래스, 특수화, 캡슐화, 클래스 버전 등 객체지향 개념으로 BPE의 여러 측면을 각각 논의하였으며, 끝으로 5장에서 결론을 맺고 있다.

Ⅱ. 목표와 접근 방법

Brooks[1982]는 정보시스템 설계에 있어서 가장 중요한 고려 사항은 개념적 무결성(Conceptual Integrity)의 유지이며 이를 바탕으로 시스템에 대한 사양이 개념적으로 단순하고 명확하게 유지될 수 있다고 지적하고 있다. 여기서 개념적 무결성의 유지 원칙은 반드시 지켜져야 할 당위적 법칙이라기 보다는 경험에 따른 휴리스틱의 성격을 가진다고 하겠다. 개념적 무결성은 그 결과로 직설성(Straightforwardness), 단순성(Simplicity), 일관성(Consistency) 등을 제공한다[Berard, 1993]. 우선, 직설성이란 인식과 이해의 대상을 투명하게 드러낼 수 있는 표현을 의미한다. 둘째, 단순성은 인간의 인식과 이해의 한계를 인정할 때 복잡성을 가장 효과적으로 처리하기 위해 문제를 분할함을 의미한다. 마지막으로, 일관성이란 인식의 결과에 대한 갈등을 최저 수준으로 유

지함을 의미한다. 따라서, 본 논문에서는 BPE의 기본 목표로서 개념적 무결성을 추구한다. 그리고 이런 목표의 당위성은 경험적 선호임을 밝혀 둔다. 그리고 개념적 무결성이 보장될 때 우리는 직설성, 단순성, 일관성 등의 세가지 결과를 기대할 수 있을 것이다.

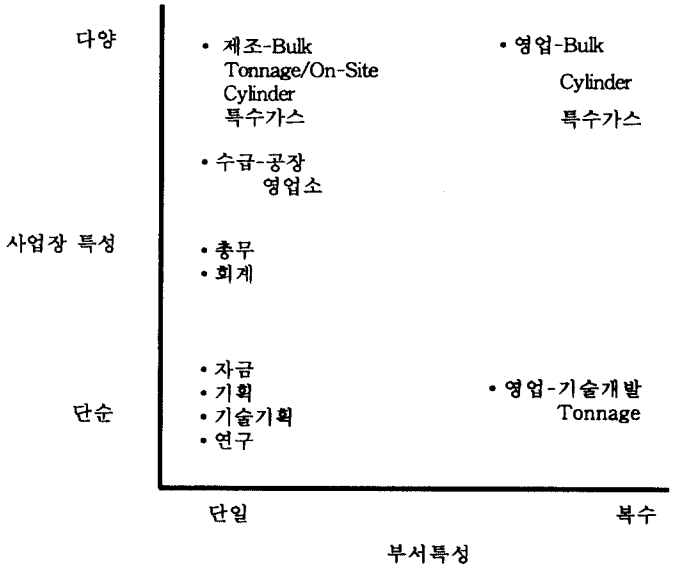
지속적인 최적화를 추구하기 위해 기업은 반드시 가차없는 일련의 리엔지니어링 노력을 기울여야 한다. 그러나, '지속적'이란 급변보다는 점진적인 것을 요구하는 측면이다. 이것은 프로세스를 설계함에 있어 좀더 개선되고 신속하게 그리고 저렴하게 한다는 것이 충분치 않음을 의미한다. 프로세스는 품질, 속도, 경제성에서 경쟁적 우위를 유지하면서 변화에 능동적으로 대응할 수 있도록 설계되어야 하는 것이다. 적응적 설계는 단기적으로는 더욱 어려울 것이다. 그러나 장기적으로는 단순한 수리만 하면 되기 때문에 오히려 용이하게 된다. 일단 적응적 시스템(Adaptive System)이 설치되면 개별적인 활동들은 전체적인 단절과 파괴 없이 지속적으로 개선될 수 있다. 따라서, "BPE의 최종 목표는 적응적 프로세스의 설계이다."라고 할 수 있을 것이다.

추상이란 복잡성을 효과적으로 이해, 처리하기 위한 방법이다. 우리는 비즈니스 프로세스의 추상의 결과로 업무 매뉴얼을 작성한다. 업무 매뉴얼은 프로세스에 대한 설명이지만 그 자체적으로도 복잡성을 내포하고 있다. 업무 매뉴얼에 포함되어야 할 항목과 항목별 내용, 전체적 구조 등을 어떻게 결정해야 할 것인가? 바로 업무 매뉴얼 타입의 정의에 대한 연구가 필요한 것이다. 더욱이 분산 조직의 경우, 이런 복잡성 문제는 새로운 차원의 복잡성을 추가하게 된다. 과연 표준화된 업무 매뉴얼을 분산 조직의 각 사업장에 획일적으로 적용할 수 있는가? 다수의 공장을 가지고 있는 기업의 경우, ISO900X 인증을 특정 공장만을 대상으로 하는 것도 이런 문제 때문이다. 우리는 업무 매뉴얼의 복잡성과 이질성을 효과적으로 처리하기 위해 업무 매뉴얼 자체에 대한 추상을 시도한다. 이런 의도에서 본 논문의 기본 접근 방법은 "업무 매뉴얼의 작성은 프로세스 클래스의 정의와 같다."라는 것이다. 이런 객체지향 패러다임에 의한 업무의 추상은 객체지향이 지향하는 모듈화 추상(Modular Abstraction), 재사용(Reuse), 캡슐화(Encapsulation), 유전관계(Inheritance)와 폴리모피즘(Polymorphism) 등의 개념을 활용할 수 있어 보다 직설적이며 단순하고 일관성 있는 업무 매뉴얼 작성을 성취할 수 있을 것이다.

