IDEF Methods for Knowledge Engineers and Evolutionary Enterprise

1. IDEF(Integration DEFinition) 방법 개요

새로운 시스템을 계획하고 확립한다는 것은 가장 어려우면서도 성공하기 힘들고, 위험한 일이라는 사실을 염두에 두어야 한다. 창안자는 새로운 시스템의 구축으로 얻게 될 이익에 관해 미온적으로 대응할 따름인 소수의 지지자를 확보할 수 있을 뿐, 낡은 제도의 유지로부터 계속적으로 이익을 얻고자 하는 모든 사람들과 적대관계에 놓일 수 있기 때문이다.

마키아벨리, "군주론", 1513 년

1.1 IDEF 방법의 목표

TQC(Total Quality Control), JIT(Just In Time), TPM(Total Productivity Management), MRP(Material Resource Planning), MIS(Management Information System), EIS(Executive Information System), DSS(Decision Support System), TQM(Total Quality Management), CE(Concurrent Engineering), CIM(Computer Integrated Manufacturing), BR(Business Reengineering), BPR(Business Process Reengineering), AM(agile Manufacturing), ABC(Activity Based Costing), ABM(Activity Based Management), ERP(Enterprise Resource Management), WfM(Workflow Management), KM(Knowledge Management), 6SIGMA, CALS(Computer Aided Logistics Support), EC(Electronic Commerce), BSC(Balanced Scored Card), CRM(Customer Resource Management), RiM(Risk Management) ….. 등 최근의 혁신적 기술이나 방법의 궁극적인 목표는 조직이나 기업의 업무방식을 변화 시키는 것이다. 위에 언급된 각각의 기술이나 방법은 기본적으로 프로세스를 분석, 설계, 리엔지니어링하는 전략을 제공함으로써 이들 프로세스와 연관된 조직, 정보, 데이터의 구조를 변화 시키는데 있으며 그 목표는

① 대응력이 강하고 유연하며 믿을 만한 시스템,
② 개발, 운용, 유지에 비용이 덜 드는 시스템,
③ 품질을 개선시킬 수 있는 시스템,
④ 개발의 리드 타임을 줄일 수 있는 시스템
⑤ 시스템 내부의 기제로 스스로 진화하는 시스템을 엔지니어링하는 것이다.

마키아벨리가 간파했듯이, 조직에 새로운 시스템을 도입할 기본적 책임이 있는 사람은 항상 어려운 상황에 직면하게 되는데 이는 새로운 시스템의 실행이 위험을 수반하기 때문이다. 위에서 언급한 기술이나 방법의 적용은 바람직한 변화를 가져오기 위해 몇 가지 근본적인 패러다임의 전환을 요구한다. 즉 기업은 사업수행 방식에 관한 기본적 가정을 변화시켜야 하는데 그러한 변화의 충격은 정확히 예측하기 어려운 것이며 프로세스의 변화는 즉각 나타나지 않는 상호 의존적 시스템의 근본적 변화를 초래할 수 있다. 따라서 프로세스의 변화는 인사, 데이터, 정보, 정책, 절차 등 관련 시스템의 모든 측면에 대한 영향을 충분히 검토할 것을 요구한다. 새로운 시스템을 구축하려는 사람들은 자신의 조직에 관하여

혹은 고객의 사업에 관하여 다음과 같은 표면적으로는 단순해 보이는 다음과 같은 질문에 답해야 한다.

① 우리가 하는 것은 무엇인가
② 그것을 어떻게 하는가
③ 이들 목표를 달성하기 위해 어떤 정보와 데이터가 필요한가
④ 시스템을 변화 시킬 때 어떤 일이 일어날 수 있는가
⑤ 기업의 정보시스템 구축을 위한 기능분석의 체계적인 표현 방법이 있는가
⑥ 기업 내부 혹은 기업간의 제조시스템이나 서비스시스템의 기능을 지원하기 위해 필요한 정보의 구조 및 역할을 정의할 수 있는 가장 적절한 방법은 무엇인가
⑦ 정보의 구조에 관한 정의를 통하여 논리적, 물리적으로 정보를 데이터베이스화 시킬 수 있는 절차 및 방법은 무엇인가
⑧ 정보시스템 구축을 위한 프로세스를 명확히 포착할 수 있는 방법과 시스템 구현 시 문제 발생의 예측 가능한 기법은 무엇인가

이는 급변하는 경영환경에서 기업의 미래체제를 구상하고 새로운 시스템의 구축을 주도해야 하는 시스템개발 관리자들에게 가장 기본적이면서도 절실한 문제다.
컴퓨터 하드웨어와 특정 소프트웨어 기술의 급속한 발전에도 불구하고 대형 정보시스템 개발이란 과제는 그러한 시스템을 엔지니어링하는 효과적이면서도 이해하기 쉬운 방법을 지속적으로 찾고 있다. 물론 정보시스템의 개발 및 구축이라는 소프트웨어 엔지니어링, 혹은 정보공학적 측면에 있어서는 많은 학자들의 연구 및 기업의 적용을 통하여 발전이 이루어지고 있다. 그러나 정보시스템의 이용을 기본으로 하게 된 1970년대 이후에는 산업시스템 공학(Industrial System Engineering)적인 관심으로 확산되었고 현재는 통합된 기업 전체의 시스템을 표현하는 방법으로서 구축 되어야 할 시스템을 표현하고, 이들 시스템 개발과 관련된 사람들간에 의사소통 도구로서 활용하며, 그 절차를 기술하는 도구로서 방법의 활용이 확대되고 있다.
솔직히 말해 방법의 개발이란 곧 언어의 개발, 즉 커뮤니케이션을 촉진시키는 강력한 도구의 개발에 다름이 아니다. 또한 방법이란 일을 하는 절차이다. 그런데 대부분의 방법 특히 IDEF에서 방법은 기본적으로 모델 혹은 묘사(설명)라는 다이어그램과 문장으로 구성된 복수의 구문적 방법으로 표현된다. 이는 방법이 사용에 있어서 명확한 의사소통과 일관된 직관적 기초를 제공하여야 한다는 필요와 경험에 의하여 발전된 것이다.

방법을 이용한 모델화 작업의 산출물로서 나타나는 '모델' 혹은 '묘사'의 예는 우리의 일상

에서 쉽게 찾아 볼 수 있다. 예를 들어 지도를 생각해 보자. 지도는 현실의 지형 상태를 2 차원 평면에 나타낸 것이므로 현실의 지형 그대로는 아니지만 현실적으로 자신이 위치하고 있는 지점을 주위와의 관련으로 파악하기 위해서는 매우 유효한 정보를 제공해 준다. 즉, 실제의 지형이라는 존재에 대해 마치 실제로 보고 있는 것과 같은 정보를 인간에게 제공해 주는 것이다. 그러한 특정 목적을 가진 정보를 전달하기 위한 표현 형식을 인공지능의 분야에서는 '모델'이라고 부르고 있다.

그런데 우리는 종종 방법론(Methodology)이나 방법(Method), 모델링(Modeling), 모델(Model) 등과 관련된 용어의 사용에서 혼동을 느낄 수 있는데 방법론은 방법 - 예를 들면 IDEF방법과 같이 - 을 개발하기 위한 이론적 토대나 학문으로서의 분야를 말하며 방법은 이와 같은 이론을 바탕으로 수행된 결과로서 산출된 구문이나 표현형식을 말한다. 또한 이러한 방법들은 대개 정해진 표현형식에 따라 현실세계의 상황을 명료하게 개념적으로 표현하기 위하여 단순화 및 추상화 시킨 형태의 모델을 작성하게 되는데 이러한 작업의 과정을 우리는 모델링 혹은 모델화라고 정의하고 있다.

위의 예에서 우리는 모델에 해당하는 지도를 만드는 작업을 모델링이라 하고 이러한 지도 작성 작업에 사용된 척도, 방위표시, 등고선 등의 표현형식을 방법이라 할 수 있다. 또한 위의 예에서는 이러한 현실세계의 지형지물을 효과적으로 표현하는 방법을 연구하는 학문을 방법론이라고 한다.

일반적으로 말해, 모델과 묘사의 목적은 의사결정을 돕는 것이다. 건축물에 있어서 청사진이 중요하게 사용되는 것과 마찬가지로 대규모 정보시스템의 개발은 시스템 개발의 선행과제로서 현재 구축되어 운용중인 시스템(AS-IS)에 대한 모델과 새롭게 구축되어야 할 시스템의(TO-BE) 모델을 필요로 하는데 이는 관련 요원들 간의 의사소통을 촉진하고 시스템의 분석, 디자인, 개발 등 시스템 라이프사이클 전체를 효과적으로 지원하기 위하여 필수적인 것이다.

그런데 기업의 새로운 정보시스템 구축이라는 명제는 정적인 상태의 지형이나 건축물과 달리 상호 유기적이고도 복합적으로 조직되어 운영되는 시스템으로서 정보시스템을 구성하는 각 단위기능의 분석 및 설계, 이를 지원하기 위한 정보의 구조 및 역할, 데이터베이스화 시킬 수 있는 절차 및 방법, 관련 프로세스의 포착 및 표현 등과 같이 우리에게 정보시스템 구축에 필요한 다양한 모델링 방법의 개발과 이를 이용하여 작성된 여러 차원의 기업 모델의 제시를 요구하고 있는 것이다.

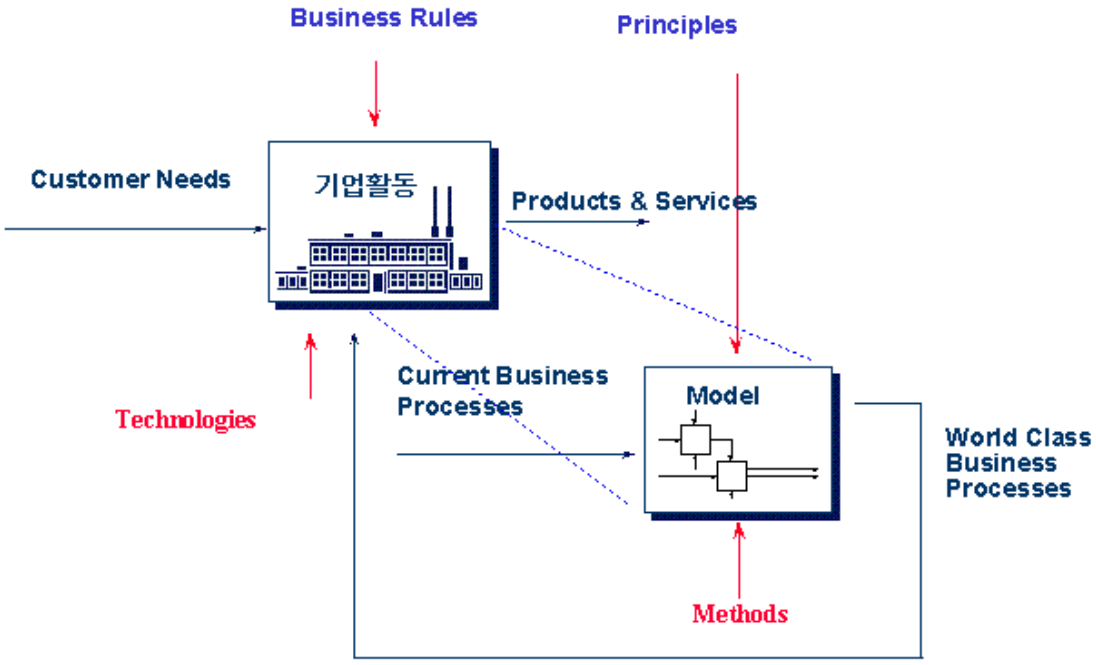


그림 1-1 : 기업활동과 모델의 관계

IDEF 방법은 기업이나 조직의 실체를 추상화하여 모델화하고(AS-IS), 작성된 모델의 체계적인 분석을 통하여 문제점을 추출하여 개선된 기업의 모델(TO-BE)을 설계할 수 있도록 개발된 시스템 분석, 설계 방법이다. 또한 IDEF는 시스템의 개발과 관련된 사람들간의 의사소통을 촉진하기 위한 언어로 개발되었으며 현재는 다음과 같은 목적으로 사용되고 있다.

① 시스템 분석, 설계, 교육, 문서화, 통합
② 합의를 도출하기 위한 의사소통 수단 지원
③ 기업의 정보 시스템 구축을 위한 업무 활동의 분석과 문제점 포착
④ 기업의 활동에 관한 업무 흐름의 명확한 표현

1.2 IDEF 방법의 구성

이러한 필요에 부응하기 위해 1976년에 미 국방부에서는 발전된 정보시스템의 구축을 전제로한 항공,우주 관련 가상의 기업모델(Virtual Enterprise Model)을 표현하기 위한 방법의 연구가 시작되었고 이 연구의 결과로 개발된 것이 IDEF(Integration DEFinition) 방법이다. IDEF는 1980년대 미 공군의 ICAM (Integrated Computer Aided Manufacturing) 프로그램에서 IDEF⌀ (Function/Activity Modeling Method), IDEF1/1X(Information and Data Modeling Method)가 개발 발표되었으며 1990년대 미 공군의 지원을 받아 Knowledge Based Systems사(KBSI)가 수행한 IICE (Information Integration for Concurrent Engineering) 프로그램에서 IDEF3 (Process Description Capture Method), IDEF4 (Object-Oriented Design Method), IDEF5 (Ontology Description Capture Method)가 개발 발표되었다.

IDEF 방법은 아래에 나타난 바와 같이 실로 여러 가지 방법으로 구성되어 있는데 이는 복잡하고 유기적으로 작용하는 시스템의 각 부문을 효율적으로 표현하고자 구성되어진 것이다.

①	IDEF⌀	Function Modeling Method
②	IDEF1	Information Modeling Method
③	IDEF1X	Data Modeling Method
④	IDEF3	Process Description Capture Method
⑤	IDEF4	Object-Orient Design Method
⑥	IDEF5	Ontology Description Capture Method
⑦	IDEF9	Business Constraint Discovery Capture Method

즉 기업의 활동을 여러 관점에서 모델화 할 수 있도록 촉진하고 사용 목적에 따른 정밀하고도 체계적인 모델의 획득에 최적화된 방법을 제공하고자 개발된 것이다. 이는 우리가 못을 박기 위해 망치를, 땅을 파기 위해 삽을 각각 개발하여 사용하듯이 하나의 방법으로 기업의 여러 활동을 모두 표현한다는 것이 부적합 하다는 것을 반영하는 IDEF 방법 개발자들의 철학이기도 하다. 삽 그 자체는 땅을 파는 것이 아니라 땅을 파는 사람을 돕는 도구인 것처럼, 방법은 과업을 더 효과적으로 달성하기 위한 도구를 인간의 정신에 제공한다. 즉 방법은 인간의 지적활동을 지원하고 촉진한다. 하지만 방법이 의사결정을 내리거나 통찰력을 창조하는 것은 아니다. 또 문제를 해결하지도 못한다. 현실세계의 상황을 통찰하고 의사결정을 내리는 이들 활동은 인간에 의해서만 가능한 것이지만 방법의 활용에 의해 발전되고 촉진되는 것이다.

방법의 중요성은 제조산업에서 오랫동안 인정 받아왔다. 이 분야에는 생산의 5M (Manpower, Methods, Materials, Machines, Money)이라는 말속에 방법이 들어있다. 물자, 기계, 화폐는 교체될 수 있다. 하지만 사람, 그리고 사람의 지식을 이용하는 방법은 사업의 성공을 결정하는 결정적인 요소이다. 실로 IDEF 방법은 20여년에 걸친 기간에 있어서 기존의 방법 개발자를 포함한 많은 연구인원과 기술의 실질적인 활용에 있어서 폭 넓은 경험을 갖춘 성원들에 의하여 개발, 발전되어왔으며 대규모의 발전적 통합정보시스템에 필요한 방법을 개발하고 시스템 개발의 "엔지니어링 원칙"을 제시하고자 하는 학문적 노력의 결과인 것이다. 본 자료에서는 현재까지 개발된 IDEF 방법을 중심으로 기술한다.

2. IDEFø 기능 모델링 방법(Function Modeling Method)

2.1. IDEFø 개요

2.1.1 기능(활동) 모델이란 ?

① 현재의 시스템이나 계획된 시스템 안에 존재하는 활동 혹은 활동간의 관계를 표현한다.
② 어떠한 상황, 관점, 목적에 따라 다이아그램, 용어해설 그리고 문장으로 이루어진 집합.

IDEFø 기능 모델링 방법은 조직이나 시스템의 의사결정, 행동, 활동을 모델링 할 수 있도록 디자인 된 방법이다. IDEFø 는 SADT 로(Structured Analysis and Design Technique) 잘 알려진 그래픽 언어에서 파생되었으며 1976 년에 개발이 시작되어 1981 년 6 월 ICAM(Integrated Computer Aided Manufacturing) Function Modeling Manual 로 미 공군에 의해 발표되었고 1986 년 미 국방부에 의해 Activity 모델링을 위한 표준 방법(Department of Defense 8020.1-M)으로 채택되었으며 1993 년 12 월 NIST(National Institute of Standards and Technology)는 미연방 정보처리 표준(Federal Information Processing Standard Publication 183-Integrated Definition for Function Modeling)으로 IDEFø 을 채택하였다. IDEFø 는 셀(Cell) 모델링 그래픽 표현방법을 바탕으로 시스템을 기능적 관점에서 분석, 커뮤니케이션할 수 있는 기능/활동 모델링 방법이다.

IDEFø 방법은 하드웨어, 소프트웨어, 사람 등이 모두 연관되어 이루어진 어떠한 시스템도 모델화 할 수 있는 방법으로 사용되어지는데, IDEFø 는 시스템 개발에 앞서 우선적으로 요구사항 및 기능을 정의하고 이러한 요구사항과 기능을 수행하는데 적절한 구현설계를 위해서 주로 사용된다. 사실 IDEFø 기능모델은 매우 간단한 방법으로 현실세계의 분석 및 시스템 설계를 위한 활동 및 활동간의 관계를 표현할 수 있다. 이러한 방법의 전반적인 이해를 위하여 우리는 자동차 조립라인의 엔진을 조립하는 공정을 상상하여보자. '자동차 엔진을 조립한다'는 활동을 그림 2-1 에 표현된 것과 같이 하나의 박스로 표현하고 '자동차 엔진을 조립한다'는 활동의 수행을 위해 소요되는 입력으로서의 조립부품을 활동으로 들어오는 박스 왼쪽의 화살표로, 활동의 결과로 산출되는 출력인 조립결과 보고서와 조립된 엔진을 활동박스의 오른쪽으로 나가는 화살표로 표현해 보자. 또한 이러한 활동의 수행과 관련된 제약 및 지침으로서 조립일정 계획 및 회사지침을 활동박스 위의 화살표로, 무엇이 활동을 수행하는가를 나타내는 메커니즘으로서 인원, 장비, 전산시스템을 활동박스 아래의 화살표로 표현해 보자. 이러한 활동의 도형적인 표현방법을 통해 우리는 활동의 수행과 관련된 입력과 출력은 무엇이고 어떤 제약조건과 무엇에 의하여 활동이 수행되는지를 직관적으로

파악할 수 있다. 셀 모델링 그래픽 표현방법이란 활동을 이러한 박스와 화살표로 이루어진 하나의 조합, 즉 셀로 생각하는 방법이다. 우리는 이러한 셀이 이를 구성하는 몇 개의 하위 셀로서 이루어져 있다고 가정할 수 있으며 아래의 예는 '자동차 엔진을 조립한다'는 셀의 분해를 통해 아래에 표현된 네 개의 활동 및 이들 활동간의 관계(입력, 출력, 제어, 메커니즘)를 다이아그램으로 나타내고 있다.

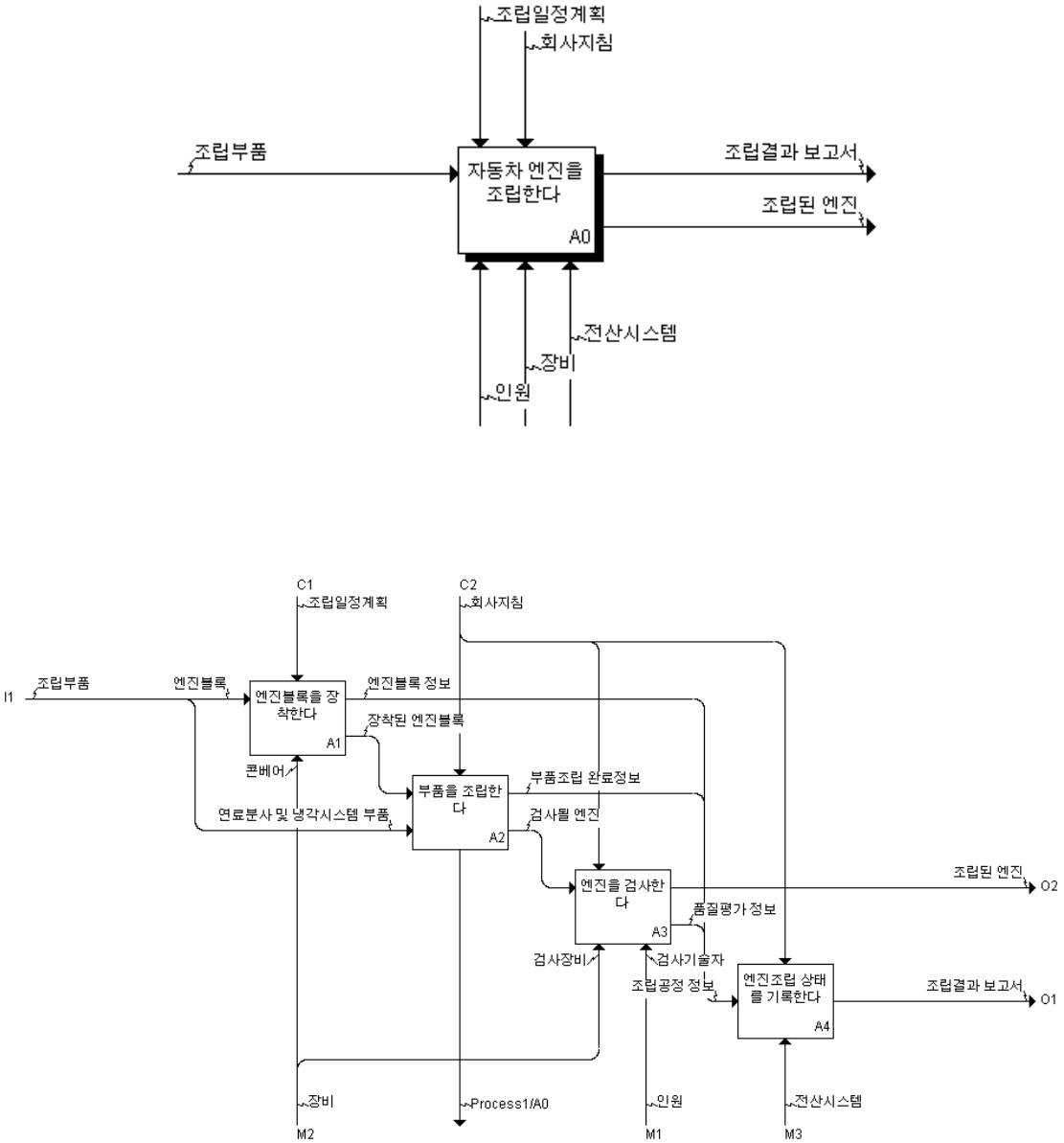


그림 2-1 : '자동차 엔진을 조립한다'는 셀에 대한 분해 다이아그램

■ IDEF⌀는 무엇인가?

① 하나의 기능(활동)모델링 방법.
② 계층적인 분해를 통한 다양한 레벨의 기능에 관한 설명을 지원.
③ 각 활동 및 이들간의 관계에 관한 모델을 구축하기위한 프로세스 및 언어를 제공.

효과적인 IDEF⌀ 모델은 시스템을 분석 조직화하고 시스템분석가와 고객사이의 효과적 커뮤니케이션을 촉진하는데 도움을 준다. 나아가 IDEF⌀ 모델링 방법은 특히 기능적 관점에 관해서 확실한 분석방법을 지원하는데 커뮤니케이션 도구로서의 IDEF⌀는 단순화된 그래픽 방법을 통하여 중요한 전문가의 참여를 증진시키고 일치된 의사결정을 향상시킨다. 분석도구로서의 IDEF⌀는 기업에 있어서 수행되는 각 기능 및 그 기능이 수행되는데 필요한 자원을 정의하고 현재 그 시스템이 정확하게 작동하는지 여부를 확인함에 있어서 모델 작업자를 지원한다. 따라서 IDEF⌀ 기능모델은 종종 현 시스템(AS-IS)의 기능분석을 위하여 시스템 개발에 앞서 우선적으로 작성되기도 한다.

■ IDEF⌀ 기능 모델은 다음과 같은 필요에 의해서 개발되어지고 사용된다.

① 기업이 현재 수행하는 주요 활동을 분석,정의하고 문서화하며 이를 통해 의사소통.
② 기업이 수행하는 활동이 상호 어떻게 연관되어지는지에 대한 이해를 촉진하며 이를 통해 관련된 활동에 새롭게 투입되는 요원들의 업무습득 기간을 단축한다.
③ 부가가치를 생산하는 활동과 그렇지 않은 활동을 구분하며 현재활동에 소요되는 시간 및 자원을 분석, 추출한다.
④ 우선적으로 개선, 발전시켜야 할 기능을 추출하고 새롭게 개선된 기업모델을 디자인하며 평가한다.
⑤ AS-IS활동의 포착 및 분석을 지원하고 TO-BE 시나리오를 위한 활동의 설계를 지원한다.

IDEF⌀ 는 기업이나 조직의 활동을 추상적인 단위의 기능으로 표현 가능토록 지원하며 각 기능간에 연관된 정보 및 자원을 기능모델 안에 표현 할 수 있도록 절차와 언어를 동시에 지원한다. IDEF⌀ 는 조직의 기능에 관련된 사항을 사용자에게 '말해 주도록' 디자인 되었는데 분석도구 혹은 커뮤니케이션 도구로서 여러 응용영역에서 성공적으로 사용되고 있다. IDEF⌀ 모델은 '조직'과 '기능'을 분리시킨다. 즉 조직 내 업무활동의 공통적이고 기능적인 연관관계를 정의 함으로서 조직에 대하여 독립적인 분석을 촉진 시킬 수 있도록 한다. 이 같은 분석방법의 접근은 업무중심의 분석을 시도 함으로서 업무개선이 이루어질 수 있도록 지원한다. 이러한 특징은 IDEF⌀ 가 시스템 구축에 있어 '요구수립' 활동을 수행하는 메커니즘으로 사용될 수 있음을 의미한다.

■ IDEF⌀는 다음과 같은 것들을 표현하고 있다.

① 기능(function, activity) – 특정분야의 의사결정, 행동, 활동
② 개체(objects) – 특정분야의 추상적 혹은 물리적인 실체
③ 기능과 기능간의 관계
④ 기능과 개체의 관계
⑤ 기능의 수행에 있어서 개체의 역할

IDEF⌀의 커뮤니케이션 촉진능력은 CIM(Computer Integrated Manufacturing)이나 CE(Concurrent Engineering)등과 같이 상호 협동적 팀 프로젝트에 대하여 효과적인 분석도구로서 활용된다. 단 IDEF⌀ 모델은 조직이나 기업이 “무엇을” 수행하는지에 관해서 표현할 뿐 그것을 어떻게, 어떤 순서로 하는지 하는 비법이나 프로세스에 관한 설명에는 도움을 주지 않는다. 기능과 연관된 특정 시기나 논리에 대하여 구체적인 설명을 얻기 위해서는 4장의 IDEF3 프로세스 모델링 방법(Process Modeling Method, Process Description Capture Method)이 필요하다.

2.1.2 기능모델과 DFD의 비교

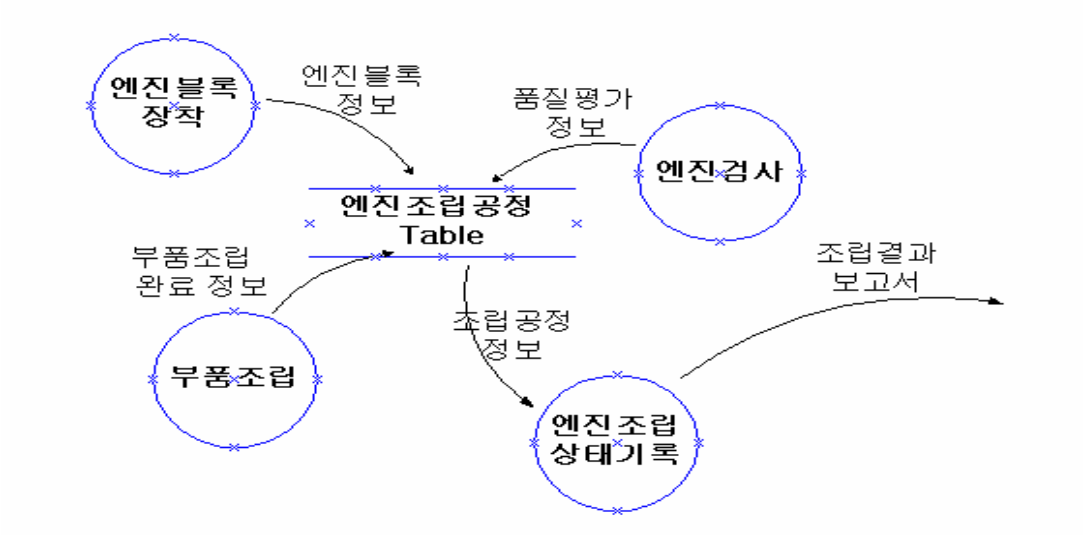


그림 2-2 : 설계담당자의 관점에서 표현된 정보중심의 업무분석 다이아그램

이는 우리가 이미 잘 알고 있는 DFD(Data Flow Diagram)와 의 비교를 통해서 기능모델의 활용성을 확인 할 수 있다. 우리는 지난 절에서 논의한 자동차 엔진 조립공정의 정보시스템 개발을 위한 DFD를 그림 2-2와 같이 표현할 수 있을 것이다. 그림 2-2에서는 세 개의 활동(엔진블록 장착, 부품조립, 엔진검사)에서 생성된 정보가 엔진조립공정 테이블에 등록되고 엔진조립상태발행 활동에서는 이를 읽어 조립결과 보고서를 발행하는 정보의 흐름을 보여주고 있다. 기능모델과 DFD의 비교를 위해 우리는 다음과 같은 과정으로 기능모델을 작성할 수 있을 것이다.

① 각 활동과 활동간의 정보 흐름을 표현한다.(그림 2-3)
② 각 활동과 관련된 현실세계의 실물의 흐름을 표현한다.(그림 2-4)
③ 각 활동의 수행을 제어(시작, 종료, 트리거)하는 사항을 추가한다.(그림 2-5)
④ 이들 활동의 수행이 무엇에 의해 수행되는가를 명시한다.(그림 2-6)

완료된 활동모델은 현실세계의 활동 및 입출력 관계 뿐 아니라 이들 활동에서 발생되는 정보, 이를 통제하는 제약 및 수행하는 메커니즘을 직관적으로 파악할 수 있도록 지원함으로써 그림 2-2의 DFD와 같이 정보시스템 설계담당자의 관점에서 표현되는 정보흐름 위주의 업무분석 다이아그램의 영역을 현실세계의 관점으로 확장하고 있다.

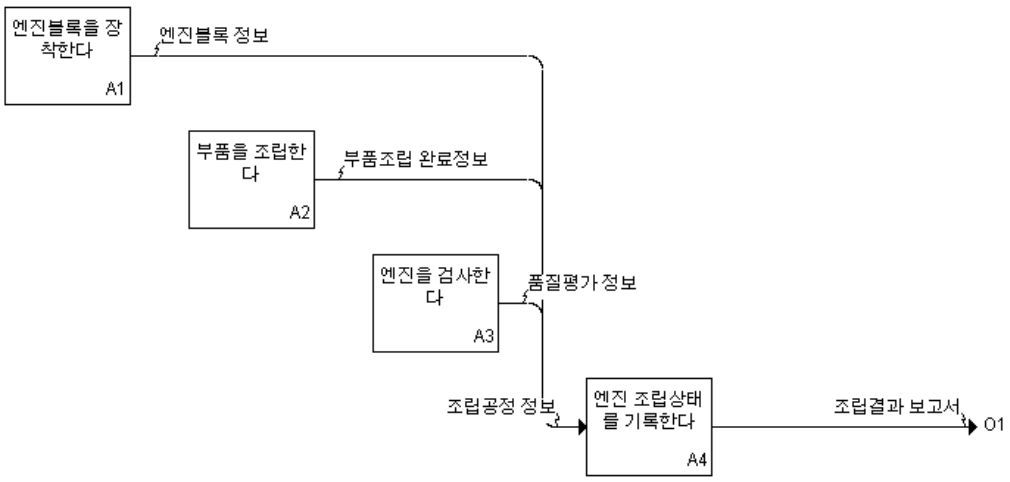


그림 2-3 : DFD와 같은 정보흐름 관점의 입출력 관계 표현

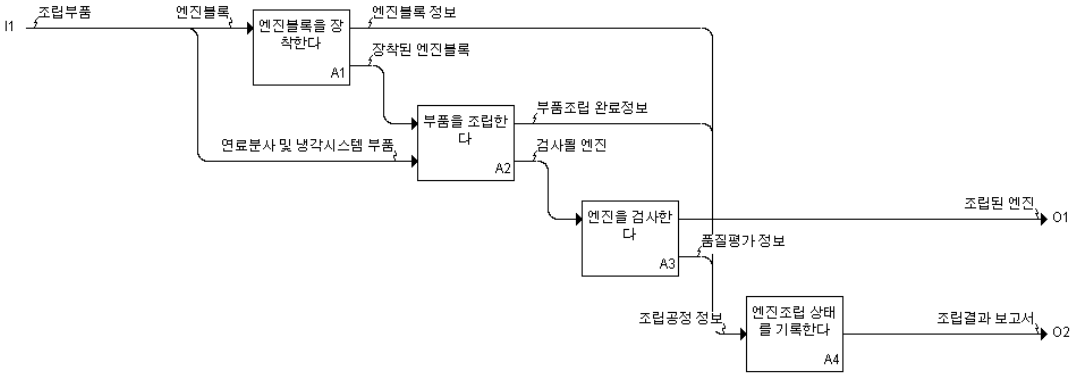


그림 2-4 : 현실세계에어 움직이는 실물의 입출력을 추가로 표현

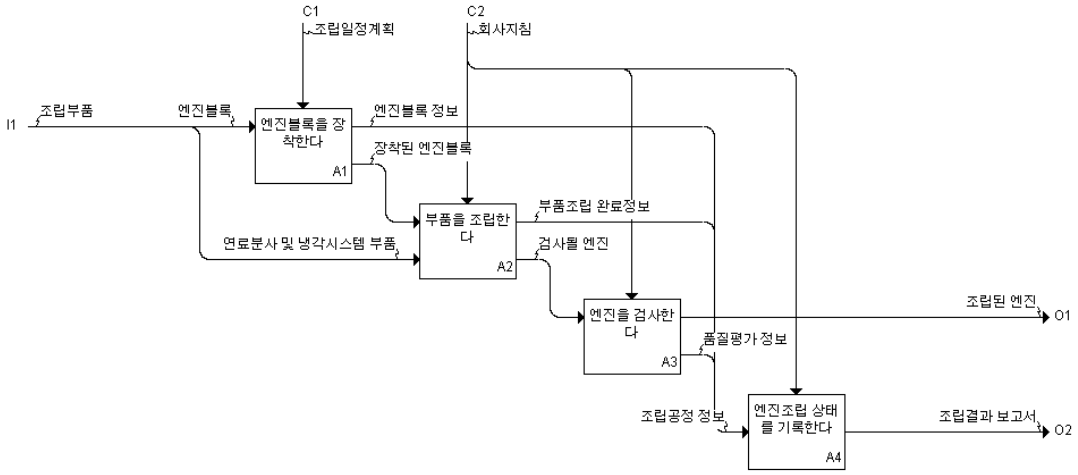


그림 2-5 : 각 활동을 제약하는 제약사항을 추가

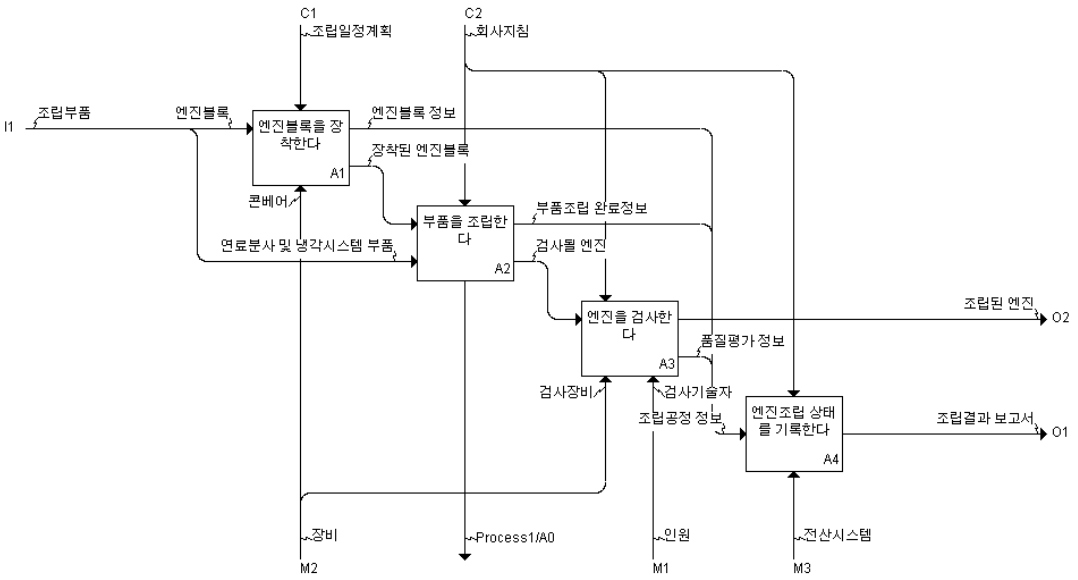
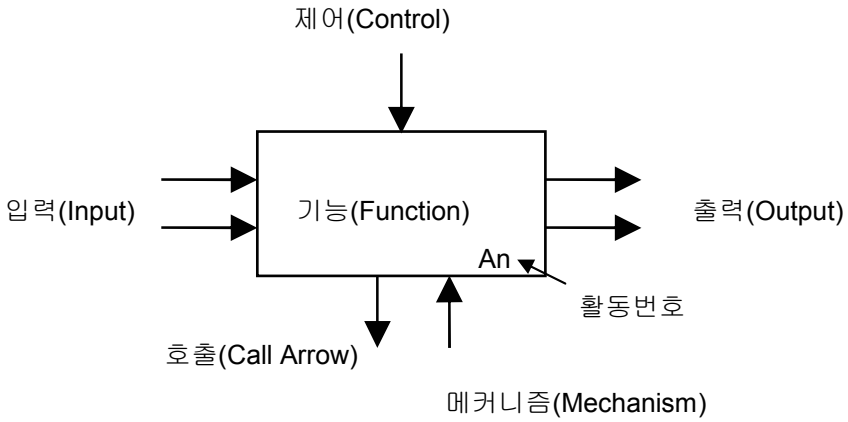


그림 2-6 : 무엇이 활동을 수행하는지를 추가

2.2. IDEFø 모델링 표현 방법

2.2.1. 활동박스 다이아그램 문법

IDEFø 기능모델을 한마디로 표현하면 기업에서 수행되어지고 있거나 수행되어야 할 기능, 또는 기능과 기능간의 관계를 그림과 문자로서 표현해 놓은 것을 말한다. IDEFø는 이러한 모델을 박스와 화살표로서 표현하며 각각의 박스를 계층적으로 분해 가능한 셀(Cell)로 가정하며 기본적 구성은 아래 그림과 같다.



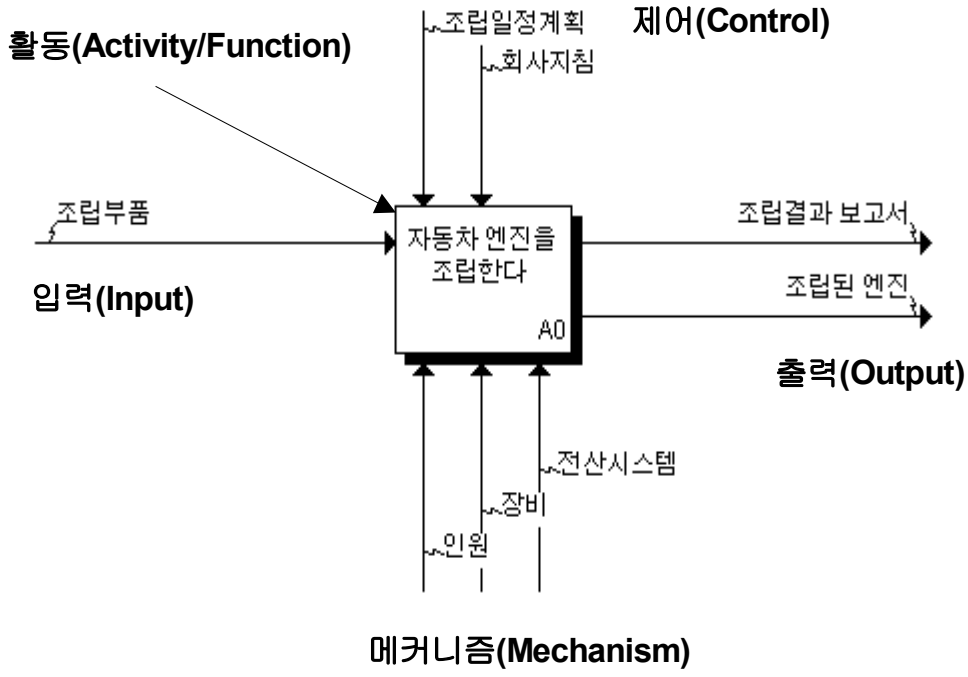


그림 2-7 : 기능박스와 ICOM

그림 2-7과 같이 IDEFø 모델 다이어그램의 기본구성은 박스형태로 표시되는 기능과 화살표로 표시되는 ICOM으로(Inputs, Outputs, Controls, Mechanisms을 말하며 이들을 Concepts라고도 함.) 구성되어 있다. 기능은 모델 작업자에 의해 관찰된 활동,작업,행동,기능을 표현하며 ICOM은 기능의 수행에 관여된 개념(개체)들을 표시하는데 그 구분은 다음과 같다.

■ 기능(활동)

기능은 박스형태로 표현되며 활동, 행동, 프로세스, 혹은 운영을 표현하는데 어떠한 상황에서 무엇이 일어나는지에 관한 서술로 사람이나, 기계, 컴퓨터 등에 의해 수행되는 기능을 표현한다.

① 기능의 이름을 나타내는 레이블은 동사나 동사구로 표현된다. (예를 들면 - 자동차를 설계한다)
② 활동박스의 오른쪽 하단에는 활동의 고유번호를 명시한다.(1 에서 6 까지)

■ ICOM(개념)

ICOM 은 기능이 수행되는데 필요한 개념들, 혹은 기능간에 관여된 개념들로서 기능과 연결된 화살표로 나타내며 명사나 명사구의 형태로 표현되는데 다음과 같은 네 가지로 구분된다.

① 입력(Inputs) : 기능박스의 왼쪽으로부터 들어가는 화살표로서 기능을 수행하는데 필요한 개체 혹은 데이터로 기능이 수행됨에 의하여 소모되거나 변형되는 것을 표현한다. (예를 들면 - 재료, 반제품, 기초데이터)
② 출력(Outputs) : 기능박스의 오른쪽으로 나오는 화살표로 표현되며 기능이나 활동의 결과로 산출되는 산출물을 말한다. (예를 들면 - 반제품, 제품, 가공된 자료)
③ 제어(Controls) : 기능박스로 위쪽에서부터 들어가는 형태로 표현되며 기능을 통제, 제어하는 제약조건, 가이드 혹은 출력을 결정하는데 필요한 제약조건 등으로 기능의 수행을 통제하거나 시작하게 한다. (예를 들면 - 작업 표준, 회사 규정, 품질표준)
④ 메커니즘(Mechanisms) : 기능박스로 밑에서부터 들어가는 형태로 표현되며 기능을 수행하는 사람 또는 개체로서 무엇에 의해 그 기능이 수행되는가 혹은 기능의 수행에 어떠한 자원이 소용되는가를 나타낸다. (예를 들면 - 사람, 기계, Robot, 정보시스템)
⑤ 호출(Call Arrow) : 기능박스의 밑으로 나오는 형태로 표현되며 활동의 수행과 관련된 보다 상세한 기술을 위하여 관련된 다이아그램을 호출하기 위하여 표현한다. Call arrow 는 동일한 리포지토리에 있는 모델에서 일어나는 다른 활동을 참조하게 한다.

