분산조직의 프로세스 추상을 위한 객체지향 모델링

(An Object-oriented Modeling for Process Abstraction in Highly Distributed Organizations)

박광호

한양대학교 경영학부

요약

기업경영을 공학적으로 추상하려는 노력은 정보시스템의 개발과정에서 자연스럽게 시작되었다. 그러나, 정보시스템 구축을 주목적으로 한 공학적 추상은 실제 업무와 커다란 격차를 가지게 마련이다. 불완전한 업무분석이 정보시스템 유지보수 비용의 주범이라는 지적은 이러한 주장을 뒷받침하고 있다. 기업경영에 대한 공학적 추상을 도구가 아닌 당위론적인 목적으로 시도하려는 것은 기업경영의 내부구조를 정립하려는 시도와 동일한 것으로 이해해야 할 것이다. 본 논문은 기업경영에 대한 공학적 추상을 위한 구체적인 방법론을 제시하고 있다. 객체지향 패러다임이 추상의 방법론으로 적용되었다.

1. 서론

최근 환경변화에 대응하기 위한 기업의 변신 노력이 비지니스 프로세스 리엔지니어링, 벤치마킹 등의 경영혁신방법론으로 추진되고 있다. 그러나, 이러한 내부변혁의 대상과 범위를 효과적으로 정하기 위해서 기업의 경영활동에 대한 명확한 분석이 선행되어야 한다. 기업의 경영활동에 대한 분석은 결국 기업에 대한 추상 모델(Abstract Model)인 비지니스 모델의 도출로 귀결된다. 여기서 추상이란 분석대상인 경영활동에 있어서 불필요한 것은 배제하고 오직 관심있는 특징만을 추출하는 과정이다. 또한 모델이란 현실세계를 일관성있게 공식적인 표기(Notation)와 도식(Diagramming)으로 가시화한 추상체다. 이런 비지니스 모델의 존재여부는 문제발견, 원인분석, 해결책도출 등 일련의 경영혁신과정의 성과를 크게 좌우한다.

현대 기업의 변화방향 중의 하나는 분권화이다(Taylor, 1992). 그러나, 기업 규모의 확장, 세계화, 사업 다각화에 따른 필연적인 조직형태인 분권화는 전사적 업무 추진과 통제 기능의 상실, 표준의 파괴, 조직간의 조정이 어려워지는 등 여러 문제점을 내포하고 있다. 따라서, 이런 분산조직에 대한 비지니스 모델의 구축은 중앙집중식 조직에 비해 평면적인 관점보다는 입체적인 관점에서 추진해야 하는 어려움이 존재하고 있다.

비지니스 모델의 구축을 위한 공학적 접근방법으로 비지니스 엔지니어링을 들 수 있는데 비지니스 엔지니어링이란 기업의 구성요소를 자동차의 부품처럼 표준화, 정형화하고 이들 경영부품을 조립하여 기업을 추상하는 것이라고 정의할 수 있다. 또한, 비지니스 엔지니어링은 경영활동을 추상하는데 있어 조직이나

구성원 등의 인간적이며 유동적인 측면을 배제하고 기능, 프로세스 등 비인간적인 측면만을 모델링하는 접근방법이다.

비지니스 모델의 기본 구성요소는 업무, 즉 프로세스이며 이에 대한 설명은 업무매뉴얼에 기록된다. 그러나, 이러한 프로세스에 대한 설명이 공식적으로 공학적 접근방법에 의해 작성된 사례를 찾아 보기 힘들다. 명확하게 조직과 구성원으로부터 프로세스를 분리하여 업무매뉴얼의 작성을 시도한 예가 없는 것 같다.