본 논문에서는 기능 분할(Function Decomposition)을 BPE의 논리적 분류 방법으로 사용하고 있다. 기능 분할은 기업의 업무를 기능상 특징을 기준으로 상하 계층 구조로 분류하는 대표적인 프로세스 정의 방법이다[Martin, 1989], [Yourdon, 1989]. 기능 분할 방법은 일반적으로 제조, 영업, 구매, 회계, 자금, 인사, 기획, 연구 등의 기능 영역을 최상위 분류로 하고 각 기능 영역을 중분류 기준인 기능으로 분류하고 마지막으로 기능별 업무, 즉 프로세스를 분류하는 이른바 3계층구조(Three-Tier Hierarchy) 분할 방법을 일컫는다. 이러한 분류는 업무의 수행이나 조직적 분장에 전혀 영향을 주지 않는 논리적인 것임을 밝혀 둔다. 물리적으로 여러 부서에 거쳐 발생하는 연계 프로세스(Cross Functional Process)의 경우에는 논리적으로 분류되어 있는 프로세스들을 선행, 후행, 병행 관계로 조립함으로써 추상할 수 있다.

분산 조직은 물리적, 논리적으로 분산되어 있는 사업장으로 구성된 기업을 의미한다. 물리적 분산 조직(Physically Decentralized Organization)은 지역적으로 분산되어 있는 다수의 사업장으로 구성된 조직을 의미하며 논리적 분산 조직(Logically Decentralized Organization)이란 제조, 영업 등 본원적 기능 영역이 각각 그 형태에 따라 분리되는 조직을 의미한다. 일반적으로 물리적 분산성은 사업장 특성으로, 논리적 분산성은 부서 특성으로 결정된다. 우리는 이런 분산성을 나타내기 위해 2차원 평면에 분산성을 투영한 분산 맵(Decentralization Map)을 작성하였다. 분산 맵의 X축은 논리적 분산성을 나타내며 Y축은 물리적 분산성을 나타낸다. 분산 맵에서 분산도가 높게 나타나는 기능 영역의 프로세스는 이질적인 특성을 내포하기 때문에 하나의 표준화된 클래스를 정의한다는데 현실적으로 무리가 있다. 따라서, 우리는 "분산도가 높은 프로세스에 대해서는 일반적인 프로세스 클래스를 기반으로 이를 각 분산 조직에 맞게 특수화한다."라는 전략을 사용하기로 한다.

본 논문에서는 본사, 4개 공장, 6개 영업소로 구성된 D사의 BPE 프로젝트를 제시될 방법론의 설명을 위해 소개한다. <그림 1>은 D사의 업무 매뉴얼 작성에 있어 도출된 기능 영역별 분산 맵으로서 제조, 수급, 영업, 총무, 회계, 자금, 기획, 기술 기획, 연구 등 기능 영역을 사업장 특성과 부서 특성에 따라 분산성을 정의한 것이다.



<그림 1> D사의 기능영역별 분산 맵

예를 들어 제조의 경우, 분산된 공장, 영업소를 공급 방식에 따라 Bulk, Tonnange/On-Site,

Cylinder, 특수 가스 등에 따라 분류하여 각각 특성을 인정하고 업무 매뉴얼을 작성하였다. D사는 9개 기능 영역에 총 322개 프로세스 클래스로 구성된 비즈니스 모델을 1995년 2월부터 1996년 1월까지 비즈니스 엔지니어링 프로젝트의 결과로 구축하였으며 이중 제조, 영업 기능 영역의 일부 프로세스 클래스에 대해서는 일반적 프로세스 클래스를 정의하여 각 공장별, 영업 부서별로 특수화하였다. 현재, D사의 프로세스 클래스는 각각 본사, 공장, 영업소 정보시스템으로 구성된 분산 정보시스템으로 구현 중에 있다.

Ⅲ. 비즈니스 모델

3.1 기본 구조

BPE에 대한 우리의 접근 방법은 프로세스를 속성과 연산으로 구성된 클래스로 추상하는 것이다. 속성은 프로세스에 대한 정보를 표현한다. 프로세스 명, 버전 번호, 발생 빈도, 참고 자료, 사용 양식 등이 속성의 예이다. 반면에 연산은 프로세스의 행동적 측면을 포착한다. 연산에 대한 구체적 실행 방법인 메소드는 다음 측면에서 기술된다.

- 절차적 측면: 무엇을 하는가(What to Do);
- 작업 흐름적 측면: 어떤 부서에서 하는가(How to Flow)
- 지침적 측면: 어떻게 하는가(How to Do);

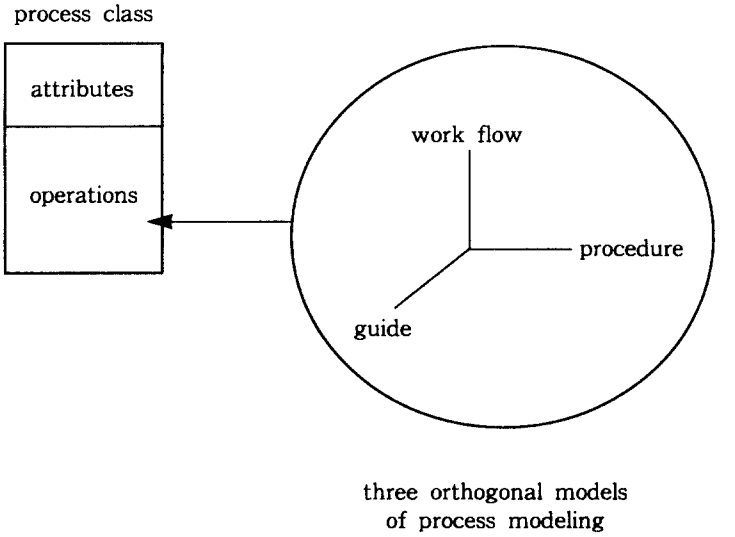
절차적 측면은 연산이 수행되는 세부 작업(Activities)를 정의하고 이들간의 순서를 추상한 것으로 작업에 대한 구체적 수행 방법은 추상 되지 않는다. 예를 들어, 용기 입고 처리 프로세스의 경우, 첫번째, 작업(연산)인 "입고 접수"에 대한 절차적 측면은 "용기 입고 접수 대장 기록"이라는 작업에 대한 정의일 뿐이다.