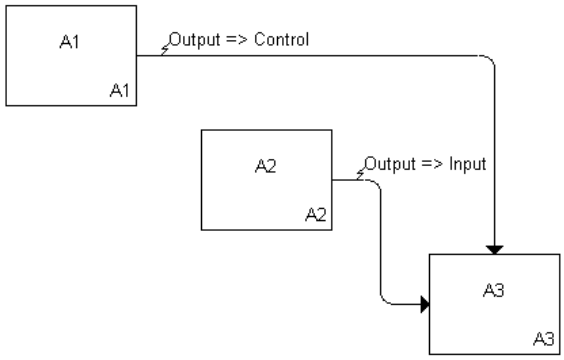
참고로 입력은 기능에 의하여 출력으로 변환되는 개념을 제어는 기능에 의하여 변환되지 않는 개념을 말하며 기능모델에 있어서 하나의 기능에는 최소한 하나 이상의 제어와 하나 이상의 출력이 있어야 한다.

■ 화살표의 표현

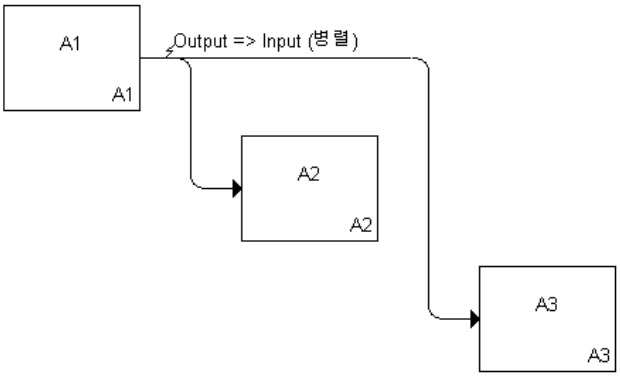
① 화살표는 수평 혹은 수직선으로 그려진다.
② 구부러진 부분은 둥글게 표현한다.
③ 각각의 화살표는 그 내용을 표현하는 이름을 가져야 하며, 이름은 전체 모델에서 유일한 것이어야 한다.

■ 연결관계의 유형

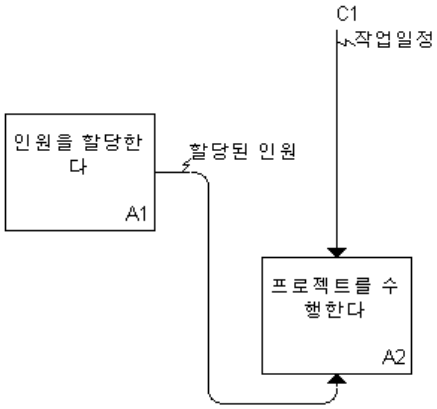
① Constraint & Input : 선행활동의 출력이 후행활동의 제어(Control) 혹은 입력(Input)으로 작용한다. 그림에서 A3 활동은 A2 의 출력이 입력으로 들어오거나 그리고/혹은 A1 의 출력에 의해 제어되어서 수행된다. A3 의 수행을 위하여 A1 과 A2 활동이 선행되어야 한다.



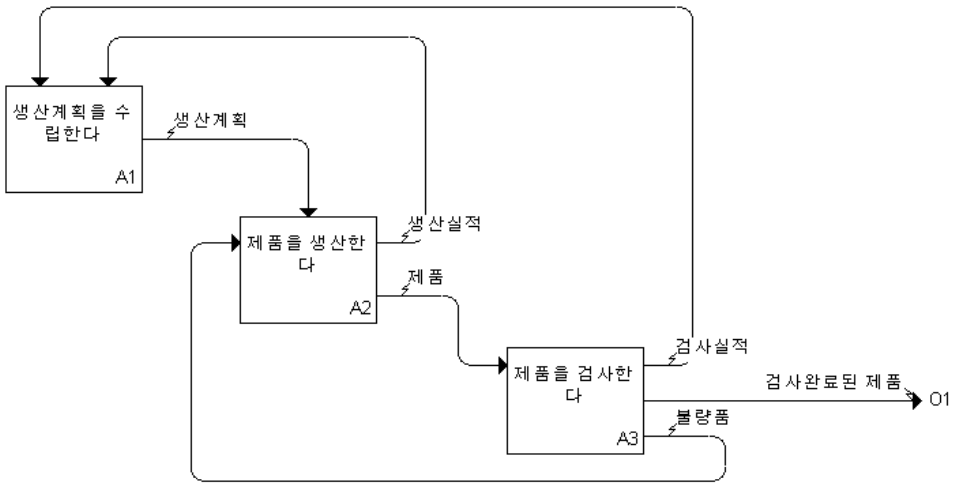
② Simultaneous : 이전 활동의 출력이 다음 활동들에 동시에 입력으로 작용한다. 주어진 조건이 만족되면서 A1 활동의 출력이 입력으로 제공되면 A2 와 A3 활동은 동시에 혹은 시간의 선후에 관계없이 수행된다. A2 와 A3 의 수행은 상호 독립적이다.



③ Mechanism : A1 활동의 출력이 A2 활동의 메커니즘으로 작동한다. A2 의 수행을 위해 A1 의 수행은 필수적이다.



④ Feedback : IDEF⌀ 기능모델은 계속적인 프로세스의 진행에 있어서 활동간의 선후관계나 시간관계의 표현을 배제하고 있다, 아래 그림은 피이드백 관계에 있어서 이러한 선후관계의 표현이 무의미 함을 보여주고 있다.



IDEFØ 는 연구된 시스템이 무엇을 하는가를 문서화 한다. 따라서 어디에서 불합리한 비용이나 노력이 소모되고 있는지를 밝혀준다. 그런데 IDEFØ 는 기능적인 측면에만 관심을 집중할 뿐 조직적인 면이나 절차적인 측면 그리고 원인 결과적인 측면의 분석은 추구하지 않는다. 앞서 논의 했듯이 IDEF⌀ 는 활동의 시간 순서적 제약을 명확하게 포착하지 않는다. 사실 상 IDEF⌀ 모델의 다이어그램에서 화살표의 선후 관계는 이러한 것을 표현하는 것으

로 생각될 수도 있지만 이와 같은 일시적 시간의 관계는 IDEF⌀ 모델에서 의도적으로 포함하지 않는다. 이는 모델의 목적에 합당한 선명성과 보편성을 확보하기 위함인데 IDEF⌀ 가 특별한 시간 순서적 프로세스의 범위 내에서 운용되는 일련의 특정 활동을 설명할 수 있지만, 분석의 목적을 위하여, 그보다는 기능간 모든 경로에 해당되는 일반화된 모델을 제공하는 것이 훨씬 유용하기 때문이다. 이들 경로는 어떤 수 만큼의 프로세스에 대한 시간 순서적 일련의 활동과 일치할 수도 있으나 일치하는 정도에 따라 부분적으로 판단되어야 하며, 특정 프로세스를 이 모델에 맞춰 실행할 의도를 가지고 구축되어서는 안 된다.

2.2.2. 기능적 분해(Functional Decomposition)

IDEF⌀ 기능 모델링에 있어서 기본이 되는 또 하나의 표현 방법은 '활동'의 수직적 해체라는 개념이다. 이는 셀(Cell) 모델링 표현방법이라고도 하는데 기업의 활동에 있어서 분석해야 될 하나의 추상화 된 기능을 규정하고 이를 계속적으로 분해하여 이를 구성하는 하위 단계의 기능간의 관계를 표현 함으로서 상위단계의 단위 셀을 구성하는 하위단계의 기능 및 기능간의 관계를 규정하는 것으로서 외부로부터 내부로의 접근방법(Outside-In Approach)을 지원하는 것이다.

■ 기능적 분해란?

① 하나의 기능을 이를 구성하는 하위 기능들로 나눔으로써 기능을 상세화 한다.
② 기능이 무엇을 수행하는지에 대한 이해와 의사소통에 요구되는 자세한 사항에 대하여 단계적, 시스템적인 설명을 지원한다.

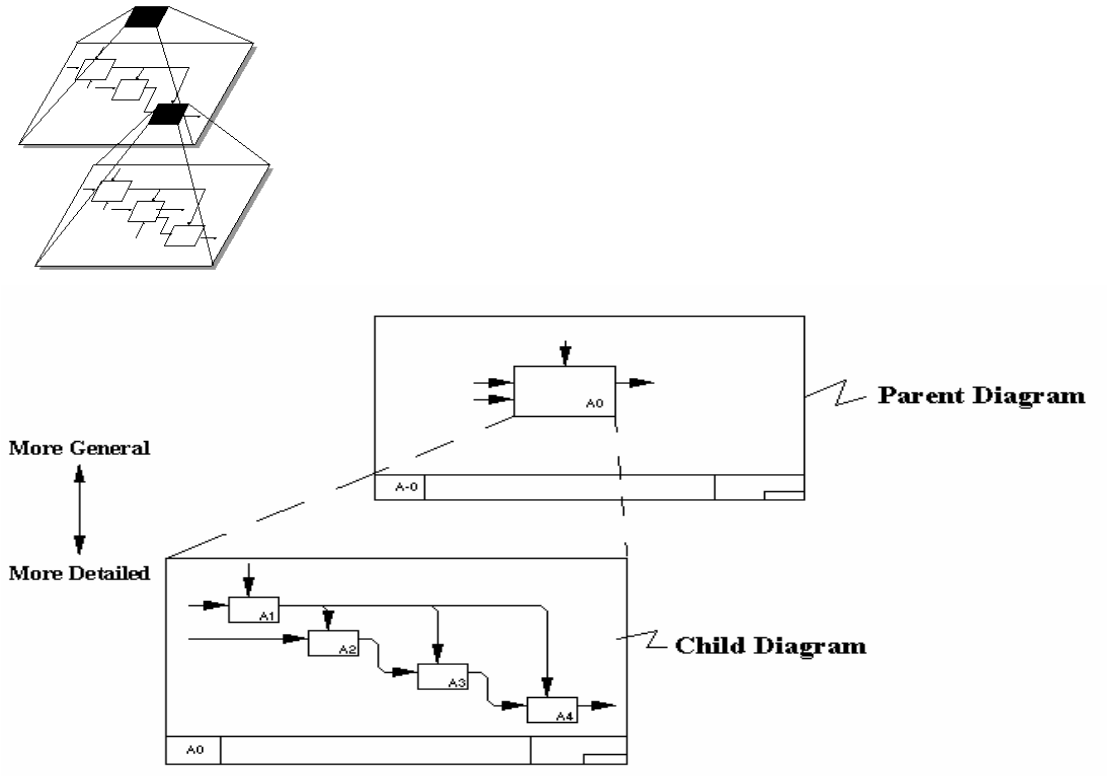


그림 2-8 : IDEF⌀ 모델의 계층적 구조(다이어그램)

즉 IDEF⌀ 최상위 레벨의 '박스'는 어떤 기능과 주변과의 경계를 자연스럽게 구분하며 그 '박스' 안에서 기능은 더욱 작은 기능으로 세분화 되어 구성될 수 있는 관계를 형성한다. 이러한 계층적 구조는 분석가의 모델활동에 있어서 하나의 모델에 대한 명백한 경계(Boundary) 설정을 용이케 함은 물론

① 모델의 목적(Purpose), 관점(Viewpoint), 범위(Context)의 관계에 대한 일관성을 제공하고
② 모델 개발에 대한 계층적, 수직적 분석 방식을 지원하며
③ 추상화 단계에 대한 개념을 제공한다.

IDEF⌀ 방법 내에서 지원되는 이러한 전략은 커뮤니케이션에 있어서 엄청난 표현 능력과 대단한 수월성을 가져온다. 이러한 수직적 해체의 개념은 실제 IDEF⌀ 모델 다이어그램의 표현에 있어서 그림 2-8 과 같은 형태로 표현된다.
모델 개발자는 IDEF⌀ 기능 모델을 구축함에 있어서 우선적으로 분석하고자 하는 대상 활동을 추상화 한 하나의 최상위 기능 박스를 설정하게 되는데 이 박스의 테두리는 개발자가 모델에서 표현하고자 하는 영역과 외부 영역과의 경계를 설정하는 것이며, 기능박스 외부와의 연관 관계를 ICOM 으로 표현 하는데 이렇게 설정된 모델의 최상위 다이어그램을 '배경(Context) 다이어그램'이라 하며 전체모델의 경계를 설정하는 다이어그램이다. 배경 다이어그램에는 전체모델을 대표할 수 있는 추상화 된 하나의 기능박스 만을 표현해야 되며 이를 다시 분해하여 표현한 하위단계의 다이어그램에서는 상위단계 기능의 경계조건 내부에서 상호 관련된 몇 개의 기능 박스로 분해 될 수 있다.

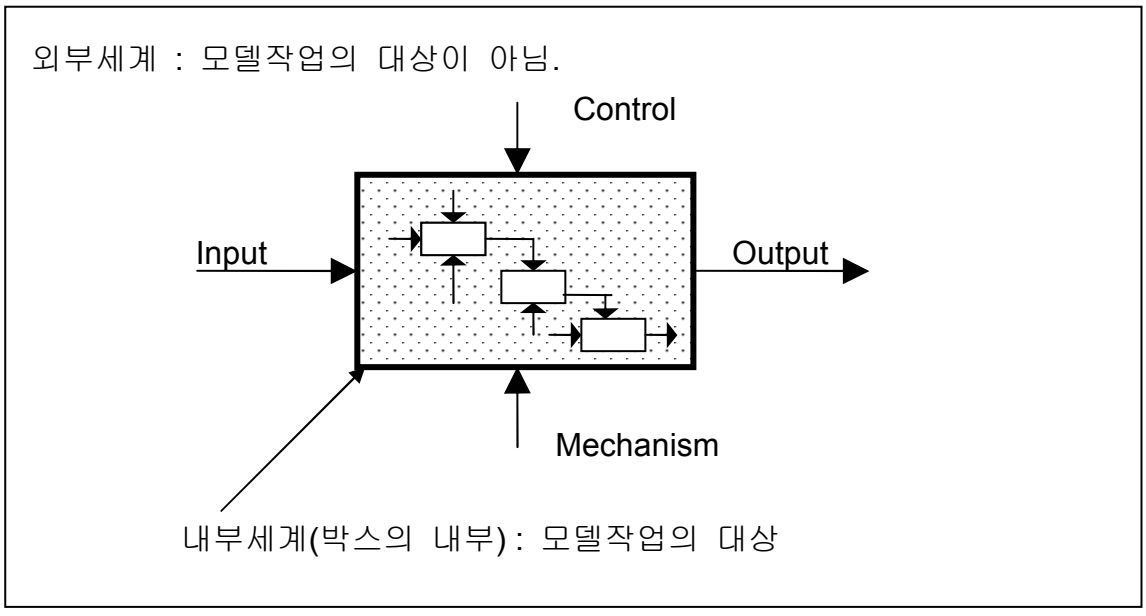


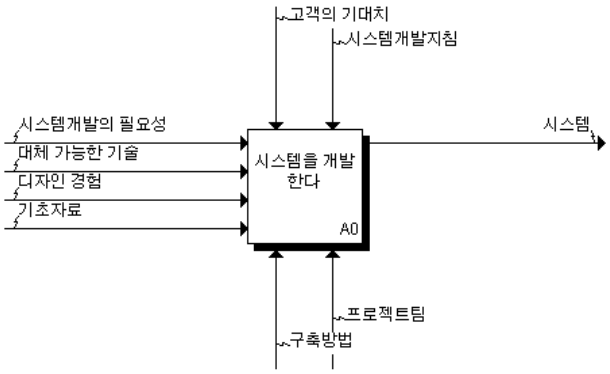
그림 2-9 : 기능박스의 경계조건

사실 IDEF⌀ 모델 다이어그램은 주어진 기능을 분해 함으로서 이를 구성하는 하위기능과 상위기능 간에 관련된 연관관계를 표현하고자 하는 것이다.

■ 기능적 분해의 원칙

① 각 활동은 구분될 수 있는 하위활동으로 구성된다.
② 오직 하나의 활동만이 표현되는 A0 다이아그램을 제외하고는 일반적으로 상위활동은 3 개에서 6 개 사이의 하위활동으로 분해된다.
③ 각 하위활동은 다시 분해되어질 수 있으며 이 때 하위활동의 상위활동이 된다.
④ 하나의 모델 안에서 각 활동은 유일하며 여러 번 사용될 수 없다.
⑤ 모델 안의 모든 다이아그램은 모델의 전체적 관점, 목적, 배경에 충실해야 한다.

예를 들어 그림 2-10 은 시스템 개발이라는 최상위 단계의 기능을 분해한 하위단계로서 세 개의 기본적인 기능들의 관계를 표현하고 있다. 즉 한 기능의 출력이 다른 기능의 입력, 제어, 혹은 메커니즘 등으로 작용하는 관계를 표현 할 수 있다. 이 때 수직적 해체의 단계는 모델의 목적에 따라 달라지나 배경 다이어그램은 모델분해에 앞서 우선적으로 설정되어야 하며 하위단계의 다이어그램은 상위단계의 기능박스에서 설정된 경계 조건의 범위를 승계한다.



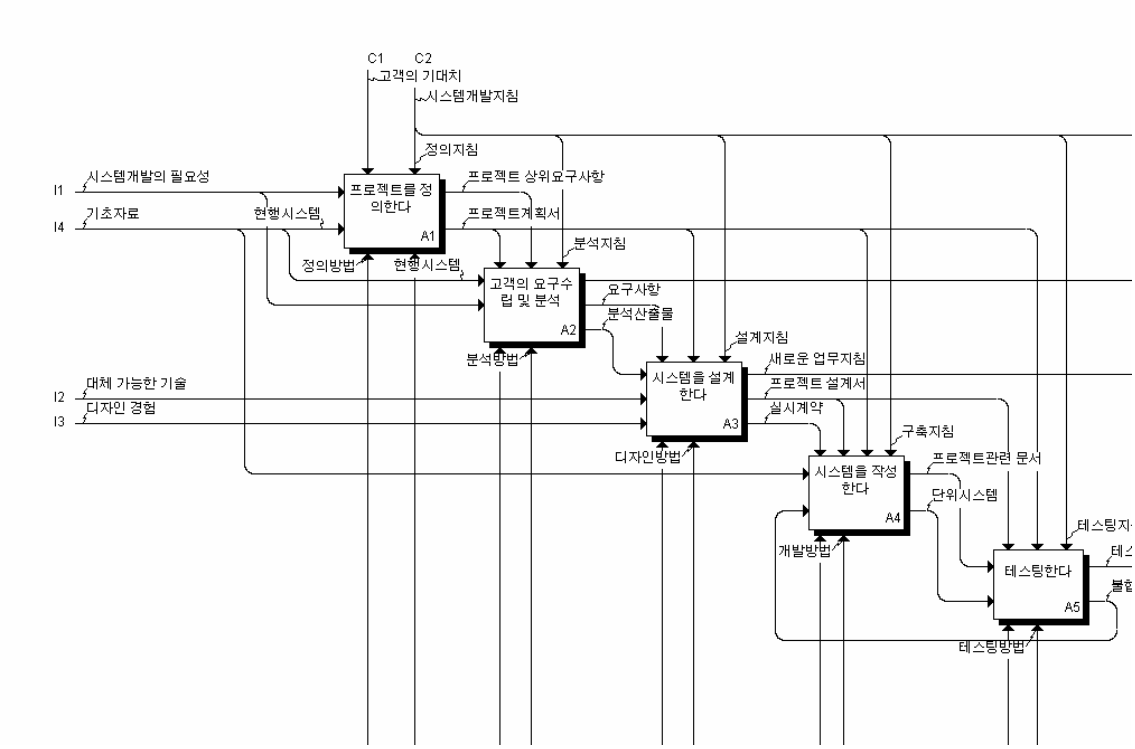
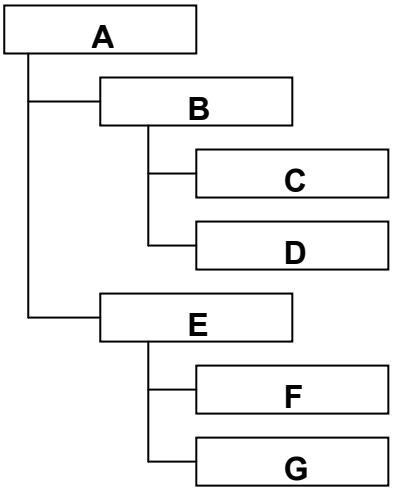


그림 2-10 : IDEF 모델에서의 기능분해의 상관관계

■ 분해 다이아그램의 레벨링 참조용어 정리

우리는 정보시스템 개발 과정에서 분해레벨에 관하여 자주 사용되는 참조용어를 다음과 같이 정리한다.



용어	설명	예
Roor	최상위 레벨의 Object	A
Leaf	최하위 레벨의 Object	C, D, F, G
Parent	하위 레벨의 Object 들을 갖는 Obect	B 는 C 와 D 의 Parent
Child	상위 레벨의 Object 를 갖는 Object	C 와 D 는 B 의 Child
Sibling	Parent 가 같은 Object 들	C 와 D 는 Sibling, B 와 E 는 Sibling
Ancestor	하위 레벨의 Object 집합을 갖는 상위 레벨 Obect	A 는 B, C, D, E, F, G 의 Ancestor
Progeny	상위 레벨의 Obect 에 속하는 하위 레벨 Object 집합	B, C, D, E, F, G 는 A 의 Progeny
Ancestry	특정 Object 의 Ancestor 가 될 수 있는 Object 들의 집합	C 의 Ancestry 는 B 와 A

그림 2-11 : 분해다이아그램의 레벨링 참조용어

2.2.3. 다이아그램 번호 부여

■ 기능의 계층구조

① 모델에서 각각의 활동은 유일하게 부여되어진 기능번호(예를 들면 – A0, A1, A21)로 구분된다.

② 각각의 활동은 이에 관련된 분해번호에 따라 유일하게 위치한다.

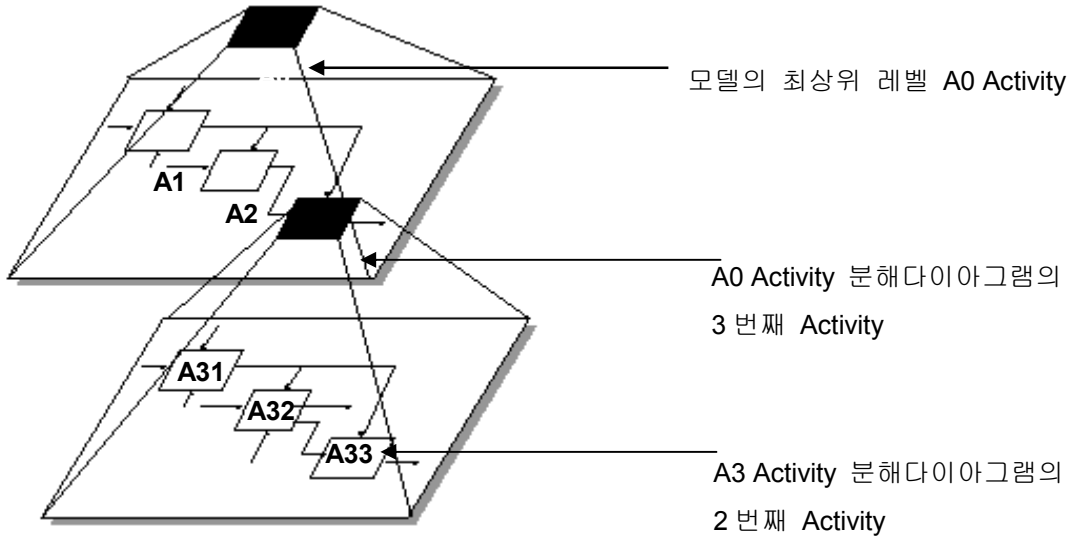


그림 2-12 : 다이아그램 번호부여 체계

기능의 계속적 분해 및 확장에 따라 다이어그램에는 많은 기능박스와 ICOM 이 계속적으로 생성 혹은 수정되며, 이때 개발자는 이들 각각을 구분하기 위한 고유번호를 부여할 필요를 가진다. IDEFø 는 이와 같은 필요에 부응하여 각각의 기능 및 ICOM 에 대한 번호부여 방법을 제시하고 있는데 그림 2-12 와 같이 배경(Context) 다이어그램의 기능을 A0 로 표기하며 여기에서 분해된 하위단계의 기능에 대하여는 A1, A2, A3...라는 번호를 부여한다. 또한 A2 기능의 하위 단계에는 A21, A22, A23...과 같이 상위단계의 번호를 포함 함으로서 각각의 기능이 상위의 어떤 기능에서 분해된 것인지를 나타낸다. 이때 기능의 번호를 'A'로 시작하는 이유는 기능의 다른 개념인 'Activity'의 'A'를 나타낸다. 모델에서 하나의 활동은 두 개의 번호를 가지고 있다. 그 첫번째는 다이아그램 레벨을 가리킨다. A0 다이아그램에서 분해되어진 활동은 상위활동에 따라 번호가 부여된다. 만일 A1 활동이 분해되어진다면 분해된 다이아그램에서는 A1 이라는 번호가 모든 활동의 번호 앞에 부여된다.

두 번째 번호는 다이아그램에서 활동의 위치에 의해 위로부터 아래로의 순서에 따라서 매겨진다. A1 다이아그램에서 순서적으로 첫번째 있는 활동에 1 이 부여된다. 따라서 이 때

붙여지는 활동의 완성된 번호는 A11 이며 두 번째 활동의 번호는 A12 가 된다. 이러한 다이아그램의 계층적 구조에 있어서 우리는 최상위 다이아그램(A0 활동을 포함하는)을 A0 다이아그램이라고 한다. 만일 A0 활동을 이루는 A1 활동이 분해된 것을 나타내는 분해 다이아그램은 A1 다이아그램이라고 간주된다.

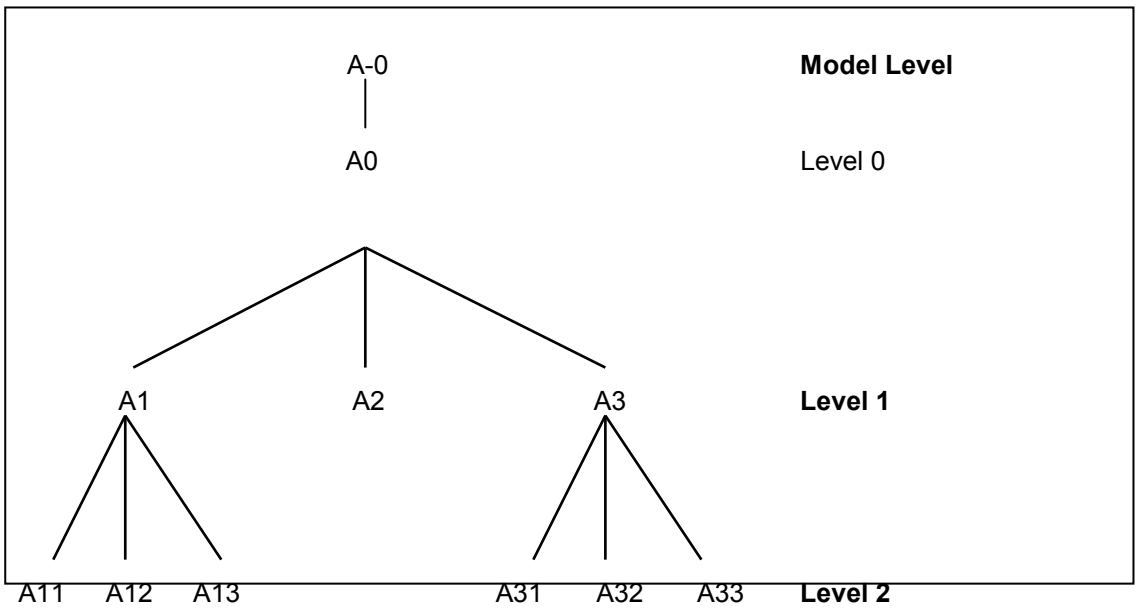


그림 2-13 : 모델의 노드 구조

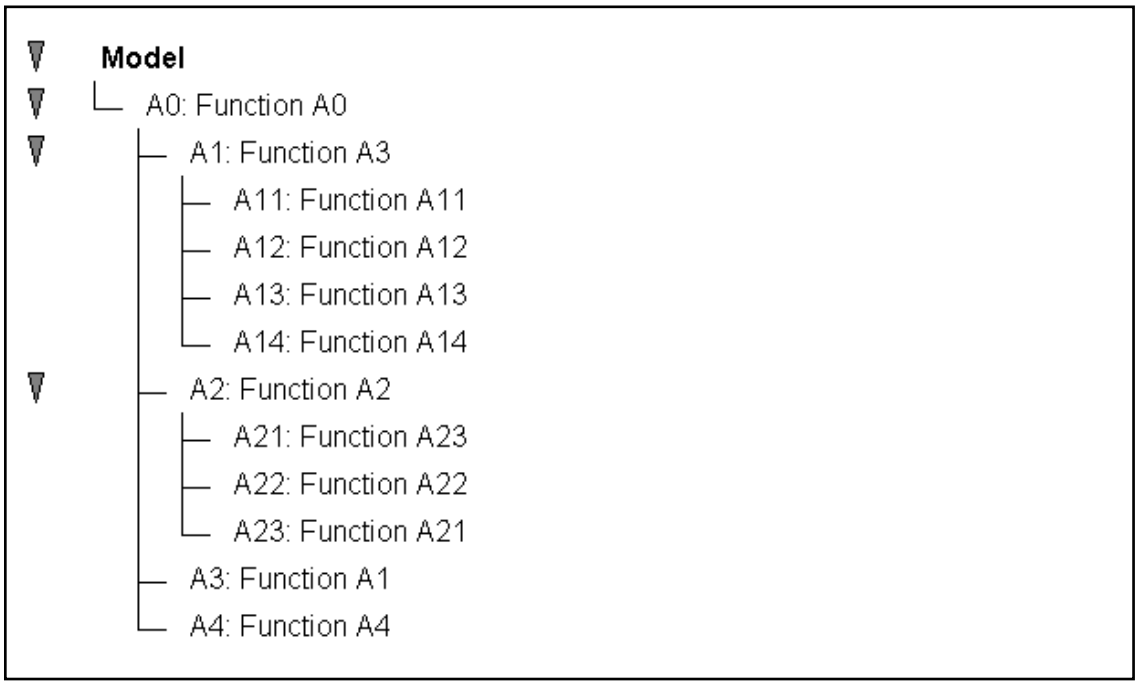


그림 2-14 : Indent 노드트리

IDEF⌀ 는 기능의 계층적 관계를 모델작업자 편의에 따라 다른 형태의 도형으로 표현하기도 하는데 그림 2-13. 과 그림 2-14 는 그 예이다.

■ 개념(concept)의 번호부여와 계층구조

① 개념(입력, 제어, 출력, 메커니즘)들은 모델 안에서 유일하게 사용되어질 필요는 없다.
② 그러나 상위활동과 하위활동 사이에서만은 유일하게 하나로 사용되어야 한다.
③ 각 개념은 글자와 숫자의 조합으로 상위다이아그램에서 어떻게 사용되어지는지를 구분한다.

ICOM 에 있어서는 입력(Input)은 I1, I2, I3..., 제어(Control)는 C1, C2, C3...와 같이 ICOM 각각에 대하여 부여 방법을 달리하고 있는데 ICOM 에 있어서는 하나의 ICOM 번호가 기능번호와 달리 전체 모델에서 고유한 식별자로 사용되는 것은 아니며 단지 표현된 ICOM 이 다이어그램과 관련된 상위단계의 기능박스에서 ICOM 중 어떤 개념으로 사용되는지, 또 그 중 몇 번째 표현된 것인지를 판독하기 위해 나타낸다. 즉 'I3'으로 표현된 번호는 이 개념이 상위 다이어그램의 연관된 기능박스에서 입력(Input)으로 사용되었으며 그 중 세 번째 그려진 화살표임을 나타낸다. 이와 병행하여 모델개발자는 한 모델에서 각각의 기능 및 ICOM 에 대한 식별이 가능하게 하도록 레이블(명칭)을 사용하게 되는데 한 모델 내에서 사용되는 레이블은 기능 과 ICOM 전체를 통하여 중복 부여되어서는 안되며 여러 이름이 동일한 기능이나 ICOM 에 대하여 부여되어서도 안 된다. 모델작성에 있어서 우리는 기능의 분해에 따른 다이아그램의 상세화와 함께 기능과 관련된 개념 또한 상세화 되어야만 됨을 느낄 수 있다. 이와 같은 기능의 상세화는 두 가지로 구분될 수 있는데 그 하나는 상위레벨에서 표현된 개념의 구체화를 위한 것이고 다른 하나는 하위레벨에서 새롭게 표현되어지는 기능과 기능의 연결을 위한 개념들이다. 아래의 그림에서 C1 로 번호가 붙은 '자재조달 지침' 개념은 기능의 분해와 함께 '자재수불 규정' 및 '소요량 산출 지침'으로 분해되었으며 상위레벨에서 제약(Control)의 첫번째로 사용되었음을 나타낸다. 또한 C2 번호가 붙은 '거래선관리 지침'은 상위레벨에서 제약으로 두 번째로 사용되었음을 나타낸다. '재고수량'은 기능의 분해와 함께 새롭게 표현된 개념이다. 개념의 계층구조에 관하여는 ICOM 의 분해와 결합에 관한 설명에서 다시 논하도록 한다.

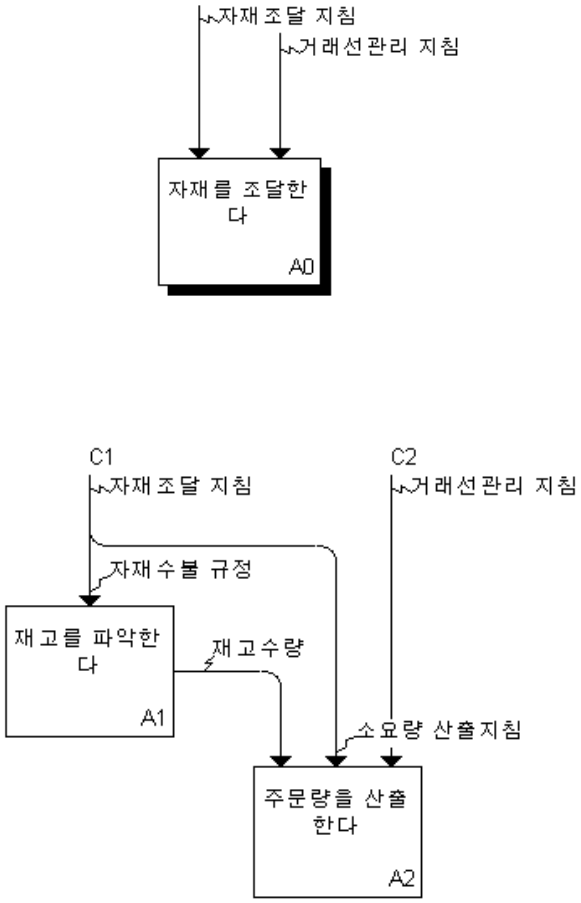

자재조달 지침
거래선관리 지침
자재를 조달한다
A0
C1
자재조달 지침
C2
거래선관리 지침
자재수불 규정
재고를 파악한다
재고수량
A1
소요량 산출지침
주문량을 산출한다
A2

그림 2-15 : 개념의 분해

2.2.4. 기능과 개념의 연관관계

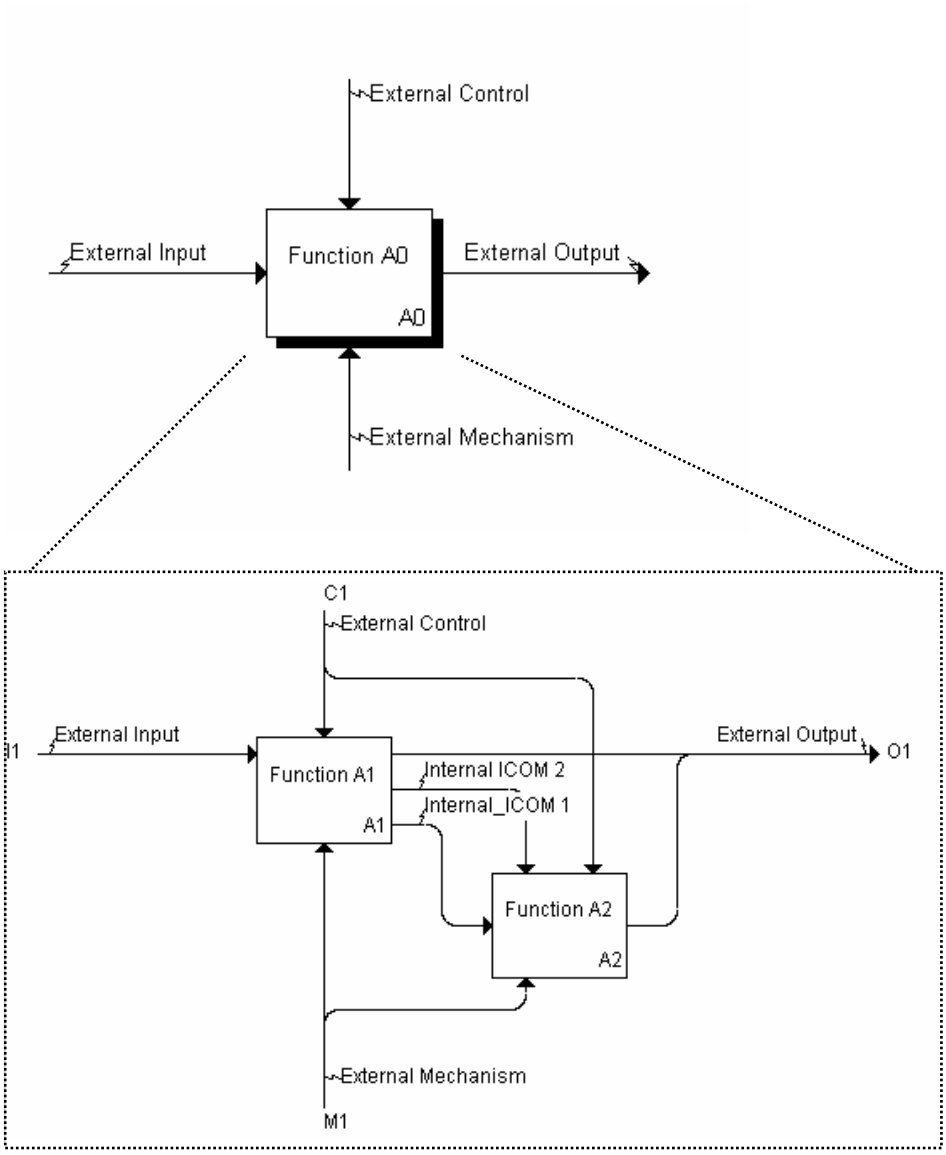


그림 2-16 : 내부 ICOM 과 외부 ICOM

이 절에서 우리는 다이어그램에서 표현되는 ICOM 의 표현방법에 관하여 알아보자.
ICOM 은 크게 내부 ICOM 과 외부 ICOM 으로 구분할 수 있는데 다이어그램에서 기능과 기능을 연결하는 ICOM 을 내부 ICOM 이라고 하며 한쪽은 기능과 연결되었으나 다른 한쪽은 아무 기능에도 연결되지 않은 ICOM 을 외부 ICOM 이라고 한다. 외부 ICOM 은 상위단계의 기능에 이미 표현된 ICOM 을 상속 받은 것이며 내부 ICOM 은 현 단계의 다이아그램에서 각 기능에 의해 새롭게 나타난 것이다. 내부 ICOM 은 일반적으로 하나의 기능에서 출력으로 사용된 ICOM 이 다른 기능의 입력이나 제어로 사용되며 때때로 메커니즘으로 사용되기

도 한다. 또한 하나의 입력, 제어, 메커니즘이 여러 개의 기능에 동시에 사용되는 경우와 하나 이상의 기능에서 같은 출력이 나오는 경우를 그림 2-16 은 보여주고 있다.

■ ICOM의구분

① 내부 ICOM : 현 단계의 다이아그램에서 각 기능에 의해 새롭게 생성됨.
② 외부 ICOM : 상위단계에서 표현된 것을 현 단계에서 상속 받음.
④ 배경다이아그램(A0 다이아그램)에서 표현된 화살표는 모두 외부(external) 다이아그램이라 한다. 이들 화살표는 전체 모델의 밖으로 나가거나 모델 밖에서 모델 안으로 들어오는 관계를 표현한다.

2.2.5. ICOM의 결합(Join) 과 분해(Split)

IDEFø 는 위에서 설명한 바와 같이 모델의 분해(Decomposition)에 있어서 기능의 수직적 해체 뿐만 아니라 각각의 ICOM 에 관한 수직적 해체도 병행된다. 즉 상위단계에서 표현된 ICOM 은 하위단계에서 분해된 기능과 함께 좀더 구체적이고 자세하게 분해되어 표현되거나 재사용된다. (예를 들면 - 상위 다이어그램에서 제품으로 표현된 출력이 하위단계에서는 제품 A, 제품 B, 제품 C 라는 상세화 된 세 개의 출력으로 표현될 수 있다). 이때 상위단계에서 상속 받은 하나의 입력, 제어, 메커니즘이 여러 개의 기능에 세분화 되어 할당되는 것을 분해(Split, Unbundle)이라 하며 하위단계의 여러 기능에서 나온 출력이 상위단계에서 표현된 하나의 출력에 할당되는 것을 결합(Join, Bundle)이라고 한다.

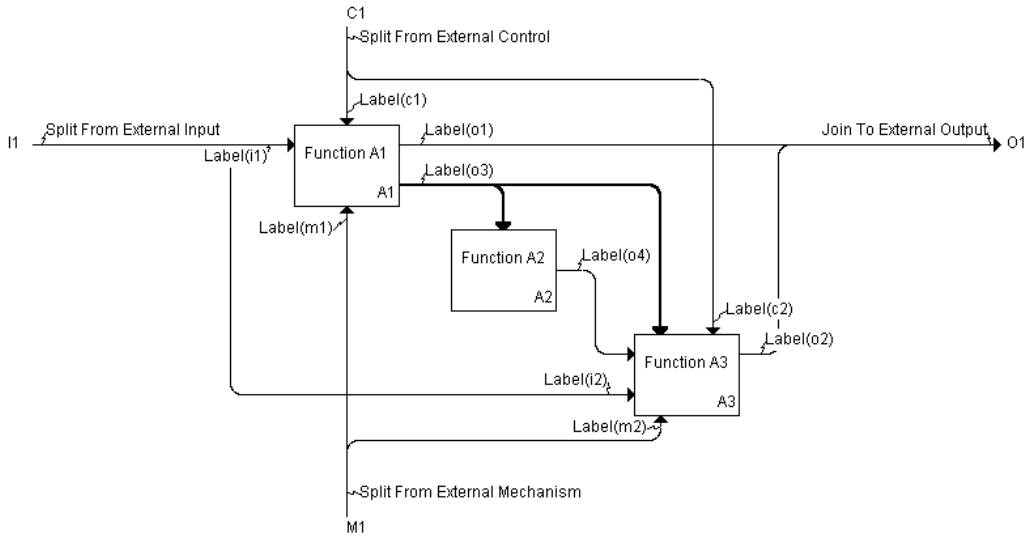


그림 2-17: ICOM 의 Join 과 Split

■ ICOM의 결합과 분해

① ICOM 의 결합(join, bundling)은 우리에게 여러 개념을 그룹화하여 하나의 커다란 개념의 세트로 만들 수 있도록 지원한다.
② ICOM 의 분해(split, unbundling)는 우리에게 일반적인 개념을 분해하여 그 것을 이루고 있는 부분들로 표현 할 수 있도록 지원한다.

아래의 그림은 상위레벨에서 표현된 Control_C 가 Control_c1 과 Control_c2 로 분해되는 과정을 보여주고 있다. 그러나 상위레벨에서 표현된 Mechanism 의 경우는 하위레벨에서 분해되지 않고 단지 A1 활동과 A2 활동으로 분해된 기능에 다시 할당됨을 보여주고 있는데 이를 개념의 분기(branch)이라 한다.