본 논문에서는 분산조직의 업무매뉴얼 작성을 위해 공학적인 접근방법인 비지니스 엔지니어링을 기초로 한 객체지향 모델링 방법을 제시한다. 분산조직의 업무매뉴얼 작성을 위한 객체모델링은 중앙집중식 조직의 경우에도 내부적 다양성이 존재하는 경우에도 적용할 수 있는 개념적 기반을 제공할 것으로 기대된다. 2절에서는 객체지향 패러다임중 본 논문에서 사용되는 개념을 요약하였고, 3절에서는 본 논문이 대상으로 하는 전제와 상황이 설명되었으며, 4절에서는 모델링의 기본 구조인 클래스의 정의를 제시하고 5절과 6절에서는 업무매뉴얼 클래스의 특수화와 클래스버전 개념을 각각 설명하였으며, 끝으로 7절에서 결론을 맺고 있다. 본 논문에서 제시되는 개념의 보충설명을 위해서 D사의 업무매뉴얼 작성 예를 제시함을 밝혀둔다. 또한 클래스정의 메소드로서 OMT(Object Modeling Technique)(Rumbaugh et al., 1992)를 사용하였다.

2. 기본 개념(Fundamental Concepts)

객체지향 패러다임의 기본 개념중의 첫째는 클래스와 객체의 관계적 정의에 있다. 일반적으로 클래스는 템플릿(Template), 설명(Description), 패턴(Pattern), 타입(Data Type), 청사진(Blueprint), 주형(Cast) 등으로 설명하고 객체는 인스턴스(Instances), 변수(Variables), 실물(Constructs), 부품(Parts) 등으로 설명하고 있는데 이 가운데 클래스와 객체와의 관계를 가장 쉽게 설명하는 예는 주형과 부품의 관계일 것이다. 즉, 주형은 틀로서 부품을 찍어 내는데, 여기서 부품의 주형이 클래스라면 주형이 찍어내는 부품들은 객체라고 볼 수 있다. 그러나, 본 논문의 의도에 맞는 설명은 컴퓨터 언어적 측면에서 본 데이타 타입과 변수의 관계일 것이다. 즉 클래스는 데이타의 구조를 정의한 데이타 타입이며 객체는 데이타 타입에 정의된 속성에 대한 구체적인 값을 가지는 변수인 것이다.

둘째, 구성관계(Aggregation)는 특수형태의 관계로 구성요소가 되는 클래스들과 이들로 조립된 클래스 사이의 관계로 부품-제품(part-whole) 관계 또는 부품(part-of, consist-of) 관계로 불리기도 한다. 구성관계의 중요한 특성은 전이성(Transitivity)이다(Rumbaugh et al., 1991). 예를 들어 A가 B의 부품이고, B가 C의 부품이면 A는 C의 부품이 된다. 또한, 구성관계는 비대칭이다. A가 B의 부품이면 B는 A의 부품이 될 수 없다.

세째, 유전관계(Inheritance)는 서로 다른 클래스들이 유사한 속성과 연산을 공유하기 위한 추상방법으로 일반화(Generalization), 특수화(Specialization) 관계로 불리기도 한다(Rumbaugh et al., 1991). 유전관계는 어떤 클래스와 그 클래스의 하나이상의 세련된 버전 사이의 관계다. 여기서 세련되는 클래스를 슈퍼 클래스, 슈퍼클래스로부터 세련된 클래스를 서브클래스라 부른다. 일반화는 슈퍼클래스가 서브클래스의 특성을 일

반화한다는 사실에서, 특수화는 서브클래스가 슈퍼클래스를 세련시켜 특수화한다는 사실에서 각각 그 개념이 만들어졌다.