작업 흐름적 측면은 프로세스가 물리적으로 어떤 부서, 어떤 담당자가 수행되는 지를 추상한 것으로 프로세스의 가장 동적인 측면을 추상한 것이다. 따라서, 리엔지니어링의 1차적인 대상이 되어 조직 개편이나 프로세스 재 설계시 변경되어야 하는 측면이다. 예를 들어, "입고 접수"에 대한 작업 흐름적 측면은 수급부 창고 담당이 처리함을 추상한 것이다.

마지막으로 지침적 측면은 프로세스를 구체적으로 어떻게 수행해야 하는 방법을 추상한 것이다. 절차적 측면이 수행되어야 할 작업에 대한 정의와 순서에 초점을 두고 있는데 비해 지침적 측면은 개별 작업을 수행하는데 필요한 지식이나 경험의 추상에 초점을 두고 있다. 예를 들

어, "입고 접수"작업의 경우, 체크 해야 할 사항이나 조심해야 할 작업 등을 추상하는 것이다.

이상의 세가지 측면은 연산의 추상에 있어 상호 보완하는 수직적 관계를 갖는다. 절차적 측면은 연산에서 수행되어야 순차적 작업에 대한 정의를 추상하고, 작업 흐름적 측면은 절차적 측면에서 정의된 작업이 조직 구조상 어떤 곳에서 수행되는가의 흐름을 추상하고, 마지막으로 지침적 측면은 절차적 측면에서 정의된 작업을 작업 흐름적 측면에서 할당된 부서의 담당자가 어떤 지식이나 경험을 가지고 수행해야 하는가를 추상하게 된다.



<그림 2> 프로세스 추상의 기본구조

3.2 프로세스 클래스

프로세스 추상의 기본 구조를 기반으로 프로세스 클래스가 정의된다. 클래스는 매우 유사한 것들의 집합이거나 부류를 대표하는 템플릿, 패턴, 청사진이다[Berard, 1993]. 클래스는 부품 클래스(Component Class)라 불리는 다른 클래스로 구성된 것으로 정의될 수도 있다. 이런 클래스를 혼합 클래스(Composite Class)라 한다. 프로세스 클래스는 혼합 클래스의 형태로 비즈니스 프로세스의 골격을 제공할 것이다.

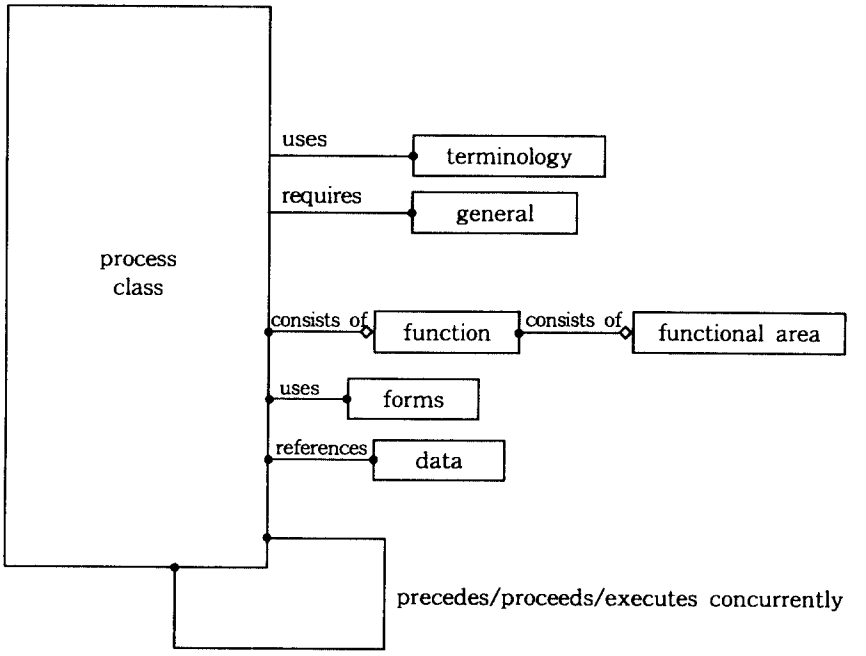
프로세스 클래스가 비즈니스 프로세스 타입을 정의한다면 프로세스 객체는 실질적으로 현장에서 수행되는 업무 사례가 되는 것이다. 즉, 하나의 프로세스 클래스에 상응하는 다수의 프로세스 객체가 업무 수행 과정에서 발생하게 된다. 프로세스 클래스를 생성하기 위한 프로세스 메타 클래스[1]의 속성과 연산은 다음과 같이 정의될 수 있다.

```
metaclass process
{attributes: $name; * process name
             $functional area; * the functional area which the process belongs to
             $function; * the function which the process belongs to
             $version; * process version number
             $code; * process identification number
             $frequency; * occurrence interval(daily, monthly, quarterly, bi-annually, annually,
                    * or anytime)
             $objectives; * reasons why the process exists
             forms; * a list of forms the process uses during the execution of the process
             reference data; * a list of forms and documents the process references
             relationships:
                  precedents;* a list of processes which must finish before the
                          * process starts
                  antecedents; * a list of processes which can start only after the process
                          * finishes
                  concurrents; * a list of processes which can be executed simultaneously
                          * with the process
             $terminology;   * technical words which are used in the process class
                          definition
             $general; * prerequisites, cautions, security rules, or checklist
 operations: subprocess_1();
             subprocess_2();
             ......
             subprocess_n()
}
```

프로세스 메타 클래스의 속성 중에 사용 양식(forms), 참고 자료(reference data), 관련업무(relationships)를 제외한 모든 속성은 클래스 속성[2)][Rumbaugh et al., 1991] 인데 클래스 속성이란 모든 객체가 공통적인 값(또는 디폴트 값)을 갖는 특수 속성이다. 따라서, 프로세스 클래스의 구체적인 인스턴스인 모든 프로세스 객체가 클래스 속성에 대해 공통 값을 가지게 된다.