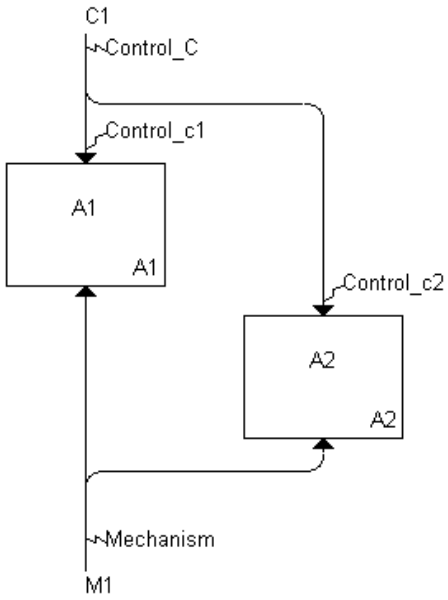


그림 2-18 : ICOM 의 분해와 분기

2.2.6 턴넬링(Tunneling)

일반적으로 배경(Context) 다이어그램이 아닌 하위단계의 다이어그램에서는 관련 상위단계의 ICOM 과 하위 ICOM 간에 연결/상속 관계를 자세히 기술해야 된다. 그러나 IDEFø 는 기능의 분해를 기반으로 하고있으며 따라서 상위단계의 ICOM 이 하위단계의 다이어그램에서 생략되거나 상위단계에서 생략된 ICOM 이 하위단계에서 표현되는 것을 허용하고 있다. 이는 기능간의 관계파악이라는 IDEFø 의 기본목적을 달성하기 위하여 허용된 사항으로 IDEFø 는 모델의 세분화에 따른 이러한 문법적 문제를 해결하고자 아래와 같은 방법으로 이를 표현한다. 이러한 턴넬링의 표현방법은 다이아그램의 단순화를 지원하는데 다이아그램 내부에 모든 것을 다 표현함으로써 야기되는 복잡성을 배제하여 활동 사이의 기능적 관계에 대한 의사소통을 촉진한다. 그러나 이러한 방법이 모델에서 불필요한 개념을 제거하려는 의도로 사용되어서는 안 된다.

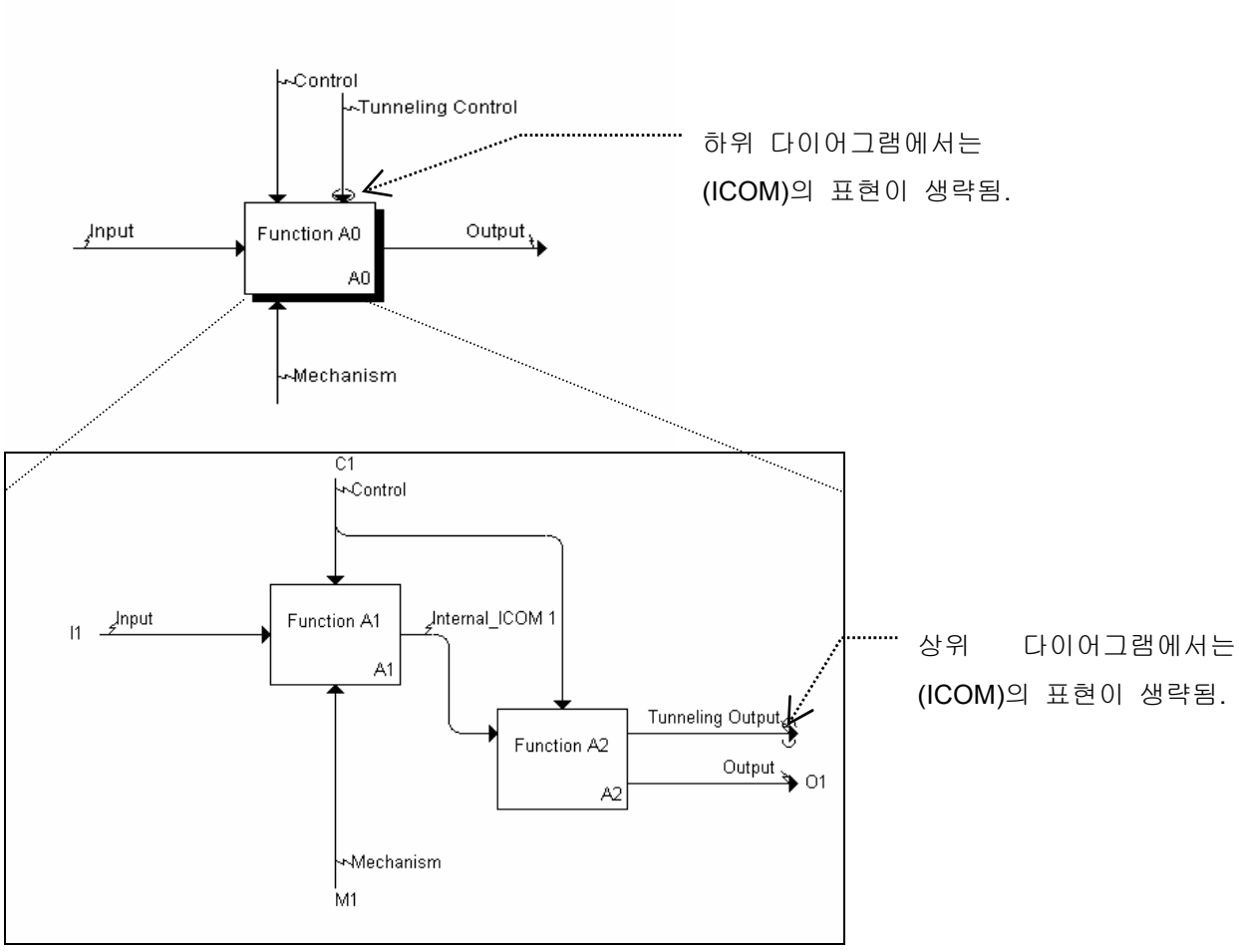


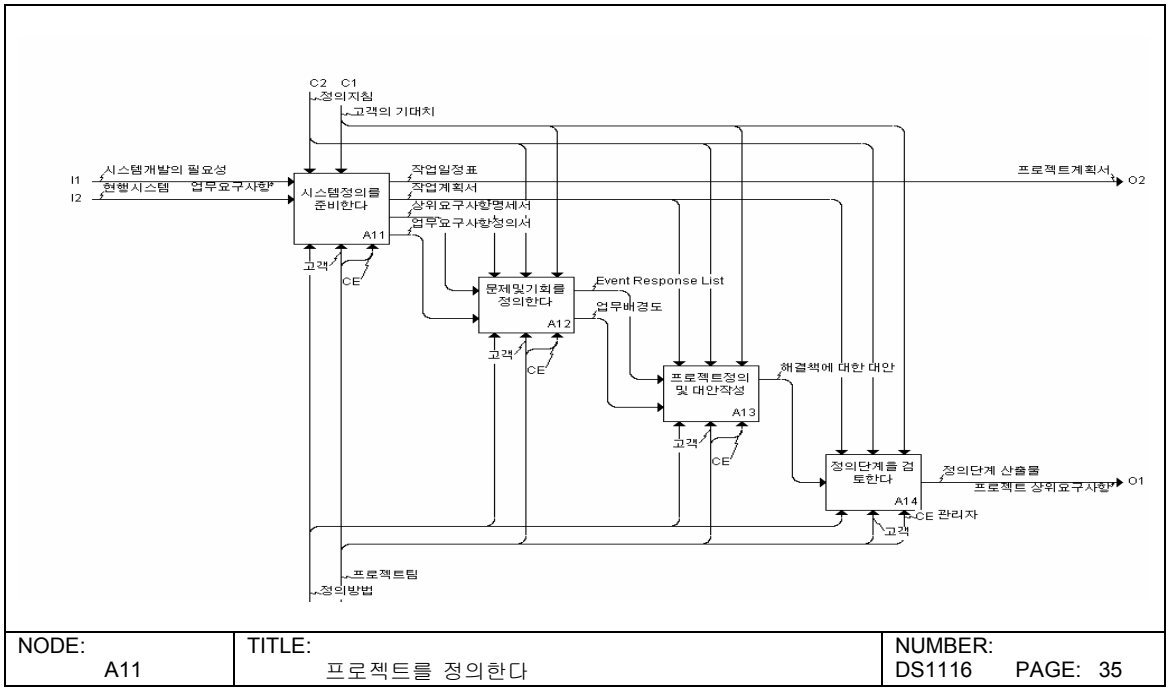
그림 2-19 : 개념(Concept)의 턴넬링(Tunneling)

■ 턴넬링의 표현방법

① 턴넬링은 개념이 상위레벨 혹은 하위레벨의 다이아그램에서 표현될 것인가 혹은 생략될 것인가를 나타낸다.
② 기능박스로 이어지는 부분쪽에 표시된 괄호는 하위레벨의 다이아그램에서 이 개념이 더 이상 표현되지 않음을 명시한다.
③ 기능박스와 연결되지 않은 부분쪽에 표시된 괄호는 상위레벨의 다이아그램에서 이 개념이 표현되지 않았음을 나타낸다.

2.2.7 IDEF 다이아그램 FROM

USED AT:	AUTHOR: 홍길동 DATE: 2000.3.21	x	WORKING	READER:	CONTEXT:
XYZ 회사	PROJECT: ICMS REV:		DRAFT		
			RECOMMENDED		
	NOTE: 1234567890		PUBLICATION	DATE	



NODE: A11	TITLE: 프로젝트를 정의한다	NUMBER: DS1116 PAGE: 35

그림 2-20 : IDEF 다이아그램 출력 Form

IDEF⌀ 기능모델의 다이아그램은 그림 2-20 과 같은 표준 다이아그램 폼의 출력을 지원하는데 각 항은 다음과 같은 내용을 포함한다.

① USED AT : 해당 모델이 사용되어질 조직이나 부서 등을 나타낸다
② AUTHOR : 모델 작성자
③ DATE : 작성일
④ PROJECT : 프로젝트명
⑤ REV : 마지막으로 수정된 일자
⑥ NOTE : 검토자에 의해 검토된 횟수를 표현한다.
⑦ STATUS
WORKING : 현재 모델을 새롭게 작성 중이거나 수정중임을 나타냄.
DRAFT : 검토자에 의하여 검토완료 된 상태. 최종 승인을 앞둔 상태.
RECOMMENDED : 다이아그램과 그 세부내용이 검토 완료되고 승인된 상태.
PUBLICATION : 모든 것이 완료되고 배포되기 위한 상태.
⑧ READER : 검토자의 이름
⑨ DATE : 검토 일자
⑩ CONTEXT : 현재의 다이아그램이 어디에서 분해된 것인지를 나타내며 항상 현재 다이아그램의 상위레벨을 나타냄. 모델(A-0)레벨 다이아그램의 Context 는 NONE 로 표시되며, A0 레벨의 경우는 TOP 으로 표시됨.
⑪ NODE : 상위활동의 활동번호
⑫ TITLE : 상위활동의 이름
⑬ NUMBER :
C-Number : C-Number 는 모델 안에서 해당 다이아그램에 대한 유일한 식별 번호이다. 일반적으로 작성자의 이니셜을 나타내는 2-3 글자와 일련번호로 구성된다. 모델이 완료되고 배포될 때 C-number 는 표준화된 페이지 번호로 변경될 수 있다.
PAGE : 해당 문서철 안에서의 순차적인 일련번호.

또한 이러한 다이아그램은 IDEF Cover Sheet 그리고 IDEF Node Index/Contents Sheet 와 함께 다음과 같은 절차로 운영된다.

① 새로운 모델을 작성한다.
② 문서담당자는 문서대장에 이를 등록 후 검토자에게 전달.
③ 검토자는 모델을 검토 후 수정이나 추가, 삭제가 필요한 부문의 의견을 기술한다.
④ 문서담당자를 통하여 전달된 검토자의 의견을 모델에 반영한다.
⑤ 수정된 모델을 다시 검토한다.
⑥ 모델작업자 혹은 검토자의 요청에 의하여 토의를 위한 모임을 갖는다.
⑦ 이러한 절차는 STATUS 항목과 NOTE 번호, NUMBER 항목에 반영된다.
⑧ 이러한 절차를 거쳐 최종 검토를 완료하고 승인을 요청한다.

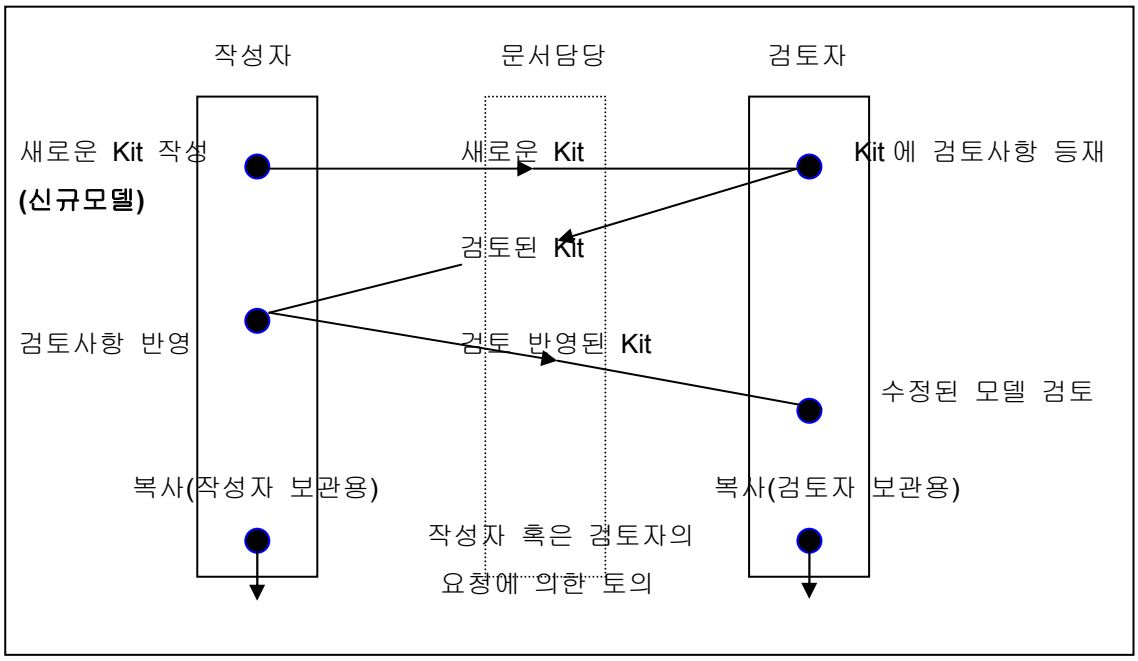


그림 2-21 : Author – Reader Cycle

2.2.8 그 밖의 구성요소

■ For Exposition Only(FEO)

아래의 왼쪽 그림과 같은 형태로 다이아그램의 분해가 생략된 경우 판독자는 입력 I1, I2, I3 와 제어 C1, C2, C3, C4 가 출력 O1, O2, O3 에 상호 어떠한 연관관계를 맺고 있는지를 알 수가 없으며 따라서 다음과 같은 추측이 가능하다.

① O1, O2, O3 는 각각 모든 입력(I1, I2, I3)이나 제약(C1, C2, C3)이 작동되지 않더라도 생산이 가능하다.
② O1, O2, O3 는 모든 입력과 제약이작동 되어야만 생산이 가능하다.

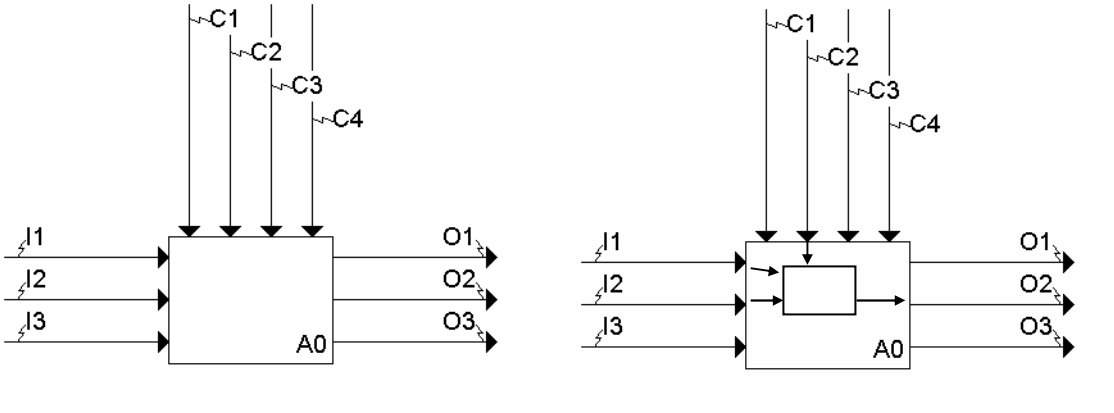


그림 2-22 : FEO

우리는 이러한 상황에서 출력에 관여하는 입력과 제어를 판독자에게 설명하기 위하여, 또는 분해되지 않은 레벨에서 관련되어지는 개념에 대하여만 서술하기 위하여 IDEF⌀ 다이아그램 문법과는 별도로 오른쪽의 그림과 같이 기술할 수 있다.

이러한 모델작업의 진행과 함께 여러분이 수행한 IDEF⌀ 의 요소들을 정의하고 분류하여 문서화 하여야 한다. 모델 내의 모든 요소들은 용어해설이 되어야 하며 문자로 이루어진 자세한 설명은 다이아그램과 관련된다. 이것은 다이아그램으로 표현될 수는 없으나 필요하거나 알려지고 이해되어야 할 사항을 표현한다. 경험적으로 볼 때 용어의 정의나 기능수행과 관련된 업무 룰 등의 문서화 작업의 병행은 시스템 개발 프로젝트의 라이프사이클 전체에 있어서 참으로 많은 효용성을 지원한다. 그러나 이러한 작업이 병행되지 않은 모델화 작업의 결과는 모델로서의 가치가 없음은 물론 무의미한 시간과 정력의 낭비일 뿐이다. 다음은 이러한 문서화 작업과 관련된 것 들이다.

■ 설명(Description)

설명은 기능이나 개념의 이름으로 쓰이는 용어의 정의는 물론 기능이나 개념이 무엇이고 무엇을 수행하는지를 객관적으로 기술한다. 이러한 설명은 모델 판독자는 물론 정보시스템 개발자에게 시스템에서 사용되는 용어정의 리스트(Terminology Definition List)와 기능정의 리스트(Activity/Process Definition List)로 전달된다. 하나의 기능이나 개념을 위한 설명은 오직 하나만 제공된다.

예 : Location 번호
– 제품을 출하하기 위하여 제품창고에 적재시 제품창고의 각 위치(Rack)를 나타내는 번호.

■ 노트(note)

모델 판독자의 이해를 돕기위해 기능이나 개념과 관련된 제약사항이나 참고사항, 업무 룰을 주로 포함한다. 이러한 업무 룰은 프로그램 개발자에게 프로그램 코딩을 위한 업무 룰로서 메소드(Method)를 지원한다. 하나의 기능이나 개념에 다수의 노트를 등록할 수 있다.

예 : Location 번호부여 방법
– Location 번호는 각 출고처(배송센터+납품처)별로 코드가 부여된다.
YYMMDD+ 9999(Sequence)로 부여된다.
포장되지 않은 농산물에 대하여는 배송센터별로 부여된다.

■ 소스(Source)

해당 자료가 어디에서 수집되었는지 누가 관리하는 정보인지를 나타낸다. 하나의 개념이나 기능에 다수의 소스를 등록할 수 있다.

예: Location 번호
- 제품관리과 창고 담당자 홍길동, Location 번호부여 결과보고서
문서번호: 980205-1

■ 속성(Property)

해당 기능이나 개념이 가지고 있는 속성을 나타낸다. 속성은 데이터의 타입과 그 값, 그리고 설명으로 나타낸다. 하나의 기능이나 개념에 다수의 속성을 등록할 수 있다.

예 : 농산물 여부

- 데이터 타입 : Boolean
- 값 : True

Frequency

- 데이터 타입 : String
- 200 회/ 일
- 주간에는 평균적으로 200 건이 처리됨.

2.3 AIØWIN 을 이용한 IDEFØ 모델의 개발

기능모델의 첫 번째 목적은 조직이 무엇을 수행하는가를 파악하고 문서화하는 것이다. 기능모델은 개별적인 활동 뿐만 아니라 각 활동의 기능하에서 수행되는 활동과 관련된 사실을 나타내는데 있다. AIØWIN의 기능모델링 지원은 이러한 작업을 쉽게 그리고 강력한 활동모델의 디자인과 구축이 가능하도록 한다. 이번 절에서는 AIØWIN을 사용하여 여러분이 직접 기능모델을 작성하는 과정을 지원한다. 또한 AIØWIN의 기능을 파악하여 이를 효과적으로 응용하는 능력을 배양하는 것에도 초점을 두어 진행할 것이다.

2.3.1 배경, 관점 그리고 목적을 확립하고 검토한다.

모델개발 프로세스는 배경, 관점, 목적을 확립하고 모델화 되어야 할 분야에 관한 정보를 수집하고, 계층구조에 있어서 상위 혹은 하위방향으로 정제하는 많은 단계로 이루어져 있다. 우선적으로 모델링작업을 위한 상황이나 배경, 목적, 과점을 정의함으로써 프로젝트의 목표 달성에 쉽게 접근할 수 있다. 이러한 세가지 요소는 모델링을 위한 구조 및 투명성을 제공하며 현재 수행되는 모델링 작업의 기준이 되며 후에 여러분의 모델을 분석하게 될 다른 모델작업자를 위한 참조가 될 것이다. 배경(context)은 모델의 범위와 경계를 설정하고 모델에 무엇을 포함할 것인가를 정의한다. 이러한 배경 다이아그램은 A0 다이아그램이라고도 하는데 만일 범위를 너무 넓게 설정하면 모델이 복잡해지고 따라서 너무 많은 노력이 소요된다, 반면 너무 작은 범위를 설정하면 보잘것없는 모델만을 개발할 수 있을 뿐으로 배경의 설정은 모델링 작업에 있어서 가장 핵심적인 단계에 속한다.

■ 배경(context)

배경은 모델의 경계를 설정하고 모델에 무엇을 포함할 것인가를 정의한다.

① 모델의 범위를 확립한다.
② 좀더 큰 범위 내에서 부분으로서의 주제를 확립한다.
③ 주변과의 경계를 설정한다.

■ 배경(context) 설정의 예 :

대부분의 조직과 같이, XYZ 회사도 각 부서에서 사용할 자재를 외주 공급자로부터 조달하는 자재부를 가지고 있다.
XYZ회사 자재부는 자재를 필요로 하는 부서에 적기에 자재를 공급하고 자재를 납품하는 공급자의 기록을 유지하는 일에 대하여 책임을 지고 있다. 마니정보시스템의 컨설팅 팀과

협의 중에 XYZ회사 자재부장은 자재부의 효율을 증진시키기 위한 활동모델을 활용할 기회를 얻게 되었다.

자재부장은 부서의 직원들이 수행할 활동모델과 활동 상호간의 관계를 구축하기 위해 컨설턴트에게 도움을 청하였다. 자재부장은 현황을 파악하고 분석하여 컨설턴트와 공동으로 부서의 비용을 줄일 방안이 제시되기를 기대하고 있었다. 결국 AS-IS를 TO-BE로 전개하는데 목적이 있다. 여기서 TO-BE는 사업을 더 효율적으로 수행하고 새로운 직원이 고용되었을 때 교육을 하기 위한 것이었다.

컨설턴트는 많은 자재부 직원들과 면담을 하고 자신의 지식에 기초하여 자재조달 활동모델을 구축하였다. 그는 자재부가 수행한 다섯 가지의 핵심기능을 정의하였는데 그것은 자재요청 접수, 자재 발주, 자재 배분관리, 거래선 관리, 자재예산의 확보 등이었다. 마지막항목을 제외한 네 가지의 활동들은 서너 개의 하위 활동들을 포함하고 있었다.

이들 하나 하나의 핵심 기능을 보면서 자재부장과 컨설턴트는 자재요청 접수과정에서 개선할 수 있는 부문을 즉시 파악할 수 있었다. 전산화 된 자재요청 시스템은 AS-IS 모델에서 자재요청을 확인하는 활동이 불필요하게 할 수 있었다.

TO-BE 시스템에서는 자재요청은 전산으로 자동화될 수 이도록 검토되었고 어떠한 필요한 확인작업도 이미 가동중인 XYZ 회사의 email시스템을 통하여 수행할 수 있었다. 이 변경의 효과는 자재담당 직원이 확인작업보다는 좀더 활동적으로 자재요청을 접수할 수 있도록 하는 시간을 확보할 수 있게 되어, 내부고객에게 자재조달 기간을 단축할 수 있게 하였고 상위 활동인 자재요청 프로세스 활동의 경비를 줄일 수 있게 되었다.

이 경과에 고무되어 자재부장은 컨설턴트에게 자재부가 수행하는 모든 활동들의 활동기준 원가산정(Activity Based Costing)을 수행할 것을 요구하였다. 이 프로젝트의 몇 가지 모델은 이것을 수행하는 컨설턴트의 노력을 문서화하였다.

■ 목적(Purpose)

모델을 개발하는 이유를 설명한다.

① 모델을 통해서 표현하고자 하는 목적을 확립한다.
② 왜 모델이 개발되는지를 정의한다.
③ 모델이 어디에 사용될 것인가를 상세화 한다.

목적은 특정 활동모델을 개발하는 이유이다. IDEF⌀ 뿐만 아니라 모델링 작업에 있어서 가장 중요하면서도 어려운 점은 모델의 레벨 사이에 일관된 목적과 관점을 유지하는 것이다. 하나의 현실세계는 목적과 관점에 따라 여러 가지 형태의 모델로 작성 되어질 수 있는 것이다. IDEF⌀ 모델링 활동을 시작하기 위하여 모델 작업자는 모델의 '목적', 그리고 활동의 설명이 정형화 될 수 있는 '관점'을 우선적으로 결정해야 한다.(이는 모델 작업에 있어서 가장 중요한 사항이다.) 목적은 모델화 작업의 목표 즉, 어떤 정보를 모을 필요가 있는가, 이 정보가 어떤 의사결정을 지원해야 하는가, 어떤 동의를 얻어야 하는가 등을 설명하는 것이다. 수용된 목적은 모델화 작업 팀에게 모델 완료의 준거를 제공한다. 즉 목적이 충족될 때 모델은 완료된다.

■ 관점

모델을 개발하는 사람/팀의 전망을 나타낸다.

① 어떠한 각도에서 무엇을 표현할 것인가를 결정한다.
② 관점은 모델을 개발하는 사람이나 그룹이 바라보는 시각을 설명한다.
③ 다른 목적과 다른 관점은 전혀 다른 모델이 된다.

관점은 모델을 구축, 검토, 파악 할 경우에 취해야 할 관점을 설명하는 것이다. 그것은 판독자가 모델을 해석하는 방법, 그리고 모델 작업자가 연구 중에 시스템에서 발생하는 기능 표현의 이상화나 추상화의 한계를 규정한다. 또한 채택된 관점은 모델의 구체적 수준과 범위를 제어하는 메커니즘을 모델화 작업 팀에게 제공한다. 이러한 목적과 관점의 일관된 유지를 위해서는 모델 작업자가 다음과 같은 질문을 정형화 하는 것도 하나의 방법이다. "이 기능이 더 고차원적인 기능의 범위에 들어가는가?", "이 기능이 모델에 대한 기존의 목적과 관점에 맞는가?"

■ 관련정보와 자료를 수집한다.

정보와 관련자료를 수집하기 위해서는 누가 어떠한 자료를 가지고있는지 이를 검토할 수 있는 사람은 누구인가를 우선적으로 파악해야 한다. 또한 관련된 업무에 있어서 이해당사자를 파악하고 이들과의 인터뷰를 통해서 이를 확인할 수 있다. 인터뷰는 아래와 같은 방법으로 진행한다.

① 조직 내의 모든 계층을 포함한다.
② 경청한다.
③ 자세한 노트를 남긴다.
④ 가능한 한 많이 수집한다.

인터뷰 후 수집된 자료를 구성하고 모델을 완료 시키기 위하여는 모델작업자와 검토자 사이에 여러 번에 걸친 자료의 교환이 필요하다. 이러한 자료키트는 모델작업자에게는 작업의 진척사항을 문서화하기 위하여, 검토자에게는 그들의 의견사항을 기록하기 위하여 , 또한 모델작업자는 검토자의 의견사항을 모델에 반영시키기 위하여 사용한다.

2.3.2 모델의 문서화

활동모델링에 대한 AIØWIN의 지원은 설계능력을 향상시키고 포괄적이고 역동적인 활동모델을 창출하는 것이다. 이번 절에서는 활동모델을 어떻게 생성하고 문서화하는 가를 다룬다. 활동모델에 개념(입력, 제어, 출력, 메커니즘)을 추가하기 위해 풀(pool)을 사용하게 된다. 리포지토리, 모델, 분해 다이아그램과 관련된 것의 문서화는 모델 작업을 위한 척도를 정의하고 리포지토리에 관련된 데이터를 제공한다.
모델구축 연습을 시작하기 위하여 우선 예제모델을 구축할 새 리포지토리의 생성이 요구된다.

■ 리포지토리, 모델, 다이아그램이란 무엇인가?

① 리포지토리는 모델링 프로젝트를 위해 잡아둔 장소.
② 프로젝트에 있는 모든 모델과 그 모델을 형성하는 모든 요소들을 포함한다.
③ 모델 요소들은 풀에 저장되는데 하나의 요소만이 하나의 풀에 존재하게 되며, 프로젝트의 모든 모델에 그것을 분배할 수 있고 필요한 만큼 수정하여 사용할 수 있다.
④ 모델은 최상위 레벨의 A-0 다이아그램과 다이아그램의 A0 활동으로부터 유래된 분해 다이아그램의 계층으로 이루어진다.
⑤ 다이아그램은 전체 모델의 한 부분을 나타낸다.
⑥ AIØWIN에는 두 가지 타입의 다이아그램이 있다. 최상위 레벨 즉 A0 다이아그램과 분해 다이아그램이 그것이다.

■ 리포지토리와 모델의 생성

① AIØWIN 시작. 다음의 메뉴 중 원하는 것을 요구하는 대화창이 나타난다 :

Open an existing repository,

Create a new repository using a template,

Create an empty repository.

② create an empty repository를 선택하고 **OK**를 클릭한다.

AIØWIN은 NONNAME.AI0라는 기본으로 제공되는 새로운 프로젝트를 생성한다.
이것은 또한 새로운 모델을 신속히 생성할 수 있도록 Model Browser Window와 Open Model 대화창을 연다.

③ Name 항목에 **자재 조달** 을 쳐 넣고 **Create**를 클릭한다.
④ 새로운 모델을 위한 Model Window를 열기 위하여 Model Window radio button을

활성화하고 **OK**를 클릭한다.

새로운 모델에 요소들을 추가하기 전에 프로젝트 산출물을 참조할 차후의 모델 작업자를 위해 잔여정보를 제공할 Repository를 서식화 한다.

■ Repository 서식화

① File 메뉴에서 *Repository Summary..*를 선택한다
Repository Summary 대화창은 프로젝트의 이름을 할당토록 한다.

프로젝트의 생성을 위한 각각의 임무를 확인한다. 그리고 프로젝트 안에서 모델링 작업의 배경을 설정한다.

리포지토리에 이름을 부여하고, 프로젝트의 배경상황(context)을 설정하며, 프로젝트의 목적(purpose)과 범위(scope)에 대한 설명을 입력한다.

② Repository Summary 대화창 내에 프로젝트 이름(Project Name), 작성자(Creator), 어디에 사용될 것인지(Used At)에 관한 항목을 채워 넣는다. (예를 들면 - **자재조달 업무의 문서화**, **홍길동**, **XYZ회사**)
③ **Description**을 클릭한다.

이는 프로젝트의 목적 과 범위를 설명할 수 있도록 문장을 쳐넣을 수 있도록 Repository Description 대화창을 연다.

이 프로젝트의 모델은 XYZ회사의 재고품목을 위한 자재관리기능을 문서화한다. 기능 모델은 재료획득기능을 위한 활동기반 비용산정(Activity Based Costing)의 기초를 제공한다.

④ Repository Summary 대화창으로 돌아가기 위해 **OK**를 클릭한다.
⑤ 대화창을 닫고 Model Browse Window로 돌아가기 위해 **OK**를 클릭한다.

모델을 문서화하고 모델에 concept과 활동을 추가하기 위해 계속하자. 활성화된 창에 나타난 새로운 모델을 나타내는 Diagram Window에 유의하자. 뒤에 숨은 Model Browse Window는 전체 repository에 생성된 모든 모델을 나타낸다.

■ 모델의 서식화

① Model menu에서 *Model Summary…*를 선택한다.

② **Purpose**를 클릭하고 Diagram Purpose 대화창에 모델의 목적을 설명하는 다음과 같은 문장을 쳐넣는다. (예를 들면 - **"자재조달 기능모델은 자재획득 공정에서 활동기반 원가분석을 수행한다"**)

③ 변경사항을 저장하고 Model Summary 대화창으로 돌아가기 위해 **OK**를 클릭한다.

④ Diagram Viewpoint 대화창에서 **Viewpoint**를 클릭하고, 만들어질 모델의 관점을 설명하는 다음의 문장을 입력한다. (예를 들면 - **"자재조달 모델은 자재부장의 관점에서 생성되어졌다."**)

⑤ 변경사항을 저장하고 Model Summary Window을 돌아가기 위해 **OK**를 클릭한다.

⑥ Model Window로 돌아가기 위해 **OK**를 클릭한다.

지금 작성된 모델에서 AIØWIN에 관해 좀더 알아보자.

■ AIØWIN이 지원하는 윈도우

AIØWIN는 활동 및 ABC모델을 보기위한 네 가지의 윈도우를 제공한다. 각 윈도우는 모델에서 서로 다른 화면을 제공한다. 본 절에서는 각 윈도우가 지원하는 사항에 대해 논의한다.

① 모델브라우져(model browser) 윈도우

모델브라우져 윈도우는 활성화 된 리포지토리에 있는 모든 모델과 모델의 모든 활동 그리고 활동과 관련된 모든 분해다이아그램을 을 보여준다. 모델브라우져 윈도우는 모델의 구조를 작성하는데 유용하다. 이 브라우저에서 할 수 있는 일은 새로운 모델, 다이아그램, 활동의 추가, 그리고 모델, 다이아그램, 활동의 드래그-앤-드롭(drag and drop), 이동 및 복사 등에 사용할 수 있다.

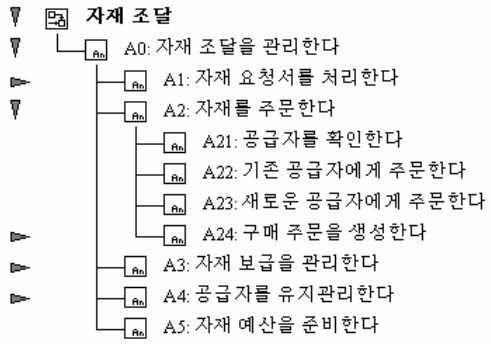


그림 2-23 : 모델브라우져 윈도우

② 다이아그램(diagram) 윈도우

다이아그램 윈도우는 표준 IDEFØ 그래픽 디스플레이를 이용하여 각 다이아그램 레벨을 보여준다. 이 윈도우는 활동과 개념 그리고 각각의 활동들 사이의 관계에서 가장 세부적인 사항의 모습을 제공한다.

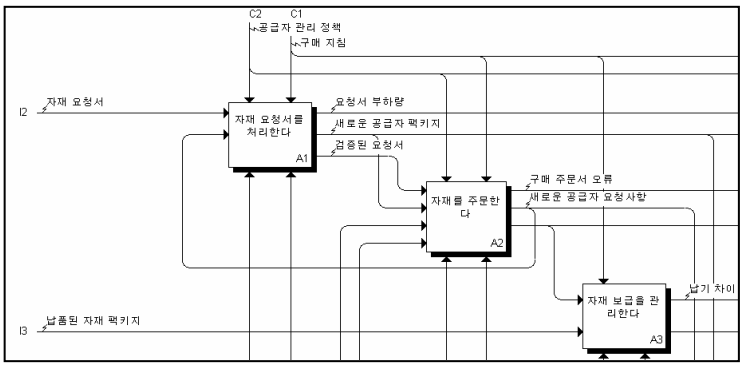
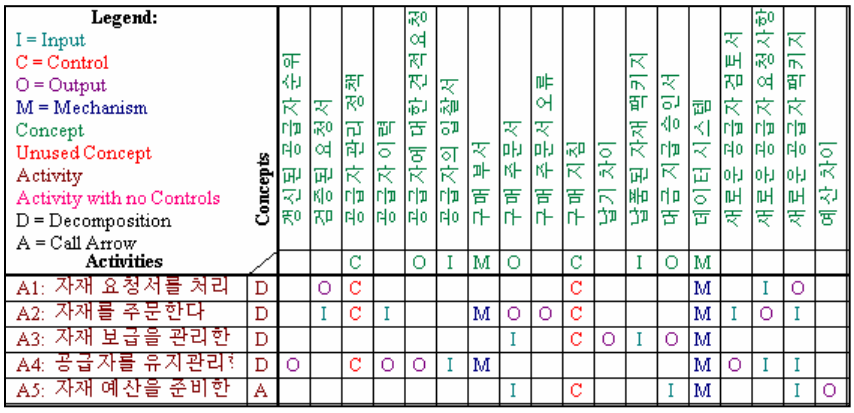


그림 2-24 : 다이아그램 윈도우

③ 활동/개념 매트릭스(Activity/Concept matrix) 윈도우

활동/개념 매트릭스 윈도우는 모델에 있는 모든 활동들과 활동과 관련된 모든 개념들을 보여준다. 매트릭스 셀은 개념들이 어떠한 활동에 의해서 어떻게 사용되는지를 나타낸다. 이 매트릭스 역시 활동들의 분해(decomposition)요소와 전이(migrated)된 개념들을 나타내고 있다.

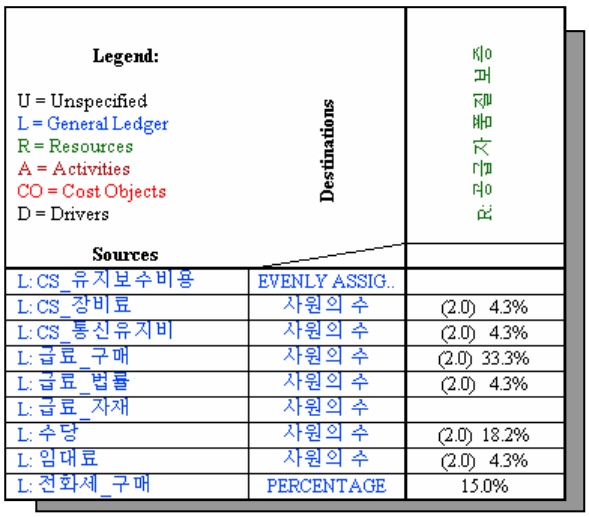


Legend: I = Input, C = Control, O = Output, M = Mechanism, Concept, Unused Concept, Activity, Activity with no Controls, D = Decomposition, A = Call Arrow	Concepts	공급자 순위	검증된 요청서	공급자 관리 정책	공급자 이력	공급자에 대한 견적요청	공급자의 입찰서	구매 부서	구매 주문서	구매 주문서 오류	구매 지침	납기 차이	납품된 자재 박키지	대금 지급 승인서	데이터 시스템	재보운 공급자 검토서	재보운 공급자 요청사항	재보운 공급자 박키지	예산 차이
Activities				C		O	I	M	O		C		I	O	M				
A1: 자재 요청서를 처리	D		O	C							C				M		I	O	
A2: 자재를 주문한다	D		I	C	I			M	O	O	C				M	I	O	I	
A3: 자재 보급을 관리한	D								I		C	O	I	O	M				
A4: 공급자를 유지관리	D	O		C	O	O	I	M							M	O	I	I	
A5: 자재 예산을 준비한	A								I		C			I	M			I	O

그림 2-25 : 활동/개념 매트릭스 윈도우

④ ABC 매트릭스(matrix) 윈도우

ABC 매트릭스 윈도우는 활성화된 ABC모델에 있는 모든 계정과목(account)과, 드라이버(driver), 할당경로를 보여준다. 매트릭스는 각기 다른 계정(account)형태를 보여주는데, 특히 detail view 옵션을 선택할 경우 각 destination에 할당된 비율 이나 값을 보여준다.



Legend: U = Unspecified, L = General Ledger, R = Resources, A = Activities, CO = Cost Objects, D = Drivers	Destinations	R: 공급자 품질보증
Sources		
L: CS_유지보수비용	EVENLY ASSIG..	
L: CS_장비료	사원의 수	(2.0) 4.3%
L: CS_통신유지비	사원의 수	(2.0) 4.3%
L: 급료_구매	사원의 수	(2.0) 33.3%
L: 급료_법률	사원의 수	(2.0) 4.3%
L: 급료_자재	사원의 수	
L: 수당	사원의 수	(2.0) 18.2%
L: 임대료	사원의 수	(2.0) 4.3%
L: 전화세_구매	PERCENTAGE	15.0%

그림 2-26 : ABC 매트릭스 윈도우

이 교재에서 우리는 각 윈도우를 작성하여 사용해 보도록 하자. 우선 모델에서 활동과 concept들을 추가해보자.

2.3.3 활동(Activity) 및 개념(Concept)의 생성과 서식화

■ 활동 및 개념의 정의 순서.

① 기능을 지원하는 개체를 추출한다.
② 기능에 관련된 개체가 수행하는 역할을 확인한다.(입력, 제어, 출력, 메커니즘)
③ 다음 레벨의 기능과 관련된 상세화 된 개체 확인한다.
④ 같은 레벨에서의 개체의 관련성을 검토한다.

■ 활동이란 무엇인가?

활동은 실세계의 기능(function)을 나타낸다. 활동모델에서 활동은 박스형태로 나타나고 동사구에 의해서 기술되어진다. 활동들은 보통 생성물 혹은 생성물의 집합을 생산하는데 요구되는 자원(입력, 제어, 자원)들을 가지고 있다. IDEFØ 방법을 적용한 다이아그램에서 각 활동은 적어도 하나의 제어(control)를 가지고 있어야 한다.
모든 조직에는 수행되어야 하는 활동들이 있으며 이번 단계에서는 조직에서 수행되는 의사결정, 행동, 활동을 추출한다. 이 때, 활동을 규정하기 위한 이름을 결정해야 하는데 활동을 지칭하는 일반적인 의미를 사용해야 하며, 우리가 공통적으로 사용하는 내용을 중시해서 명시한다. 이러한 활동은 다음과 같은 일련의 순서를 통해서 이루어진다.

■ 대상활동 정의 순서

① 의사결정, 행동, 활동을 추출한다.
② 활동의 이름을 결정한다.
③ 활동의 목록을 만든다.(비슷한 이름끼리, 같은 개체와 관련된 것 끼리)
④ 검토사이클로 검증한다.
⑤ 대상이 될만한 개체를 확인한다.
⑥ 개체와 관련된 정보를 추출한다.
⑦ 핵심활동과 관련된 개체(ICOM)의 이름을 지정한다. 명사나 명사구로 표현된다.
⑧ 리스트를 작성한다.(종류별, 관련부문별, 사용되는 관계별)
⑨ 검토 사이클을 통한 검증을 수행한다.

리포지토리와 모델을 생성하였으면 활동을 다이아그램에 추가하기위한 준비가 된 것이다. 활동 또는 개념을 다이아그램에 추가하는 것은 그 엘리먼트(element, 기능이나 개념 등)에 대한 특별한 사건을 정의하게 한다. AIØWIN에서 한 사건은 두 가지의 중요한 특성에 의해서 구별된다. 한 엘리먼트가 발생하는(예를 들면 개념과 활동의 그룹, 그것들의 활동,

둘 사이에 존재하는 제약) 모델의 상황배경, 그리고 엘리먼트의 코스팅(costing) 정보가 그것이다.

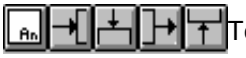
AIØWIN의 아키텍쳐는 활동모델의 구축을 위해 두 가지의 선택조항을 지원한다. 모델 작성에 필요한 모든 엘리먼트를 우선 수집함으로써 하향전개(top down) 방식의 작업을 할 수 있다. 집을 짓는 것과 같이, 특정용도에 맞게 재료들을 자르기 전에(2x4s, 4x6s, 4x8s, 등) 필요로 되는 재료를 모으는 것이다.

물론 직접적으로 다이아그램에 엘리먼트를 추가하는 것과 같이 사건을 직접 정의하는 상향식(bottom up) 방식으로 할 수도 있다. 집 짓는 것과는 달리 AIØWIN은 풀(pool)에서 이에 맞는 조각들의 치수를 입력함으로써 프로젝트 전기간을 통하여 이들의 재사용이 가능하다. 매번 사용할 때마다 새로운 사건에 맞게 조각들을 수정할 수 있다. 이러한 방법으로 실제 비즈니스 시스템을 보다 정확하게 나타낼 수 있는 모델을 작성하는 능력을 촉진함으로써, AIØWIN는 모델링 활동을 보다 능률적이고 효과적으로 진행할 수 있도록 지원한다.

다음 목표는 모델을 생성하고 모델에 엘리먼트를 추가하는 것이다. 엘리먼트와 관련된 특성(엘리먼트의 이름, 작성자, 문서화를 위한 설명, 노트, 엘리먼트를 위한 소스) 역시 모델에 있어서 엘리먼트의 사건과 관련이 있다.
프로젝트를 수행하는데 있어서 모델과 함께 목적이나 배경을 서술적으로 표현하는 것은 하나의 척도로서 같이 작업하는 팀원들에게 도움이 될 것이다. 작업의 서류화는 역시 후에 작업을 참조할 다른 모델작업자에게 정보를 제공하게 된다.