마지막으로 폴리모피즘(Polymorphism)이란 한 연산이 다양한 구현을 가지는 것을 말한다(Berard, 1993). 즉, 하나의 정의에 대해 여러가지 형태로 구현된 메소드가 존재하는 것이다. 또한 폴리모피즘이란 하나의 프로세스에 대해 다수의 버전이 존재하는 것을 의미하기도 한다. 따라서 폴리모피즘은 해머와 챔피(Hammer and Champy, 1993)가 동일한 프로세스에 대해 여러가지 버전이 존재해야 한다는 주장을 구현할 수 있는 개념이기도 하다. 예를 들어, 신용대출승인 프로세스는 컴퓨터로 자동처리 될 수 있는 평범 케이스와 담당자가 직접 처리해야 하는 난이 케이스, 그리고 전문가의 지원이 필요한 특별 케이스별로 따로따로 처리 절차가 규정되어 있다. 폴리모피즘이 유전관계와 함께 내포하는 중요한 개념은 무시 개념(Overriding Concept)이다. 이는 서브클래스가 슈퍼 클래스의 메소드를 무시하고 자신의 독자적인 메소드를 구현할 수 있음을 의미한다.

3. 전제(Premises) 와 상황(Context)

브룩스(Brooks, 1982)는 시스템 설계에 있어서 가장 중요한 고려사항은 개념적 무결성(Conceptual Integrity)의 유지이며 이를 바탕으로 시스템에 대한 사양이 개념적으로 단순하고 명확하게 유지될 수 있다고 지적하고 있다. 여기서 개념적 무결성의 유지원칙은 반드시 지켜져야 할 당위적 법칙이라기 보다는 경험에 따른 휴리스틱의 성격을 가진다고 하겠다. 개념적 무결성은 그 결과로 직설성(Straightforwardness), 단순성(Simplicity), 일관성(Consistency) 등을 제공한다. 우선, 직설성이란 인식과 이해의 대상을 투명하게 드러낼 수 있는 정직한 표현정도를 나타내는 기준이다. 둘째, 단순성은 인간의 인식과 이해의 한계를 인정할 때 복잡성을 가장 효과적으로 처리할 수 있는 전략이다. 마지막으로 일관성이란 인식의 결과에 대한 갈등을 최저 수준으로 유지하기 위한 조건이다.

본 논문의 기본 전제는 첫째, 목적측면에서 개념적 무결성을 추구한다는 것이다. 그리고 이런 목적의 당위성은 경험적 선호임을 밝혀 둔다. 그리고 개념적 무결성이 보장될 때 우리는 직설성, 단순성, 일관성 등의 세가지 결과를 기대할 수 있을 것이다.

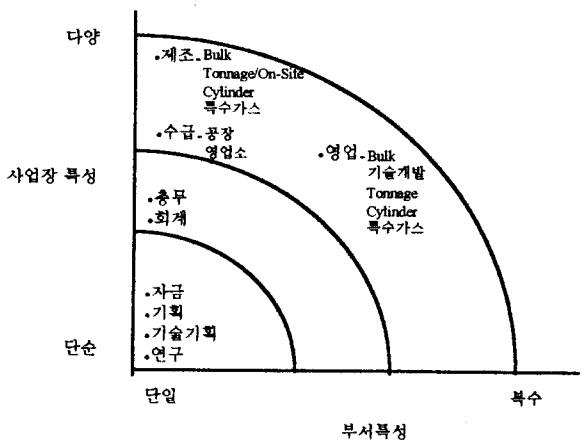
추상이란 복잡성을 효과적으로 이해, 처리하기 위한 방법이다. 업무매뉴얼은 프로세스에 대한 설명이지만 그 자체적으로도 복잡성을 내포하고 있다. 업무매뉴얼에 포함되어야 할 항목과 항목별 내용, 전체적 구조 등을 어떻게 결정해야 할 것인가? 바로 업무매뉴얼 타입의 정의에 대한 연구가 필요한 것이다. 더우기 분산조직의 경우, 이런 복잡성 문제는 새로운 차원의 복잡성을 추가하게 된다. 과연 표준화된 업무매뉴얼을 분산조직의 각 사업장에 획일적으로 적용할 수 있는가? 우리는 업무매뉴얼의 복잡성을 효과적으로 처리하기 위해 업무매뉴얼 자체에 대한 추상을 시도한다. 즉 "업무매뉴얼의 작성은 클래스의 정의와 같다"라는 전제하에 업무매뉴얼 타입을 정의함으로써 업무매뉴얼의 작성에 있어 개념적 무결성을 유지하는 것이다. 이런 객체지향 패러다임에 의한 업무매뉴얼의 추상은 객체지향이 지향하는 모듈화 추상(Modular

Abstraction), 재사용(Reuse), 캡슐화(Encapsulation), 유전관계와 폴리모피즘 등의 개념을 활용할 수 있어 보다 직설적이며 단순하고 일관성 있는 업무매뉴얼 작성을 성취할 수 있을 것이다.