1) 클래스 정의에서 *로 시작되는 내용은 comment임.
2) 속성명이 $으로 시작되는 속성

클래스의 속성은 설명적 속성(Descriptive Attributes), 명명 속성(Naming Attributes), 참조 속성(Referential Attributes)로 분류하는데[Shlar and Mellor, 1992], 프로세스 클래스의 속성 중 업무명($name), 발생 빈도($frequency), 목적($Objectives) 등은 설명적 속성이다. 코드($code), 버전($version)은 명명 속성이며 용어($terminology), 일반 사항($general), 기능 영역($functional area), 기능($function), 사용 양식(forms), 참조 자료(reference data), 관련업무(relationships)-선행(precedents)-후행(antecedents)-병행(concurrents) 등은 참조 속성이다. 참조 속성은 부품 클래스와의 관계를 내포하므로 <그림 3>과 같이 프로세스 클래스 관계도에서 속성에 따른 타 부품 클래스와의 관계를 설정할 수 있다.



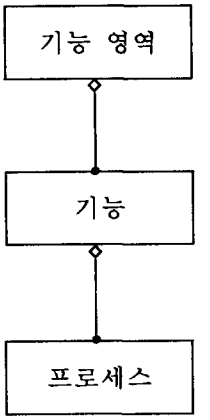
<그림 3> 프로세스 관계도

이미 언급했듯이, 프로세스 클래스의 정의를 위해 3계층 구조 기능 분할 방법을 채택할 때 우리는 기능 영역 클래스와 기능 클래스를 정의하게 된다. 기능 영역 클래스와 기능 클래스의 속성과 연산은 다음과 같다.

class	functional area
{attributes:	name; * functional area name

```
                functions; * a list of functions which belongs to this functional area
operations:     scan(); * to scan functions which belongs to this functional area
                register(); * to add new functions to this function area
                delete() * to delete existing functions from this function area
}
class           function
{attributes:    name; * function name
                functional area; * the functional area which this function belongs to
                processes *a list of processes which belongs to this function;
 operations:    scan(); * to scan processes which belongs to this function
                register(); * to add new processes to this function
                delete(); * to delete existing processes from this function
}
```

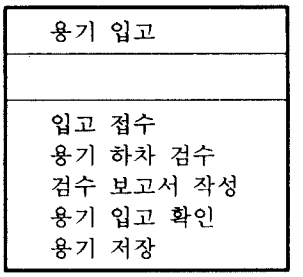
기능 영역, 기능 클래스는 비즈니스 모델의 구조적 측면을 설명하며 하부 구성요소에 대한 분류, 검색, 등록, 삭제 등 관리 목적의 연산을 제공한다. 기능 영역, 기능, 업무 매뉴얼 클래스의 3계층구조는 <그림 4>와 같이 객체지향의 구성 관계로 나타낼 수 있다. 각 기능 영역은 다수의 기능으로 구성되어 있으며 다시 각 기능은 다수의 프로세스로 구성되는 것이다.



<그림 4> 기능 분할의 3계층 구성 관계

프로세스 클래스의 연산은 업무의 수행 절차상 나타나는 단위 작업, 즉 서브 프로세스를 의미한다. 예를 들어 용기 입고 프로세스의 연산은 <그림 5>와 같이 입고 접수, 용기 하차 검수,

검수 보고서 작성, 용기 입고 확인, 용기 저장 등의 서브 프로세스로 정의하게 된다 프로세스 클래스의 연산들은 일반적으로 상호 순차적 흐름 관계를 가진다는 특징이 있다.



<그림 5> 용기 입고 프로세스 클래스의 연산 정의

프로세스 클래스의 연산은 비즈니스 프로세스의 행동적 측면을 설명한다. 연산의 수행 방법인 메소드를 완전히 포착하기 위해 우리는 세가지 수직적 측면에서 추상한다.

3.3 절차 모델

절차 모델(Procedure Model)은 프로세스 연산의 "무엇을 하는가(What to Do)" 측면을 설명한 것이다. 이 모델의 기본 체계는 IPO(Input-Process-Output)이다. 메소드를 설명하기 위해 입력 자료는 무엇이며 이를 어떤 세부 작업들이 처리하여 최종적으로 어떤 출력이 생성되는지를 명확히 설명하는 것이다. 그러나, 주로 입력과 출력에 강조를 두며 각 세부 작업을 어떻게 수행하는지 까지는 설명하지 않는다.

절차 모델은 데이타 흐름도나 프로세스 의존도로 도식화되며 상세한 내역은 슈도 코드와 같은 자연어를 사용하여 기술한다. 절차 모델은 프로세스 자체에 초점을 두며 조직이나 인간적 측면, 예를 들어, 조직 부서명이나 담당자 명 등은 철저히 배제한다. 절차 모델은 정보시스템의 구현 시점에서 프로그램 모듈 구조를 정의하는데 사용된다.