■ 활동의 추가

활동은 실제 세상에서 일어나는 행동, 기능, 운영 등을 나타낸다. 이들은 활동의 이름 및 다이아그램에서의 활동순서 뿐 아니라 활동의 상위다이아그램을 정의하는 숫자가 표시되는 상자형태로 나타내진다. A0활동은 모델에서 최상위 레벨을 표시하는 활동을 나타내며 전체 모델을 정의한다. 활동이름은 동사구로 붙여져야 한다. 활동은 한 모델에서 단 한번만 발생한다. 각 다이아그램은 세 개에서 여섯 개의 활동으로 구성된다.
AIØWIN의 다이아그램 툴 박스를 이용하여 활동을 추가하게 된다. 엘리먼트는 툴 바의 버튼, 메뉴, 그리고 키보드를 이용하여 추가할 수 있다. 툴 박스를 이용한 활동의 추가는 활성화된 Quick Detailing 설정에 따라 구분된다.
Quick Detailing이 활성화(툴 바에서  을 누름)되었을 때, 사용자 프롬프트 없이 엘리먼트에 이름을 붙이는 모드로 모델의 새로운 활동과 개념을 추가할 수 있다(AIØWIN은 엘리먼트의 기본 이름을 할당할 것이다).

지금 Diagram Tool box로 활동을 추가하여 보자.  Tool box button은 Toolbar button과 같이 동일한 작업을 수행한다.

① 활동을 추가하기 위해서 Tool box에서 **Add Activity**를 클릭한다. Activities 대화창에서 **자재 조달을 관리한다** 를 등록한 후 **Create New**를 클릭한다
② Activities 대화창을 빠져 나오기 위해서 **OK**를 클릭한다. 이때 활동이 모델에 추가된 것을 나타내기 위해 스크린에 표현될 것이다.

■ 풀(Pool)이란 무엇인가?

활동과 개념들은 풀 안에 저장되어 있다. 풀은 프로젝트를 통하여 재사용과 분배를 위해 엘리먼트들(예를 들면 - 활동, 개념, 소스, 노트 등)을 저장하고 있는 프로젝트 레벨의 리포지토리를 말한다. 풀의 형태는 모든 활동을 수집하고 몇 개의 팀이 구축된 시스템의 엘리먼트를 확인하는 그룹토의(brainstorming) 기능을 촉진함으로써 하향방식(Top-Down, 예를 들면, 다이아그램 구축에 소요되는 모든 활동과 개념을 종합적으로 수집하여 프로젝트를 시작할 수 있다)의 프로젝트 구축을 위한 기능을 제공한다.

Activity Pool에서 Activity를 추가하기 위하여 다음과 같은 작업을 한다.

① Pool menu에서 Activity...를 선택한다.
② Activity Pool 대화창의 Name field에 **자재를 주문한다**를 등록한 후 **Create New**를 클릭하거나 **Enter**를 친다. Activities list에 활동이 추가될 것이다. 마찬가지로 같은 절차로 다음의 활동들을 추가한다.

공급자를 유지 관리한다 **자재요청서를 처리한다**
자재 보급을 관리한다 **자재 예산을 준비한다**

③ **Done**을 클릭한다.

■ 개념의 추가

이 절에서는 개념이 무엇인지를 정의하고 모델의 활동에 개념을 추가하는 방법에 대하여 알아본다. 개념은 활동에 의해서 사용되거나 생성되는 개체와 정보를 말한다. 이들은 화살표를 가진 선으로 나타나고, 이름이 부여되며 상위다이아그램에서 그것들이 사용됐느냐에 따라서 ICOM중 하나의 부호가 부여된다. 상위활동에 관여된 각 개념은 분해다이아그램에서 자동적으로 나타난다. 개념은 입력(Input), 제어(Control), 출력(Output), 메커니즘 (Mechanism) 과 같이 활동에 의해서 사용되는 개념의 각각의 용도에 근거하여 활동에 추가된다.

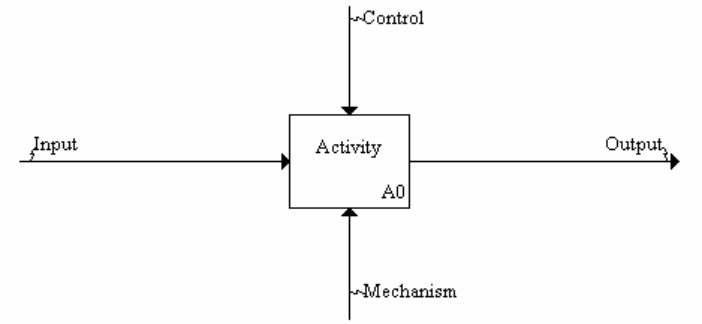


그림 2-27 : ICOM

입력은 활동에 의해서 변형되거나 소모되는 것이다. 입력은 스크린의 왼쪽편의 화살로 표시된다.
제어는 활동을 통제, 제어하는 자원이며, 활동박스 위에 붙은 화살표로 표시된다.
출력은 활동에 의하여 사용된 자원(입력, 제어, 메커니즘)에 의하여 생산되며, 활동박스 오른쪽의 화살표로 표시된다.
메커니즘은 사람 및 기계 등의 수단을 말하며, 활동상자 아래에 표시된다.

■ Activity에서 Concept의 직접추가

다시 한번 모델에 개념의 추가를 위해 몇 개의 선택사항(예를 들면, 메뉴, 툴 바, 키보드 명령 등)이 제시된다. 우선 Add Concept 대화창을 이용하여 활동박스에 개념을 직접 추가하자.

① Model Window에서 **자재조달을 관리한다** 를 선택한다.
② Input을 추가하기 위하여 toolbox 안에서 **Add Input**을 클릭한다. 나타난 Add Concepts 대화창에 **자재 요청서**를 처넣고 **Create New**를 클릭한다.

③ **자재 요청서** 를 선택하여 **Add Concepts**를 클릭한다. 선택된 input은 자동적으로 활동에 추가된다.

④ Add Concepts 대화창을 닫기 위해 **OK**를 클릭한다.

⑤ 우선 Option menu에서 *Quick Detailing…*을 선택하여 다음의 선택된 부분이 활성화 시킨다.

Quick Detailing

Prompt for Name on Create

⑥ Quick Detailing Options 대화창을 닫기 위해 **OK**를 클릭한다. Toolbar에  가 활성화 되었는지 확인한다. 활동에 개념들을 추가하기 위해 Diagram Toolbox또는 toolbar button를 클릭하였을 때 Concept Name 대화창이 보일 것이다.

⑦ **자재조달을 관리한다**가 선택된 상태에서 control을 추가하기 위해 toolbox에서 **Add control**을 클릭한다. 나타난 Concept Name 대화창에 **구매 절차**를 쳐넣고 **OK**를 클릭한다.

⑧ Output을 추가하기 위해 toolbox에서 **Add Output**을 클릭한다. 나타난 Concept Name 대화창에 **구매 주문서**를 쳐넣고 **OK**를 클릭한다.

⑨ 마지막으로 Mechanism을 추가한다. Mechanism을 추가하기 위해 toolbox에서 **Add Mechanism**을 클릭한다. Concept Name 대화창에 **데이터 시스템**을 쳐넣고 **OK**를 클릭한다.

모델 창에서 현재 활동이 그것에 첨부된 4개의 개념을 갖고 있음을 주목하자.

■ 개념 풀에서 개념의 추가

여러분은 개념 풀에 개념을 저장하고, 차후에 활동에 개념을 할당하기를 원하게 될지도 모른다.

① Pool menu에서 *Concept…*을 선택한다.

② Concept 대화창의 Name field안에 다음의 concept들을 등록한 후 **Create New**를 클릭하거나 **<Enter>**를 친다. concept들은 대화창 안의 Concept 리스트에 추가될 것이다.

공급자 입찰서	**공급자 견적요청서**	**자재 부서**
자재업무 수행 데이터	**선적서류**	**구매 부서**
공급자 관리정책	**대금 지급 승인서**	

③ Model window로 돌아가기 위해 **Done**을 클릭한다

활동들과 개념들이 관련된 풀에 추가되면 다음 레벨은 모델에 이들 엘리먼트들을 추가하는 것이다. 풀에 이미 존재하는 개념들을 추가하려면 우선 ***Quick Detailing…***을 비활성화 시켜야 한다.

■ 풀로부터 모델에 개념을 추가한다.

① Toolbar에서  를 **toggling**함으로써 Quick Detailing을 비활성화 시킨다. 그리고 활동의 추가를 시작한다.(버튼이 튀어나온 상태)
② Model 윈도우에서 **자재 조달을 관리한다** 를 선택한다.
③ Input을 추가하기 위해 toolbox에서 **Add Input**을 클릭한다. Add Concepts창이 나타나면 <Ctrl>+click을 사용하여 Concept list field에서 다음의 concept들을 한꺼번에 선택한다.

선적서류 **공급자 입찰서**

④ **Add Concepts**를 클릭한다. List로부터 concept들이 살아지는데 이는 concept들이 model에 추가되는 것을 나타낸다. 대화창은 계속 열려있고 model에 더 많은 concept들의 추가를 허용할 것이다.
⑤ Type box에서 **Control** radio button을 활성화한다. 그리고 Concepts list field에서 **공급자 관리 정책** 을 선택한다.
⑥ **Add Concepts**를 클릭한 다음 output을 추가한다.
⑦ Type box 안에 **Output** radio button을 활성화한다. 그리고 Concept list field에서 다음의 concept들을 다중선택하기 위해 <Ctrl>+click한다.

자재업무 수행 데이터 **공급자 견적요청서** **대금지급 승인서**

⑧ **Add Concepts**를 클릭한다. 끝으로 mechanism들을 추가한다.
⑨ **Mechanism** radio button을 활성화한다. 그리고 Concept list field에서 다음의 concept들을 한번에 선택하기 위해 <Ctrl>+click한다.

자재 부서 **구매 부서**

⑩ **Add Concepts**를 클릭한다
⑪ Add Concepts 대화창을 빠져나가 Model Window로 돌아가기 위해 **OK**를 클릭한다.

concept들은 Model Window에서 활동에 추가된다.

모델에 활동과 개념들을 추가하면 다이아그램은 다음과 같이 나타난다.

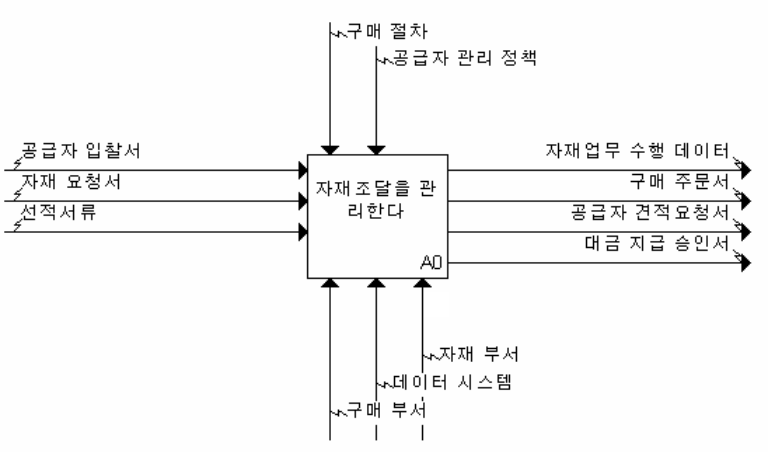


그림 2-28 : '자재조달을 관리한다' 의 Context 다이아그램

2.3.4 계층구조를 만든다. – 활동의 분해와 개념의 이전

■ 분해(Decomposition)란 무엇인가?

분해는 각 활동의 더 세부적인 조감도 이다. 분해다이아그램은 상위(모)활동에서 일어나는 하위 또는 자 활동들을 포함한다. 하나의 활동은 단 하나의 분해를 갖는다.

모델은 계층적 구조로 구성되어 있는데, A0레벨에서부터 A0레벨에 관한 보다 많은 정보를 제공하는 분해(decomposition)다이아그램의 연속으로 이루어져 있다. 모델윈도우는 모델계층에서 단 하나의 다이아그램을 보여준다. 이 절에서는 A0레벨 활동을 구성하는 하위 활동들을 생성하게 된다. 분해다이아그램을 생성함으로써 다양한 다이아그램을 “page through” 하게 된다. AIØWIN에서 개념들은 활동의 분해와 같이 여러 개의 세부적인 개념들로 분해될 수 있다. 또한 하위레벨의 여러 개의 유사개념이 상위레벨의 하나의 개념으로 모아질 수 있도록 하는 표현도 지원된다. 상위개념이 여러 개의 세부 개념들을 포함한다는 것을 보여주기 위해 몇 개의 개념들을 하나의 개념으로 묶을(bundle) 수 있다. 다음 단계로는 각 활동과 관련되어진 요소(개념)들을 할당하는 것이다.

우리의 경험으로 보아 IDEFø 모델을 작성함에 있어서 모델의 계층적 분할에 따르는 수치적 제약은 모델의 일관성 및 판독성을 증진한다. 즉 하나의 기능이 다음 단계에서 수십 개의 기능으로 분해되어 표현 된다든지, 하나의 기능에 수십 개의 화살표(Concept)가 그려진다는 것은 흡사 전자부품의 회로도 같아 모델의 판독성을 감소시킴은 물론 모델 자체의 추상화 단계 설정이 부적절 함을 반영하는 것이다. 그렇다고 하나의 기능이 계속하여 두개의 기능으로 분해되는 것도 필요 없이 너무 많은 계층적 구조만 형성할 뿐이므로 피해야 한다. 특히 모델링작업 팀과 같이 다수의 멤버에 의해 추진되는 모델화 작업에 있어서는 이를 제한 하기 위한 약속이 선행되어야 하는데 이는 퍼즐과 같이 멤버 각자가 개발한 모델이 통합 될 때 참으로 중요한 문제로 부각된다. 따라서 이러한 약속의 예로서 하나의 기능이 분해 될 수 있는 범위(예를 들어 3 개 - 6 개)와 하나의 기능에 연결 될 수 있는 화살표의 수적 제한(예를 들어 - 동일한 개념은 6 개 이하)을 미리 설정하는 것이 중요하다. 이러한 작업은 아래와 같은 순서로 진행된다.

■ 계층구조작업 순서

① 활동을 계층구조 안에 수집한다.
② 같은 개체를 대상으로 하는 활동을 수집한다.
③ 가능하면 계층구조를 지양한다.
④ 활동 그룹의 이름을 정한다(필요하다면)

⑤ 하나의 그룹 안에 적어도 활동이 3개 이상 6개 이하가 되도록 노력한다.
⑥ 그룹별로 생략된 멤버를 확인한다.(가능하다면)

계층구조에는 하나의 활동을 이루는 부분으로서의 하위 활동들이 있는 한편 하나의 활동을 종류별로 분류가 가능한 경우로 나눌 수 있다. 우리는 이와 관련된 활동의 계층구조에 있어서 후자의 경우를 지양하고자 다음과 같은 사항을 검토한다.

■ 종류의 계층구조를 부분의 계층구조로 전환

① 이름과 관련된 것을 결정한다.
② 용어를 통일한다.
③ 다이아그램을 단순화 한다.
④ 생략된 활동을 확인하는 모델작업자를 지원한다.
⑤ 대표되는 분류명 혹은 구성을 위한 이름을 작성한다.
⑥ 전문가와 함께 검증한다.

■ 분해(Decomposition) 윈도우에서 세부 활동의 추가

다음 단계에서는 분해다이아그램을 추가하여 A0활동에 관한 보다 세부적인 사항을 추가한다. Title bar는 분해의 이름과 함께 상위(모) 활동의 번호와 이름을 보여준다. 다이아그램에 필수 활동수량(IDEF0방법에 따라 3~6개 사이)을 추가하기 전까지는 그것은 임의의 위치에 나타내어진다. 분해윈도우는 왼쪽에서 오른쪽으로 숫자가 부여된 활동에 따라 계단형태로 보여질 것이다. 활동들은 AIØWIN의 Drag-and-Drop 기능을 사용하여 모델윈도우 내에서 이동이 가능하다.
상위활동에 붙어있는 개념들은 자동적으로 분해다이아그램에 나타나게 된다. 그러나 어떤 활동에도 이들 개념들이 붙어있지는 않을 것이다. 하위활동에 이 개념들 중에 하나가 첨부되는 것은 개념이전(migrated concept)을 생성한다. 그래서 상위활동의 다양한 수행(개념의 다른 조합은 상위활동에 첨부되어 사용한다.)을 전달하고 배분하게 된다.

■ Decomposition에서 활동의 추가와 배치

① Toolbar에서 **자재 조달을 관리한다** activity을 선택하고  를 클릭한다. AIØWIN은 즉시 새 다이아그램을 위해 새로운 Model Window를 열 것이다.
② Diagram Toolbox에서 **Add Activity**를 클릭한다. 그리고 Activities 대화창에서 Activity Pool로부터 다음의 활동을 다중 선택한다.

공급자를 유지 관리한다	**자재를 주문한다**	**자재를 요청한다**
자재 보급을 관리한다	**자재 예산을 준비한다**	

③ Model Window로 돌아가기 위해 **OK**를 클릭한다.

Modeling 연습에서 활동을 재배치하기 위해서는 drag-and drop 기능을 사용한다.

④ 아래 나열된 각 활동을 선택한다. 그리고 지정된 위치에 그것을 잡아서  커서가 보일 때 끌어다 놓는다. (A1은 맨 왼쪽에, A5는 맨 오른쪽에)

자재 요청서를 처리한다, A1	**자재를 주문한다**, A2	**자재 보급을 관리한다**, A3
공급자를 유지 관리한다, A4	**자재 예산을 준비한다**, A5	

■ 활동에 개념을 추가하기

A0활동에 붙여진 개념들이 분해레벨에서 자동적으로 보여진다는 것에 유의한다. 그러나 이들 개념들은 어떠한 활동에도 붙어있지 않다. 이 개념들은 필수적으로 상위 다이아그램과 같은 형식의 개념으로 나타난다. – 괄호가 상위다이아그램과 분해 다이아그램 내부의 끝에 위치하고 있음을 주목하라. 분해다이아그램내의 활동들에 이 개념들을 붙임으로써, 상위다이아그램에서 하위다이아그램으로 개념의 이전을 정의할 것이다.

다른 AIØWIN 기능과 같이 활동에 개념들을 붙이기 위하여는 몇 개의 선택조항이 있다. 활동에 개념들을 신속히 붙이기 위해서는 우선 Toolbar Button ICOM을 toggling하여 개념의 형태를 정의하여야 하고, 개념을 선택한다. 그리고 대상 활동에 <Shift>+click을 한다.
A/C matrix Window는 프로젝트에서 활동과 개념사이의 관계를 강조하는 모델에서 또 하나의 view를 제공한다. Model Window와 같이 A/C Matrix Window는 활성 모델의 한 레벨만을 보여준다. 그러나  혹은  을 클릭하면서 다른 레벨들 간의 이동을 할 수 있다.

A/C Matrix Window에서 활성화된 다이아그램의 모든 세부적인 개념들이 수평축에 나열되어 있다. 활성화된 다이아그램의 세부 활동들은 수직 축에 나열되어 있다. 각 개념들의 바로 아래 cell은 그것들이 상위다이아그램에서 어떻게 사용되었는지를 알려준다.(ICOM으로) 그래서 이 Window는 상위다이아그램에서 사용하는 것과 각 개념의 지역적 사용을 구별하게 한다

A/C matrix Window의 이점은 개념을 활동에 손쉽게 붙일 수 있다는 것이다 개념과 활동 사이에 상호 관계를 표시하는 matrix cell을 클릭함으로써 활동과 개념을 신속히 붙일 수

있다. 개념이 활동에 붙여지면 개념의 머리문자(예를 들면 - Control의 표시로 "C") 가 해당 Matrix Cell에 나타날 것이다. 글자 색은 matrix legend에 표시된 색 규정과 같게 될 것이다.

■ Model Window에서 활동에 개념 붙이기

① Toolbar에서 button 이 "C"로 바뀔 때까지  를 토글한다. Button은 4가지의 다른 concept type으로 순환하며 바뀐다.

② **공급자 관리 정책** control을 선택한다. 그리고 **자재 요청서를 처리한다** 활동을 <Shift>+click 한다. concept은 이 활동에 control로 붙을 것이다.

③ **공급자 관리 정책** control을 **자재를 주문한다** 와 **공급자를 유지 관리한다** activity에 control로 <Shift>+click하여 붙인다. Decomposition 다이아그램에서 **구매 절차** control을 activity에 붙여보자.

④ **구매 절차**를 선택한다. 그리고 다음의 활동들 위에<Shift>+click한다.

자재 요청서를 처리한다. **자재를 주문한다**

자재 보급을 관리한다 **자재 예산을 준비한다**

작업이 완료되면 model은 다음 그림과 같이 나타날 것이다

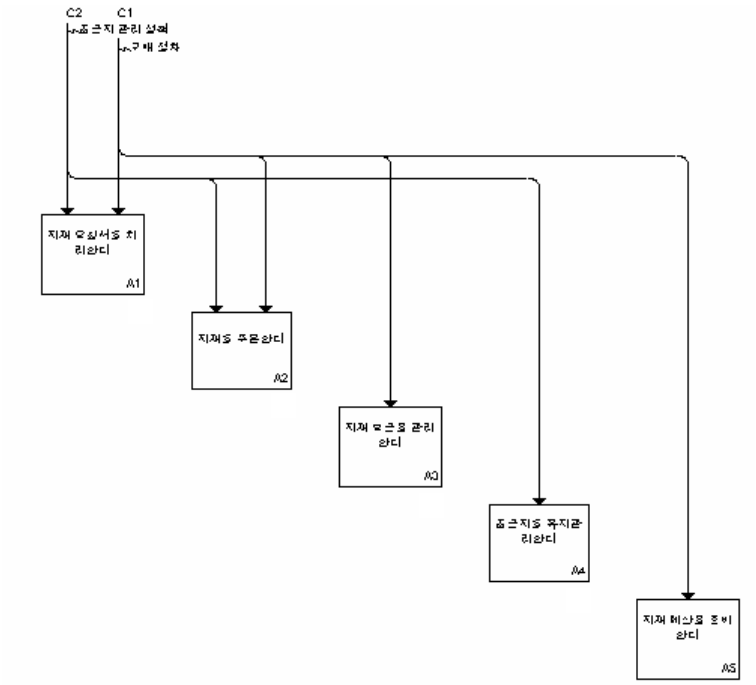


그림 2-29 : Control이 추가된 모델

■ Activity/Concept Window에서 Concept을 활동에 붙이기

Activity/Concept Matrix Window에서 concept을 활동에 붙이는 것을 끝내자. 우선 Input을 붙여보자

① A/C Matrix Window을 열고 Toolbar에서  를 클릭한다

② 붙일 concept의 형태를 정해기 위해 Toolbar에서  를 클릭한다

③ 아래 표에 보여지는 concept과 activity 사이의 교차점을 표시할 matrix cell을 클릭한다

Activities	Inputs
A1 자재 요청서를 처리한다	**자재 요청서**
A3 자재 보급을 관리한다	**선적서류**
A4 공급자를 유지 관리한다	**공급자 입찰서**

다음 mechanism을 붙여보자.

④ Ribbon toolbar에서  을 클릭한다. 그것들 사이의 교차점을 표시하는 matrix cell을 클릭하여 다음의 mechanism들을 아래 활동에 추가한다

Activities	Mechanisms
A1 자재 요청서를 처리한다	**데이터 시스템**
	자재 부서
A2 자재를 주문한다	**데이터 시스템**
	구매 부서
A3 자재보급을 관리한다	**데이터 시스템**
	자재 부서
A4 공급자를 유지 관리한다	**데이터 시스템**
	구매 부서
A5 자재예산을 준비한다	**데이터 시스템**
	자재 부서

이제 output을 추가하여 보자.

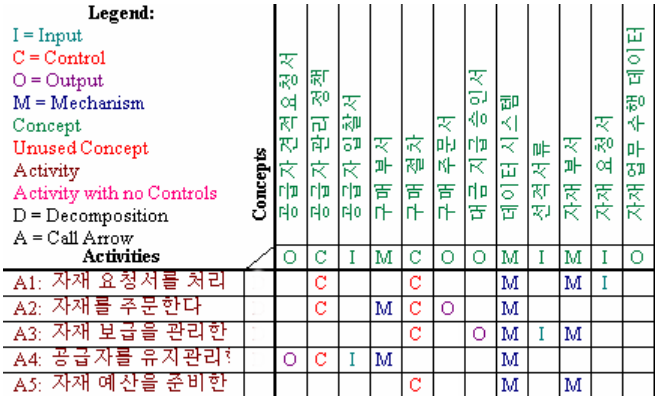
⑤ Toolbar에서  를 클릭하고 아래 활동에 다음 output을 추가하여 보자

Activities	Outputs
A2 자재를 주문한다	**구매 주문서**

A3 자재 보급을 관리한다 **대금지급 승인서**

A4 공급자를 유지 관리한다 **공급자 견적요청서**

작업이 끝나면 A/C Matrix는 다음과 같이 보여질 것이다.



■ 다이아그램 정교화

지금까지는 단순히 분해다이아그램의 활동들에 상위레벨로부터 이전된 개념들 만을 붙여왔다. 추가로 우리가 붙여왔던 개념들은, 활성화된 다이아그램의 범위밖에 또 다른 활동에 의해서 생성되거나 사용된 것을 의미하는 경계조건 외부의(boundary) 개념(단지 한쪽만 붙어있는 끝을 가진 개념) 들이다. 또한 기능모델은 활동들 사이에 존재하는 제약조건 들을 나타낼 수 있기 때문에 다이아그램의 내부개념(internal concept - 양끝이 붙어있는 개념)들을 추가할 필요가 있다.

다이아그램작업을 계속하기 위해 Model Window로 돌아가자.  을 클릭하여 Model Window로 돌아간다. Model을 정교화하기 위해 다이아그램에 더 많은 output의 추가를 시작한다. Quick Detailing은 여전히 비활성화 되어있을 것이다.

■ Model에 Output의 추가

① A1 **자재 요청서를 처리한다** 활동을 선택하고 Diagram Toolbox에서 **Add Output**을 클릭한다.

② Add Concept 대화창이 나타나면 Name field에 **요청서 부하량** 를 입력하고 **Create**를 클릭한다. 목록에 새로운 concept이 나타날 것이다.

③ Concept이 선택된 동안에 **Add Concept**을 클릭한다. 활동에 output이 추가될 것이다.

④ 같은 방법으로(대화창이 열린 상태에서) 아래 concept을 output으로 생성하여 붙인다.

신규 공급자 선정요청서 **검증된 요청서**

⑤ 대화창을 닫기 위해 **OK**를 클릭한다. concept들은 Model Window에 output으로 Process Material Requests에 붙인 상태로 나타난다.

⑥ 같은 방법으로 아래 활동들의 다음 output을 추가한다.

Activities	Outputs
A2 자재를 주문한다	**구매 주문서 오류**
	새로운 공급자 요청사항
A3 자재 보급을 관리한다	**납기 차이**
A4 공급자를 유지 관리한다	**새로운 공급자 평가서**
	공급자 이력표
A5 자재 예산을 준비한다	**예산 변동사항**
	자재부문의 예산

■ Internal Concept의 정의

Model Window와 <Shift>+click방법을 이용하여 internal concept을 정의하자.

① "I"가 단추표면에 보일 때까지 Toolbar에서  를 토글한다. Input으로 concept들을 추가할 것이다.

② Output **신규 공급처 선정요청서**(자재 요청서를 처리한다로부터의 output)를 선택하고 다음 활동에 <Shift>+click한다.

자재를 주문한다 **공급자를 유지 관리한다** **자재예산을 준비한다**

각 활동 위에 <Shift>+click함과 동시에 AIØWIN은 다이아그램을 다시 그리게 할 것이다.

③ Output **검증된 요청서**(자재 요청서를 처리한다로부터의 output)를 선택하고 **자재를 주문한다**에서 <Shift>+click한다.

다이아그램에 정의한 제약 조건들에 유념한다. 자재를 주문한다 활동이 Output 신규 공급자 선정요청서를 생성할 때까지 자재를 주문한다, 공급자를 유지 관리한다, 자재예산을 준비한다 활동의 활동은 발생하지 않는다. 다시 말해서 이 각각의 활동들은 개별적으로

내포된 잠재 Output이 어떠한 개념의 조합으로부터 생겼는지를 결정하기 위해 활동의 분해를 생성하여야 한다. 분해된 활동에 붙여진 개념들은 분해다이아그램에 상속될 것이며 다이아그램에서 개념들을 활동들 사이에 다시 분배할 것이다.

자재를 주문한다 활동으로 이동하자. 그리고 output **구매 주문서** 와 **새로운 공급자 요청사항**으로부터 internal concept들을 생성하자.

① **구매 주문서**를 선택하고 다음의 활동에서 <Shift>+click한다.

자재 보급을 관리한다 **자재 예산을 준비한다**

② **새로운 공급자 요청사항** 을 선택하고 다음 활동에서 <Shift>+click한다.

자재 요청서를 처리한다 **공급자를 유지 관리한다**

다음, **자재 보급을 관리한다**로부터 output으로 생성된 **대금지급 승인서**로부터 internal concept을 생성하자.

③ **대금지급 승인서**를 선택하여 **자재 예산을 준비한다** 위에 <Shift>+click한다.

마지막으로, **공급자를 유지 관리한다**의 output **새로운 공급자 평가서**와 **공급자 이력표**로부터 internal concept을 생성한다.

④ **새로운 공급자 평가서**를 선택하고 activity **자재를 주문한다**에서 <Shift>+click한다.
⑤ **공급자 이력표**를 선택하고 **자재를 주문한다**에서 <Shift>+click한다.

internal concept을 포함하는 다이아그램의 부분은 다음 그림과 같다.

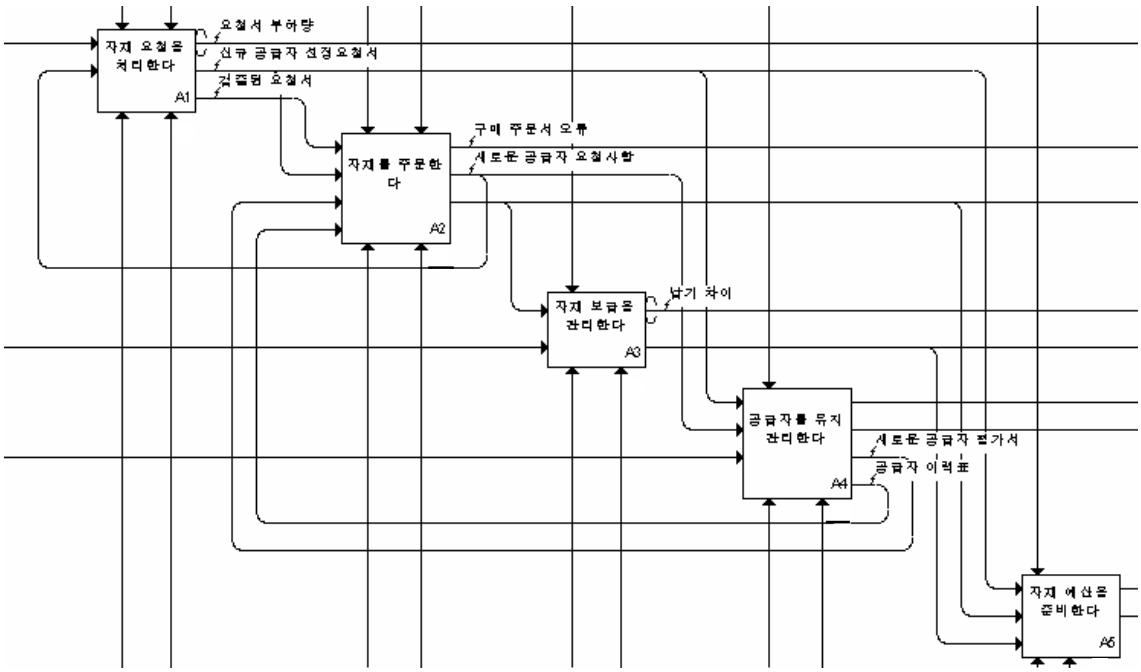


그림 2-30 : 정교화 작업으로 작성된 모델

■ 개념의 그룹화(Bundling Concepts)

다이아그램에서 아직 활동과 연결되지 않은 **자재업무 수행 데이터** concept을 가지고 있음을 인식하라. 이 Concept은 A0활동(concept의 코드에 유념하라)의 output으로 붙어 있다. 그리고 번들(bundle) 안에서 parent concept으로 그것을 사용할 것이다. AIØWIN에서, 상위 활동의 분해관계와 유사한 계층이라는 것을 보여주기 위해 concept들을 함께 번들할 수 있다. 번들링 Concept은 몇 개의 concept을 보다 큰 하나의 concept "셑"으로 그룹핑하는 것을 허용한다.

번들은 두 가지 형태로 나눌 수 있다.

① 번들의 분해(Split Bundles)는 한 concept으로부터 유래된 concept(input, control, mechanism)들을 묶는다. 이의 수행을 위해 활동의 그룹들이 동일한 input, control, mechanism을 요구하는 경우를 나타낸다.
② 번들의 결합(Join Bundles)은 하나 또는 그 이상의 활동을 떠난 output들을 묶는다. 이는 활동의 그룹이 같은 output을 생성하는 경우를 나타낸다.

■ 개념의 생략(Tunneled Concept)

다이아그램의 경계조건(context) 밖으로부터 다이아그램의 안으로 들어오거나 또는 다이아그램 경계조건 밖으로 나가는 개념들을 tunneled concept 이라 한다. 상위다이아그램에서는 사용되지 않고 다이아그램 안쪽의 하위다이아그램에서만 사용되거나 상위다이아그램에서는 사용되었지만 하위다이아그램에서는 사용되지 않는 개념들을 말한다. 턴넬링 표현방법은 concept이 활동과 연결되지 않는 끝에 있다. 즉 활동의 분해 다이아그램에 사용되지 않은 concept은 다이아그램의 경계조건 밖에 턴넬링 된다.

예에서 상위 concept로 자재업무 수행 데이터를 이용하여 output 요청서 부하량, 구매 주문서 오류, 납기 차이, 그리고 예산 변동사항을 번들링 한다. 그렇게 함으로써 자재업무 수행 데이터가 다른 concept을 포함한다고 이야기할 수 있다.

■ Bundling Concepts

① **자재업무 수행 데이터** Concept를 선택한 후 Diagram Toolbox에서 **Add Bundle**를 클릭한다.
② 나타난 Edit Bundle 대화창에서 Eligible Concept 목록에서 다음 Concept에다 <Ctrl>+clicking하여 다중 선택한다.

예산 변동사항 **납기 차이**
구매 주문서 오류 **요청서 부하량**

③ 위에 concept들을 Bundle 목록에 있는 Concepts로 이동하기 위하여 

를 클릭한다
④ 대화창을 닫기 위하여 **OK**를 클릭한다.

자재업무 수행 데이터가 선택한 네 개의 하위concept의 parent로 작용됨을 알 수 있다.

2.3.5 모델의 복사(Copying), 분해(Splitting), 병합(Merging)

■ 모델의 복사(Copy)

Model Browser window는 리포지토리에 있는 모든 모델, 이들과 연관된 다이아그램에 관련된 활동들을 보여준다. 이 과에서는 Model Browser Window를 사용하여 TO-BE 모델로 변경하기 위한 복사된 모델을 어떻게 생성하는지를 발견하게 될 것이다.

Model Browser Window은 모델의 계층을 보여준다. 모델 안에서 분해와 활동들 그리고 모델자신을 볼 수 있다. 엘리먼트를 확장하거나 없애기 위해 엘리먼트 타이틀을 레프트 클릭한다.
AIØWIN 는 세 가지의 Model Browser Window 을 제공한다.

① 노드리스트(Nodelist)
② 수평 노드트리(Horizontal Nodetree), 그리고
③ 수직 노드트리(Vertical Nodetree).

노드리스트는 다양한 다이아그램 레벨을 구별하기 위해 톱니 모양의 형태를 취한다. 필수적으로 모델 자신의 rolled-up 버전인 활동모델의 최상위 레벨은 A0활동에 대한 보다 세부적인 정보를 제공하는 다이아그램 레벨을 유지하면서 A0활동을 포함한다. 수평 또는 수직 노드트리는 리포지토리 계층을 보여주는 방법을 대체적으로 제공한다. 적당한 Toolbar button()을 클릭함으로써 신속히 View를 바꿀 수 있다.

① 자재조달 모델의 Model Browser을 열기 위해  를 클릭한다
② 자재조달 모델 부근에 있는  를 클릭한다. 그리고 마우스 버튼을 누른 상태에서 아이콘을 끌어 노드리스트의 빈 곳에 놓는다.
③ 복사할 모델을 정확하게 지시하면서  커서 가 보일 때 모델을 놓는다.

AIØWIN은 기본적으로 새로운 모델을 생성한다. 적당한 모델 엘리먼트에 인접 화살을 클릭 또는 Toolbar에  를 클릭함으로써 각 다이아그램 레벨을 확장할 수 있다. 계층이 당초의 모델과 동일하다는 것에 유의한다.

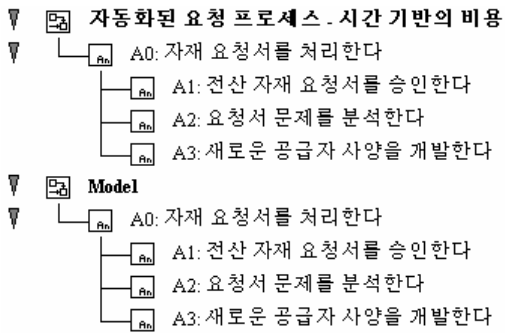


그림 2-31 : 모델브라우져 윈도우에서 모델이 복사된 모습

모델을 복사는 시스템에서 제안된 실행의 TO-BE 버전을 복사된 모델에 반영하는데 있어서 수정이 용이하도록 지원한다. 활동, 개념, 그리고 관계를 추가하거나 변경, 삭제할 수 있다. 한편 새로운 코스팅 모델에 들어가 TO-BE 실행을 위해 그 원가를 계산할 수 있다.

■ 모델의 분할(Splitting Models)

① 팀 별의 모델링 활동을 위해 모델작업 과제를 나눈다.
② 너무 큰 모델을 좀더 관리하기 쉬운 하위모델로 줄인다.
③ 하위모델을 같거나 다른 프로젝트에서 사용할 수 있도록 한다.

단일활동 또는 단일활동으로부터 유래된 다이아그램 계층으로부터 새로운 모델의 생성을 위해 모델을 분할한다. 모델을 분할할 때, 선택된 활동을 복사하고, 그것과 관련된 분해(decomposition)에 따라서 새로운 모델을 생성한다.

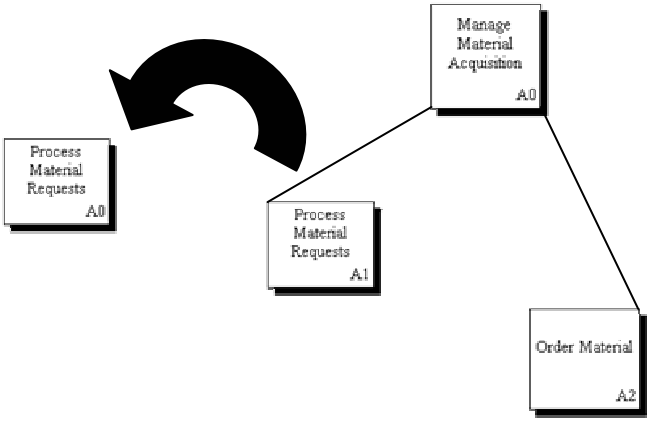


그림 2-32 : A1 활동이 분할 작업을 통해서 새로운 모델생성

■ 모델의 병합(Merging Models)

① 분리된 모델을 하나의 통합된 모델로 합친다.
② 사용자가 정의한 기준에 따라 중복된 모델의 정보가 자동적으로 수집된다.
③ 같은 혹은 다른 프로젝트로부터 모델을 병합한다.
④ 그룹의 모델링 프로젝트를 위해 적절한 협조를 지원한다.

특정한 다이아그램 레벨에서 리포지토리에 존재하는 2개의 모델을 통합하기 위해 모델들을 병합할 수 있다. 모델을 어떻게 활성화된 다이아그램 안에 병합하느냐는 세 개의 선택조항이 있다.

① A0활동과 그 분해(decomposition)요소를 추가한다.
② A0활동과 그 분해요소와 선택된 활동을 바꾼다.
③ A0활동과 그 분해요소를 선택된 활동의 분해다이아그램에 추가한다.

2.3.6 미연방표준과의 호환성(FIPS Compliance)

KBSI는 FIPS표준의 개발에 통합된다. 그리고 AIØWIN는 기능성과 표시에서 포괄적이고 자동적인 FIPS compliance을 제공한다.

■ AIØWIN는 다음 사항을 제공한다.

① Non-compliance를 방지하기 위해 자동적인 배경 점검
② Kit forms
③ 활동 box 번호부여
④ Decomposition identifier

AIØWIN는 활동박스 번호부여와 decomposition identifiers의 FIPS 표시를 선택할 수 있도록 지원한다.

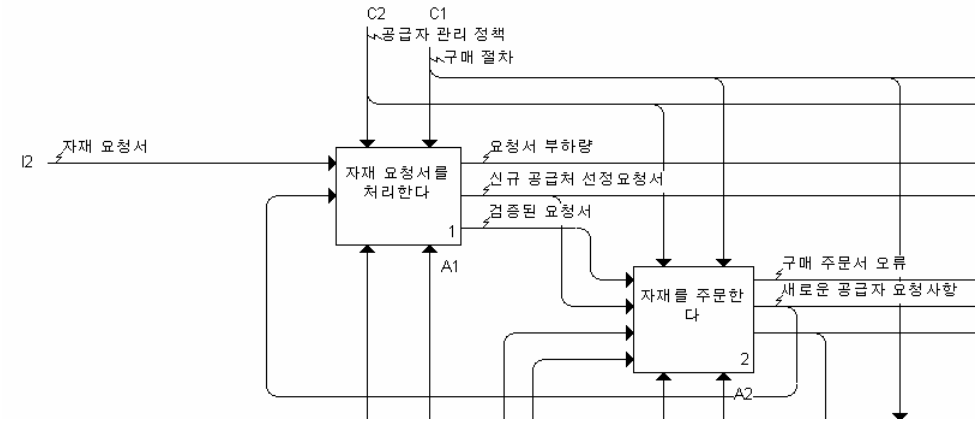


그림 2-33 : FIPS 사양 선택에 따른 모델의 변화

2.4 IDEFø 모델링 지침

2.4.1 모델링 절차

모델 개발 프로세스는 배경, 관점, 목적을 확립하고 개선되고, 모델화 되어야 할 분야에 관한 정보를 수집하고, 계층구조에 있어서 상위 혹은 하위방향으로 정제하는 많은 단계로 이루어져 있다.

■ 경계와 관점, 목적을 확립한다.

① 경계를 설정한다.
- 무엇이 모델 안에 혹은 밖에 있는지를 결정한다.
- A-0 를 결정한다.

② 무엇을 나타내고 무엇을 나타내지 말 것인가, 누구의 관점에서 작성되는가를 결정한다.
③ 무엇이 모델작업 완료의 기준인가를 결정한다.
- 어떠한 의사결정이 되어야 하는가를 정한다.

여러분은 한번의 작업으로 이러한 모든 것을 완벽하게 끝낼 수는 없을 것이다. 여러분은 이러한 모델을 작성하는 과정에서 이러한 것들을 정련할 수 있을 것이다.

■ 정보와 자료들을 수집한다.

① 출처와 적합한 검토자를 밝혀낸다.
② 이해 관계자들을 밝혀낸다.
③ 인터뷰를 실시한다.
- 조직 내부의 모든 계층을 포함한다.
- 경청한다.
- 자세한 노트를 남긴다.
- 가능한 한 많이 수집한다.

④ 수집된 자료를 구성한다.
⑤ 작성자-검토자의 검토 사이클을 진행한다.

■ 모델 작성자 – 검토자 사이클

모델을 완료 시키기 위하여는 모델 작업자와 검토자 사이에 여러 번에 걸친 자료의 교환이

필요하다. 이러한 자료키트는 모델작업자에게는 작업의 진척사항을 문서화하기 위하여, 검토자에게는 그들의 의견사항을 기록하기 위하여, 또한 모델작업자는 검토자의 의견사항을 모델에 반영시키기 위해 사용한다.

■ 대상이 될 활동을 추출한다.

① 의사결정, 행동, 활동을 추출한다.
   - 모든 조직에는 수행되어야 하는 활동들이 있다.
② 활동의 이름을 신중하게 결정한다.
   - 일반적인 의미를 사용한다.
   - 우리가 공통적으로 사용하는 내용을 중시한다.
   - 학문적인 것 보다는 실질적인 이름을 사용한다.
③ 목록을 만든다.
   - 비슷한 이름끼리 리스트 한다.
   - 같은 개체와 관련된 것 들을 리스트 한다.
④ 검토 사이클로 검증한다.