분산조직은 물리적, 논리적으로 분산되어 있는 기업을 의미하며 기본적으로 상호 배타적인 영역을 전제로 한다. 물리적 분산조직(Physically Distributed Organization)은 지역적으로 분산되어 있는 다수의 사업장으로 구성된 조직을 의미하며 논리적 분산조직(Logically Distributed Organization)이란 제조, 영업 등 본원적 기능영역이 형태에 따라 분리되는 조직을 의미한다. 우리는 이와 같은 분산조직에 있어 물리적 분산성은 사업장 특성으로, 논리적 분산성은 부서 특성으로 각각 나타내고 이를 기준으로한 2차원 평면에 분산성을 투영한 분산 맵(Distributedness Map)을 작성하였다.

기업의 업무는 일반적으로 상하 계층구조로 설명하는 것이 일반적인데 대표적인 방법은 업무의 기능상 특징을 기준으로 분류하는 기능분할(Function Decomposition) 기법이 있다(Martin, 1989). 기능분할 방법은 기업경영을 생산, 영업, 구매, 회계, 자금, 인사, 기획, 연구 등의 기능영역을 최상위 분류로 하고 각 기능영역을 중분류 기준인 기능으로 분류하고 마지막으로 기능별 업무, 즉 프로세스를 분류하는 이른바 3계층구조(Three-Tier Hierarchy) 분할방법을 일컫는다. 이러한 분류는 업무의 수행이나 분장에 전혀 영향을 주지 않고 다만 개념적인 편리성을 위한 것임을 밝혀 둔다.

(그림 1)은 D사의 업무매뉴얼 작성에 있어 도출된 기능영역별 분산 맵으로서 제조, 수급, 영업, 총무, 회계, 자금, 기획, 기술기획, 연구 등 기능영역을 사업장 특성과 부서 특성에 따라 분산성을 정의한 것이다. 예를 들어 제조의 경우, 분산된 공장, 영업소를 공급방식에 따라 Bulk, Tonnange/On-Site, Cylinder, 특수가스 등에 따라 분류하여 각각 특성을 인정하고 업무매뉴얼을 작성하였다.



(그림 1) D사의 기능영역별 분산 맵

4. 업무매뉴얼 클래스 정의

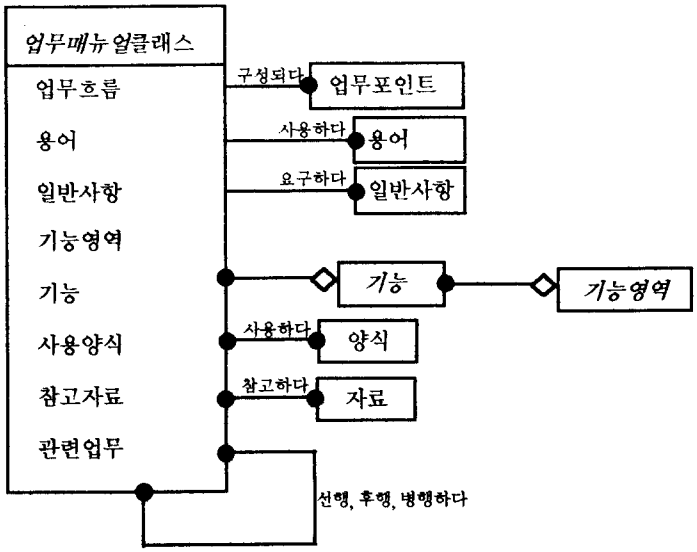
*업무매뉴얼*클래스는 기업 경영의 기본 구성단위인 프로세스에 대한 설명서로서 업무의 타입에 대한 정의라고 할 수 있다. *업무매뉴얼*클래스가 업무에 대한 재반 규정, 패턴, 청사진이라고 한다면 실질적으로 현장에서 수행되는 업무사례는 *업무매뉴얼*객체가 되는 것이다. 즉 하나의 *업무매뉴얼*클래스에 상응하는 다수의 *업무매뉴얼*객체가 업무수행과정에서 발생하게 된다.