3.4 작업 흐름 모델

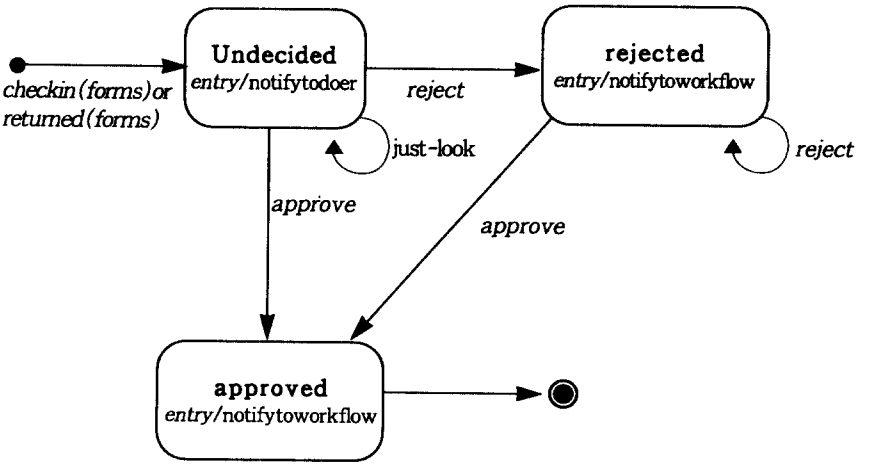
작업 흐름 모델(Work Flow Model)은 비즈니스 프로세스를 수행하는 인간적 활동의 순서를 표현한 것이다[Taylor, 1995]. 그러나, 우리는 이러한 작업 흐름 모델의 기본 정의를 확장하여 프로세스가 착수부터 종료까지 수행되는 경로상의 담당자와 부서를 포함하기로 한다. 이 모델을 구현한 그룹웨어는 기안과 정보를 사전에 정해진 업무 경로에 따라 전달하게 된다.

작업 흐름을 추상하기 위해 다음과 같은 작업 흐름 메타 클래스가 정의된다.

```
metaclass    workflow
{attributes: $name; * workflow name
             $process; * the process which this workflow corresponds to
             $head; * the head workflow_point where the process starts
             $next workflow; * the next workflow to flow into after this workflow finishes
             $previous workflow; * the preceding workflow from which the control is passed
                                 * to this workflow
             current; * the current workflow_point where the process is being executed
 operations: move(); * to make this workflow to proceed
             register(); * to add new workflow points to this workflow
             delete(); * to delete existing workflow points from this workflow
             passtonextworkflow(); * to pass the control to the next workflow
             returntopreviousworkflow(); * to return the control to the previous workflow
}
```

또한, 개별적인 작업 흐름 포인트 메타 클래스는 다음과 같이 정의된다.

```
metaclass    workflow_point
{attributes: $name; * workflow_point name
             $workflow; * the workflow which this workflow_point belongs to
             $next; * the next workflow_point
             $doer; * the person who is responsible for this workflow_point
             forms; * a list of forms which this workflow_point must process
             status; * the status information of this workflow_point(approved, rejected, undecided)
 operations: checkin(); * to initiate processing at this workflow_point
             returned(); * to reinitiate processing at this workflow_point
             approve(); * to approve the current matter
             justlook(); * to look at the current matter and do not decide anything
             reject(); * to reject the current matter
}
```



<그림 6> workflow_point의 상태도

일반적으로 상태 전이도(State Transition Diagram)[Harel, 1987]는 실시간 시스템과 같이 객체의 속성 값 변화를 추적하기 위해 작성하는 모델링 기법으로 경영정보시스템의 분석/설계에는 사용하지 않는다. 그러나, 작업 흐름 모델은 업무처리 결로상 특정 의사결정 지점에서의 상태변화를 추적할 필요가 있으므로 <그림 6>과 같은 상태 전이도를 작성하여 작업 흐름 포인트의 상태 변화를 추상하였다.

3.5 지침 모델

지침 모델(Guide Model)은 프로세스를 수행하는 담당자의 전문지식이나 경험을 포착한 것이다. 따라서 이 모델은 무엇을 하는가 보다는 전문가가 프로세스를 어떻게 수행하는지를 설명한 것이다. 지식 공학이 공식적으로 전문가의 지식을 획득하기 위해 요구된다. 그러나, 모든 메소드가 전문지식을 요구하지 않으므로 지침 모델을 모든 메소드에 대해 구축하지 않는다.

우리는 프로세스의 연산을 다음과 같이 분류하였다.

- 의사결정(DM: Decision Making): 인간 의사결정을 요구하는 서브 프로세스
- 대인활동(HI: Human Interaction): 인간적인 접촉을 수행하는 서브 프로세스
- 분석(AN: ANalysis): 데이타의 분석을 수행하는 서브 프로세스
- 작업(OP:OPeration): 기계와 같은 물리적 대상에 대한 작업을 수행하는 서브 프로세스
- 작성 (RE: REporting): 양식을 작성하는 서브 프로세스
- 거래처리(TP: Transaction Processing): 수시로 발생하는 거래 상황을 기록, 확인, 계산, 발행,

접수, 인쇄하는 서브 프로세스

이상의 연산 분류에 따라 지침 모델의 구축 여부가 결정된다. 연산이 보다 지식 위주일수록 보다 엄격한 지식 공학이 요구된다. 따라서 의사결정형 연산이 가장 많은 지식 공학을 요구하며 거래처리형 연산이 가장 적은 지식 공학을 요구하게 된다.

지침 모델은 다양한 방법으로 표현된다. 첫째, 가장 보편적인 지식 표현은 IF-THEN 규칙이다. 각 서브 프로세스에 대해 비즈니스 규칙을 발견하고 이를 전문가시스템[Rich and Knight, 1991]으로 개발하게 된다. 두번째, 규칙을 획득하기 어려울 경우, 사례를 수집하여 이를 사례베이스에 저장하고 차후에 업무를 수행할 경우 이를 활용하게 된다.