■ 대상이 될만한 개체를 추출한다.

① 관련된 개체를 추출한다.
② 중요활동과 관련된 많은 개체의 이름을 지정한다.
   - 서술적인 용어는 일반적인 이름으로 변환될 필요가 있다.
   - 명사 혹은 명사를 사용한다.
   - 서술되는 단어의 상태를 주의하라.
③ 리스트를 작성한다.
   - 종류별로 작성한다.
   - 관련된 부문별로 작성한다.
   - 사용되는 관계별로 작성한다.
④ 검토 사이클을 통한 검증을 한다.

■ 기능의 계층구조를 작성한다.

① 기능을 계층구조 안에 수집한다.
② 같은 개체를 대상으로 하는 기능을 수집한다.
③ 가능하면 같은 종류별로 이루어진 계층구조를 지양한다.
④ 기능그룹의 이름을 정한다(필요하다면).

⑤ 하나의 그룹 안에 적어도 기능이 세 개 이상 여섯 개 이하가 되도록 노력한다.
⑥ 그룹별로 생략된 멤버를 확인한다(가능하다면).

■ 부분을 이루는 계층과 종류를 이루어는 계층을 수립한다.

① 이름과 관련된 것을 결정한다.
② 용어를 통일한다.
③ 다이아그램을 단순화한다.
④ 생략된 기능을 확인하는 모델작업자를 지원한다
⑤ 대표성을 가진 혹은 복합적인 이름을 작성한다.
⑥ 전문가와 함께 검증한다.

■ 개체들을 정의한다.

① 기능과 관련된 개체들을 정의한다.
② 기능에 관련된 개체가 수행하는 역할의 정의한다.
- 입력(Input)
- 출력(Output)
- 제어(Control)
- 메커니즘(Mechanism)

③ 상세화된 하위레벨에 관련된 개체를 검토한다.
④ 같은 레벨에서의 개체간의 관련성을 검토한다.

■ 다이아그램을 구성한다.

① 현재의 구성관계로부터 어떤 다이아그램을 작성할 수 있는가?
② 불완전하거나 모순되거나 맥락이 맞지 않는 사항을 찾는다.
③ 관계가 생략된 키를 분석한다.
④ 원본 자료로부터 가능한 최선의 스토리를 완성한다.
⑤ 전문가와 검토한다.

■ 상위방향과 하위방향의 정렬을 실시한다.

① 계층구조 안에서 다이아그램을 분류한다.
② 인터페이스의 일관성을 검토한다.
③ 경계조건은 명확하게 정의되었는가 검토한다.

- 이 때 상위방향으로 재검토 한다.

④ 모델의 목적에서 요구되는 정보가 세부적인 사항에 포함되고 있는가를 검토한다.

- 이 때 하위방향으로 재검토 한다.

⑤ 전문가와 검토한다.

■ 모델을 병합한다.

① 분리된 모델을 하나의 통합된 모델로 합친다.
② 이 때 사용자가 정의한 기준에 따라 중복된 모델의 정보가 자동적으로 수정된다.
③ 같은 혹은 다른 프로젝트로부터 모델을 병합한다.
④ 그룹의 모델링 프로젝트를 위해 적절한 협조를 제공한다.

■ 모델을 분할한다.

① 팀별의 모델링 활동을 위해 모델화 할 작업과제를 나눈다.
② 너무 큰 모델을 좀더 관리하기 쉬운 범위의 모델로 줄인다.
③ 하위모델을 동일 프로젝트나 다른 프로젝트에서 사용할 수 있도록 한다.

■ 결과를 적용한다.

① 모델사용자 그리고 이해당사자와 함께 정기적인 검토를 수행한다.
② 양쪽 그룹으로부터 승인을 득한다.
③ 모델 작성을 문서화 한다.
④ 추가적인 정의를 위한 요구사항을 수집한다.
⑤ 모델작성을 위한 요구사항을 수집한다.(요구사항 정의, 자료수집 및 조직, 훈련)

■ 모델을 유지한다.

① 모델 자체의 수명은 직접적으로 모델을 사용하는 빈도이다.
② 모델이 계속적으로 유용한 것이 되기 위해서는 관리되어야 한다.
③ 모델을 읽는 것만으로는 팀이 모델을 개발하기 위해 습득한 모든 지식을 의사소통 할 수 없다. 따라서 팀의 멤버가 모델 전체에 걸쳐 관심을 갖고 기초자료 및 모델 개발과정을 검토해야 한다.

2.4.2 모델의 개발 지침

IDEFø 모델의 품질을 평가함에 있어서 가장 유용한 방법은 모델을 판독하고, 그것이 2 분 미만의 시간에 가능한지를 결정하는 것이다. 대형모델의 경우에는 이것이 두시간 이상 걸릴 수도 있다. 판독자가 그 시간 내에 IDEFø 방법에 의해 모델화된 환경을 이해하고 표현된 것을 설명할 수 있다고 느끼면, 모델로서의 가치가 있는 것이다. 만일 이틀 동안의 면밀한 연구를 통하여 충분히 이해할 수 있는 모델은 훌륭한 IDEFø 모델이 되지 못하며 커뮤니케이션 및 직관적 이해라는 모델의 중요 목적을 상실하는 것이다. 다이아그램의 구성은 다음과 같은 검토사항을 포함한 순서로 진행한다.

■ IDEFØ 다이아그램의 이해

성공적인 활동 모델링은 IDEFØ 다이아그램을 어떻게 판독하느냐, 어떤 정보가 모델로 투입되고 생산되는가에 대한이해, 그리고 모델 안에서 각각의 요소가 어떤 역할을 하는가에 대한 이해에 따라 달려있다. 다음은 IDEFØ 다이아그램의 이해를 위한 지침이다.

① 탑-다운 방식으로 읽는다.
② 기능은 무엇이 수행되어야 하는가를 나타낸다. 따라서 기능은 능동적인 동사구 형태로 표현되어야 한다.
③ 화살표의 한쪽 끝이 이어져 있지 않은 경우는 개념이 다이아그램의 범위 밖에서 제공되던가, 소모, 사용된다는 것을 나타낸다.
③ 호출 화살표(call arrow)는 자체적으로 수행되는 하나의 시스템을 호출하는 것을 말한다.
④ 화살표의 상호관계에 있어서 하나의 활동이 다른 활동의 입력, 제어, 혹은 자원으로 사용되는 경우 이때 후속 활동은 이전의 활동에 종속적이라는 것을 나타낸다. 그러나 이것이 시간적이거나 방법적인 사항을 포함하는 것은 아니다.
⑤ 하나의 다이아그램 내의 기능들이 동시에 수행되는 것이 아니듯이 다양한 입력, 제어, 출력, 자원(ICOM)이 동시에 수행된다는 것을 나타내는 것은 아니다.

■ 최상의 행동지침

① 첫째로 흐름이 아닌 제어와 제약사항을 생각하라. 다이아그램의 구조는 순차적인 관계와 상관없이 그들이 가지고 있는 연관관계를 보여주는 것이다. 다이아그램은 어떤 것이 먼저 수행되는가와 상관없이 정확한 사항만 말하면 된다. 다음은 이와 관련된 지침들이다.

② 다이어그램이 난해하다는 것은 당신이 다른 단계의 세부사항 일부를 넣었다는 것을 의미한다.
③ 의심스러운 개념을 버려라.
④ 견실한 추상화는 미숙한 세부화보다 더욱더 명료하고 파워풀 하다.
⑤ 입력(이것이 변경되었는가?)으로서 분명치 않으면 통제에 포함시켜라. 의심 나면 통제로 간주하라.
⑥ 입력이나 출력은 기능에 의하여 수행(조작, 가공, 생산))되는 것이다.
⑦ 제약은 왜 그렇게 기능이 수행되아야 하는가를 나타낸다.
⑧ 메커니즘은 기능이 무엇에 의하여 수행되는지를 나타낸다.

■ 모델링 체크리스트

비즈니스 엔지니어링을 용이하게 하기위해서는 다음사항을 수행한다.
① 무슨 활동이 수행되는지 정의하라.
② 이러한 활동들을 수행하기 위해 무엇이 필요한지를 정의하라.
③ 현재 시스템의 무엇이 올바른지를 결정하라.
④ 현재 시스템의 무엇이 틀렸는지 결정하라.
⑤ 활동 인터페이스(객체와 데이터)를 정의하라.
⑥ 활동에 관련된 비용을 포착하라.

의사소통을 용이하게 하기위해, 아래와 같은 사항을 수행한다.
① 도메인 전문가 이해를 향상시켜라.
② 공감대 의사결정을 용이하게 하라.
③ 효과적인 팀 활동을 촉진하라.

■ 모델 품질 체크리스트

① 완결성
- 이름, 설명, 목적, 그리고 관점 등 모든 필드의 내용이 표현되었는가?
② 간결성
- 대상 독자에 맞게 모든 용어가 적절하게 사용되었는가 ?
- 어떤 모델의 요소가 중복되지는 않았는가 ?
- 모든 요소가 명쾌하게 구별되는가 ?
③ 일치성
- 모델 전체에 걸쳐서 용어는 통일되었는가 ?
- 모든 모델 요소들은 모델화된 시스템에서 추적 가능한가 ?

④ 정확성

- 모델은 모델화된 시스템에 대한 정확한 설명이 되는가?
- 적용된 관계 및 제약사항은 시스템의 관계 및 제약사항으로 추적이 가능한 가 ?

④ 판독성

- 검토자에게 모델은 명확한가 ?
- 모델에 의해 전하고자 했던 정보는 정확하게 문법에 맞게 표현되었는가 ?

2.4.3 모델의 연관도(Coupling)와 응집도(Cohesion) 평가

IDEF⌀ 다이아그램에서 Coupling 은 데이터의 독립성과 개념의 분해 정도를 파악하기 위하여 하나의 개념이 어떤 기능에 의하여 어떻게 사용되는지를 파악하기 위한 방법이며 Cohesion 은 기능의 분해에 있어서 기능의 분해 정도를 파악하기 위하여 적절한 레벨링이 되었는지 또한 같은 레벨의 다이아그램에서 표현되는 기능들간에 어떤 관계가 유지되고 있는지를 파악하기 위한 방법이다. 이러한 연관성을 파악하는 이유는 Coupling 이나 Cohesion 의 측정치를 기준으로 개발된 모델의 복잡성 정도를 측정할 수 있다는 것이다.
경험적으로 볼 때 IDEF⌀ 다이아그램은 Cohesion 의 정도가 증가되고 Coupling 의 정도가 줄어들수록 모델의 판독성은 증가되고 복잡성은 줄어든다. 따라서 Coupling 과 Cohesion 의 정도를 파악하는 매트릭스는 판독성과 복잡성을 평가할 수 있는 유용한 측정방법이다. 다양한 타입의 Coupling 과 Cohesion 을 구분하기 위한 척도나 기준을 갖는 것이 모델작업자에게는 판독이 용이하고 복잡하지 않은 모델을 구축하기 위한 유용한 도구가 될 것이다.

특히, Coupling 의 정도가 낮을 때 우리는 AS-IS 에서 TO-BE 모델로의 전환을 쉽게 할 수 있는데, IDEF⌀ 다이아그램에서 모듈간의 연결은 데이터 인터페이스로 생각되며, 따라서 같은 가정아래 모듈간의 연결되는 데이터 인터페이스 개수에 제약을 둠으로서 가능한 한 적은 수의 모듈이 서로 연관되도록 한다. 시스템 공학적인 측면에서 이야기 되는 '주어진 모듈 셑으로부터 알려진 정보는 다른 모듈로부터 은닉(혹은 분리) 되어져야 한다.'는 데이터 은닉의 원칙도 같은 상황을 설명한다. 정보 은닉의 잇점은, Coupling 을 줄이는 것과 같이 불필요한 부분의 수정 없이 하나의 모델이나 시스템 설계 안에서 부분적인 수정을 용이하게 지원한다. 즉 데이터의 은닉과 Coupling 간에는 직접적인 연관관계가 있는데 특정 모델에서의 Coupling 의 정도를 추정하는 것은 정보의 은닉정도를 측정하는 효과를 가져다 준다.

■ Coupling의 종류와 측정

전통적으로 시스템공학적인 연구에 있어서 기능간의 Coupling 은 두 가지 관점으로 분석되고 있다. 그 첫 번째는 연결관계에 있어서 데이터가 어떻게 사용되는냐(natural of the connection) 하는 측면에 관한 것이고 두번 째는 연결관계에 있어서 데이터의 연결구조(structure of the connection)에 관한것이다. 그림 2-34 는 두가지 관점의 분석을 설명하고 있다.

이들 두가지 분류에서 사용되는 용어는 그림에서 찾을 수 있으며 이들에 대한 설명은 다음과 같다.

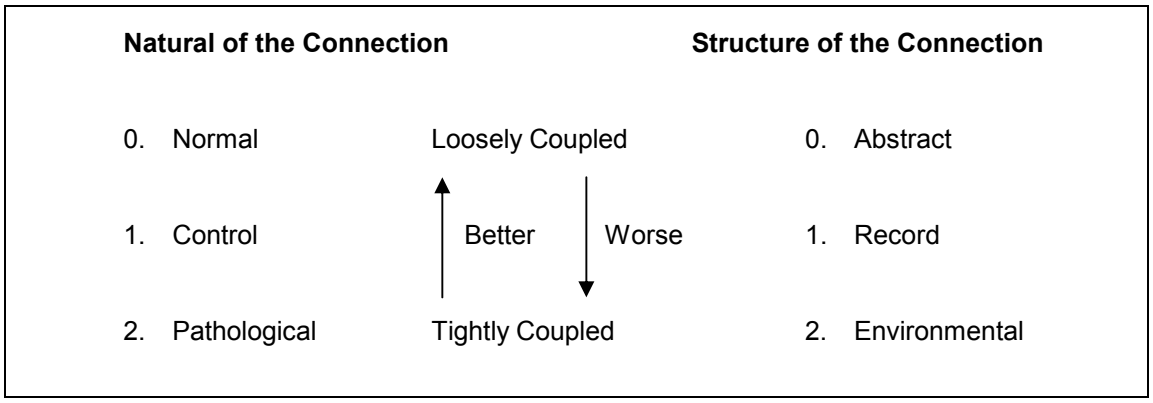


그림 2-34 : Coupling 의 종류와 측정

■ 데이터 사용 관점의 연결관계(Natural of Connections)

데이터 사용 관점의 연결관계는 표현되는 다이아그램이 하위 다이아그램을 가지고 있을 경우에 즉 최 하위 다이아그램이 아닌 경우 관련된 기능간에 데이터 인터페이스의 복잡성을 파악하는데 적절하다.

① Pathological : 가장 바람직하지 않은 연관관계

만일 하나의 기능이 다른 기능이 제공하는 정보를 참조하여 수행될 때, 두개의 기능이 'pathlogical couple' 되었다고 한다. 이러한 상황은 데이터 은닉의 원칙이 적용되지 않았을 때 발생한다.

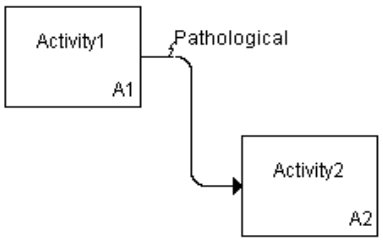


그림 2-35 : Pathological Coupling

② Control

하나의 기능에서 발생한 데이터의 흐름이 다른 기능의 control 로 영향을 미칠 때, 두 개의 기능은 'control couple' 되었다고 한다. 이와 같은 coupling 은 "기록이 실패하면 에러 메시지를 발행한다"와 같은 예정된 기능에 따라 플래그(flag)를 생성하는 기능을 주시함으로서 발견될 수 있다.

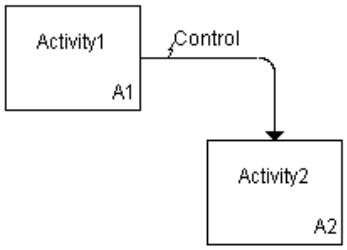


그림 2-36 : Control Coupling

③ Normal

두개의 기능간에 포함된 내용에 있어서 데이터의 의존적인 연관관계가 로 존재하지 않거나 혹은 하나의 기능과 다른 기능간에 데이터의 흐름이 control 로 영향을 미치지 않을 때 이들 두 기능은 'normally couple' 되었다고 한다. 이러한 타입의 연결은 가장 바람직한 종류의 연결인데, 기능간의 상호 커뮤니케이션에 있어서 전체적으로 데이터를 명쾌하게 보여주고 하나의 기능이나 상호간에 포함된 내용을 참조하지 않기 때문이다.

■ 구조적 관점의 연결관계(Structure of the Connection)

구조적 관점의 연결관계는 상위 다이아그램의 개념이 하위 다이아그램에서 상속될 때 분해의 정도를 파악하는데 적절하다.

① Environmental

두 기능들이 각각의 박스에 관하여 명쾌하게 상세화되지 않은 채로 모두 하나의 공통된 데이터 소스와 연관될 때 우리는 이를 'environmentally couple' 되었다고 한다.

이와 같은 상황은 그림 2-37 의 다이아그램에서 찾아볼 수 있다. Environmental coupling 의 문제점은 다음과 같은 결과를 가져올 수 있다.

- 데이터 은닉의 원칙을 위배한다. 오직 필요한 데이터만 억세스 하는 것이 제한되지 않는다.
- 지역화(localization)의 원칙에 위배된다. 데이터가 연관된 모듈과만 상관되지 않는다.
- 에러의 발생이 확률적으로 높아진다. 주변환경의 변화가 예상치 못한 모든 기능들에 영향을 미칠 수 있다.

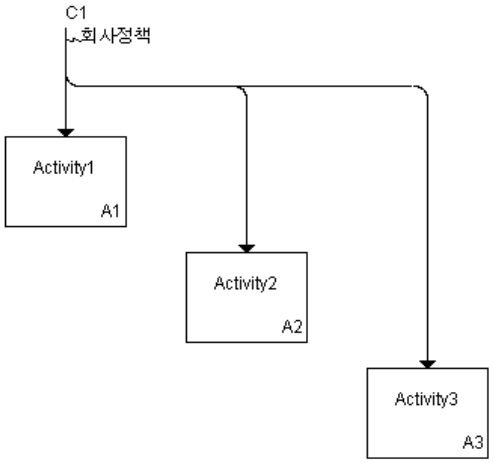


그림 2-37 : Environmental Coupling

② Record

Environmental 과 같이 모든 인터페이스가 공통된 데이터 셑으로부터 직접 연관되지는 않았지만 그렇다고 모두 상세화되어 연결되지도 않은 경우. IDEF⌀ 에서 Rcord Couple 의 예가 그림 2-38 에 나와 있다. 이와 같은 종류의 인터페이스에서는 모든 기능박스에 연결된 데이터 요소를 상세화하는 것이 좋다.

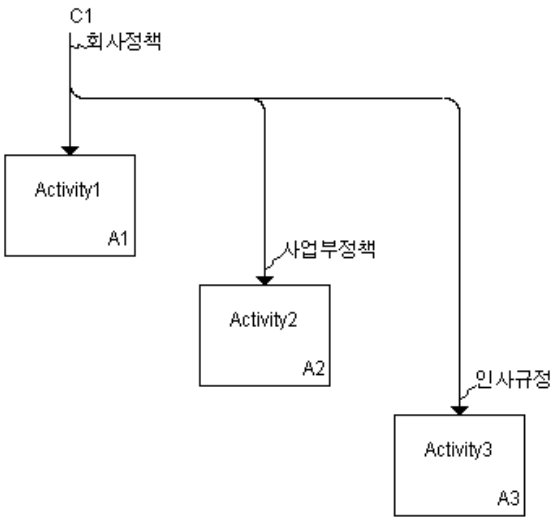


그림 2-38 : Record Coupling

③ Abstract

공통적으로 연관되어진 데이터 셑이 연결되어진 엘리먼트에 따라 명쾌하게 상세화 된경우를 우리는 abstract coupling 이라고 한다. IDEF⌀ 다이아그래에서는 상위 다이아그램의 데이터 화살표가 하위레벨에서 각 기능별로 상세하게 분해되어 연결된 관계를 abstract couple

이라고 한다. 상위레벨의 화살표는 하위레벨의 화살표가 포함하고 있는 요구되는 내용을 완벽하게 이해시킬수 있는 추상화된 용어를 사용하는 경우 데이터 은닉의 효과가 있다.

■ Cohesion의 종류와 측정

Coupling 이 개념을 중심으로 데이터 이용이나 분해의 관점이라면, Cohesion 은 하나의 기능에서 분해된 즉 최 하위 다이아그램 레벨에서 표현된 기능간의 연결관계 혹은 결합과 관련된 사항에 관한 것이다. 이론적으로 최소한 일곱가지의 바인딩 타입이 구분될 수 있는데 일반적인 IDEF⌀ 의 예를 통하여 각각의 타입에 대하여 설명하도록 한다.

Coupling 이 데이터의 은닉과 연관되어지는 것과 같이 Cohesion 은 지역화(Localization)와 연관되어진다. Localization 은 상호 밀접한 엘리먼트들을 물리적으로 근접한 범위 안에 구성시키기 위한 원칙인데 – 이러한 원칙을 바탕으로 같은 레벨의 다이아그램에 표현된 기능들이 어떠한 관계로 상호 연관되어져 있는가를 파악할 수 있다. 이러한 Localization 은 만일 같은 정보가 흐터진 많은 곳에 걸쳐 위치할 경우에 나타날 수도 있는 중복성을 최소화 하는데 있어서 도움을 준다. 예를 들면, 제조시스템의 기능모델에 있어서 AS-IS 에서 만들어질 수 있는 제조 서브시스템을 생각해 보자. 생산관리와 자재관리에 관련된 기능들을 분리된 기능모델로서 지역화 시키는 것이 기능적으로나 물리적으로 관계된 기능들과 데이터를 TO-BE 모델로 쉽게 전환할 수 있도록 지원한다.

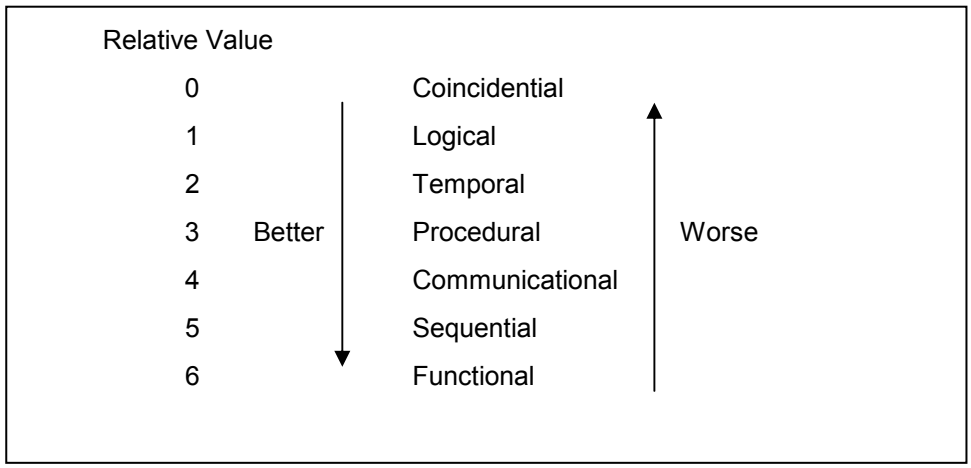


그림 2-39 : Cohesion 의 종류와 측정

① Coincidental: 가장 부적절한

Coincidential 은 하나의 다이아그램 안에서 기능들 간에 연결된 관계가 없거나 아주 적을 때 나타난다. 이러한 상황은 IDEF⌀ 다이아그램에서 화살표로 표현되는 데이터 간에 서로 거의 관련이 없을 때 발생한다. 이러한 상황의 극단적인 예가 아래 그림에 나타나 있다.

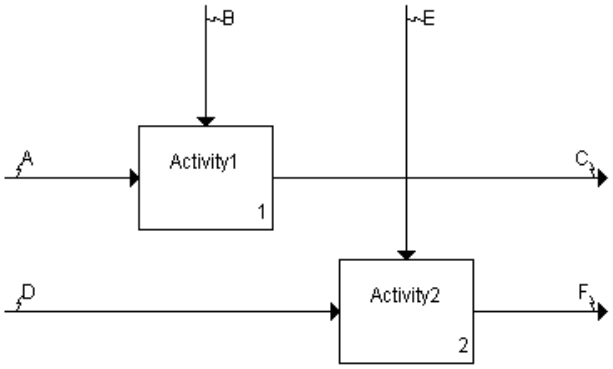


그림 2-40 : Coincidental Binding(Cohesion)

② Logical

Logical 은 데이터와 기능이 공통의 클래스나 엘리먼트 셑으로 합쳐지는 것으로 보여지는 것 같지만 경우에 따라서 이러한 관계가 형성되는 상황에서 발생한다.

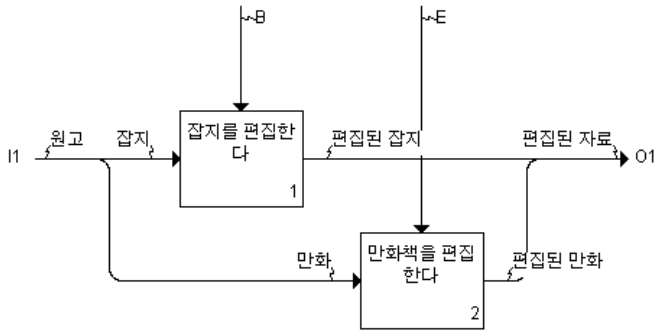


그림 2-41 : Logical Binding

③ Temporal

기능들이 같은 시간이나 데이터를 동시에 사용하는 경우, 순차적이 아닌 병행적으로 수행되는 경우 Temporal 이 나타난다. 이러한 다이아그램은 연관된 모든 기능을 초기화하는 작업과 같은 곳에서 종종 찾아볼 수 있다.

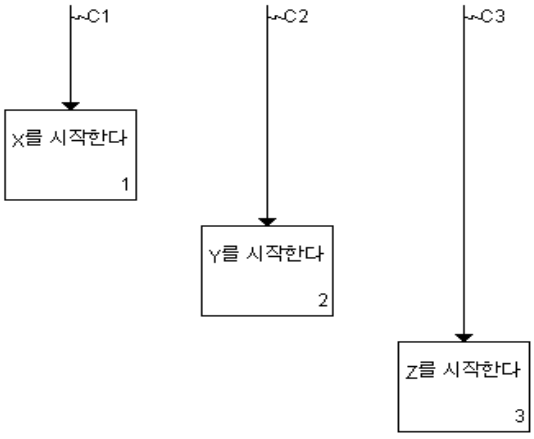


그림 2-42 : Temporal Binding

④ Procedural

Procedural 은 동일한 사이클이나 프로세스를 수행하기 위하여 기능이 공통의 프로세스로 합쳐지는 경우에 나타난다. 여기서 공통의 프로세스는 의사결정을 위한 분기 혹은 진행에 있어서 필요한 순차적인 절차의 경우이다.

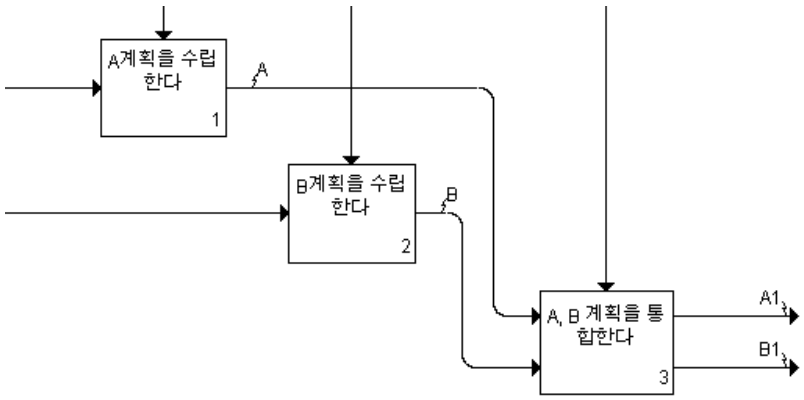


그림 2-43 : Procedural Binding

⑤ Communicational

Commnicational 은 두 개 이상의 기능이 동일한 입력 데이터을 이용하거나 출력하는 경우에 발생한다. 이러한 상황은 개념의 추가적인 분해를 통해서 해결될 수 있다.

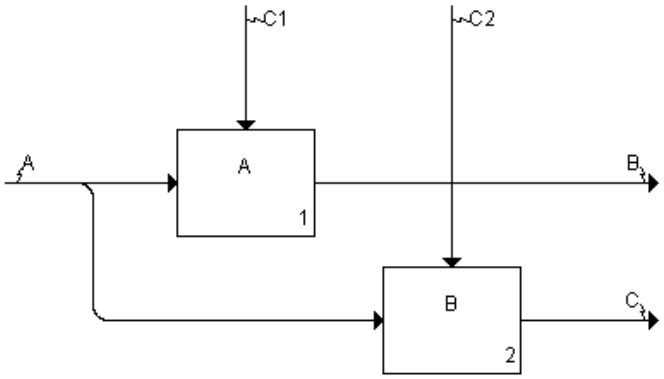


그림 2-44 : Communicational Binding

⑥ Sequential

Sequential 은 하나의 기능에서 나온 출력이 다음 기능의 입력으로 작용하는 순차적인 경우를 말한다. 이러한 상황은 Functional 에 가장 가까운 상태이며 지금까지 설명된 다른 Cohesion 보다 높은 수준이다.

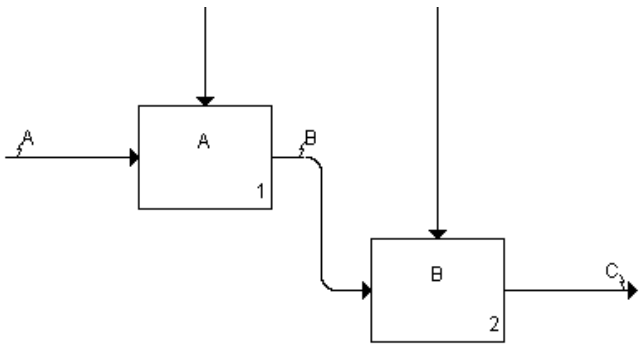


그림 2-45 : Sequential Binding

⑦ Functional

Functional 은 연관된 기능이 수행되는데 있어서 필수적인 사항으로 작동되는 경우를 말한다. 하나의 기능이 기동되기 위하여 다른 기능의 출력이 제약(control)으로 작동되는 경우가 여기에 해당된다.

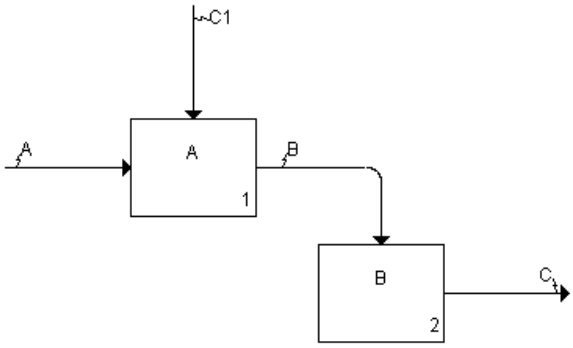


그림 2-46 : Functional Binding

응집도	Cohesion Name	기능	데이터
0	Coincidential	임의적	임의적
1	Logical	같은 셑이나 종류의 기능들 (예: 모든 입력을 수정)	같은 셑이나 종류의 데이터
2	Temporal	같은 시간동안에 수행되는 기능들(예: 모든 기계의 설정치를 초기화)	같은 기간동안에 사용되는 데이터
3	Procedural	같은 절차나 반복중에 발생하는 기능들(예: 일차 편집을 수행한다)	같은 절차나 반복중에 사용되는 데이터
4	Communicational	같은 데이터를 사용하는 기능들	같은 기능에 의하여 활성화되는 데이터
5	Seqential	어떤 동일 데이터를 순차적 변화 시키는 기능들	순차적인 기능에 의하여 변화된 데이터
6	Functional	하나의 기능을 수행하는데 결합된 기능	하나의 기능에 관련된 데이터

*. 번호가 높을수록 바람직함.

그림 2-47 : Cohesion 의 기능과 데이터 관점

■ Cohesion의 정도를 높이기 위한 검토

우리는 경험적으로 다음과 같은 사항을 검토함으로써 Cohesion 의 정도를 높일 수 있다.

① 복합된 문장이나, 쉼표를 포함하거나, 하나 이상의 동사를 포함하거나 할 수 밖에 없는 다이아그램은 낮은 레벨의 Cohesion 을 갖는다. 이러한 것들은 아마도 Sequential 이나 Communicational, 혹은 Logical Cohesion 을 형성한다.
② 첫번 째, ~한 다음, 그리고, 그런다음, ~을 시작한다, 단계, ~할 때, ~때 까지 그리고 모든 ~에 대하여 등과 같은 시간적인 표현을 포함하는 문장으로 표현된 다이아그램은 대부분 Temporal 이나 Procedural Cohesion 이 된다. 그러나 간혹 Sequential Cohesion 이 되는 경우도 있다.
③ 동사와 관련된 오브젝트(국어에 있어서는 주어)가 한 개가 아닐 경우 다이아그램은 대부분 Logical Cohesion 이 된다.

바람직한 품질의 모델을 개발하기 위하여 우리는 가능한 한 Cohesion 의 레벨을 Communicational 이상으로 유지해야 한다.

■ Coupling 과 Cohesion의 평가

① IDEF⌀ 에 있어서 Coupling 과 Cohesion 은 사실상 같은 개념이다. 그러나 실질적으로는 모델에 있어섯 다른 부분에 대하여 적용되는데, Coupling 은 상위 레벨의 다이아그램이나 기능들간에 복잡성을 평가하기 위하여 사용되는 반면, Cohesion 은 같은 상위기능에서 분해된 하나의 다이아그램 안에 표현된 기능간의 연결관계나 응집도를 측정하기 위하여 사용된다.
② 상위 다이아그램에서 높은 Coupling 을 찾아 냄으로서 우리는 부적절한 연관관계의 단서를 찾을 수 있다. 이러한 연관관계는 하위 다이아그램에서 낮은 Cohesion 을 형성하게 만드는데, 모델작업자는 복잡한 Coupling 을 제거하기 위하여 상위 기능을 다시 분해함으로서 모델의 품질을 높이고 이해하기 쉬운 상위(Parent):하위(Child) 다이아그램 셑을 만들 수 있다.
③ Coupling 과 Cohesion 의 측정 값은 더해질 수 있는데, 만일 하나의 다이아그램이 ‘Sequential(5 점)’이면서 동시에 ‘Communicational(4 점)’ 이라면 단순히 ‘Communitional(4 점)’ 보다는 바람직한 것이다(Cohesion 이 높은 정도가 더욱 바람직하므로). 그러나 연결관계가 ‘Control(1 점)’과 ‘Record(1 점)’가 결합된 상태라면 단순히 Control 이거나 Record 하나의 경우보다 바람직하지 않다(Coupling 은 낮을수록 좋다).

3. IDEFØ 기능 모델의 Activity Based Costing(ABC) 확장

이 장의 목적은 IDEFØ 기능모델과 Activity Based Costing 의 장점을 이용하여 통합된 ABC 를 수행하기 위한 방법을 소개하는 것이다. SmartABC™ 는 IDEFØ 기능모델을 전환하여 ABC 분석을 용이하게 수행하도록 지원하는 Knowledge Based Systems, Inc.(KBSI)의 통합된 ABC Tool 이다. SmartABC™ 는 다음의 세 가지 구성 요소를 가지고 있다:

① 기능모델 부문 - KBSI 의 AI⌀WIN™ ,
② Excel™ - Microsoft 사의 Spread Sheet, 그리고
③ SmartABC™ – KBSI 에 의해 만들어진 것인데, AIØWIN™ 의 IDEFØ 기능모델과 Excel™ 의 ABC 분석 Sheet 사이의 양방향 자료변환을 지원한다.

이 글은 IDEFØ 기능모델과 통합된 ABC 를 수행하기위한 방법을 소개한다. 이 글은 또한 SmartABC™ 를 사용해서 어떻게 기능모델과 통합된 ABC 를 수행하는가에 대하여 논의할 것이다. 이 글은 다음과 같은 목적을 가진 사람들에 의해 사용되어 질 수 있다.

① ABC 를 처음 접하는 사람들은 ABC 를 배우기 위해 이 글을 활용할 수 있다.
② ABC 에 경험이 있는 사람들은 기능모델을 ABC 와 통합하는 것에 관해 그리고 다음 단계인 Business Process Reengineering(BPR)에서 ABC 를 효과적으로 활용하는데 있어서 이 글을 활용할 수 있다.
③ 기능모델 개발자들은 어떻게 기능모델로부터 ABC 모델을 생성하는지를 배우기 위해 이 글을 활용할 수 있다.
④ SmartABC™ 의 사용자들은 SmartABC™ 기초가 되는 방법론을 배우기 위해 이 글을 활용할 수 있다.

세가지 작업이 이 글에서 다루어질 것이다.

① ABC 개념과 방법론을 소개한다
② 활동을 기반으로 한 통합된 비용/기능분석 수행의 이점을 소개한다.
③ SmartABC™ 의 사용을 위한 방법들을 문서화한다.

3.1 Activity Based Costing(ABC) 이란 무엇인가?

Activity Based Costing(ABC)은 비용에 근거한 관리 및 의사결정 지원을 위해 가장 널리 사용되고 있는 기술들 중에 하나로 빠르게 자리잡고 있다. 전통적인 회계방식은 투자자들의 의사결정 지원을 위해 자산 기반의 보고서로서 재고를 평가하는 것에 관심을 가져왔다. 이러한 방법은 주요 비용 요소들을 추출함에 있어서 각각의 상품에 소요되는 비용을 직접 추적할 수 있다면(전통적인 직접 비용들) 의사결정을 위해 유용한 정보가 될 수가 있었다. 그러나 오늘날 기업에서 발생되는 대부분의 비용은 직접비용이 아닌 경우가 적지 않다. 그리고 전통적으로 운영되던 간접비의 일괄적인 배분에 의해 계산된 각 상품의 원가는 실제 소요되는 비용과는 다른 결과를 종종 보여주기도 한다.

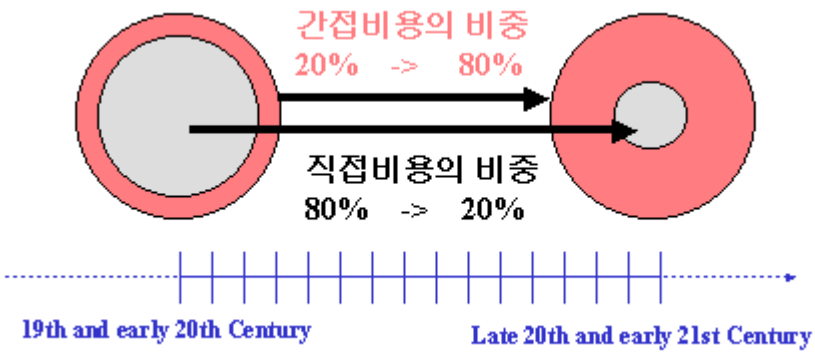


그림 3-1 : 직접비용과 간접비용의 비중

특히 서비스업종이나 공공기업과 같은 조직에서 경영관리 판단 근거로서의 전통적 재고평가 회계기술의 사용은 불가피하게 부적절한 의사결정에 이르게 될 것이다. ABC 는 조직에서 비용을 발생시키는 원인이 되는 계량화할 수 있는 요소들(ABC 용어 중에 “코스트 드라이버”)에 초점을 맞추는데 각각의 그러한 요소들이 전통적인 원가구성의 데이터에 미치는 비용발생 영향을 결정하는데 사용된다. 이와 같은 방법의 원가구성은 코스트 드라이버(Cost Driver)와 원가와의 관계에 의해 비용에 영향을 미치는 사항을 분석하고 대안을 개발할 수 있는 대체 시나리오의 개발을 지원한다.

3.1.1 Activity Based Costing의 개념들

ABC 의 중심 개념은 모든 활동들은 자원을 사용한다는 것이다. 또한 이런 자원의 사용은 결국 비용을 소비하게 되며 이는 회계장부상에 기재되는 계정별 비용항목으로 할당된다. 다른 하나의 중심개념은 이러한 활동은 활동에 소요되는 비용을 기반으로 부가가치를 생산하기 위한 것인데 이러한 부가가치를 생산하기 위한 활동의 원가를 산출하는 것이다. 아

래의 그림에서 우리는 직관적으로 '자재조달' 활동이 '재고관리 시스템' 과 '자재조달 부서' 라는 자원을 사용하고 있으며 이러한 활동의 결과로 세가지(A,B,C) 제품의 자재를 조달하는 부가가치를 산출한다는 것을 알 수 있다. 이러한 각각의 A,B,C 제품의 자재조달의 원가를 산출하고 ABC 원리를 사용하기 위해서는 우리가 ABC 의 도움을 받아 의사결정하기를 원하는 모든 활동들을 확인해야 한다. 우리가 "무엇"을 하는지에 대한 이해 혹은 활동에 대한 분석은 현대적 조직에 있어서는 매우 일반적인 활동이지만, 활동에 반하여 조직적 단위로서 비용을 수집하는 전통적 회계시스템에서는 이를 정확히 반영하고 있지 않다. 인사부서에 의해 수행되는 직원의 고용과 같은 주요 활동들은 외부로부터 새로운 인력을 채용하는데 필요한 활동을 요구하는데 이러한 활동에 필요한 시간은 비용을 소모하는 것이다. ABC 의 수행은 하나의 활동을 수행하는데 필요한 비용을 결정하는 것이다.

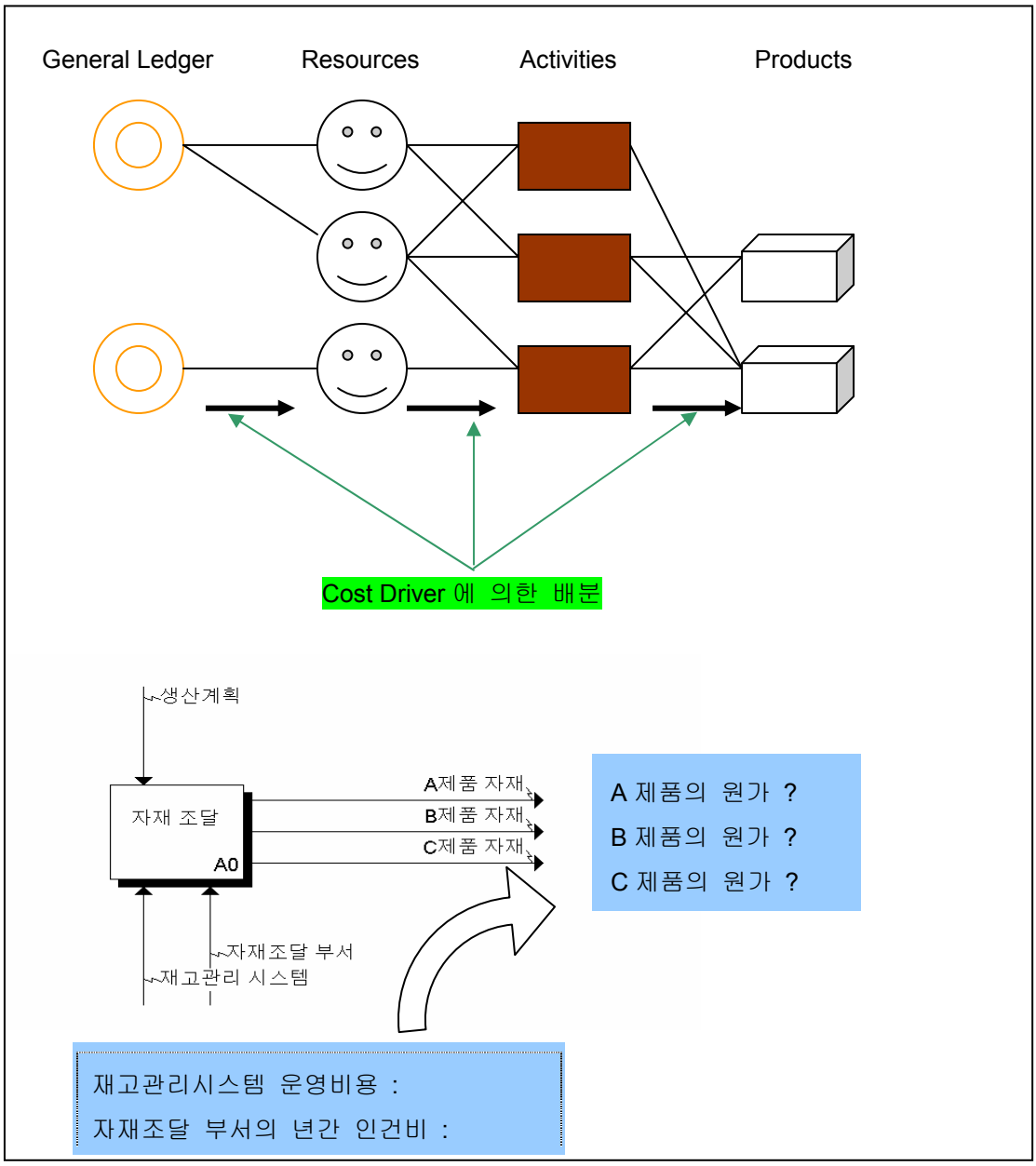


그림 3-2 : Activity Based Costing 의 비용 배분 구조

3.1.2 Cost Drivers

Cost Driver 는 비용을 발생 시키게 하는 인자들이다. Cost Driver 는 트리거(trigger), 혹은 활동이 수행되어지게 하거나 비용 발생/증가의 원인이 되는 그러한 것들이다. 예를 들면, "자재 요구시점을 결정한다" 라는 활동을 고려해 보자. 이 예는 생산일정이 변할 때마다 그러한 계산을 다시 할 필요가 있다. 따라서 "생산일정 변화의 수" 는 "자재 요구시점을 결정한다" 의 Cost Driver 가 될 수 있다. 이 예가 암시하는 것은 각각의 활동은 유일한 Cost Driver 를 가져야 한다는 것이며 그렇지 않을 경우 그 활동은 더욱 분해되어질 필요가 있다는 것을 나타낸다. 반대로 활동분석의 유일한 목적이 ABC 를 지원하기 위한 것이라면 활동은 더 분해될 필요가 없다.