*업무매뉴얼*클래스를 생성하기 위한 *업무매뉴얼*메타클래스의 속성과 연산은 다음과 같이 정의될 수 있다.

메타클래스: *업무매뉴얼*
{ **속성**: $*업무명*: 명사+동사형의 업무내용을 대표할 수 있는 이름;
$*기능영역*: 업무에 대한 2레벨 상위의 분류;
$*기능*: 업무에 대한 1레벨 상위의 분류;
$*버전*: 업무매뉴얼의 개정번호;
$*업무코드*: 업무에 대한 분류코드;
$*발생빈도*: 업무의 발생빈도로 (수시, 일, 월, 분기, 반기, 년) 중의 하나임;
$*업무흐름*: 업무의 착수에서 종료까지의 흐름;
$*목적*: 업무의 수행목적에 대한 설명;
사용양식: 업무의 수행과정에서 사용하는 양식의 리스트;
참조자료: 업무의 수행과정에서 참조하는 양식, 책자 등의 리스트;
관련업무: *선행*: 업무의 착수이전에 완료되어야 할 업무의 리스트;
후행: 업무의 종료이후에 수행되어야 할 업무의 리스트;
병행: 업무의 수행과정에서 동시에 수행되는 업무의 리스트;
$*용어*: 업무의 수행과정에서 사용하는 용어와 정의;
$*일반사항*: 업무의 수행에 필요한 사전 지식, 주의사항, 안전조항, 체크리스트;
연산: *서브 프로세스_1*{업무의 처리과정 중 첫번째 서브 프로세스의 내역};
서브 프로세스_2{업무의 처리과정 중 두번째 서브 프로세스의 내역 };
.....
서브 프로세스_n{업무의 처리과정 중 마지막 서브 프로세스의 내역 }
}

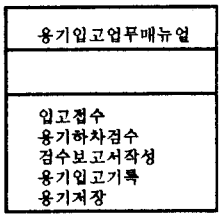
*업무매뉴얼*메타클래스의 속성중에 *사용양식, 참고자료, 관련업무*를 제외한 모든 속성은 클래스 속성(Rumbaugh et al., 1992)으로서 *업무매뉴얼*클래스에 대한 구체적인 인스턴스인 모든 *업무매뉴얼*객체가 공통값을 가지게 된다. 그리고 이러한 *업무매뉴얼*메타클래스의 정의에 따라 *업무매뉴얼*클래스가 정의되는 것이다.

클래스의 속성은 설명적 속성(Descriptive Attributes), 명명속성(Naming Attributes), 참조속성(Referential Attributes)로 분류하는데(Shlaer and Mellor, 1992), *업무매뉴얼*클래스의 속성중 *업무명, 발생빈도, 목적, 업무흐름, 용어, 일반사항* 등은 설명적 속성이며 이중 *업무흐름, 용어, 일반사항*은 혼성데이타타입(Composite Data Type)을 가지는 속성이다. *업무코드, 버전*은 명명속성이며 *기능영역, 기능, , 사용양식, 참조자료, 관련업무-선행-후행-병행* 등은 참조속성이다. (그림 2)는 *업무매뉴얼*클래스의 속성에 따른 타 부속 클래스와의 관계도이다.