지식을 전문가로부터 수집하는 것은 쉬운 일이 아니다. 따라서 가장 쉬운 방법은 사례 분석일 것이다. 이미 언급된 대로 프로세스 객체는 프로세스 클래스의 인스턴스이다. 프로세스가 실제로 수행될 때마다 프로세스 객체는 생성된다. 이런 프로세스 객체가 축적되면 사례 분석이 수행될 수 있다. 충분한 사례 분석 후, 사례베이스가 구축되고 유사한 업무의 수행 시 과거 프로세스 객체는 검색되어 참조된다. 사례베이스의 구축과 운용을 위해 프로세스 메타 클래스에 다음 연산이 추가된다.

```
metaclass    process
{            ...
operations:  process[3] (); * to create a new process object
             match(); * to match and return most similar past cases with a given current case
             ......
}
```

Ⅳ. 일반적 프로세스 클래스, 특수화, 캡슐화, 버전

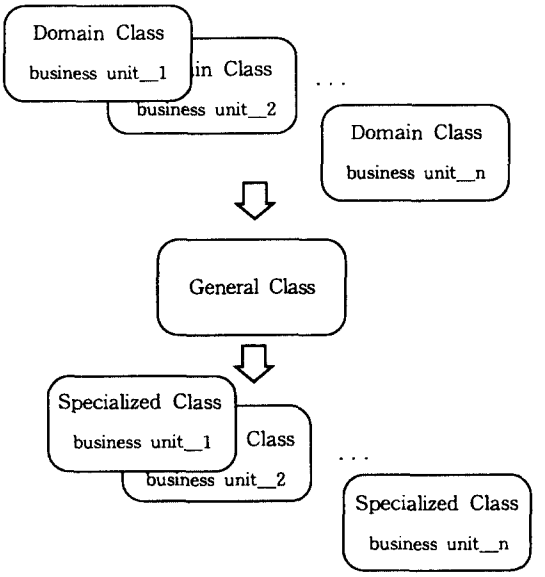
4.1 일반적 프로세스 클래스

프로세스 클래스를 정의한 후 우리는 분산 조직을 위한 비즈니스 모델 구축을 위한 방법이 필요하게 된다. 획일적 표준이란 대부분의 분산 조직에서 가능하지 않기 때문에 참조 모델을 구축하여 이를 기준으로 각 사업장별로 수정하는 것이 현실적일 것이다. 참조 모델은 각 사업장이

3) 클래스명과 같은 이름을 가지는 연산은 Constructor로서 객체를 생성한다.

각각 자신에 맞게 수정할 수 있는 일반적 프로세스 클래스(General Process Class)로 구성된다.

그러나, 어떻게 참조 모델의 일반적 프로세스 클래스를 도출할 수 있는가의 문제가 남게 된다. 우리는 일반적 프로세스 클래스를 <그림 7>과 같이 3단계를 거쳐 도출하였다. 첫째, 각 사업장별로 도메인 프로세스 클래스를 정의한다. 이 도메인 프로세스 클래스는 사업장별 특성을 그대로 반영하게 된다. 예를 들어, 각 사업장별로 상이한 양식과 데이타를 사용할 수 있으며 한 사업장에서 수행하는 연산을 다른 사업장에서는 사용하지 않을 수 있는 것이다. 두번째 단계로, 도메인 프로세스 클래스는 일반적 프로세스 클래스로 표준화된다. 일반적 프로세스 클래스는 모든 도메인 프로세스 클래스의 특성을 반영할 수 있는 가장 포괄적이며 일반적이어야 한다. 마지막으로 일반적 프로세스 클래스를 기준으로 각 사업장은 다시 각자의 프로세스 클래스를 재정의하게 된다.



<그림 7> 분산조직의 비즈니스 모델 구축절차

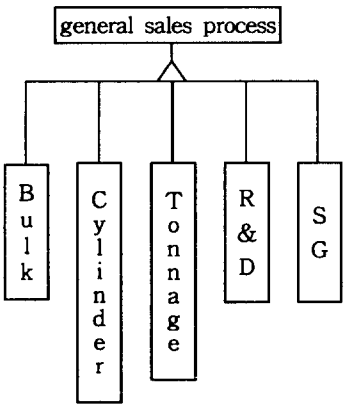
4.2 특수화

분산 조직의 경우 이미 지적한 바와 같이 각 사업장 프로세스는 사업장에 따라 각각 다른 특성을 지니게 된다. 따라서, 각 업무별로 전사적 표준 프로세스 클래스를 정의하는 것은 현실적으로 무리가 따르게 된다. 따라서, 이러한 분산 조직의 경우, 분산 맵을 작성하고 각 사업장별로 특성을 반영하는 프로세스 클래스를 작성하는 것이 바람직할 것이다. 그러나, 같은 프로세스 클

래스는 기업의 비즈니스 프로세스 타입으로서 가능하면 전사적으로 공통된 형태를 갖는 것이 개념적 무결성을 유지할 수 있게 될 것이다. 이러한 목적하에 객체지향의 유전 관계 개념을 적용하여 하나의 표준 안으로부터 특수화된 업무 매뉴얼을 작성하는 방법을 선택할 수 있다. 따라서 사업장 프로세스 클래스는 다음과 같이 정의된다.

```
metaclass    local_process $general process class * the general process class from which
                                                  * this local_process is specialized
{attributes:  ... ; * additional attributes may be defined here;
 operations:  subprocess_1(overriding method);
              subprocess_2(overriding method);
              ...
              subprocess_n(overriding method)
}
```

사업장별 프로세스 클래스는 일반적 프로세스 클래스를 사업장에 맞게 특수화 시킨 프로세스 클래스이다. 사업장 프로세스 클래스는 기본적으로 일반적 프로세스 클래스의 속성과 연산을 유전 받게 된다. 그러나, 속성의 값은 기본값을 무시하고 독자적인 값을 가질 수 있으며 연산의 구현인 메소드, 즉 서브 프로세스에 대한 내역 또한 독자적으로 구현할 수 있다. 여기서 중요한 것은 속성 명이나 연산 명은 반드시 동일하게 사용해야 한다는 것이다. <그림 8>은 D사의 분산 맵에 따른 영업 기능 영역에 대한 일반적 프로세스 클래스에 대한 사업장 프로세스 클래스의 유전 관계를 나타내고 있다.