활동의 비용에 영향을 미치는 인자들 사이에서 기초적인 차이가 있을 때는 그 활동은 더 분해되어져야 한다. 예를 들면, 많은 회사에서는 한번의 주문에 의해 판매되는 상품들의 수가 같지 않은 경우가 대부분인데 "외상매출 금액을 검토한다" 와 같은 활동은 주문의 수에 의해 구동 되지만 "주문된 상품을 기록한다"는 주문된 상품의 종류(수)에 의해 구동 되어 진다.

3.1.3 활동

활동이란 생산적인 목적을 위해 수행되어진다고 간주되는데 이러한 목적은 ABC 분석에서 Cost Object(산출물, 활동모델의 Output)로 표현된다. Cost Object 의 정의는 ABC 로부터 예상되는 의사결정지원 항목에 따라 달라진다. 일반적으로 주어진 ABC 분석 프로젝트에서 Cost Object 들은 고객에게 제공되는 상품이나 서비스들이다. 또는 그것은 "내부적인 교육을 제공한다" 혹은 "저장 위치를 결정한다" 와 같은 내부적 서비스들도 될 수 있다.
ABC 가 프로세스 개선활동의 지원을 위해 사용될 때 , 활동들은 부가가치 활동(Value Added)과 부가가치가 없는 활동(Non Value Added)으로 분류되어 질 수 있다. 이러한 분류는 분석에 적용되는 Cost Object 들에 의해 결정되어 진다.

3.1.4 직접/간접 비용

직접비용은 자재 그리고 직접인건비와 같이 각각의 상품과 서비스에 직접적으로 할당 할 수 있는 것이다. 이런 비용들은 직접적으로 제품이나 서비스에 축적되어 지며, ABC 를 사용할 때 우리는 의사결정 프로세스에서 이러한 비용을 활동에 배분하는 것을 배제한다. 간접비용은 상품이나 서비스에 직접적으로 할당 되지 않는 비용이다. 현대의 조직들에서 이러한 비용은 기업이 지출하는 전체 비용 중에서 많은 부분을 차지하는 추세이다. 이러한 간접비용의 분석이 ABC 의 목표이다.

활동들은 Cost Object 들을 생산하기 위해 수행되며 활동들은 '자원(resource)'의 소모를 통하여 제공된다. 전통적인 회계시스템은 자원이 소모한 비용을 특정한 방법으로 수집하는 것인데 이것으로 하나의 Cost Object 의 실제 비용을 정확하게 결정하기 어렵다. ABC 는 전통적인 총계정으로부터 Cost Object 에 대한 원가를 산정하고, 재분류하고, 할당하는 증명된 기술들을 제공한다.

활동들은 직접적으로 돈을 소비하지 않는다. 그것들은 실제로 시간과 같은 자원들을 소비한다. 하나의 활동이 소모하는 비용을 결정하기 위해서 우리는 활동에 사용되어지는 자원들을 확인할 필요가 있다. 그런데 문제는 활동과 관련되어진 것으로 밝혀진 자원의 비용이 – 예를 들면 하나의 거래선을 개발하기 위해 구매담당자가 소모하는 시간의 비용 – 총계정원장과 같은 일반적인 비용원천에서는 추출할 수 없다는 것이다. 총계정에서 정의된 자원들은 "구매부서의 급료", 그리고 "컴퓨터장비 리스비용" 과 같은 항목으로 제한되어 있는 것이다.

경험적으로 볼 때 한 활동에 소요되는 정확한 자원의 비용을 추출하기위해 그룹별로 추출된 자원비용을 할당하는 것은 종종 필요하다. 예를 들면, '급료', '수당'으로부터 '구매관련 인건비'를 분류할 수 있고, '구매관련 인건비'를 관련된 사람의 역할에 따라 다시 발주담당과 계약담당자의 인건비 비용으로 분류할 수 있다.

3.2 기존 Activity Based Costing 의 문제점

Activity Based Costing 은 조직에서 다양한 객체(object)들의 실제 비용을 결정하거나 예산편성 및 계획을 위한 강력한 도구다. 그것은 발생되는 비용과 이러한 비용 발생에 영향을 주는 인자들에 관한 정보를 제공한다. 한편, ABC 는 관련된 업무개선의 노력이 집중되어야 할 곳이 어디인지에 관한 정보를 제공하지만 업무개선 그 자체를 수월하게 수행하는데 필요한 정보는 제공하지 않는다.

업무개선의 성공적인 수행을 위해서는 비용정보와 같이 기능에 관한 정보가 필요하다. 조직이 사업을 수행하는 방법에 있어서 문제영역을 추출하고 인식하기 위해서, 왜 활동이 수행되어 지는가, 활동의 Output 이 무엇인지, 어디에서 Output 이 사용되어 지는지 등에 관한 정보가 추가적으로 활동에 관한 비용정보를 외에 필요하다.

전통적인 ABC 는 활동을 '부가가치 활동과 부가가치가 없는 활동', '중요한 활동 혹은 부차적인 활동', '임의의 활동과 꼭 필요한 활동' 등으로 분류함으로써 어느 정도 업무개선을 위한 지원을 제공한다. 그러나 이러한 분류들은 매우 주관적이며 분류를 하는 사람의 관점에 의존하게 되는데 작업장 내에서의 자재이동은 작업장 관리자에게는 부가가치 활동으로써 분류될 수 있다. 그의 관점에서 보면 부품이 기계에 장착되는 것과 작업하기 위해 준비하는 것은 부품이 창고에 있는 것보다 더욱 가치를 가지는 것이다. 즉 기계로 부품을 옮기고, 장착하는 것은 그것을 팔고 이윤을 얻는데 필요한 작업들을 수행 가능하게 하는 활동들이다. 그러나 마케팅 관리자는 작업장 안에서의 자재 이동을 부가가치가 없는 활동으로서 분류하게 될 것이다. 이와 같이, 누군가 자재이동 활동을 없애는 수행 가능한 대안적인 방법으로 찾을 때까지 자재 이동은 필요한 것으로써 분류되어 질 수 있다. 따라서, 활동들을 '부가가치 활동 또는 부가가치가 없는 활동', '임의의 또는 꼭 필요한 활동' 등으로 분류하는 것은 별로 유용한 것은 아니다. 그런 정보들은 업무개선의 노력이 어디에 집중되어야 하는지를 지시해 줄 수 있을 뿐이며, 그 업무를 수행하는 보다 낳은 방법을 결정하는 데에는 그리 유용하지 않다.

조직의 이윤을 증가 시키기 위해서 우리는 다음과 같은 질문에 대답할 수 있어야 한다.

① 어떻게 우리의 부가가치가 없는 활동들을 감소 시킬 수 있는가
② 부가가치 활동들을 수행하는 보다 나은 방법은 없는가
③ 어떤 활동들을 제거할 수 있는가
④ 활동을 수행하지 않는 것이란 무엇을 의미하는가

이러한 분석을 용이하게 하기 위해서 분석가는 왜 하나의 활동이 수행되어 지는가, 그 활동들은 어떠한 활동에 영향을 받는가, 활동을 대체할 수 있는 다른 방법이나 대안에 관한 정보를 수집해야 한다. 따라서 비용에 관련된 ABC 정보와 함께 위에 정의된 정보와 같은 연관된 정보의 보충이 필요하다. SmartABC™ 개발 동기의 하나는 업무개선 활동의 수행에 필요한 정보를 제공하는 것이다. 이것은 기능적 모델링 구성요소(AIØWIN™)와 프로세스 모델링의 구성요소(PROSIM™)를 통해 용이하게 이루어 진다.

AIØWIN 은 기능 관점으로부터 모델링 활동을 용이하게 지원한다. - 하나의 활동에 Input 은 무엇인가? Output 은 무엇인가? Output 이 사용되는 곳은 어디인가? 등등. AIØWIN™ 에서 지원하는 방법은 IDEFØ 이론에 근거한다. PROSIM™.은 활동에 관한 프로세스의 관점의 설명을 포착하는 것을 가능하게 한다. – 직원채용의 순서는 무엇인가? 그 활동들을 수행하는 어떤 다른 대안적인 방법들이 있는가? 등등..
PROSIM™ 은 IDEF3 방법을 기반으로 한다. IDEF3 에 관한 설명은 다음 장에서 찾을 수 있다.

3.3 새로운 Activity Based Costing 방법론

ABC 를 수행하기 위한 새롭게 제안된 방법론은 다음의 절차들을 포함한다.

3.3.1 I단계 : 과제의 목적과 경계를 정의한다

우선 조직은 ABC 수행의 범위를 정의하는 것이 필요하다. - 전체 조직을 대상으로 수행할 것인지, 특별한 부서를 혹은 어떤 공장을 대상으로 수행할 것인지. 조직은 과제의 목표를 명확하게 결정해야 한다. - ABC 가 활동들의 비용결정을 하는데 수행될 것인지, BPR 을 위해 수행될 것인지, 생산비용의 결정을 위해 수행될 것인지 등등,
이 글에서의 ABC 방법론은 BPR 의 관점에서 설명되어 질 것이다. BPR 활동들과 상품의 비용결정을 비롯해 업무를 수행하는 더욱 효과적이고 더 나은 결정을 지원하는 것을 포함한다. 만약 목적이 단순히 활동과 상품의 비용 결정이라면 과제의 수행은 그 목적이 이루어진 후에 바로 멈춰져야 한다.

3.3.2 II단계 : 활동들을 정의한다.

여기에서 조직은 I 단계에서 결정한 과제의 경계 안에서 수행되는 다양한 활동들을 정의할 필요가 있다. 이 단계에서 우리는 다음과 같은 것들을 밝혀낸다.

① 기능 그리고 입력, 출력, 제어, 메커니즘(Inputs, Outputs, Controls, Mechanisms)으로 불리어지는 개체(object) , 자원들의 연관관계
② 활동들, 자원들 그리고 객체들간의 계층 적 구조들.

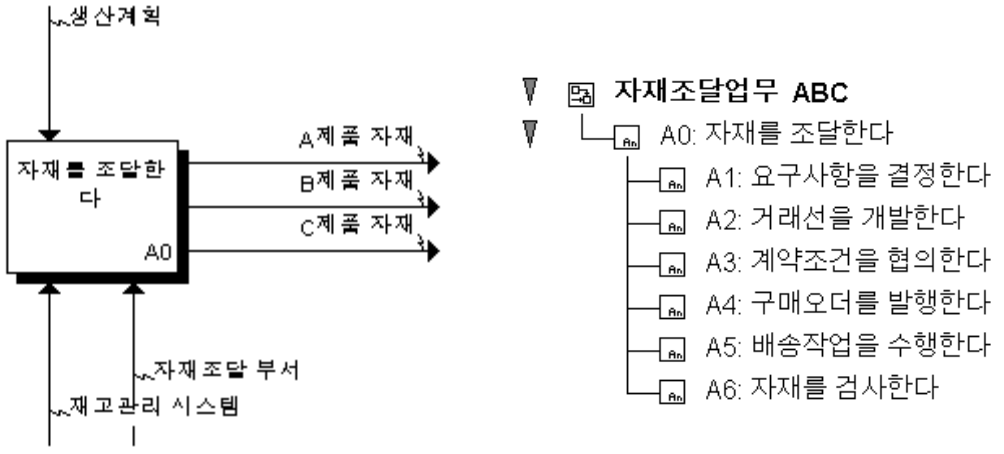


그림 3-3 : ABC 를 위한 Context 다이아그램

3.3.3 III단계 : 기능모델(IDEFØ)을 만든다.

III 단계에서 우리는 활동의 기능적 모델을 만들기 시작한다. AIØWIN^{TM} 은 IDEFØ(기능 모델링 방법론)에 근거한 모델링 Tool 이며 활동들의 기능모델을 만드는데 사용될 수 있다. 이러한 기능모델을 만드는 것은 다음과 같은 장점들을 가진다:

① II 단계에서 만들어진 간결한 모델의 결과를 서류화 한다.
② 그림을 사용해서 활동들과 자원들 그리고 개체들 사이의 관계들을 명확하게 정의하는 것을 도와준다.
③ 전개된 기능적 모델로부터 AIØWIN^{TM} 은 대강의 ABC 모델을 생성할 수 있다. 이 ABC 모델은 완성된 ABC 모델을 만드는데 시작점으로써 사용되어 진다.
④ 이와 같은 정확한 AIØWIN^{TM} 모델 전개 절차는 어떤 정보의 불일치나 생략을 밝혀내고 그런 문제가 발생되지 않도록 지원한다.

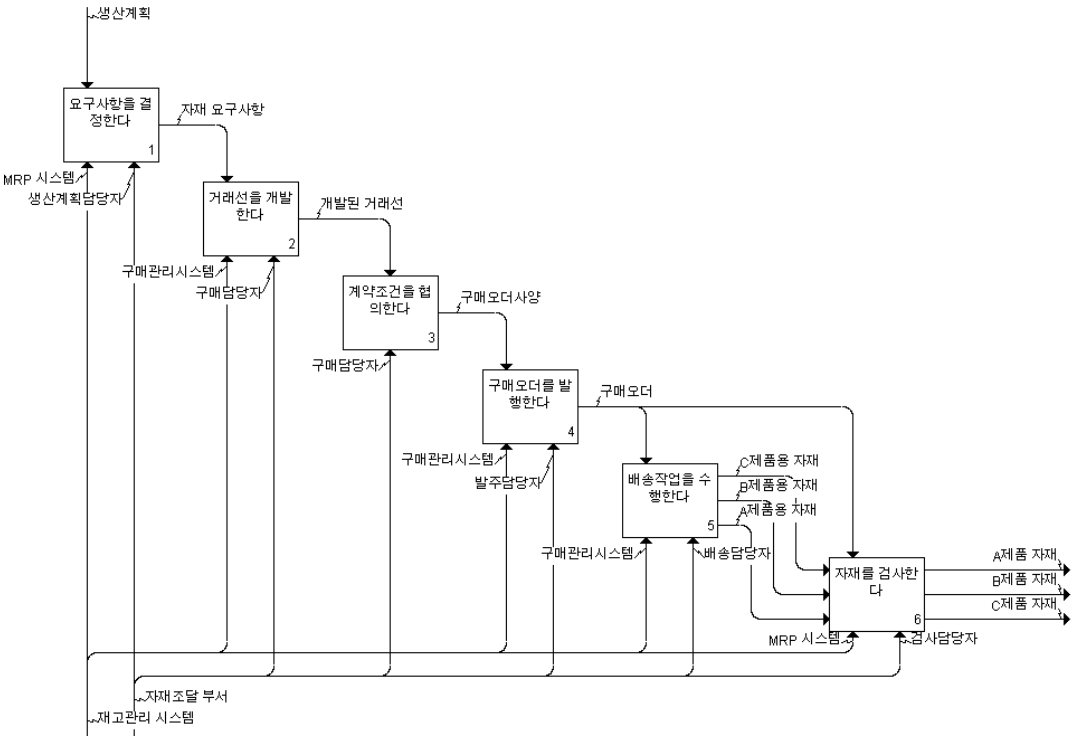


그림 3-4 : ABC 를 위한 기능모델의 구조

3.3.4 IV단계 : 전개된 기능적 모델로부터 기본적인 원시 ABC모델을 생성한다.

초기의 ABC 모델은 AIØWIN™ 에서 전개된 기능적 모델로부터 생성되어 질 수 있다. 이 초기 ABC 모델은 기능적 관점에서 만들어진 정보의 ABC 모델이다. 이것은 ABC 모델을 완성하기 위해 비용과 연관된 정보를 보완시킬 필요가 있다. 이 초기의 ABC 모델을 보기위한 이상적인 곳은 AIØWIN 의 ABC Matrix View 이다. 이 ABC Matrix View 는 초기의 ABC 모델을 다듬고 세부적인 비용정보를 가진 초기 ABC 모델의 보완을 용이하게 한다.

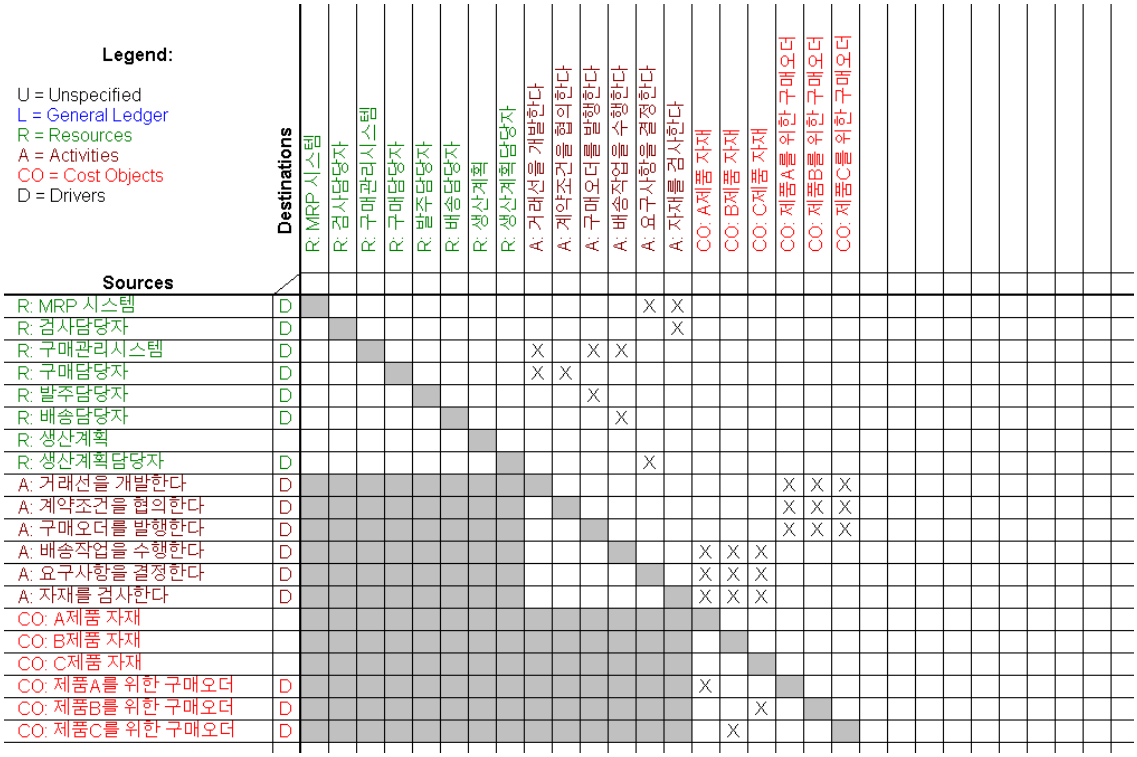


그림 3-5 : ABC Matrix View 사례

3.3.5 V단계 : 초기의 ABC 모델을 개선한다.

초기의 ABC 모델에서 활동들, 자원들 그리고 Cost Object 들은 기능적 관점으로부터 생성되어진 것이다. 이것들은 비용관점에 맞게 조정되어질 필요가 있다. 이러한 조정은 활동, 자원들 그리고 Cost Object 들의 추가, 삭제, 병합을 포함한다. 예들 들면, 기능적으로 중요한 활동이 그것과 연관된 낮은 비용을 가지고 있다고 하자. 그런 경우, 조직의 모델 설계자는 비용모델 관점에서 그 활동을 계속 유지시키느냐 혹은 다른 활동들과 병합함으로써 그 모델을 단순화시키느냐 하는 것을 결정할 필요가 있다. 기능적 모델에서 Output 들은 초기 ABC 모델에서 Cost Object 로 변화될 것이다. 모델 설계자는 그러한 Cost Object 들을 모두

유지시키느냐 혹은 몇몇 Cost Object 를 생략하느냐 하는 것을 결정해야 한다. 유사하게 기능모델에서 계층적 구조는 기능적 관점에 근거할 것이다. 모델 설계자는 이러한 계층적 구조가 비용 모델 관점에서 다시 구성되어야 할 필요가 있는지를 결정할 필요가 있다.

기능적 모델링은 정의된 경계 내부의 모든 활동들을 정의한다. 대부분의 경우, 이 단계에서 비용관점과 연관성이 없거나 덜 중요한 활동이나 자원들 그리고 Cost Object 들의 생략이나 병합과 같은 변경사항만이 수행되어질 필요가 있다. 하나의 개념(자원,Cost Object)의 중요성은 과제의 목적에 그것이 어느 정도 중요사항인가에 달려있다. 만약 그 과제가 활동들과 Cost Object 들의 실제 비용을 결정하는 것을 목표로 하는 회계담당자에 의해 수행되어 진다면, 그는 모델에서 나타나는 모든 비용을 포함하기를 원하게 될 것이다. 만약 그 과제가 BPR 노력의 한 부분으로써 엔지니어에 의해 수행되어 진다면, 그는 높은 비용을 발생시키는 항목에만 관심을 가지게 될 것이다. 이때 엔지니어는 복잡하고 정확한 모델을 만드는 것보다 프로세스 리엔지니어링 분석을 위한 보다 간단하고 유용한 모델을 만들기를 더 원할 것이다. 전통적으로 ABC 는 회계결산, 비용정산 그리고 예산편성을 위해 개발되어 진 것이다. 이러한 모델들은 정확성을 중요하게 생각하는 경향이 있으며 이러한 정확성은 그들의 요구사항과도 관련이 되어있다. 하여튼 ABC 의 기본 원칙으로써 합리적 방법의 간접비용 계산은 다른 영역들과 연관을 가진다. 어떤 사용자들은 그들의 특별한 요구에 근거한 ABC 모델을 만드는 것을 선호할 것이다. AIØWIN™ 는 이와 같은 관점에 의존하는 ABC 모델 생성을 용이하게 지원한다. 하나의 기능모델로부터 각각의 사용자들은 각자 다른 초기 ABC 모델들을 만들 수 있고 또한 그들의 요구들에 근거해 독립적으로 이를 발전시켜 나갈 수 있다. 다시 말해 각자의 요구와 관점에 근거해서 초기 ABC 모델에서 필요한 항목을 추가, 편집 그리고 삭제할 수 있다. 하나의 기능모델로부터 발전된 다른 ABC 모델들은 다른 관점에 근거한 하나의 비용모델에 대한 다른 버전으로 해석되어 질 수 있다. 또한 이러한 과정에서, 모델 내부의 활동들은 '부가가치 활동 과 부가가치 없는 활동' 또는 '임의의 활동과 꼭 필요한 활동' 등의 분류와 같이 다른 차원으로 분류되어진다.
AIØWIN™ 의 Cost Matrix View 는 초기의 ABC 모델을 변화 시키기 위한 가장 좋은 환경을 지원한다.

3.3.6 VI단계 : Cost Driver를 정의한다.

Cost Driver 는 산출물(products)에 의해 소요되는 관련된 자원이나 활동의 빈번함, 세기의 정도를 나타낸다. Cost Driver 는 비용을 발생 시키게 하는 인자들이다. Cost Driver 는 자극유인(trigger), 혹은 활동이 수행되어지게 하거나 비용 발생, 증가의 원인이 되는 그러한 것들이다.

예들 들면, "구매오더를 발행한다" 는 활동의 Cost Driver 는 "구매오더의 횟수"가 될 수 있

을 것이다. 이것은 구매오더 발행 활동의 비용은 구매주문 횟수에 의해 좌우되는 것을 의미한다. 각각의 개념은 하나의 Cost Driver 만 가져야 되며 그렇지 않을 경우 그 개념은 더 분해될 필요가 있다. 반대로, 만약 분석의 유일한 목적이 ABC 를 지원하는 것이라면 한번 유일한 Cost Driver 가 밝혀진 개념은 더 이상 분해할 필요가 없다. AIØWIN™ 에서 Cost Driver 는 각각 같은 양으로 배분되던지(Evenly assigned), 퍼센트(%), 혹은 사용자가 임의로 정의(User defined)하는 형태로 분류할 수 있다.

예들 들면, 아래의 그림에서 구매오더 발행활동의 년간 비용이 100 원인 경우, 만약 Evenly assigned Cost Driver 가 "구매오더를 발행한다"는 활동의 Cost Object C1 과 C2 에 할당되어 진다면 구매오더 발행의 비용은 두 Cost Object C1, C2 에 50 원씩 똑같이 배분된다. 다른 한편, 만약 퍼센트(%) Driver 가 사용되어 진다면 구매오더 발행 활동의 비용은 두 Cost Object C1, C2 에 할당된 비율로서 분배되어지는데 이 경우 경험적으로 C1 과 C2 의 비율이 각각 70%와 30%라면 비용은 두 Cost Object C1,C2 에 70 원과 30 원으로 배분된다. 추가적으로 만약 "구매오더의 수"가 Cost Driver 로써 사용되어 진다면, 구매오더 발행 활동의 비용은 주문되어지는 각 C1, C2 의 구매오더의 수에 근거해서 두 Cost Object 에 분배되어 지고 C1,C2 의 년간 구매오더 수가 각각 100 과 300 이라면 C1,C2 에 배분되는 활동의 비용은 구매오더수의 비율에 의해 각각 25 원과 75 원이 될 것이다. Cost Object 의 비용 결정만이 ABC 과제의 목적인 경우를 제외하면 우리는 모델에서의 모든 활동과 자원 그리고 Cost Object 를 밝혀내고 Cost Driver 를 정의해야 하며 초기 ABC 모델에 이를 등록해야 한다.

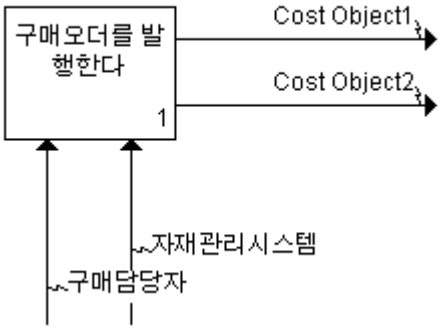


그림 3-6 : Cost Driver 를 이용한 비용 배분

3.3.7 VII단계 : 배분 경로(Assignment Paths)를 정의한다.

ABC 는 계정비용을 발생시키거나 이와 관련되어질 수 있는 항목들에 할당하는 것과 연관된다. 예를 들면, 구매주문에 소요되는 비용은 이의 수행과 관련된 다양한 객체들에 할당되

어 지는데 비용의 할당은 할당경로와 Cost Driver 를 통해 이루어진다.
이때 할당경로는 원천계정(그것의 비용이 할당되는 계정 - 위의 예에서는 “구매오더 발행”)과 목적계정(비용들이 할당되는 계정 - 위의 예에서 “구매오더 발행”에 의해 생산되는 다양한 Cost Object)으로 구분된다.

Cost Driver 는 수행되어야 할 분배 기준이나 척도를 밝혀준다. 즉, 구매오더 발행의 비용이 다양한 Cost Object 에 똑같이 할당될 것인지, 비율(%)로서 배분될 것인지, 또는 어떤 기준에 의해 배분될 것인지를 정의해야 한다. Cost Driver 와 할당경로(원천계정과 목적계정)를 정의하는 것은 비용의 적절한 배분과 조직에서 발생하는 다양한 비용들의 설명을 가능하게 한다. 그 할당 경로는 AIØWIN™ 의 Cost Matrix View 에서 편리하게 정의되어 질 수 있다.
Driver 의 설정과 할당경로 정의로써 기본적인 ABC 모델의 구조는 완성된다. 이 단계에서, 우리는 초기 ABC 모델로부터 완벽한 구조를 가진 ABC 모델로의 전환을 완료했다. 다양한 원천계정의 값을 정의함으로써 ABC 모델은 완성되어 질 것이다.

3.3.8 VIII단계 : 각각의 할당 경로에 Driver 값을 정의한다.

할당경로에 있어서, 만일 Driver 가 *evenly assigned* 로 되었다면 원천계정의 비용은 각각의 목적계정에 똑같이 분배되어 진다. 만약 Driver 가 *비율(%)*인 경우 모델 설계자는 각각의 할당경로에 비율(%) 값을 명세할 필요가 있다. 유사하게 Driver 가 *User defined* 로 정의된 경우 모델 설계자는 각각의 할당경로에 대해 그 Driver 와 연관된 비중을 명세해야 한다. 이러한 비율(%)이나 비중들은 원천계정에서 다양한 목적계정에 비율을 계산하는데 사용되어 진다. 각 Driver 값의 명세는 다음과 같이 사용되어 진다.

경험적 데이터 : 우리가 관심을 가지는 특정 기간동안에 객체 A 를 위한 주문건수가 100 이고, 객체 B 를 위한 주문건수가 150 인 경우, 만약 그 Driver 가 "주문건수"라면, Driver 값은 객체 A 와 B 의 각각의 할당경로에 100 과 150 으로 할당되어 진다.

관찰과 측정 : 작업자 A 는 평균적으로 생산품 P1 에 대하여 80%의 시간을 소비하고 생산품 P2 에 20%를 소비한다. 이것은 시간 분석을 통해 나온 것이다. 두 생산품의 비율(%) Driver 를 위한 Driver 값은 80 과 20 으로 각각 설정된다.

인터뷰와 평가 : 이 단계에서 우리는 작업자와의 인터뷰를 통해서 기계 M1 이 생산품 P1 을 처리하는데 소비하는 시간이 얼마인지 그리고 생산품 P2 를 얻기 위해 얼마나 많은 시간이 필요한지에 관한 정보를 얻는다. 두 생산품의 "소비된 시간"이라는 Driver 값은 이런 값들로 설정되어질 수 있을 것이다.

시뮬레이션 : 시뮬레이션은 모델 안에서 사용되어질 다양한 Driver 값을 결정하고 검증하는 강력한 방법 중에 하나다. 예들 들면, 작업장의 시뮬레이션을 통해 한 사람의 작업자가 다양한 활동에 소비하는 시간이 결정되어 질 수 있다. 이러한 값들은 비용모델의 적절한 배분경로에 있어서 Driver 값(양)으로 사용되어 질 수 있다. 시뮬레이션은 또한 비용모델의 결과를 검증하는데도 사용된다. 비용모델링과 시뮬레이션 통합은 IDEF Intelligent-Workbench™ 에서 지원 될 것이다. 비용 모델링에서 시뮬레이션 역할의 전망은 3.5 절에서 다루어질 것이다.

위에서 언급한 적당한 기술을 사용해서 다양한 배분경로를 위한 Driver 값이 비용 모델에서 결정되어지고 입력되어 진다. 이것은 AIØWIN™ 의 Cost Matrix View 나 SmartABC™ 에 의해서 생성되는 Excel™ Sheet 에서 편리하게 수행될 수 있도록 지원하고 있다.

3.3.9 IX단계 : 비용분석 기간을 결정하고 비용 데이터를 수집한다.

조직은 ABC 에 의해서 분석될 대상기간을 결정해야 한다. 이때 조직은 과제의 목표에 근거해서 수행되어져야 할 비용분석 대상기간을 결정해야 한다. 적절한 비용정보가 수집 가능해야 하는데 계획된 기간의 데이터가 적절치 않은 경우 분석대상 기간의 변경이나 추정치의 사용을 필요로 할 수도 있다.

비용정보의 가장 좋은 원천은 대개 총계정 원장이다. AIØWIN™ 는 총계정으로부터 비용정보를 포착할 수 있도록 지원한다. 관심기간에 대한 다양한 총계정 항목과 비용은 AIØWIN™ 의 총계정(General Ledger) View 에서 정의되어 질 수 있다. 총계정 원장에서 추출된 비용은 Cost Driver 와 배분경로를 통해 ABC 모델의 자원 및 활동항목에 할당되어질 수 있다. 이런 방법으로 ABC 모델에 비용정보를 반영시킬 수 있다. 이 단계는 AIØWIN™ 의 Cost Matrix View 나 SmartABC™ 에서 수행되어 질 수 있다.

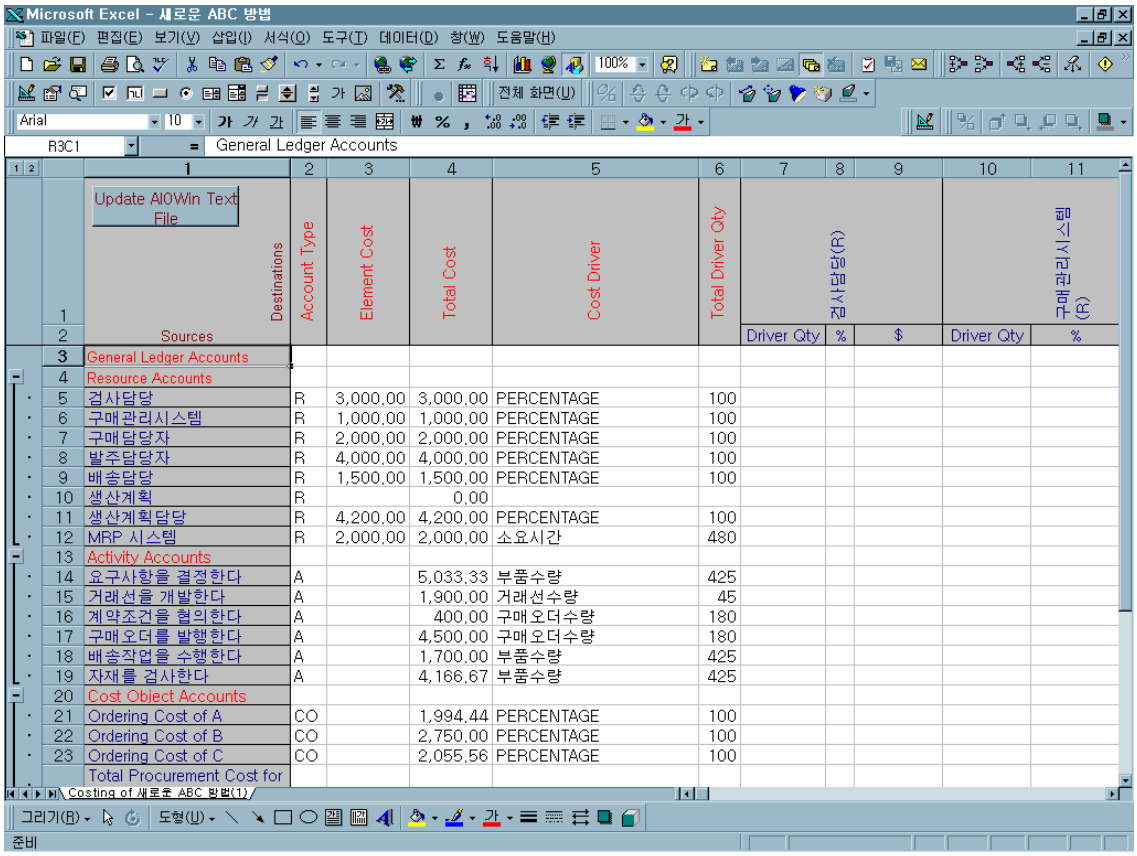


그림 3-7 : Excel 로 전환된 ABC 구조

3.3.10 X단계 : 관심을 가지고 있는 다양한 Cost Object들의 비용을 결정한다

IX 단계 작업 이후 ABC 모델은 그 전에 SmartABC™ 로 변환되어지지 않았다면 변환되어져야 한다. SmartABC™ 는 총계정 원장의 계정별 비용이나 모델의 다른 원천계정 비용, 그 외에 다양한 배분경로 및 Cost Driver 에 근거해서 다양한 Cost Object 의 비용을 계산하게 될 것이다. 우리가 수행하는 과제의 목표가 조직의 비용 구조를 결정하는 것이라면 우리는 과제를 이 단계에서 종료 시킬 수 있을 것이다. BPR 을 위한 추가적인 ABC 수행절차는 다음 절에서 상세하게 설명되어 질 것이다.

3.4. Business Process Reengineering(BPR) with ABC

3.4.1 비용과 기능적 정보의 통합에 의한 BPR

ABC 에서 활동은 각 활동의 중요성과 관련성에 관한 정보로서 '부가가치 활동 혹은 부가가치가 없는 활동', 또는 '꼭 필요한 혹은 생략 가능한 활동' 등으로 분류되어 진다. 이 아이디어는 부가가치가 없는 활동을 감소시키고 부가가치 활동을 증가 시키기 위한 것이다. 유사하게 이 목적은 하나의 조직에서 없어도 되는 활동을 감소시키는 것이다. 그러나 이러한 분류들은 3.3 절에서 강조했던 것처럼 매우 주관적인 것으로 BPR 은 왜 활동이 수행되는지 그 활동은 무엇을 하는지, 그 활동의 결과는 무엇인지 그리고 이러한 활동의 결과가 다른 활동이나 조직에 어떠한 영향을 미치는가와 관련된 보다 정확한 정보가 필요하다. 이것은 IDEFØ 기능 모델링에서 정확하게 다루어지는 정보의 형태인데 IDEFØ 방법론은 수행되는 다양한 활동들에 대한 정보로서, 왜 그것들이 수행되는가(활동의 Output 관점에서), 어떻게 그것들이 수행되는가(Mechanism 의 관점에서), 그리고 그 활동에 영향을 주는 인자는 무엇인가(Control 의 관점에서)에 관한 정보를 획득, 관리할 수 있도록 지원한다. 이러한 정보에 비용정보가 추가될 때 BPR 을 수행하는 분석가는 많은 비용을 소모하는 제품(서비스)을 추출할 수 있고, 왜 그리고 어떻게 그것들이 수행되어 지는지에 과하여 연구할 수 있으며, 이러한 정보로부터 업무를 수행하는 보다 나은 방법을 이끌어낼 수 있다. 따라서 활동에 대한 기능과 비용의 통합된 관점은 BPR 의 수행을 한층 용이하게 지원한다.

SmartABC™ 버전 1.0 은 이러한 활동에 대한 기능과 비용의 통합적 관점을 제공한다. 비용모델은 SmartABC™ (3.3.10 절)에서 생성되어진 후에 기능모델의 구성요소로써 다시 변환될 수 있는데 이는 SmartABC™ 모델의 비용과 관련된 정보를 AIØWIN 의 기능적 정보로 새롭게 갱신 시킨다. SmartABC™ 모델에서 AIØWIN 모델로의 변환을 위한 방법론은 3.7 절에서 설명되어 질 것이다.

3.4.2 프로세스 모델링을 통한 BPR

오늘날 ABC 의 중요한 한계는 아마도 업무가 한정적이고 정해진 방법에 의해 수행된다고 가정해야 된다는 것 때문일 것이다. 사실 어떤 활동을 수행하거나 목표를 달성하기 위해서는 다양한 방법이 있을 수 있다. BPR 은 업무의 수행에 요구되는 프로세스들을 수행하는데 있어서 잘 알려진 최선의 방법을 결정하는 프로세스다. IDEF3 는 미 공군에 의해 개발되어진 프로세스 모델링 방법론으로서 다양한 선택을 가능토록 하는데 관한 정보의 표현과 포착을 용이하게 한다. 업무수행에 있어서 최상의 방법을 결정하는 것 중 하나는 조직의 비용구조에 미치는 영향을 알기 위해 다른 선택 대안들을 평가해 보는 것이다. 그리고, 비용

관점에서 가장 좋은 경우를 선택하는 것이다.

PROSIM 은 KBSI 에 의해 개발된 IDEF3 모델링 Tool 이다. ABC 활용과 함께 사용되는 PROSIM 의 프로세스 모델링 기법은 비용과 관련된 관점에서 업무를 수행하는 대안적 방법의 분석을 용이하게 함으로써 BPR 수행을 위해 필요한 하부구조(Infrastructure)를 제공한다. SmartABC™ 는 BPR 과 ABC 에 보다 확장된 환경을 제공함으로써 PROSIM™ 과 SmartABC™를 통합하게 될 것이다.

3.5. ABC 에서 시뮬레이션의 역할

시뮬레이션은 모델링과 분석을 위한 강력한 도구이다. 시뮬레이션은 불확실하거나 임의적인 정보를 다루거나 복잡한 시스템의 활동관계를 단순화하여 분석하는데 유용하다. ABC 의 수행에 있어서 시뮬레이션은 두 가지 목적을 위해 사용되어진다.

3.5.1 시뮬레이션을 이용하여 Driver 값(양)을 결정한다.

ABC 는 원천계정의 비용을 다양한 목적계정에 할당하기 위해 배분경로와 Driver 값을 사용한다. ABC 모델의 비용구조는 다양한 할당경로에 의해 할당된 Driver 값에 크게 의존하는데, 예를 들면 자원계정 R1 이 소모하는 년간 비용이 10,000 이라고 가정하자. 이 자원은 두 가지 다른 활동계정 A1, A2 에 할당되어지는데, %(비율) Driver 를 위한 Driver 값은 관찰 결과 각각 30%, 70%로 결정되었다. 이러한 경우에 활동계정 A1 과 A2 의 비용은 각각 3,000 과 7,000 으로 결정되어 진다. 유사하게 Driver 값이 45%와 55%로 입력되어 졌다면 활동 계정 A1 과 A2 의 비용은 $4,500 과 $5,500 가 될 것이다. 이 때 활동계정 A1 과 A2 의 비용은 차례로 다른 계정들에 할당되어지게 되는데(예를 들면 Cost Object) 이때 초기 Driver 값의 편차는 비용모델 전체를 통해 영향을 미치게 된다.

따라서, 알맞은 Driver 값을 결정하는 것은 적합한 ABC 모델을 얻기 위해서 매우 중요한 사항이다. 비용의 할당은 어떤 원리에 근거해야 하며 무원칙하고 임의적인 할당이 되어서는 안 되는데 ABC 에서 이는 알맞은 Driver 를 사용하고, 알맞은 Driver 값을 사용함으로써 보장된다. Driver 는 비용이 할당되어지는 것에 관한 원칙을 밝혀낸다. 위의 예에서 자원 R1 의 비용은 두 활동들에 관해 소비된 시간에 근거해서 활동 A1 과 A2 로 할당되어질 수도 있을 것이다. 알맞은 Driver 가 선택되어지면 알맞은 Driver 값을 선택하는 것이 필요하다. 알맞은 Driver 가 선택됐지만 그 Driver 의 값이 정확한 값으로부터 벗어나 있다면 ABC 모델의 결과는 잘못된 의사결정을 유도할 것이다. 따라서 소요한 시간이 자원 R1 의 Driver 라면 두 활동 A1 과 A2 의 수행을 위해 필요한 자원 R1 의 소비시간에 관한 적절한 정확성을 가지고 결정되어야 한다.

알맞은 Driver 값을 결정하기 위한 다양한 방법은 3.4 절의 8 단계에서 소개되었다. 아마 시뮬레이션은 Driver 값을 결정하는 가장 강력하고 유연한 도구일 것이다. 예를 들면, R1 에 의해 수행되어지거나 R1 이 영향을 주는 인자들에 관한 활동의 시뮬레이션 모델이 만들어질 수 있다. 적당한 Mechanism 은 모델에서 활동 A1 과 A2 에 소비된 시간을 측정하기 위한 사항들을 지원한다. 수행된 그 시뮬레이션 수행의 결과들은 ABC 모델에서 Driver 값을 초기화 하는데 사용되어 질 수 있다. PROSIM^{TM} 은 Lanner Group 의 시뮬레이션 소프트웨어

인 WITNESS® 실행을 위해 프로세스 모델로부터 시뮬레이션 코드를 자동적으로 생성한다. 현재의 SmartABC™ 버전에서 ProSim™ 은 관련된 시뮬레이션 모델을 만듦으로써 Driver 값을 결정하는데 사용되어 질 수 있다. 그리고 나서 모델 설계자는 ABC 모델에서 직접 그 결과들을 갱신할 필요가 있다. SmartABC™ 버전 2.0 은 SmartABC™ 와 ProSim™ 을 통합하게 되는데 ProSim™ 에서 Driver 값을 결정하고 ABC 모델로 자동적으로 갱신하는 것이 가능하게 될 것이다.