(그림 2) 업무매뉴얼 관계도

*업무매뉴얼*클래스의 연산은 업무의 수행절차를 다시 분할했을 때 정의되는 서브 프로세스를 의미하며 세부 처리절차를 메소드로 설명하게 된다. 예를 들어, 용기입고 업무매뉴얼의 경우, (그림 3)과 같이 입고접수, 용기하차검수, 검수보고서작성, 용기입고기록, 용기저장 등의 서브 프로세스를 연산으로 정의하게 된다. *업무매뉴얼*클래스의 연산들은 일반적으로 상호 순차적 흐름관계를 가진다는 특징이 있다.



(그림 3) 용기입고업무매뉴얼클래스의 연산정의

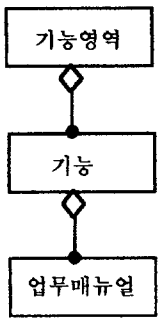
이미 소개된 3계층 구조 기능분할 방법을 채택할 때 우리는 *기능영역*클래스와 *기능*클래스를 정의하게 된다. *기능영역*클래스와 *기능*클래스의 속성과 연산은 다음과 같다.

클래스: *기능영역*
{ 속성: *기능영역명*: 명사형의 기능영역을 대표할 수 있는 이름;
기능:기능영역에 속한 기능의 리스트;

연산: *기능검색*{기능영역에 속한 기능에 대한 검색방법 };
기능등록{신규 기능의 등록방법 };
.....
기능삭제{기존 기능의 삭제방법 }
}

클래스: *기능*
{ 속성: *기능명*: 명사형의 기능을 대표할 수 있는 이름;
기능영역:기능이 속한 기능영역;
업무매뉴얼: 기능에 속한 업무매뉴얼의 리스트;
연산: *업무매뉴얼검색*{기능에 속한 업무매뉴얼에 대한 검색방법 };
업무매뉴얼등록{신규 업무매뉴얼의 등록방법 };
.....
업무매뉴얼삭제{기존 업무매뉴얼의 삭제방법 }
}

기능영역, *기능*클래스는 비지니스 모델의 구조적 측면을 설명하기 위한 것으로 하부 구성요소에 대한 분류, 검색, 등록, 삭제 등 관리를 주요 연산으로 제공하는 것을 목적으로 한다. *기능영역*, *기능*, *업무매뉴얼* 클래스의 3계층구조는 (그림 4)와 같이 객체지향의 구성관계로 나타낼 수 있다. 각 기능 영역은 다수의 기능으로 구성되어 있으며 다시 각 기능은 다수의 업무매뉴얼로 구성되는 것이다.

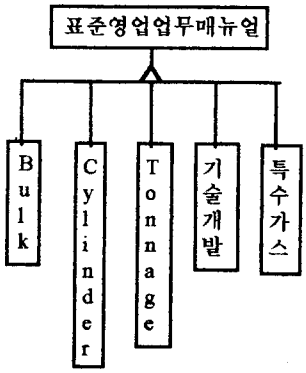


(그림 4) 기능분할의 3계층 구성관계

5. 업무매뉴얼의 특수화

분산조직의 경우 이미 지적한 바와 같이 업무매뉴얼은 사업장에 따라 각각 다른 특성을 지니게 된다. 따라서, 각 업무별로 전사적 표준 업무매뉴얼을 작성하는 것은 현실적으로 무리가 따르게 된다. 따라서, 이러한 분산조직의 경우, 분산 DB을 작성하고 각 사업장별로 특성을 반영하는 업무매뉴얼을 작성하는 것이 바람직할 것이다. 그러나, 업무매뉴얼은 기업의 비지니스 데이타 타입으로서 가능하면 전사적으로 공통된 형태를 갖는 것이 개념적 무결성을 유지할 수 있게 될 것이다. 이러한 목적하에 객체지향의 유전관계 개념을 적용하여 하나의 표준안으로부터 특수화된 업무매뉴얼을 작성하는 방법을 선택할 수 있다.