<그림 8> D사의 사업장 업무 매뉴얼 클래스 유전 관계

4.3 캡슐화

기업은 전통적으로 사업장, 사업 본부, 부서, 그 외 기능적 그룹 등을 단위로 조직화해 왔다. 이런 단위 원칙을 유지하면서 조직은 결속시키는 공통적인 목적이나 기능을 가지게 된다. 더욱이 조직은 타부서의 직접적인 간섭을 배제시키기 의해 업무 활동이 명령 계통에 따라 수행되도록 캡슐화(Encapsulation) 해 왔다.

동일한 원칙은 프로세스에도 적용된다. 프로세스는 서로 명확한 인터페이스를 통해서만 상호 의사소통할 수 있으며 직접적인 액세스를 막고 있다. 프로세스 클래스에 내포된 정보나 연산의 수행 방법은 클래스 외부에는 노출되지 않는다. 그러면 우리는 이런 캡슐화로 무엇을 얻는가? 프로세스 클래스의 내부 설계의 변화는 외부 환경, 즉 사용자에게 영향을 주지 않는다는 것이다. 이것은 클래스 내부 변화를 지역적인 영향에 그치게 하고 해당 클래스의 핵심에만 영향을 주게 한다.

4.4 클래스 버전

버전(Version)이란 어떤 타입이나 원본(Original)의 변형으로 정의되는데[Mish, 1988] 객체지향에서는 일반적으로 객체가 속성에 대한 다른 값을 가질 때, 즉 객체의 상태가 달라질 때 이를 개별적인 객체로 기록하기 위해 사용한다[Andleigh and Gretzinger, 1992].

본 논문에서는 이런 객체 레벨의 버전 개념과는 달리 클래스 버전 개념을 도입한다. 클래스 버전은 클래스 속성에 대한 값이 바뀌고 연산에 대한 메소드가 변경될 때 이를 각각 다른 클래스 상태, 즉 클래스 버전으로 정의하여 관리하는 것이다. 클래스 버전 개념으로 프로세스 클래스의 변경에 따른 라이프사이클 관리가 가능하게 되었다. 그리고 이를 위해 다음과 같은 클래스 버전 관리 클래스가 정의된다:

```
class          class_version
{attributes:   process; * the process class for which this class version manages versions
               versions; * a list of versions of this process class
operations:    scan(); * to scan versions which have been registerd to this class_version
               register(); * to add new versions to this class_version
               delete(); * to delete versions from this class_version
}
```

V. 결 론

본 논문에서 분산 조직을 위한 객체지향 비즈니스 프로세스 엔지니어링 방법이 제시되었다. 이 방법의 중심 산출물인 비즈니스 모델은 프로세스 클래스로 구성되어 있다. 일반적 프로세스 클래스는 각 사업장별로 특수화된다. 일반적 프로세스 클래스의 경계 안에 사업장 프로세스 클래스의 정의를 제한함으로써 개념적 무결성이 유지될 수 있다. 따라서, 분산 조직의 이질성을 효과적으로 처리하며 체계적인 프로세스 모델링이 가능하게 된 것이다.

본 논문에서 정의하고 있는 프로세스 클래스는 속성보다는 연산의 비중이 더 높으며 객체지향 분석 단계에서 도출될 콘트롤 객체[Jacobson, 1995]나 엔티티 객체의 연산으로 다시 추상될 것이다. 또한, 프로세스 클래스는 객체를 도출하기 위한 문제 기술로 사용하는 사용 케이스(Use Case)[Jacobson, 1995]나 시나리오[Rumbaugh, et al., 1991]와 같은 의미로 볼 수도 있다. 그러나, 프로세스 사양을 단순히 객체를 도출하기 위한 사전 자료로 인식하던 기존의 객체지향 분석 방법에 비해 본 논문은 프로세스 자체를 클래스로 정의함으로써 엔티티 객체의 연산이나 콘트롤 객체를 추상하는데 있어 보다 직설적이며 단순하고 일관된 방법을 제시하고 있다고 볼 수 있다. 이상에서 본 논문의 주요 공헌은 다음과 같이 요약할 수 있다.

- 비즈니스 프로세스 자체를 클래스로 추상하여 프로세스로부터의 비즈니스 객체 분석, 더 나아가 정보시스템 구현까지 매듭 없는 전개(Seamless Transition)를 가능하게 하였다.
- 프로세스의 복합적 측면을 세가지 수직적 모델의 구축으로 완벽하게 포착하였다.
- 분산 조직을 위한 일반적 프로세스 클래스를 구축할 수 있는 단계적 방법이 제시되었다.
- 클래스 유전 관계를 통해 통제된 범위 내에서 사업장별 프로세스 클래스를 정의하는 실용적 프로세스 모델링 방법이 제시되었다.
- 프로세스 클래스의 정의로 적응적 프로세스의 정의가 실현되었다. 프로세스 클래스는 캡슐화 되어 내부 변화가 외부에 최소의 영향을 미치게 된 것이다.

그러나, 객체지향 비즈니스 프로세스 엔지니어링 방법을 적용하는 과정에서 발견된 몇 가지 보완점이 있다. 첫째, 프로세스의 분류, 도출에 대한 명확한 기준이 제시되어야 한다는 것이다. 프로세스별로 복잡성, 난이성 등의 수준차가 심한 경우가 종종 발견되었기 때문이다. 둘째, 지침적 측면에서 사용한 사례 수집, 분류 방법이 현실적으로 프로세스의 착수 시점에서 종료 시점까지 발생한 작업이나, 작성된 양식을 추적해야 하는 어려움이 있어, 이를 공식적인 지식 획득(Knowledge Acquisition) 방법으로 추상하여 공식 문서로 작성하는 것이 요구되었다. 마지막으로, 프로세스 추상의 결과로 작성된 업무 매뉴얼(프로세스 클래스)을 정보시스템 개발자가 객체

지향 방법론을 적용하지 않을 경우, 단순한 업무 분석서 이상의 의미를 부여하지 않아 정보시스템 개발에 직접적인 도움을 주지 못하게 되어, 프로세스 클래스에서 정보시스템의 클래스를 도출하는 구체적인 방법이 제시 되어야 할 것이다.