3.5.2 ABC모델의 시뮬레이션 기반 검증

ABC 분석에 사용될 수 있는 시뮬레이션의 다른 영역은 ABC 모델을 검증하는 것이다. 예를 들어 어떤 ABC 모델이 활동 A1 이 수행되는데 150 의 비용이 소요되고 제품 P1 의 생산에 300 의 비용이 소요된다는 것을 나타낸다고 하자. 우리는 이를 위한 시뮬레이션 모델을 만들 수 있고 그 시뮬레이션의 결과는 ABC 모델의 결과를 검증하는 데 사용되어 질 수 있다. ABC 모델은 그 결과가 시뮬레이션으로부터 나온 결과와 합리적으로 일치하고 모순이 없다면 유효하다. 물론 이것은 시뮬레이션 모델이 (ABC 모델이 자체적으로 적절하다는 것을 포함하여) 그 자체로 유효하다는 가정을 전제로 한다. 시뮬레이션 모델의 유효성을 보장하기 위한 잘 개발된 기술과 절차가 있으며 이러한 기술들은 그 시뮬레이션 모델을 검증하는데 사용되어 질 것이다. 이것은 또한 역으로 ABC 모델을 검증하는데도 사용되어 질 수 있다.

3.6 기능적 모델 변환을 위한 비용 모델

기능모델로부터 ABC 모델을 생성하기 위한 절차는 3.4 절에서 소개되었다. SmartABC™ 는 또한 ABC 모델로부터 기능모델로의 역 변환(reverse translation)을 지원한다. 이 특징은 다음과 같은 작업에 유용하다.

① 기존의 ABC 모델로부터 기능모델의 생성, 그리고
② 3.5 절에서 설명되어진 것처럼 비용과 기능적 관점의 통합에 의한 BPR 의 수행.

ABC 모델을 기능모델로 변환하기 위한 방법론은 아래에 설명되어 있다.

3.6.1 I단계 : 초기 기능모델을 생성한다.

SmartABC™ 에서 변환되어질 비용모델을 기술한 후 "AIØWIN™ Update"를 선택한다. SmartABC™ 는 선택된 비용모델을 초기기능 모델로 자동적으로 변환하게 할 것이다. 초기 기능 모델을 가지고 작업하기 위한 최적의 장소는 SmartABC™ 의 Matrix View 이다.

3.6.2 II단계 : 초기 기능모델에서 개념들의 특성들을 다시 정의한다.

비용모델과 기능모델사이의 역 변환은 몇 가지 기본적 가정들에 근거한다. 이러한 가정들은 비용정보에 근거한 하나의 객체 기반의 기능적 특성에 관하여 추측 가능한 최고의 방법이다. 예들 들면, ABC 모델에서 하나의 자원계정은 기능모델에서 Mechanism 으로 변환되어질 수 있다. 이것은 이치에 맞는 기본적 가정이라고 보여지며 대부분의 일반적 상황에서 유효하다. 그러나 어떤 경우에 자원은 Control 로써 전환되어지는 것도 가능하다. 초기 기능모델의 모든 개념들은 검토되어져야 하고 적절한 전환으로 수정이 수행되어지는 것도 필요하다.

3.6.3 III단계 : 추가적인 활동들과 개념들을 추가한다.

비용 모델링과 기능 모델링은 활동을 보는 두 가지의 다른 관점이다. 하나의 관점에서 필요로 하는 중요한 것은 다른 관점에서는 중요하지 않다. 예들 들면, 하나의 활동이 기능적으로 매우 중요하다고 할지라도 비용이 낮은 활동들은 ABC 모델에서 생략될 수 있다. 마찬가지로 이런 불완전한 정보가 기능모델로부터 비용모델로의 변환에서 관련된다고 해도 그것은 역 변환에서는 뚜렷하지 않다. 이것은 기능모델이 일반적으로 보다 완성적이고 포

괄적이기 때문이다. ABC 모델로부터 기능모델을 생성할 때 사용자는 초기 기능모델에 나타나는 기능적으로 중요한 활동들과 개념들을 확인하고 검토할 필요가 있다. 어떤 활동이나 개념이 기능모델에서 빠져있다면 AIØWIN™ 의 Matrix View 에서 이러한 것들을 모델에 추가해 줄 수 있다.

3.6.4 IV단계 : 계층적 구조를 다시 정의한다.

비록 ABC 방법론이 계정센터(Account Center)의 형태로 계층적 구조를 지원하기는 하지만 그것이 계층적 구조를 강제하지는 않는다. 즉, ABC 모델 작업자는 어떤 계층적 구조도 사용하지 않고 모든 활동과 자원, 그리고 상품을 추상화 된 단일 레벨의 모델에서 표현할 수 있는데 이것은 적합한 ABC 모델로써 받아들여 진다. 그런데 이러한 비용모델로부터 생성된 기능모델은 유일한 계층적 구조의 수준에서 모든 개념들을 표현하게 될 것이다. 그러나 IDEFØ 와 같은 기능 모델링 기법들은 하나의 조직활동을 표현하고 서류화 하기 위해서 뿐 아니라 그것의 분석을 위해 설계된 것이다. 즉, IDEFØ 모델은 인간들에 의해 연구되어지고 분석되어지기 위한 것이기 때문이다. IDEFØ 는 사람이 모델을 분석하기 어렵게 되는 것을 피하기 위해 하나의 유일한 계층적 구조에서 너무 많은 개념들을 위치시키는 것을 피하도록 하고있다. 따라서, 비용모델로부터 생성된 초기 기능모델이 한 레벨에서 너무 많은 개념들을 가지고 있다면 모델 작업자는 더 높은 수준의 추상화로 개념을 추가함으로써 계층적 구조를 다시 정의할 필요가 있다. 경험적으로 볼 때 하나의 레벨에 6 개 이상의 활동을 동시에 표현하거나 또는 하나의 활동에 연관된 Input, Output, Control, Mechanism 이 각각 6 개 이상으로 표현되는 것은 피하는 것이 좋다. 만일 작성된 모델이 이러한 규칙을 만족시키지 못한다면 활동과 개념들을 보다 추상화 된 개념의 형태로 통합하는 것이 필요하다.

3.6.5 V단계 : 활동간의 상호작용을 다시 정의한다.

IDEFØ 는 하나의 활동의 Output 이 다른 활동의 Input, Mechanism, Control 이 되는 활동들 사이의 상호작용을 포착한다. 대부분의 이러한 상호작용은 ABC 모델에서는 포착되지 않는다. 따라서, 비용모델로부터 생성된 초기 기능모델에서는 나타나지 않을 것이다. 이러한 활동간의 상호작용은 AIØWIN™ 의 Cost Matrix View 나 다이아그램에서 정의되어져야 한다.
V 단계 이후에 기능모델은 완성되고, AIØWIN™ 에서 다이아그램의 형태로 보여질 수 있게 된다.

3.7 기능모델에서 비용모델로의 전환사례

제조업체인 XYZ 주식회사는 A,B,C 세가지 제품을 생산하고 있다. 생산계획 부서는 매달 생산될 각각의 제품 수량에 대한 생산 일정계획을 준비한다. 생산을 위해 필요한 자재들은 자재관리 부서에서 구매한다. 생산부서는 생산계획 부서에 의해 준비되어진 일정에 근거해서 원자재를 최종 생산품으로 변환하며 창고에 저장한다. 마지막으로 마케팅 부서는 최종 생산품을 고객에게 판매한다.

작년의 줄어든 시장 점유율은 경영진으로 하여금 제품 A, B 그리고 C 의 생산비용을 줄일 필요성을 인식하게 해주었다. 또한, 경영진은 각각의 상품에 대해 정확하고 믿을만한 원가 자료를 원하고 있다. 경영진은 제품 A 가 전통적으로 수입의 중요 원천이 되어왔지만 그리 이윤이 되는 것은 아니라고 생각하고 있다. 경영진은 제품 A 대신에 제품 C 의 생산량을 보다 많이 늘리는 것이 좋을 것이라고 생각하고 있다. 어쨌든 경영진들은 세가지 제품 A, B, C 의 생산량을 결정하기 위한 자료로서 A,B,C 각각에 대한 정확한 생산비용과 각각의 이익 기여도를 알고자 한다.

전통적으로 각 제품의 비용은 각 제품에 직접적으로 투입되는 재료비에 근거해서 결정되어졌다. 모든 간접비용은 재료비의 비율에 따라 제품에 할당되어졌다. 경영진은 이러한 비용 산정 방식은 각각의 제품에 따라 발생하는 실제 비용이나 예측되는 이익을 정확하게 제공하지 않는다는 것을 깨달았다. ABC 방법의 적용을 검토하던 XYZ 회사의 경영진은 ABC 가 그들의 문제를 해결해 주는지 파악해 보기로 결정했다.

이러한 배경으로 하나의 ABC 연구가 XYZ 회사에서 수행되어 졌다. 3.3 절에서 제안된 10 단계 프로세스는 SmartABC^{TM} 를 사용해서 수행되어졌다. 다음은 그 수행사례를 아래와 같이 요약한다.

3.7.1 I단계 : 과제의 경계와 목표를 정의한다.

직접적인 재료비에 덧붙여 제품원가의 두 가지 중요한 요소는 자재조달에 소요되는 비용과 생산비용이다. 자원적 제약을 고려해서 과제의 범위는 시험적인 ABC 시도로서 자재조달 비용을 결정하는 것으로 제한했다. 이러한 시험적 ABC 시도가 성공적으로 완성 된 것으로 판명된 후에 생산부문을 비롯한 다른 전체영역으로 ABC 분석을 확장하기로 결정하였다. 따라서, 시험적인 ABC 분석의 목표는 세 가지 제품(A,B,C)의 자재조달 결정하는 것이다.

3.7.2 II단계 : 활동, 자원, Output 그리고 이에 대한 계층적 구조를 정의한다.

ABC 분석을 위해 XYZ 사는 구매와 생산계획 부서를 인터뷰했으며 자재조달의 프로세스에서 수행되어지는 활동들을 결정했다. 다양한 활동들이 정의되었는데 :

'자재를 조달한다'는 활동은 다음과 같은 활동을 포함하고 있었다.

① 요구사항을 결정한다.
② 거래선을 개발한다.
③ 계약조건을 협의한다.
④ 구매오더를 발행한다.
⑤ 배송작업을 수행한다.
⑥ 자재를 검사한다. 이며

이러한 자재조달 활동을 수행하는 자원들은 :

'재고관리 시스템' 과 '자재조달 부서'가 있으며 이는 다시 다음과 같은 자원으로 구성된다.

① MRP 시스템,
② 구매관리 시스템
③ 생산계획담당.
④ 구매담당자,
⑤ 발주담당자,
⑥ 배송담당,
⑦ 검사담당 이다.

정의된 자재조달 활동의 Output 은 검수 'A,B,C 각각의 제품에 대한 자재'이고 이를 이루는 각각의 활동의 Output 은 다시 아래와 같다. :

① 자재 요구사항,
② 개발된 거래선,
③ 구매오더 사양,
④ 구매오더,
⑤ 공급된 자재,

3.7.3 III단계 : 기능모델을 만든다.

자재조달 프로세스의 기능모델은 II 단계에서 밝혀진 정보를 이용해서 AIØWIN™ 에서 만들어졌다. 이 단계에서는 어떻게 그것들이 다른 활동들과 연관되어 있는지 활동, 자원 그리고 활동의 Input, Output, Control 그리고 Mechanism 의 관점에서 다른 객체들 사이의 상호관계와 형식을 갖추게 되었으며, 활동과 개념(Input, Output, Control, Mechanism)의 계층 구조들이 결정되어 지고 모델화되었다. 그 결과 IDEFØ 모델은 그림 3- 8 과 그림 3-9 와 같이 작성되었다.

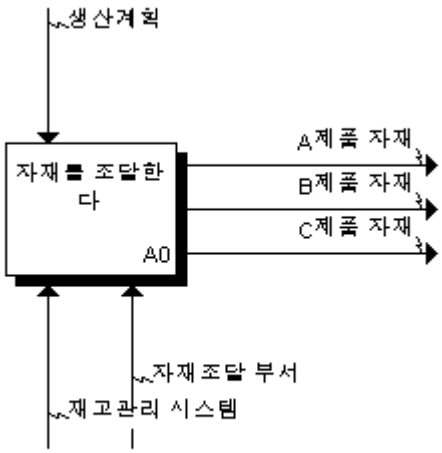
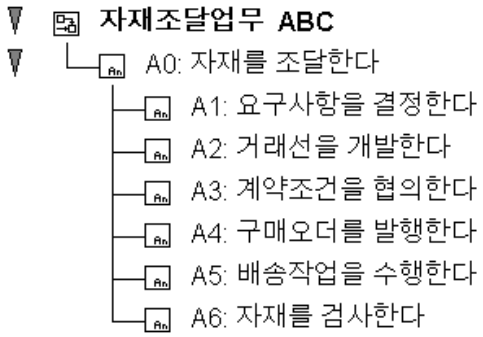


그림 3-8 : IDEFØ 모델의 최상위 레벨(Context Diagram)



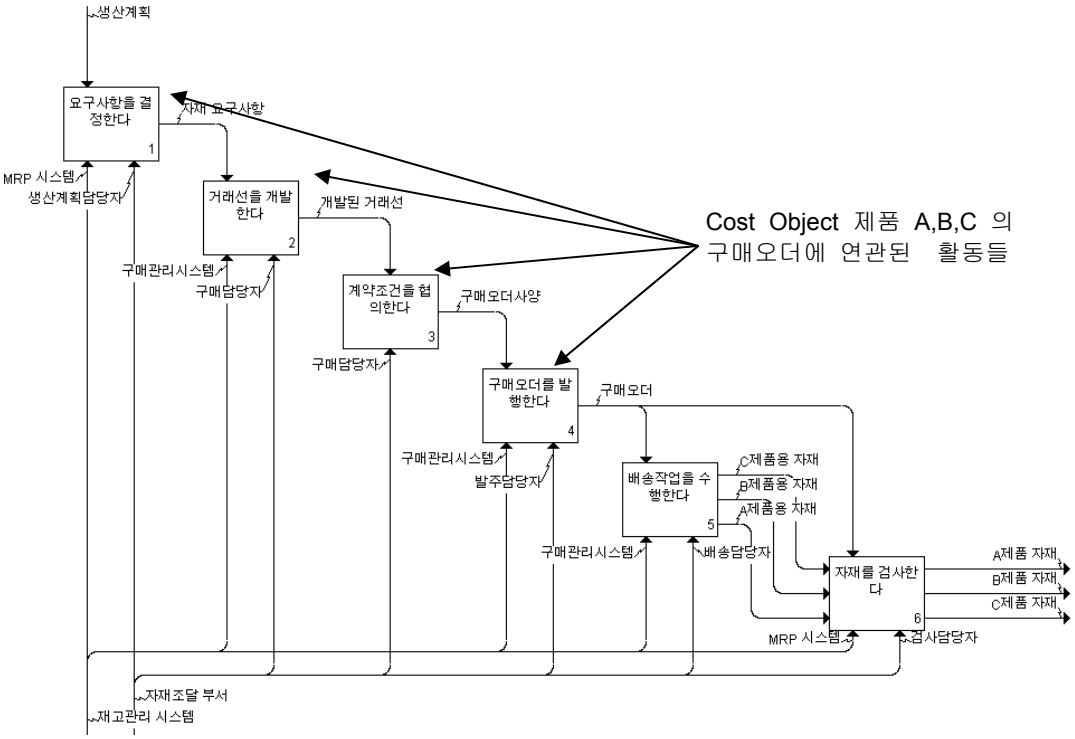


그림 3-9 : 분해된 IDEFØ 모델

3.7.4 IV단계 : 초기 ABC 모델을 생성한다.

III 단계에서 전개된 기능적 모델로부터 AIØWIN™ 을 이용하여 초기 ABC 모델을 생성한다.

ABC Matrix 보기

① Project Menu 에서 *Open Model…*을 선택한다 .

② Open Model 대화창에서, **자재조달**을 선택, Window Type group 박스에서 **ABC Matrix** 를

선택, 그리고 **OK** 를 클릭한다.

(주: 만일 여러분이 이미 자재조달을 선택하였거나, 다른 AIØWIN 윈도우를 Open 하였다면 단순히 그 모델에서 ABC matrix 를 보기위해  를 클릭하면 된다.

우리의 기능모델에서 ABC matrix 는 각 활동과 개념에 해당하는 기본 ABC 타입만을 리스트한다는 것을 주의하라. 비용모델에서 각각의 활동과 개념은 Source 와 Destination list 양쪽에 각각 그들의 엘리먼트에 해당하는 ABC 타입으로 나타난다. 이렇게 나누어져 표시된 리스트는 엘리먼트를 할당(assignment)하기 위하여 Source 와 Destination 양쪽에 모두 명시할 수 있도록 해준다. 자 이제 우리는 새로운 활동과 개념을 추가해 보자.

3.7.5 V단계 : 초기 ABC 모델을 다시 정의한다.

ABC Model에 개념 추가

다음으로, 우리의 모델에 새로운 개념을 만들어 추가해 보자. 우리는 모델에 cost object 를 먼저 추가하기로 한다. 첫번째 레벨은 개념을 위한 ABC 타입을 지정하는 것이다.

① Element menu 에서 **Cost Object** 를 선택한다.
② Element menu 에서 *Add Concept…*를 선택한다.
③ 이때 나타난 Concept 대화창에서, Name field 에 **제품 A 를 위한 주문비용**을 등록하고 **Create New** 를 클릭한다.
④ 같은 절차로, **제품 B 를 위한 주문비용** 과 **제품 C 를 위한 주문비용을** 추가한다.
⑤ 리스트에서 새로운 개념을 선택한 후 **OK** 를 클릭한다. ABC 모델에 개념을 추가하면 대화창이 닫힐 것이다. 이 때 새로운 개념은 ABC 타입이 cost object 인 그룹으로 등록될 것이다.

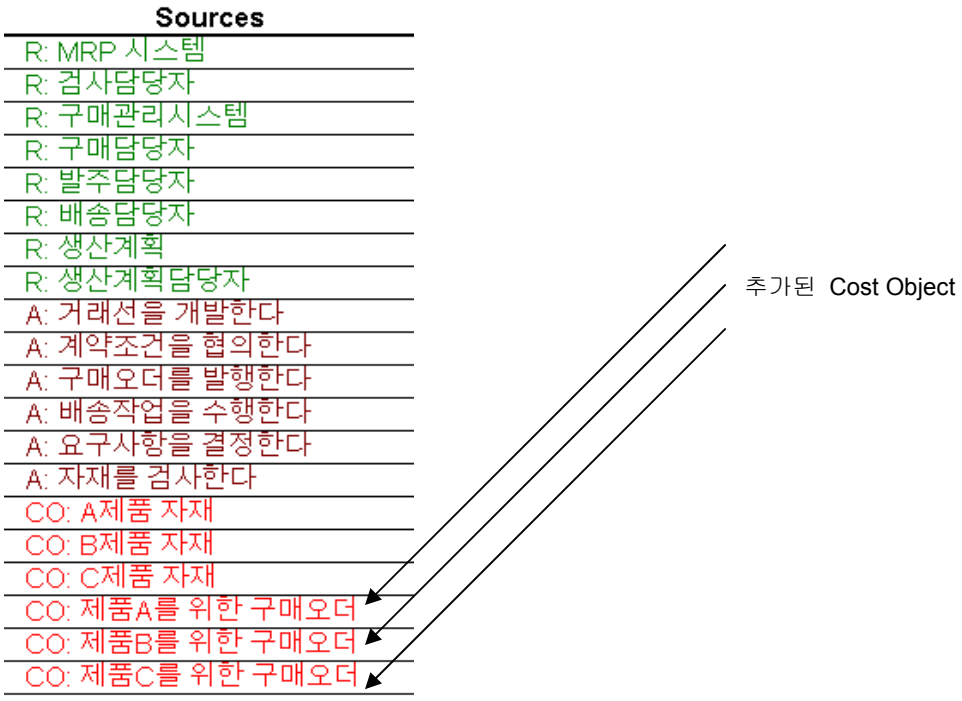


그림 3-10 : Cost Object

생성된 ABC 모델은 AIØWIN™의 Cost Matrix View 에서 다시 정의되어졌다. AIØWIN™에 의해 생성된 세 가지 Cost Object(A, B, C 각각에 소요되는 총 자재조달 비용)에 추가해서 세 가지 추가적인 Cost Object(A, B, C 의 구매오더)가 그 모델에 추가되어졌다. 이러한 Cost Object 들은 자재 주문에 소요되는 높은 비용때문에 중요하게 생각되어 졌다. XYZ 사의 경영진은 주문 비용을 감소시키기 위해서 주문 활동에 소요되는 각각의 제품에 대한 정확한 비용을 알고 싶어했다. ABC 모델의 남아 있는 구조(활동,자원 그리고 그들의 계층적 구조)는 비용 모델에서 요구되는 사항들과 일치된 상태로 남겨져 있다.

3.7.6 VI단계 : Cost Driver를 정의한다.

비용모델의 각각의 계정은 유일한 Driver 가 정의되어지며 다른 계정들에 할당되어 진다. 하나의 계정이 하나의 Driver 이상을 가진다면 그 계정은 분해될 필요가 있다. 그러한 분해들은 XYZ 사의 과제에서는 필요하지 않다. 다양한 계정들에 정의되어진 Cost Driver 는 표 3-1 에 요약되었다. 정의된 Driver 는 AIØWIN™의 Driver Pool 에 입력되었다.

계 정	DRIVER
MRP 시스템	소요시간
검사담당자	%
구매관리 시스템	%
배송담당자	%
발주담당자	%
구매담당자	%
생산계획담당자	%
거래선을 개발한다	거래선의 수
계약조건을 협의한다	구매오더의 수
구매오더를 발행한다	구매오더의 수
요구사항을 결정한다	부품의 수
배송작업을 수행한다	부품의 수
자재를 검사한다	부품의 수
제품 A 를 위한 구매오더	%
제품 B 를 위한 구매오더	%
제품 C 를 위한 구매오더	%

표 3-1 : 계정들과 그것들의 Driver 들

드라이버(Drivers) 생성과 할당

드라이버(Driver)는 비용모델에서 비용의 원천(source)과 관련이 있는데, 비용이 어떻게 Source 로부터 Destination 으로 할당되는지를 정의한다. 이러한 할당은 특정 Driver 타입에 따른다.

① Source 와 관련된 Destination 마다 똑같이 할당한다.
② 사용자에 의해 정의된 비율(percentage)에 따라 할당한다.
③ 사용자가 정의한 Source 와 연관된 계량 가능한 수에 따라 할당한다.

AIØWIN 에서 ABC 모델은 두 가지 타입의 드라이버를 가질 수 있다. 기본(Default) 드라이버는 사용자가 프로젝트를 새로 만들거나 추가할 때 각 AIØWIN 프로젝트 및 드라이버에 자동적으로 연관된다. 이들 두 가지 드라이버는 – Evenly assigned 와 Percentage – 드라이버의 각 Destination 에 Assignment path 에 따라 Evenly(예를 들면, 각각의 Destination 은 Source account 의 값을 동일한 비율로 분할 한다.) 혹은 당신이 정한 비율(percentage)의 값을 분배할 수 있도록 한다. 당신이 스스로의 드라이버를 생성할 때, 여러분은 드라이버의 Assignment path 에 대한 각각의 Destination 에 대하여 분배수량을 결정할 수 있다.
구매오더를 작성한다 활동과 관련된 드라이버는 세 가지 타입의 구매오더를 생성하는 비용에 영향을 줄 수 있다. 우리의 비용모델에서, 구매오더의 수량은 구매오더와 관련된 비용의 인자로 작동한다.

기본값으로, 우리의 ABC 모델은 Detailed View 에서 디스플레이 되는데, 이는 우리에게 우리의 Source 와 관련된 모든 드라이버들과 각각의 연관된 Destination 에서 Percentage/혹은 수량이 얼마나 할당되고 있는지를 보여준다. 우리는 이 연습문제를 시작하기 위하여 Detailed View 를 비활성화 시킬 것이다.

① Legend 를 Right-click 한 후 , *Detailed View…*를 다시 비 활성화 시킨다. 우리는 다음으로 우리의 모델과 관련된 필요한 드라이버들을 새로 만들 것이다.
② 구매오더를 발행한다 활동의 바로 오른쪽에 위치한 셀(드라이버)을 클릭한다.
드라이버 대화창을 선택하면 Driver pool 에서 Activity source 에 관련된 새로운 드라이버의 등록이나 기존의 드라이버를 선택할 수 있다.
③ Name field 에 **구매오더의 수**를 등록한다. **Shared radio button** 을 활성화 시키고, **Create New** 를 클릭한다.
Shared 의 활성화를 명시하는 것은, 여러분이 Source 에서 Destination 으로 Assignment path 를 작성할 때, Destination 으로 Assignment path 되는 다른 모든 것 들도 같은 수량을 공유한다는 것을 나타낸다.

④ New driver 를 선택하고 **OK** 를 클릭한다. 대화창이 닫히고 드라이버가 활성화된 Source 에 연결될 것이다.

이제 '구매오더를 발행한다' 활동과 관련된 Driver cell 은 "D"라는 표시가 나타나는데 이는 드라이버가 Source,와 연결되어 있음을 나타낸다.

⑤ Legend 를 Right-click 한다. 그리고 *Detailed view* 를 활성화 시킨다.

자 이제 남아있는 Source account 에 드라이버를 연관시켜 보자. 구매담당 자라는 resource 는 거래선을 개발한다 와 계약조건을 협의한다 활동의 Source account 로 작동됨을 기억하기 바란다. 우리는 다음 레벨에서 구매담당자라는 Resource 에 드라이버를 할당 할 것이다.

(주: 여러분은 Viewing field 에서 작업할 때 Scroll bar 를 이용하여 Matrix 의 밑이나 오른쪽에 있는 Source 나 Destination 을 볼 수 있다. 여러분이 이들 Matrix cell 위에서 커서를 움직일 때 Status bar 는 Source 와 Destination 이 교차되는 지점을 디스플레이 한다.

⑥ Source list 의 구매담당자 바로 오른쪽에 위치한 Driver cell 을 클릭한다.
⑦ 드라이버 대화창을 선택한 후, 리스트에서 **PERCENTAGE** 를 선택한 후 **OK** 를 클릭한다.

대화창이 닫히고, Activity window 로 돌아간다. 지금 Cell 에 나타난 PERCENTAGE 는 드라이버가 Source 와 연관되어 있다는 것을 나타낸다.

⑧ 같은 절차로 다음 테이블에 남아있는 Source 에 드라이버를 선택한다.

계 정	DRIVER
MRP 시스템	**소요시간**
검사담당자	**PERCENTAGE**
구매관리 시스템	**PERCENTAGE**
배송담당자	**PERCENTAGE**
발주담당자	**PERCENTAGE**
생산계획담당자	**PERCENTAGE**
거래선을 개발한다	**거래선의 수**
계약조건을 협의한다	**구매오더의 수**

요구사항을 결정한다	부품의 수
배송작업을 수행한다	부품의 수
자재를 검사한다	부품의 수
제품 A 를 위한 구매오더	PERCENTAGE
제품 B 를 위한 구매오더	PERCENTAGE
제품 C 를 위한 구매오더	PERCENTAGE

표 3-2 : 계정별 드라이버

자 이제 우리는 Source 에 Driver 를 할당하였다. 다음은 Assignment path 를 작성한다.

3.7.7 VII단계 : 배분경로를 정의한다.

한번 Cost Driver 가 정의되어지면, 배분경로는 다양한 계정들 사이에서 정의되어 진다. 배분경로는 원천계정의 비용을 할당하기 위한 구조를 제공한다. 배분경로의 정의는 원천계정과 목적계정 둘 다를 정의 하는 것을 포함한다. XYZ 사의 계정 배분경로는 표 3-2 로 요약되어 진다.

3.7.8 VIII단계 : Driver값을 정의한다.

Driver 값은 하나의 주어진 원천계정에 연관된 모든 배분경로와 관련된 비용할당 비중을 나타낸다. 하나의 원천계정이 하나의 이상의 목적계정을 가진다면 그 Driver 값은 다른 목적계정들에 할당되어질 수 있는 원천계정 비용의 비율로서 표시한다. Driver 값을 결정하는 다른 메커니즘은 3.4.절에서 소개했다. XYZ 사의 예에서, Driver 값은 표 3-3 에 요약되어 있다.

3.7.9 IX단계 : 비용분석의 기간을 결정하고 비용 데이터를 수집한다.

과제를 위한 이상적인 기간은 한 달로 결정되었다 1 월 한달간의 비용이 수집되고 ABC 분석을 위해 사용되었다. 이러한 값들은 표 3-3 에 요약 되었다. 다양한 원천계정의 비용은 AIØWIN™ 의 Cost Matrix View 에서 입력되어 진다.

원천 계정들(비용)	DRIVER	목적 계정들	DRIVER 값들
MRP 시스템 **(2000)**	소요시간	요구사항을 결정한다	**200**
		자재를 검사한다	**280**
검사담당자**(3000)**	%	자재를 검사한다	**100**
구매관리 시스템 **(1000)**	%	배송작업을 수행 한다	**20**
		구매오더를 발행한다	**50**
		거래선을 개발한다	**30**
배송담당자**(1500)**	%	배송작업을 수행한다	**100**
발주담당자**(4000)**	%	구매오더를 발행한다	**100**
구매담당자**(2000)**	%	계약조건을 협의한다	**20**
		거래선을 개발한다	**80**
생산계획담당자**(4200)**	%	요구사항을 결정한다	**100**
거래선을 개발한다	거래선의 수	제품 **A** 를 위한 구매오더	**15**
		제품 **B** 를 위한 구매오더	**20**
		제품 **C** 를 한 구매오더	**10**
계약조건을 협의한다	구매오더의 수	제품 **A** 를 위한 구매오더	**50**
		제품 **B** 를 위한 구매오더	**70**
		제품 **C** 를 위한 구매오더	**60**
구매오더를 발행한다	구매오더의 수	제품 **A** 를 위한 구매오더	**50**
		제품 **B** 를 위한 구매오더	**70**
		제품 **C** 를 위한 구매오더	**60**
요구사항을 결정한다	부품의 수	**A** 제품 자재 조달	**100**
		B 제품 자재 조달	**150**
		C 제품 자재 조달	**175**
배송작업을 수행한다	부품의 수	**A** 제품 자재 조달	**100**
		B 제품 자재 조달	**150**
		C 제품 자재 조달	**175**
자재를 검사한다	부품의 수	**A** 제품 자재 조달	**100**
		B 제품 자재 조달	**150**
		C 제품 자재 조달	**175**
제품 **A** 를 위한 구매오더	%	**A** 제품 자재 조달	**100**
제품 **B** 를 위한 구매오더	%	**B** 제품 자재 조달	**100**
제품 **C** 를 위한 구매오더	%	**C** 제품 자재 조달	**100**

표 3-3 : 배분경로, Driver 값 그리고 비용

할당경로(Path Assignments) 생성 및 드라이버 값 정의

두 엘리먼트 사이의 Assignment 는 Assignment 의 Destination 이 Source 비용의 일부를 사용하고 있음을 나타낸다. 즉 Source 의 드라이버는 비용이 Destination 에 할당되는 방법을 결정한다.

이들 Assignment path 를 위하여, 우리의 작업을 역으로 Cost object(제품 A 를 위한 구매오더, 제품 B 를 위한 구매오더, 제품 C 를 위한 구매오더)에서 시작하여 Source account 인 구매오더를 발행한다 사이에 Path 를 할당하는 것부터 시작하자.

① Source 인 구매오더를 작성한다 와 Destination 제품 A 를 위한 구매오더 사이에 놓인 빈 Cell 을 클릭한다. 이 때 나타나는 대화창은 사용자에게 Source 에 대한 Destination 을 추가하거나, Destination 에 대한 값을 명시하고, 혹은 Destination 을 제거할 수 있도록 한다.

② Destination group box 의 값으로 Quantity field 에 **50** 를 등록한 후 **Update** 를 클릭한다.

field 에 예측되는 값을 등록함으로써 우리는 50 개의 구매오더가 destination 에 할당됨을 가리킨다. 그리고 이 숫자가 비용의 흐름을 조정하는 것이다.

③ 대화창을 빠져 나오기 위하여 **Done** 을 클릭한다. 그리고 matrix 를 본다.

④ Source 인 구매오더를 작성한다 와 제품 B 를 위한 구매오더 Destination 사이에 있는 cell 을 클릭한다.

⑤ 이때 나타나는 Assignment Editor 대화창에서 Quantity field 에서 **70** 을 등록하고 Total Allocation group box 에서 **Done** 을 클릭하여 예측치를 갱신한다. Source 로부터 할당되는 전체 수량에 대한 것으로 %field 가 변경됨을 참고해야 한다. 제품 A 를 위한 구매오더의 %값 또한 새로운 Destination 에 할당되는 값을 반영하기 위하여 변경됨을 명심해야 한다.

대화창을 닫기 전에, 마지막 Destination 에 Assignment path 를 추가한다.

⑥ **Add Destination** 을 클릭한다. 이 결과로 Source 에 대한 연관 있는 Destination 의 리스트가 나타난다.

(주 : Source 의 ABC 타입은 여러분이 어떤 Assignment path 를 만들 수 있는가에 제약을 준다. 이와 같이 여러분이 source 와 destination 사이에 assignment path 를

만들었다면 관련되는 기능모델에 있어서 source 와 destination 의 계층구조 상에서도 역시 assignment path 에 제약을 가하게 될 것이다.)

⑦ 리스트를 Scroll down 한 후 **제품 C 를 위한 구매오더**를 선택한다. 그리고 **OK** 를 클릭한다. 선택된 Destination 이 Destination list 에 나타날 것이다.

⑧ Destination list 에서 **제품 C 를 위한 구매오더**를 선택한다. 그리고 Quantity field 에 **60** 을 등록한다. Total Allocation group 박스 안에 있는 숫자를 갱신하기 위하여 **Update** 를 클릭한다. source 로부터 할당된 총수량에서의 percentage 를 반영하기 위하여 % field 가 바뀌었음을 참고하라. Destination 제품 A 를 위한 구매오더와 제품 B 를 위한 구매오더의 %값도 새로운 destination 에 할당된 수량을 반영하기 위하여 변하였다.

⑨ 대화창을 닫기 위하여 **Done** 를 클릭한다.

자 이제 우리는 우리의 활동과 Cost object 들 사이에 Assignment path 를 만들었고, 우리의 다음 작업은 Resource 로 작동하는 Source 계정들과(구매관리 시스템, 발주담당자) 구매오더를 작성한다 활동 사이에 Assignment path 를 설정하는 것이다.

우리는 Assignment path 에 기본(default) 드라이버인 퍼센티지(percentage)를 쓸 것이고 따라서 SOURCE 의 전체 수량 중 명시된 Percent 가 Destination 에 할당 될 것이다.

⑩ Source 인 발주담당자와 Destination 인 구매오더를 발행한다 사이에 교차되는 빈 Cell 을 클릭한다.

⑪ 이때 Assignment Editor 대화창이 나타나는데, Destination group 박스의 Value % field 에 **100** 을 등록한 후 **Update** 를 클릭한다. Total Allocation group box 의 %field 가 이를 반영하기 위하여 수정되었음을 참고하라.

⑫ 대화창을 닫기 위하여 **Done** 를 클릭하고, Active window 로 돌아간다.

⑬ 같은 절차를 이용하여, 모델 안에 있는 다른 Resource account 로부터 구매오더를 발행한다 로 Assignment path 를 설정한다. 다음의 테이블은 관련된 각 Source 에 대하여 Values for Destination group box 안에 있는 Assignment Editor 대화창의 %field 에 등록하기 위한 %할당치를 나타내고 있다.

Source	% Allocation
구매관리 시스템	**50%**

⑭ 대화창을 닫기 위해 **Done** 을 클릭한다. 그리고 Active window 로 돌아간다.

우리가 모델에서 포착한 것들은 자원계정(구매관리 시스템 및 발주담당자)에서 세 개의 Cost object(제품 A 를 위한 구매오더, 제품 B 를 위한 구매오더, 제품 C 를 위한 구매오더)에 이르는 시스템에서의 비용의 흐름이다. 구매관리 시스템의 총 비용 중 50%는 구매오더를 발행한다에 할당되었으며, 실질적으로 제품 A 를 위한 구매오더를 발행하는데 는 구매관리 시스템 비용의 13.8%의 노력이 소요되었다. (50% x 50 / (50+70+60) = 13.8%) 세 가지의 구매오더가 '구매오더를 발행한다' 활동에 의해 생성되었다. 이런 경우에, 생성된 구매오더의 수는 활동의 비용을 발생케 하며, 활동에 의해 생성되는 전체 구매오더 수량 중 각각의 구매오더의 수량으로 이들의 생성 비용을 가늠할 수 있도록 한다.

⑮ 위와 같은 방법으로 표 3-3 에 있는 나머지 Assignment Path 를 모두 작성한다.

원천계정의 비용등록

① Source 계정의 구매관리 시스템 셀을 right-click 한 후 나타난 항목 중 *Edit Assignments...*를 선택한다.
② 나타난 Assignment Editor 박스에서 Value 항목에 원천계정의 비용으로 표 3-3 에 나타난 **1000** 을 등록한다.
③ Done 을 클릭한 후 Active 창으로 되돌아 온다.
④ 표 3-3 에 나타난 MRP 시스템, 검사담당자, 배송담당자, 발주담당자, 구매담당자, 생산계획 담당자의 원천계정 비용을 위와 같은 방법으로 등록한다.

3.7.10 X단계 : 다양한 Cost Object들의 비용을 결정한다.

현재 리포지토리의 저장 및 종료

① File menu 에서 *Save As* 를 선택한다.
② File name field 에서 당신의 리포지토리(예를 들면 - Tutorial.ai0)를 위한 이름을 등록한 후 Ai0win\example 디렉토리를 Browse 한다.
③ **OK** 를 클릭한다.

SmartABC를 위한 ABC모델의 Exporting

① File menu 에서 Export 를 선택한다.
② Format of Output File 박스에서 AIØWIN.(*TXT)를 선택한다.
③ Export Text Option 박스에서 모든 옵션을 선택한 후 OK 를 클릭한다.

④ Export 된 File 을 저장하기 위한 디렉토리와 File 명을 등록한 확인을 클릭한다.

여러분이 **OK** 를 클릭하면, Job Status 대화창이 나타나고, Export 의 진행상태를 여러분에게 알려줄 것이다. 여러분은 Stop 을 눌러서 이를 중단할 수 있다. Export 가 완료되면, 대화창이 나타나는데, Export 가 성공적으로 완료되었음을 여러분에게 알려준다.
대화창을 닫기 위해 **OK** 를 클릭하면 Active window 로 돌아간다.

이제 ABC 모델은 SmartABC™ 가 Import 할 수 있는 형태로 Export 되었다. 여러분은 Export 된 file 을 SmartABC™ 에서 Import 할 수 있으며 여러분의 필요에 따라 이를 수정할 수 있다. SmartABC™ 는 AIØWIN™ 으로부터 Export 된 ABC 모델의 구조에 근거하는 다양한 Cost Object 들의 비용을 자동적으로 계산한다. 다양한 Cost Object 들의 계산된 비용은 표 3-4 에 요약되었다.

Cost Objects	비 용
제품 A 의 구매오더(를 발행하기 위하여 관여된 총 비용)	**1894.44**
제품 B 의 구매오더	**2638.89**
제품 C 의 구매오더	**2166.67**
A 제품의 총 조달 비용(요구사항 결정부터 자재검사까지)	**4482.68**
B 제품의 총 조달 비용	**9521.24**
C 제품의 총 조달 비용	**6696.08**

표 3-4 : ABC 결과의 요약

이로써 XYZ 사의 실험적 ABC 노력이 성공적으로 완성되었다 제품 A 의 조달비용은 다른 상품들 보다 현저히 낮은 것으로 밝혀졌으며 XYZ 사 경영진은 B 와 C 의 조달비용을 줄이는데 BPR 노력을 집중하기로 결정했다. 반면에, 그 경영진은 제품 A 를 보다 많이 생산하기로 결정했다.(예상외로 다른 제품에 비해서 조달비용이 가장 낮은 생산품이기 때문이다.)

4. IDEF3 프로세스 모델링 방법론(Process Description Capture Method)

4.1 IDEF3 개요

4.1.1 프로세스 모델이란 ?

제임스 스와츠(James B. Swartz)는 "…..가치전달 시스템에서의 여러가지 일하는 방법들 ….."이라고 정의하고 있는데 이는 다시 말하자면 다음과 같은 것이다

① 프로세스의 표현 그리고 시간의 관점에 따라 표현된 요소들
② 프로세스 안에서 존재하는 의사결정 논리

상황(Situation)이나 프로세스(Process)를 설명하는 가장 일반적인 커뮤니케이션 메커니즘 가운데 하나는 사건이나 활동을 시간의 연속적 순서로 표현하는 이야기(Story)이다. IDEF3는 이러한 형태의 설명적 활동(Story)을 표현하기 위하여 특별히 개발된 시나리오 지향적 프로세스 흐름의 모델화 방법(Process-driven process flow modeling method)으로서 주어진 환경에서 어떤 상황이나 사건의 원인과 결과에 대한 전문가의 설명을 바로 포착, 표현할 수 있도록 구축되었다. IDEF3 은 미 공군의 주도하에 Knowledge Based Systems Inc.가 수행한 IICE(Information Integration for Concurrent Engineering) Program 에서 개발되었으며 1992년 5 월 IICE Process Description Capture Method(IDEF3)로 발표되었다. IDEF3 의 목표는 특정 시스템이나 조직이 작동되는 방식에 관하여 그 부분에 대한 전문가의 지식을 표현하는 구조적 방법을 제공하는 것이다.

■ 왜 프로세스 모델을 작성하는가 ?

프로세스 모델은 프로젝트의 사실 발견을 위한 시스템 분석 인터뷰에서 나오는 원시 데이터를 기록, 분석하는 시스템적 방법을 제공하는데 다음과 같은 것이 가능토록 지원한다.

① 기업에서 중요 운영 시나리오에 따라 정보 자원이 조직에 미치는 영향을 결정한다.
② 중요한 공유 데이터(특히 생산.엔지니어링.유지 보수.제품 정의 데이터)의 상태와 라이프 사이클에 영향을 주는 의사결정 절차의 문서화 메커니즘을 제공한다.
③ 데이터 구조 관리와 통제 정책 결정의 변화에 관한 정의를 내린다.
④ 시스템 디자인과 디자인 트레이드 오프(trade-off) 분석을 지원한다.
⑤ 시뮬레이션 모델의 생산을 지원하는 강력한 메커니즘을 제공한다.
⑥ IDEFØ 기능 모델의 개발을 위한 유용한 정보를 제공한다.

⑦ 사실.의사결정 시점.업무 분류 등을 명확하게 정의하는 메커니즘을 제공함으로서, 실시간 통제를 이룩하는 소프트웨어 디자인을 위한 프로세스 매핑(mapping)을 촉진한다.

⑧ 사용자 관점에서의 요구 사항과 개발하는데 필요한 데이터를 명확하게 정의하는 방법론을 분석가에게 제공한다.

⑨ 전문가 시스템 개발에 필요한 전문가의 관점을 포착하고 표현한다.

프로세스 모델링의 기본적인 전제는 개선의 노력이 제품 그 자체가 아닌 프로세스에 집중되어야 한다는 것이다. 그 것은 제품이 아니라, 제품을 만드는 프로세스만이 회사의 장기적인 성공을 보장할 수 있기 때문이다.