*사업장업무매뉴얼*은 *업무매뉴얼*클래스를 사업장에 맞게 특수화시킨 업무매뉴얼이다. *사업장업무매뉴얼*은 기본적으로 표준업무매뉴얼(예, 본사 기준)의 속성과 연산을 유전받게 된다. 그러나, 속성의 값은 기본값을 무시하고 독자적인 값을 갖을 수 있으며 연산의 구현인 메소드, 즉 서브 프로세스에 대한 내역 또한 독자적으로 구현할 수 있다. 여기서 중요한 것은 연산의 이름은 반드시 동일하게 사용해야 한다는 것이다. (그림 5)는 D사의 분산 맵에 따른 영업 기능영역에 대한 *표준업무매뉴얼*클래스에 대한 *사업장업무매뉴얼*클래스의 유전관계를 나타내고 있다.



(그림 5) D사의 *사업장업무매뉴얼*클래스 유전관계

6. 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 객체레벨의 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. 그리고 이를 위해 *기능*클래스에 *업무매뉴얼버전등록* 연산을 추가해야 할 것이다.

7. 결론

본 논문에서 제시된 업무매뉴얼 작성에 대한 객체지향 모델링은 개념적 무결성 유지를 목적으로 하고 있다. 기존의 업무매뉴얼이 조직구조와 구성원을 배제하지 않은 비공학적 방법으로 작성되었다면 비지니스 엔지니어링에서의 업무매뉴얼은 업무에서 인간적 측면을 최대한 배제시키고 순수하게 업무만을 설명하려는 공학적 노력이라고 하겠다. 이런 노력을 실현하기 위해서는 개념적으로 완전한 패러다임을 사용하는

것이 중요하다. 객체지향 패러다임에 의한 *업무매뉴얼*클래스의 정의는 바로 공학적인 프로세스 추상을 위한 기본적 체계를 제공하였다고 할 수 있다.

D사는 전사적인 *업무매뉴얼*클래스의 정의와 이를 통한 업무정형화의 실현, 그리고 최종적인 정보시스템 구축을 추진하고 있다. 이제 업무의 공학적 체계구축하에 이를 현업에 접목시키기 위한 2단계 작업에 착수하였다. 이는 실사례, 정보활용, 안전조항, 비상대처, 평가방법 등 업무담당자의 실질적인 업무수행과정과 상황을 첨가하여 *업무매뉴얼*객체를 작성하려는 작업으로 해석할 수 있다.

참고문헌

Andleigh, P., and Gretzinger, M., Distributed Object-oriented Data Systems Design, Prentice-Hall, Englewood Cliffs, NJ, 1992.

Barker, R., and Longman, C., Case*Method- Function and Process Modeling, Addison-Wesley, Reading, MA, 1992.

Berard, E., Essays on Object-oriented Software Engineering, Prentice Hall, Englewood Cliffs, NJ, 1993.

Brooks, F. P., The Mythical Man-Month, Addison-Wesley, Reading, MA, 1982.

Hammer, M. and Champy, J., Reengineering the Corporation, Harper Business, New York, NY, 1993.

Jacobson, I. et al, Object-oriented Software Engineering, Addison-Wesley, Reading, MA, 1992.

Jacobson, I. et al, Business Process Reengineering with Object Technology, Addison-Wesley, Reading, MA, 1995.

Martin, J., Information Engineering Book II, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Mish, F. C., Editor in Chief, Webster's New Collegiate Dictionary, Merriam Webster Inc., Springfield, MA, 1988.

Rumbaugh, J. et al, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Shlaer, S. and Mellor, S. J., Object Lifecycles, Yourdon Press, Englewood Cliffs, NJ, 1992.

Taylor, D., Object-oriented Information Systems- Planning and Implementation, John Wiley & Sons, New York, NY, 1992. .

Wirf-Brock, R., Wilkerson, B., and Wiener, L., Designing Object-Oriented Software, Prentice-Hall, Englewood Cliffs, NJ, 1992