객체지향 패러다임에 기반을 둔 BPE는 추상 과정은 거치지만 프로세스에 대한 명확하고 완전한 이해를 목적으로 한다. 분산 조직의 경우, BPE의 산출물인 비즈니스 모델을 근간으로 비즈니스 프로세스 리엔지니어링, 벤치마킹, 정보시스템 개발을 성공적으로 실현할 수 있을 것이다. 본 논문에서 제시된 분산 조직을 위한 일반적 프로세스 클래스의 사업장별 특수화 개념은 다수 기업으로 구성된 그룹의 경우에 기업별 프로세스 클래스의 특수화로 확장될 수 있을 것이다. 실제로 D그룹은 D사와 같은 전형적인 분산 조직에 비즈니스 모델을 구축한 후 공통 업무(총무, 회계, 자금, 영업 등)는 D그룹 계열 회사의 비즈니스 엔지니어링 프로젝트에서 재사용, 확산되고 있다.

참 고 문 헌

1. 고일상, "객체지향 프로세스 모델링을 이용한 비지니스 프로세스 관리시스템의 구현", 추계학술대회 논문집, 한국경영정보학회, 1996, pp. 499-513.
2. 권태형, 연광호, 최은희, "프로세스 모델링을 위한 개념적 틀", 추계학술대회 발표논문집, 한국경영과학회, 1995, pp. 233-248.
3. 김성근, 김진수, 황순삼, "정보지향 프로세스 모델링 도구의 개발 및 적용" , 추계 학술 대회 논문집, 경영정보학회, 1996, pp. 459-473.
4. Andleigh, P., and Gretzinger, M., *Distributed Object-oriented Data Systems Design*, Englewood Cliffs, N.J.: Prentice-Hall, 1992.
5. Baja, A., and S. Ram, "A Business Process Model Based on a Comprehensive Content Specifications", *Proceedings of the America's Conference on Information Systems*, Pheonix, AZ, August 16-18, 1996.
6. Berard, E., *Essays on Object-oriented Software Engineering*, Englewood Cliffs, N.J.: Prentice Hall, 1993.
7. Brooks, F. P., *The Mythical Man-Month*, Reading, M.A.: Addison-Wesley, 1982.
8. Camp. R. C., "Benchmarking: The Search for the Best Industry Practices that Lead to Superior Performance", *ASQC Quarterly*, Milwaukee Press, 1989.
9. Curtis, B., Keller, M. I., and Over, J., "Process Modeling", *Communications of the ACM*, Vol. 35, No. 9, Sep. 1992.
10. Hammer, M. and Champy, J., *Reengineering the Corporation*, New York, N.Y.: Harper Business, 1993.
11. Harel, D., "Statecharts: A Visual Formalism for Complex Systems", *Science of Computer Programming*, Vol. 8, 1987, pp. 231-274.
12. Hradesky, J. L., *Total Quality Management Handbook*, New York, N.Y.: Mcgraw-Hill, 1995.
13. Jacobson, I. et al., *Business Process Reengineering with Object Technology*, Reading M.A.: Addison-Wesley, 1995.
14. Karagiannis, D., "BPMS - Business Process Management Systems", *SIGOIS Bulletin*, Vol. 16, No. 1, August 1995, pp. 10-13.
15. Martin, J., *Information Engineering Book II*, Englewood Cliffs, N.J.: Prentice-Hall, 1989.
16. Mish, F. C., *Editor in Chief, Webster's New Collegiate Dictionary*, Springfield, MA.

Merriam Webster Inc., 1988.

17. Rich, E. and Knight, K., *Artificial Intelligence*, 2nd ed., New York, N.Y.: McGraw-Hill, 1991.
18. Rumbaugh, J. et al., *Object-Oriented Modeling and Design*, Englewood Cliffs, N.J.: Prentice-Hall, 1991.
19. Scherr, A. L., "A New Approach to Business Processes", *IBM Systems Journal*, Vol. 32, No. 1, 1993.
20. Shlaer, S. and Mellor, S. J., *Object Lifecycles*, Englewood Cliffs, N.J.: Yourdon Press, 1992.
21. Taylor, D., *Business Engineering with Object Technology*, New York, N.Y.: John Wiley & Sons, 1995.
22. Yourdon, E., *Modern Structured Analysis*, Englewood Cliffs, N.J.: Prentice-Hall, 1989.

<Abstract>

Object-oriented Process Engineering for Decentralized Organizations

Kwangho Park

The demands for continuous process optimization require a radical rethinking of how information systems are designed and constructed. Information systems must be capable of sustained, graceful change in response to evolving business requirements. This proposition is supported by the fact that information systems have helped increase productivity only when they were built to support new and better ways of conducting the business. In such context, business process engineering(BPE) is recognized the first but most critical stage in developing information systems as well as in launching business improvement and innovation projects. However, more often, there exist great gaps and inconsistencies between the results of BPE and the business itself, which turn into huge maintenance overhead during an information system lifecycle. In order to solve such an ever lasting problem, this paper presents a new process abstraction method based on the object-oriented paradigm. Three orthogonal models, procedure, guide, and work flow are constructed in our engineering process. The method aims at especially BPE for decentralized organizations. Although discussions focus on decentralized organizations, the method is general enough to be applied easily to other types of organizations without difficulties.