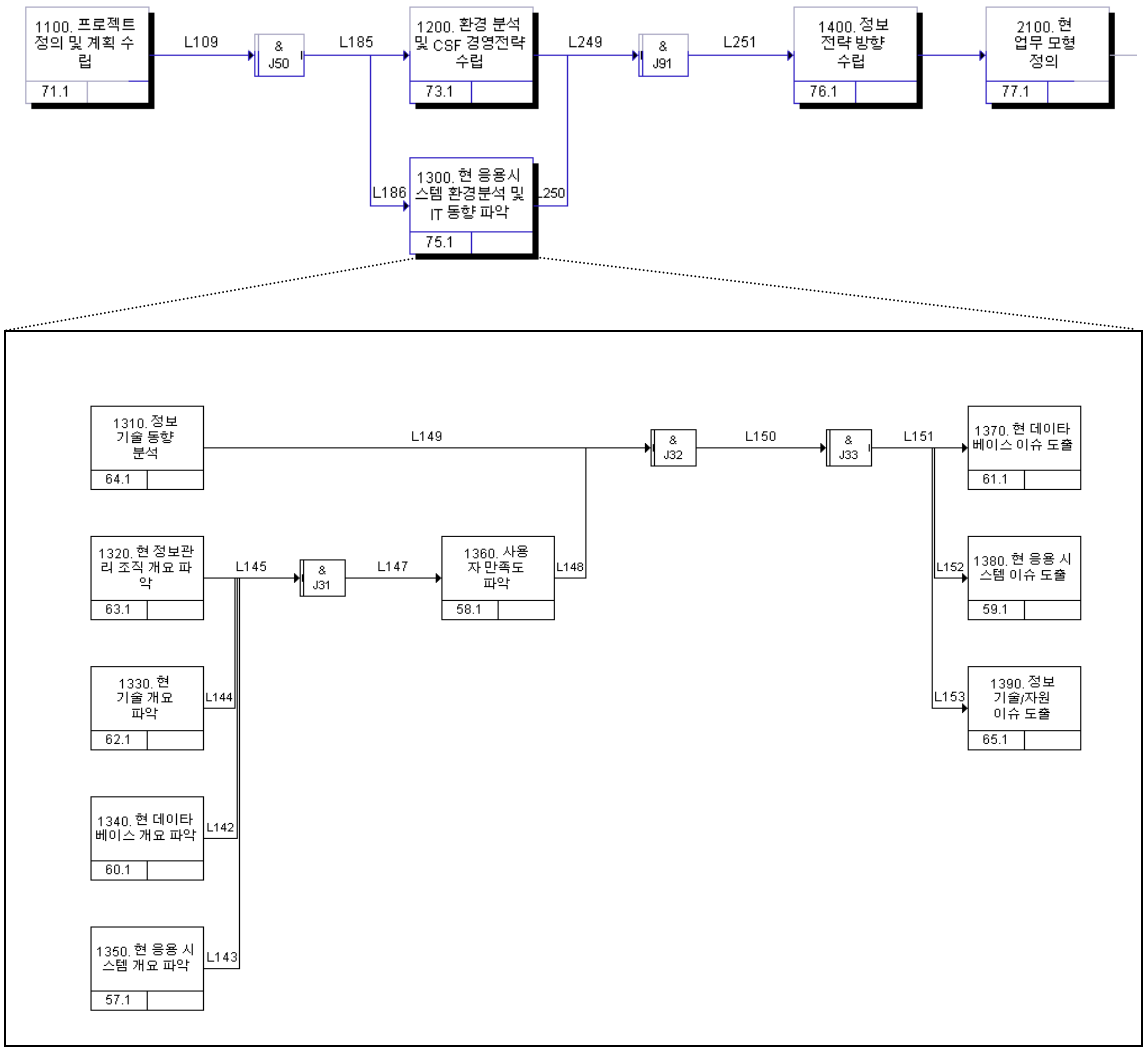


그림 4-1 : IDEF3 프로세스 모델의 예

■ IDEF3는 무엇인가 ?

IDEF3 모델링 방법론은 크게 두 가지의 분리된 방법으로 이루어져 있다.

① 프로세스 설명 포착방법
② 개체상태 전환 설명방법

IDEF3 은 시스템이 '어떤 방법으로 어떻게 되어야 한다'는 설명을 표현하고자 하는 필요에 의해 개발 되었으며 시스템에 관한 여러 사용자의 관점을 표현하고 조합하기 위한 언어로서, 시스템이 '주어진 조건에서 무엇을 얼마나 할지를 예측'하는 수치적 이상치를 알아내기 위한 시뮬레이션 언어군(즉 QGERT, SLAM 등) 과는 다른 목적에서 개발 되었다. 시뮬레이션 언어는 시간별로 달라지는 시스템 자원의 행동을 표현하고, 수량-모델-기반적(math-model-based) 시뮬레이션의 특성에 대한 틀을 제공한다. IDEF3 는 이에 반하여 시스템에 관한 구조적, 논리적 모델을 사용자에게 지원 함으로서 프로젝트 구성원간의 커뮤니케이션을 촉진하고 구축 되어야 할 시스템의 표현을 위한 구조적 틀을 제공한다. IDEF3 에는 두 가지 모델화 양식이 존재한다. 프로세스 흐름 묘사(Process Flow Description), 그리고 객체 상태의 전이 묘사(Object State Transition Description)이다. IDEF3 프로세스 모델링 방법은 기존의 혹은 제안된 시스템의 '행동적' 측면에 관한 중요 전문가의 지식을 파악하려는 시스템 개발자들이 사용한다. IDEF3 를 이용하여 파악된 프로세스 지식은 시나리오의 전후관계 속에서 구조화 되며, IDEF3 는 시스템의 묘사를 위한 직관적 지식 습득 도구의 역할을 수행한다. 일반성과 단순성을 촉진하기 위하여, 시스템의 단일 관점을 채택하여 모든 일시적 논리를 확실히 제거하는 IDEF∅모델과 달리, IDEF3 는 기업 프로세스에서 발생되는 일시적 원인과 결과에 관한 여러 사용자의 설명을 수용, 지원한다. 결과적으로 IDEF3 설명은 분석이나 설계를 위한 모델이 구축될 수 있는 구조적 지식 기반을 제공한다.

■ IDEF3는 무엇을 표현하는가 ?

프로세스 흐름 묘사는 조직 내에서 '일하는 방식'에 관한 지식을 파악하려는 것이다. 객체 상태 전환 묘사는, 객체가 특정 프로세스를 통과할 경우 발생 가능한 전환 상태를 표현한다. 프로세스 흐름 묘사와 객체 상태 전환 묘사는 모두 묘사를 구성하는 정보 단위를 내포한다. 이들 모델 실체(entities)는 IDEF3 설명의 기본적 단위를 구성한다.

이 결과 다이어그램과 텍스트로 이루어지는 '묘사(Description)'는 다른 IDEF 방법에서 만들어지는 '모델(Model)'과는 상이한 개념인데 이 차이는 중요하다.

앞서 검토했듯이 모델은 주어진 현실 세계 시스템의 중요한 측면을 모방하도록 디자인 된,

객체. 소속. 관계에 대한 하나의 개념적 시스템이다. 아주 현실적 의미에서 모델은, 현실적 시스템의 가정화. 단순화를 중심으로 구축된 시스템 그 자체이며 그것은 모델이 현실적으로 적용되는 예측된 상황의 범위 내에서 확실히 유효하다고 가정하고 있다. 따라서 모델은 그 유용성을 보증하기 위하여 완전해야 하며 내부적으로 일관되어야 한다.

그러나 묘사는 일반적으로 불완전하다. 예를 들어 시스템 분석 활동에 있어서 분석자에게 제공되는 각 부문 전문가의 설명은 설명자 자신의 관점에서 인식된 사실에 대한 표현일 뿐이며 이는 한정된 범위 내에서만 일반적인 모델로서 전환되어질 수 있다. 즉 이러한 묘사에서는 전문가는 전문 영역 밖에 있는 프로세스나 사건에 대하여 일부를 알 수는 있겠지만 그 전체적 사실을 알 수 없는 경우가 많으며, 전문가와 관계가 없거나 그 순간 단순히 잊혀졌기 때문에 주어진 설명에서 생략될 가능성이 있다. 묘사는 단지 불완전하지만 진실되다고 가정된 우리의 주변 세계에 관한 사실과 믿음에 대한 기록일 따름이다. 모델이 일정한 척도와 규칙에 따라 작성되어 공식적으로 인정된 지도라면 묘사는 여러분이 알고있는 한 지점에서 주변의 다른 지점으로 이동하기 위한 경로를 순차적(시간) 서술적으로 기술하는 시나리오(약도)와 같다. 시스템 분석 활동의 목적이 현 시스템의 기능, 정보, 프로세스에 관련된 사실과 관계를 정확하게 표현하는 모델의 구축에 있다면 이에 앞서서, 유력한 전문가에 의해 제공되는 묘사의 수집은 필수적인 것이다.

4.1.2 FLOW CHART와 IDEF3 프로세스 모델의 비교

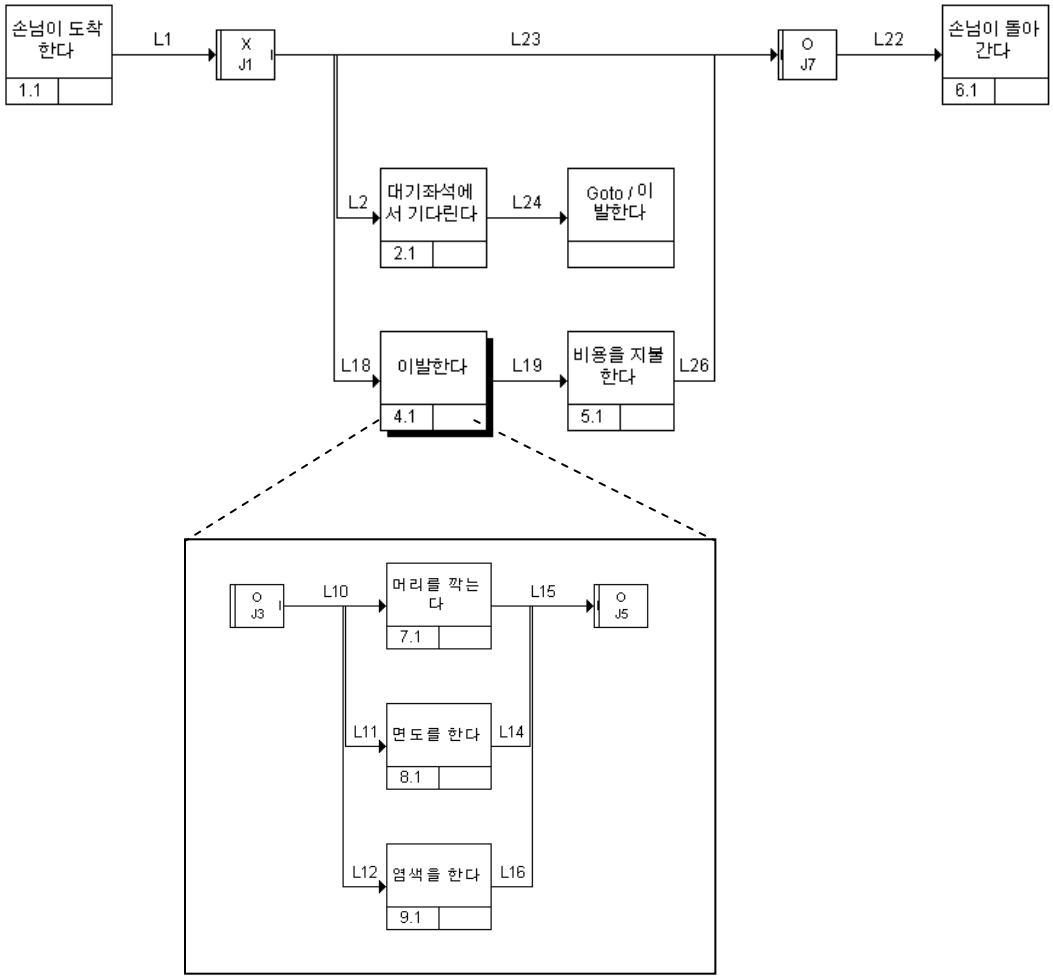


그림 4-2 : 이발소업무 프로세스

우리는 플로우차트(Flow Chart) 와의 비교를 통해서 더욱 쉽게 IDEF3 프로세스 모델을 이해할 수 있다. 그림 4-2 의 IDEF3 프로세스 다이아그램이 나타내고 있는 내용은 우리가 일반적으로 접할 수 있는 이발소의 프로세스를 이발소 주인의 관점에서 나타내고 있다. 구현된 시나리오의 내용을 이발소 주인과의 인터뷰를 바탕으로 서술적으로 나타내면 다음과 같다.

① 이발소에 손님이 도착한다.
② 도착한 손님은 대기하는 손님이 많은 경우 그냥 돌아가기도 하고
③ 대기좌석에 앉아 기다리는 경우 도 있는데 기다리다가 차례가 되면 이발을 한다.
④ 대기하는 손님이 없는 경우 도착한 손님은 바로 이발대에 앉아 이발을 시작한다.
⑤ 이발소에서 제공하는 이발 서비스는 세가지로 나눌 수 있는데 특별한 순서 없이 손님

의 요청에 따라 머리를 깎거나, 면도를 하거나, 염색을 하는 것이다. 물론 손님에 따라 이중 두가 지나 세가지를 모두 하기도 한다.

⑥ 이발을 마친 손님은 비용을 지불한 후 이발소를 떠난다.

우리는 위의 IDEF3 프로세스 다이아그램을 통하여 이발소에서 발생하는 업무프로세스를 파악할 수 있다.

여기서 우리는 IDEF3 의 특성중 하나인 두 개의 논리적 분기를 찾아 볼 수 있는데 그 하나는 도착한 손님이 그냥 돌아가던지, 대기좌석에 앉아 기다리던지, 아니면 바로 이발대에 앉아 이발을 시작하던지 중에서 한가지만 선택할 수 있다는 배타적 선택논리(J1 으로 표시된 작은 박스)가 그 하나이고 다른 하나는 이발을 시작한 손님이 제공되는 서비스(커트, 면도, 염색) 중에서 한가지 이상을 원하는바에 따라 선택이 가능하고 이들 서비스가 동시에 수행되어야 하는가를 나타내는 시간적 동시성의 논리이다(J3 로 나타난 작은 박스). IDEF3 프로세스 다이아그램은 프로세스의 흐름에서 우리가 익히 알고있는 논리게이트(AND/OR/XOR)와 같은 프로세스의 분기의 구조와 함께 이러한 분기 이벤트의 동시성 여부(Synchronous / Asynchronous)를 표현 할 수 있는 방법을 지원 함으로써 프로세스 흐름의 확실한 시간적 논리적 표현이 가능토록 한다.

IDEF3 의 또 다른 하나의 특성은 프로세스의 분해를 통한 레벨구조의 지원인데 그림 4-2 는 '이발을 한다'는 프로세스를 분해하여 레벨화 함으로써 특정 프로세스의 계층화 구조를 보여주고 있다. 이러한 계층구조의 지원은 모델 작업자로 하여금 추상화된 다양한 레벨을 조작할 수 있도록 지원하여 프로세스 표현의 융통성을 확장한다. 또한 IDEF3 프로세스 다이그램은 이러한 시나리오적 다이아그램을 바탕으로 각 프로세스에 관여되는 객체들에 대한 다음과 같은 정보를 수집하는데 이는 이러한 프로세스를 기반으로 구축되어질 전문가의 지식을 수집하고 객체중심의 프로세스 표현방법인 시뮬레이션 모델생성의 기초를 제공한다.

① 어디에서 프로세스가 수행되는가?
② 누가 혹은 어떤 장치가 이러한 프로세스의 수행에 관여하는가?
③ 프로세스의 수행으로 생성되거나 변형되어질 부품이나 정보는 어떤것인가?
④ 프로세스의 수행으로 인하여 가공되는 부품이나 정보의 상태는 어떻게 변하는가?

이와 같은 프로세스와 관련된 객체의 정보를 표현하는 구체적인 방법은 다음 절에서 논하도록 한다.

IDEF3 프로세스 다이아그램과 Flow Chart 의 비교를 돕기위해 위에 그림 4-2 를 다시 그림 4-3 과 같이 Flow Chart 로 표현했다.

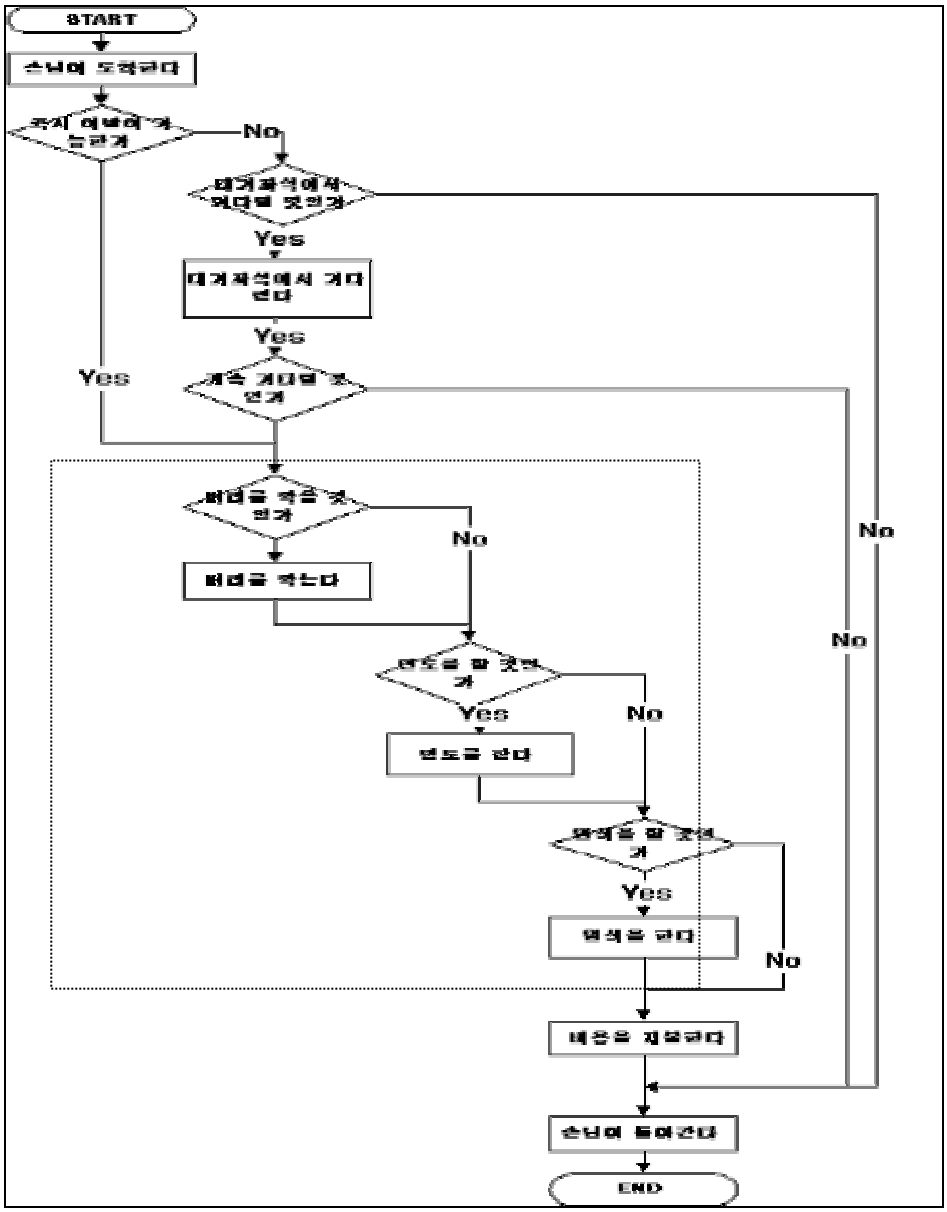


그림 4-3 : FlowChart

■ Flow Charting 과 Process Modeling 의 차이점

Flow Charting	Process Modeling
• 모호한 방법의 프로세스 논리전개 • 추상화 된 다양한 레벨을 포착할 수 없다. • 프로세스 내부에서 개체의 정보를 제공하지 않는다.	• 모호하지 않은 문법의 프로세스 논리전개 • 추상화 된 다양한 레벨의 포착 • 프로세스에 개체와 시뮬레이션 자료를 등록할 수 있다

4.2 IDEF3 모델링 표현 방법

IDEF3 프로세스 흐름 묘사는 특정 시나리오의 전후관계 속에서 행동간에 관계되어진 네트웍을 파악한다. 이 묘사의 목적은 특정한 문제 해결(혹은 발생) 상황의 전후 관계 속에서 특정 조직 내의 일 처리 방식을 보여 주는 것이다. IDEF3 는 프로세스 묘사를 위한 경계 조건과 초점을 확립하는 기본적 조직 구조로서 '시나리오'를 활용한다. 이러한 특성은 인간이 어떤 상황 속에서 경험하거나 관찰하여 알게 된 것이나, 예정하고 있는 계획에 관한 내용을 활동의 시간적 순서에 의해 묘사하려는 인간적 경향에 기반을 두고 있다. 프로세스 묘사의 전후관계 속에서 생각과 표현을 정리하려는 이러한 자연적 경향은 어떤 설정된 조건에서 행동의 구성을 표현하는 역할로서, '외부적 관점'에서 볼 때 구현 가능한 시스템의 디자인을 제안하기 위한 비공식적 틀(Framework)에서 시나리오의 광범위한 사용을 촉진시켰다. 이러한 개발 방식을 '외부 조건 지향적 디자인' 방식이라고 하며 실행 결과 새로운 시스템 디자인을 위한 효과적 메커니즘이란 점이 계속적으로 입증되었다.

4.2.1 다이아그램 문법

IDEF3 프로세스 흐름 다이어그램은 다음과 같은 네 가지의 요소로 구성되며 그림 4-4 는 IDEF3 프로세스 흐름 다이어그램의 예를 보여 주고 있다.

① 행동 단위(UOB, Units of Behavior) : 박스 형태로 표현되는 행동 단위
② 연결(Links) : 화살표로 표현되는 UOB 의 선후 연관 관계
③ 접속(Junctions) : UOB 의 분기 및 결합 시 사용되는 시간, 논리적 관계
④ 참조 사항(Referents) : UOB 와 관련된 부가적 내용을 도형적으로 표현하는 박스

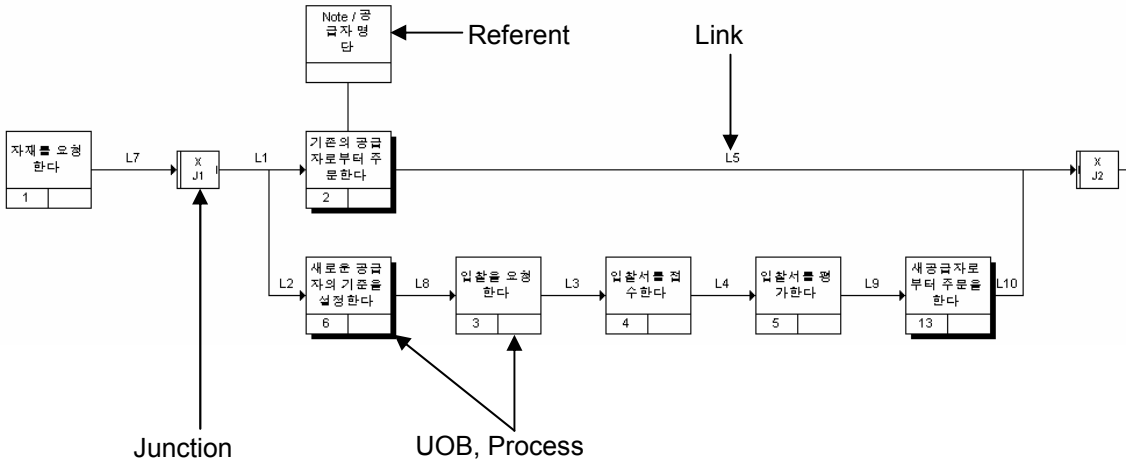


그림 4-4 : IDEF3 Process Flow Diagram 의 예

■ 4.2.1.1 UOB(Unit of Behavior)란 무엇인가?

프로세스는 현실세계 시스템에서 발생하는 기능이나 절차의 일반적인 표현으로서 IDEF3 에서 프로세스는 다음과 같은 것들을 표현한다.

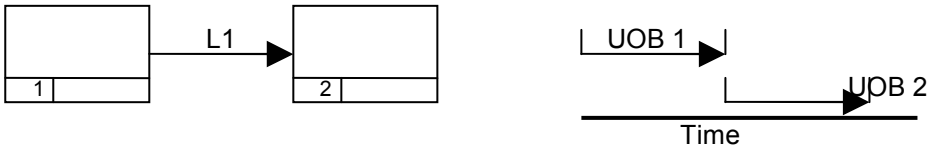
기능(Functions)	행동(Actions)	프로세스(Processes)
활동(Activities)	행위(Acts)	운영(Operations)
사건(Events)	시나리오(Scenarios)	의사결정(Decisions)
절차(Procedures)		

IDEF3 프로세스 흐름 다이어그램은 그 주변 구조에 따라 기능. 활동. 행위. 프로세스. 운용. 사건. 시나리오. 의사결정. 절차 등으로 정의 될 수 있는 UOB (Units of Behavior)를 기본적 문장 구성 단위로 한다. 주어진 시나리오의 전후관계 속에서 사용되는 각 UOB 는 이들 시나리오를 구성하는 표현 행위로서, 각 UOB 는 사건의 인지된 상태 혹은 주어진 시나리오와 연관된 개체의 변화상태에 따라 세상에 대한 특정한 관점을 표현한다. IDEF3 에서, 각각의 UOB 는 다이아그램 내에서 단위행위를 나타내는 이름이 붙은 하나의 박스로 표현된다. 하나의 식별번호는 UOB 가 생성될 때마다 순차적으로 붙여지며 각 박스의 왼쪽 아래에 표시된다. 그림 4-4 에서 우리는 시나리오를 구성하는 요소로서 '자재를 요청한다', '기존의 공급자로부터 주문한다', '새로운 공급자의 기준을 설정한다'와 같은 사건의 시간적 흐름을 표현하는 UOB 를 발견할 수 있다.

■ 4.2.1.2 링크(Link) 란 ?

각 프로세스 흐름 다이아그램의 엘리먼트 사이에 연결된 링크는 모델 시스템 안에서 각 엘리먼트 간의 관계를 나타낸다. 이들의 목적은 프로세스간의 순간적, 논리적, 협력적, 혹은 자연적인 제약관계를 표현하기 위함이다. UOB 간의 관계는 Link 와 Junction 을 통하여 서로 연결되는데 IDEF3 에서 Link 는 다음과 같이 분류된다.

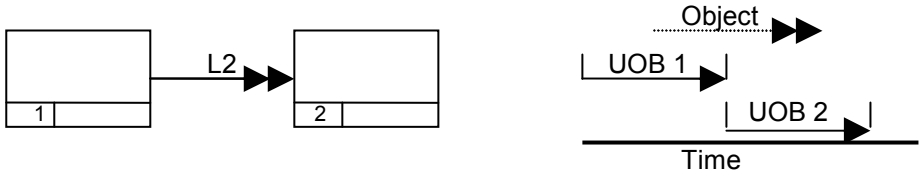
① 시간적 선행(Simple Precedence or Temporal Precedence) Link



시간적 선행 Link 는 하나의 프로세스 타입의 인스턴스와 다른 것 사이에 명료한 선후절차

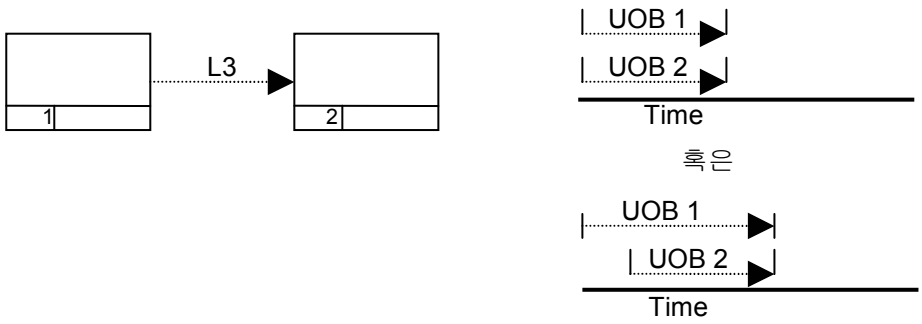
가 있는 것을 나타낸다. 각 선행 프로세스는 후행 프로세스가 시작되기 전에 종료되어야 하며, 후속 UOB 의 수행에 앞서 선행 UOB 가 완료됨이 확실할 경우로 실 선의 화살표로 표시된다.

② 개체의 흐름(Object Flow) Link



개체의 흐름 Link 는 두 프로세스 인스턴스 사이에 특정 개체(object)가 관여하고 있음을 나타낸다, 시간적 선행 Link 와 같은 관계나 프로세스의 수행과 함께 선행 UOB 에서 새롭게 생성된 중요 개체가 후속 UOB 로 이동 될 경우로 개체의 이동 여부가 후속 UOB 수행에 전제 조건으로 작용할 경우에 사용되는데 실 선에 두개의 화살표로 표시한다. 개체의 흐름 Link 로 그려지지 않았다고 해서 두 프로세스 사이에 어떤 개체도 같이 관여되고 있지 않다는 것은 아니다.

③ 관계적(Relational) Link



관계적 Link 는 일시적 선행 Link 와 객체의 흐름 Link 에 의해서는 수용되지 않는 제약을 파악하기 위해 제공되는데 연결된 엘리먼트 사이에 연결관계가 정하여지지 않았음을 나타낸다. 즉 전후 UOB 간의 완료 관계가 명확치 않을 경우를 표현하며 점선으로 표시된다. 이 링크 타입은 시간적 선행 링크와 같이 미리 정하여진 의미는 없다. 대신에 관계적 링크는 하나의 프로세스가 다른 프로세스가 시작되기 전에 시작되거나 늦어도 후행 프로세스와 같이 종료됨을 나타내고 있다. Link 의 번호는 다이어그램에 등록된 순서에 따라 L1,L2,L3 ... 와 같이 유일한 번호 체계를 갖는다.

■ 4.2.1.3 접속(Junction)

접속(Junction)은 프로세스 흐름에 있어서 UOB 의 분기/결합 관계를 표현하는데 접속은 하나의 UOB 가 두개 이상의 후속 UOB 로 분기되는 Fan Out(Divergence) Junction 과 두개 이상의 선행 UOB 가 하나의 UOB 로 결합되는 Fan In(Convergence) Junction 이 있다. junction 의 왼쪽에 나타난 짧은 수직선은 Fan-in Junction 을 오른쪽에게 표시된 짧은 수직선은 Fan-out Junction 을 표시한다. 또한 이러한 접속은 각각 분기와 결합의 논리적 관계를 표현하기 위하여 AND, OR, XOR(다이아그램에서 각각 &, O, X 로 표현됨)로 구분되며 시간적 동기성 여부를 나타내는 Synchronous 와 Asynchronous(junction 박스 좌우의 수직선으로 표현되는데 Synchronous 는 양쪽에 수직선이 Asynchronous 는 왼쪽에만 수직선이 있다)로 구분된다.

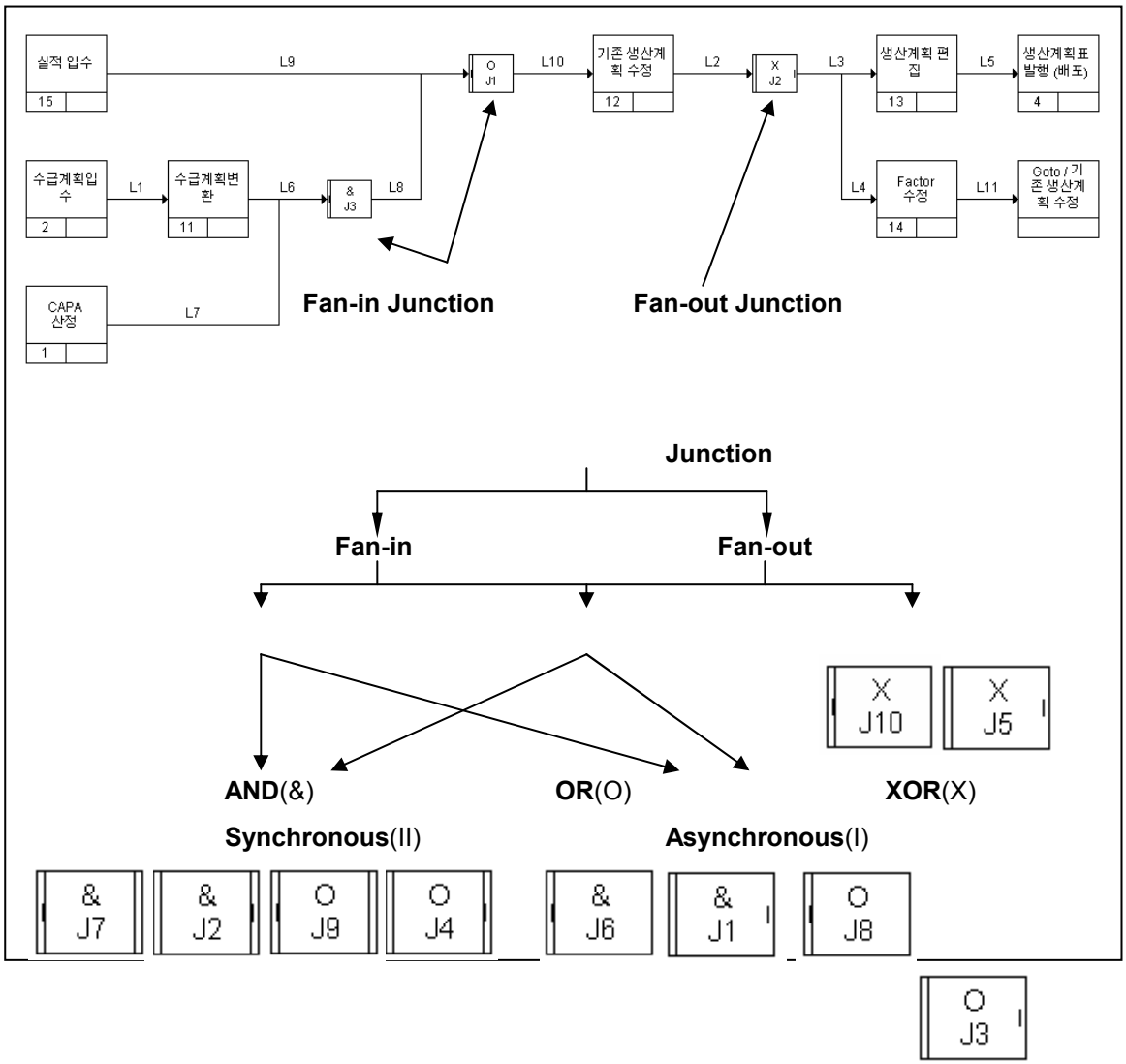
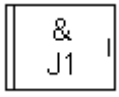


그림 4-5 : Junction 의 구조

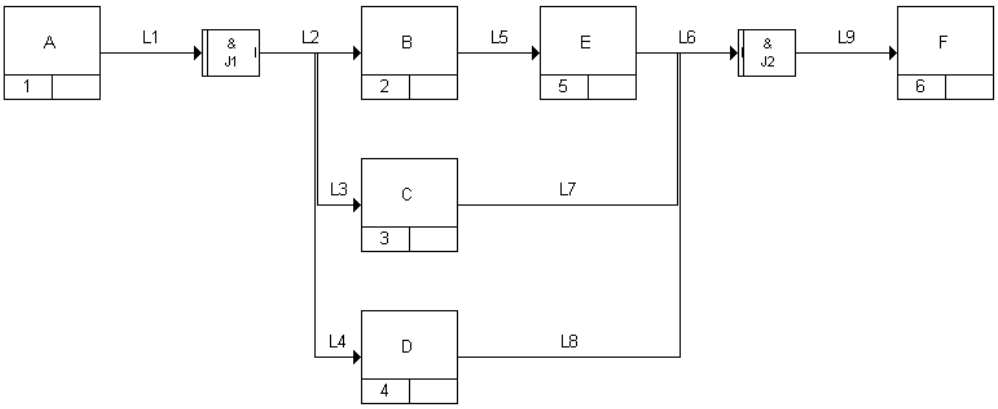
관련된 UOB 간의 분기 및 결합에 있어서 논리적 관계 및 시간적 동기/비동기성 관계를 도형으로 표현하기 위하여 IDEF3 프로세스 다이아그램에서는 발생 가능한 순차적인 모델로서 다음과 같은 5 개의 논리적 타입을 지원한다.

① Asynchronous “AND” 접속은 흐름이 계속되기 위하여 이 접속의 앞이나 뒤에 연결된 모든 프로세스가 수행될 것임을 – 비록 모든 것이 동시에 수행될 필요는 없지만 – 나타낸다. 그림에서 J1 junction 은 프로세스 A 의 수행에 이어서 프로세스 B, C, D 가 동시에 수행될 필요는 없지만 모두 수행되어야 한다는 것을 나타내며 J2 junction 은 프로세스 F 의 수행에 앞서 프로세스 E, C, D 가 모두 수행되어야 한다는 것을 나타내고 있다. 물론 E, C, D 가 동시에 완료될 필요는 없다.



(Fan Out) 선행 프로세스의 수행은 모든 후속 프로세스를 기동한다.

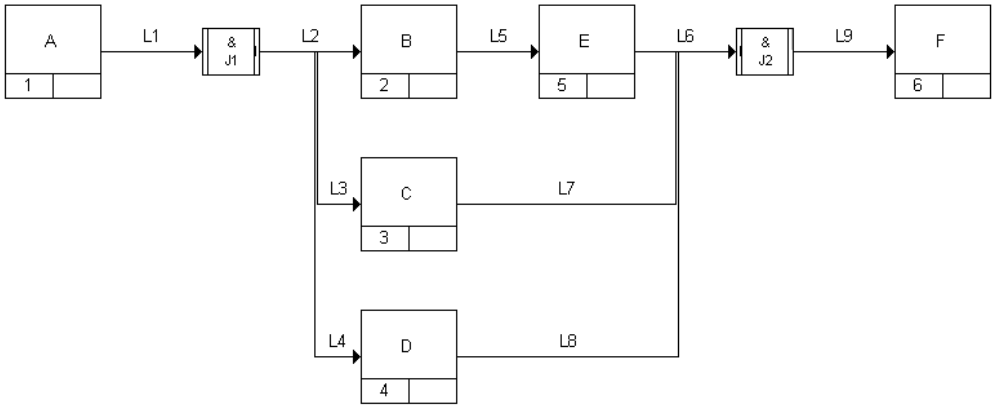
(Fan - In) 모든 선행 프로세스가 종료되어야 후속 프로세스가 수행된다.



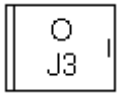
② Synchronous “AND” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 모든 프로세스가 동시에 모두 수행되어야 함을 나타낸다. 그림에서 J1 junction 은 프로세스 A 의 수행에 이어서 프로세스 B, C, D 가 모두 동시에 수행되어야 한다는 것을 나타내며 J2 junction 은 프로세스 F 의 수행에 앞서 프로세스 E, C, D 가 모두 동시에 완료되어야 한다는 것을 나타내고 있다.

(Fan Out) 선행 프로세스의 수행은 모든 후속 프로세스를 동시에 기동한다.

(Fan - In) 모든 선행 프로세스가 동시에 종료되어야 후속 프로세스가 수행된다.

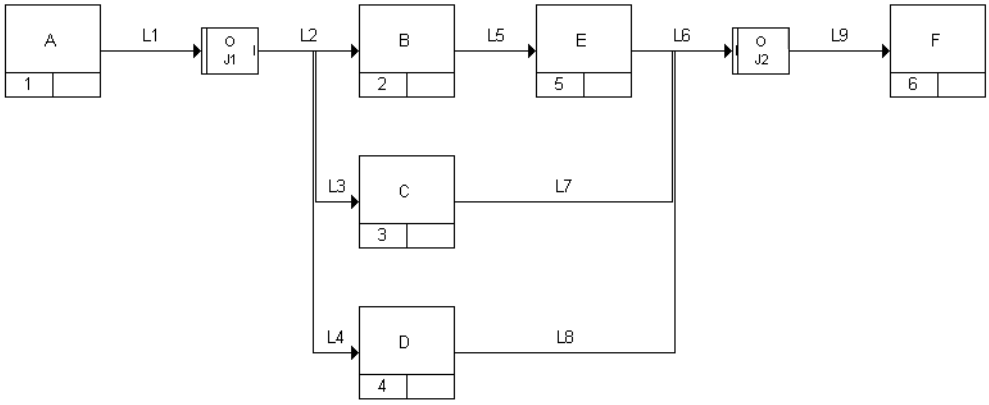


③ Asynchronous “OR” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 하나 이상의 프로세스가, 동시에 수행될 필요는 없지만, 수행되어야 함을 나타낸다. 그림에서 J1 junction 은 프로세스 A 의 수행에 이어서 프로세스 B, C, D 중 하나 이상의 프로세스가 수행되어야 한다는 것을 나타내며 J2 junction 은 프로세스 F 의 수행에 앞서 프로세스 E, C, D 중 하나이상의 프로세스가 완료되어야 한다는 것을 나타내고 있다. 물론 시간의 동시성은 배제하고 있다.



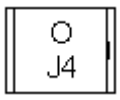
(Fan Out) 선행 프로세스의 수행은 하나 이상의 후속 프로세스를 기동한다.

(Fan - In) 하나 이상의 프로세스가 완료되어야 후속 프로세스가 수행된다.

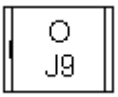


④ Synchronous “OR” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 하나 이상의 프로세스가 동시에 수행되어야 함을 나타낸다. 그림에서 J1 junction 은 프로

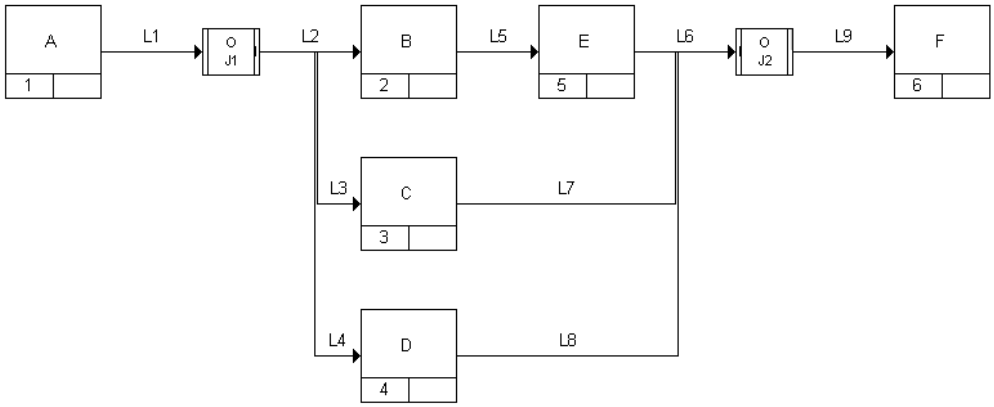
세스 A 의 수행에 이어서 프로세스 B, C, D 중 하나 이상의 프로세스가 동시에 수행되어야 한다는 것을 나타내며 J2 junction 은 프로세스 F 의 수행에 앞서 프로세스 E, C, D 중 하나이상의 프로세스가 동시에 완료되어야 한다는 것을 나타내고 있다.



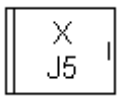
(Fan Out) 선행 프로세스의 수행은 하나 이상의 후속 프로세스를 기동한다.



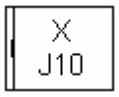
(Fan - In) 둘 이상의 선행 프로세스가 동시에 완료되어야 후속 프로세스가 수행된다.



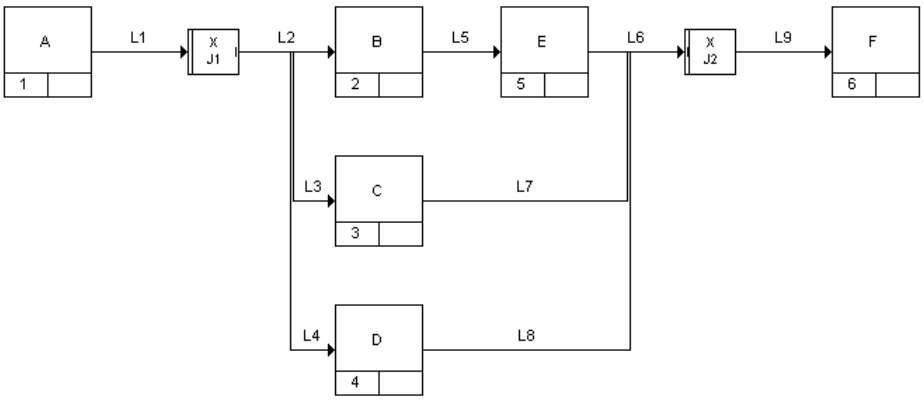
⑤ "XOR" 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 오직 하나의 프로세스만이 수행될 수 있음을 나타낸다. 그림에서 J1 junction 은 프로세스 A 의 수행에 이어서 프로세스 B, C, D 중 하나의 프로세스만이 수행 가능하다는 것을 나타내며 J2 junction 은 프로세스 F 의 수행을 위하여 프로세스 E, C, D 중 하나만 완료되면 가능하다는 것을 나타내고 있다. 물론 배타적 접속에 있어서 시간의 동시성은 용납되지 않는다.



(Fan Out) 선행 프로세스의 수행은 단지 하나의 후속 프로세스만을 기동한다.



(Fan - In) 단지 하나의 프로세스만 완료되면 후속 프로세스가 수행된다.

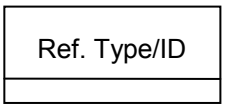


■ 4.2.1.4 참조(Referent)

IDEF3 프로세스 흐름 다이어그램은 판독자의 주의를 위하여 각 UOB 와 관련된 중요 정보를 'Referent'라는 도형으로 표시 할 수 있도록 지원한다. Referent 는 각각의 UOB 가 수행됨에 있어서 관련되는 정보를 박스 형태의 도형으로 지원 함으로서 프로세스 다이어그램 판독자가 UOB 에 관여된 중요 정보를 인지할 수 있도록 한다.

각 Referent 는 Referent Type/ID 형태로 표현되는데 Referent 의 종류에는 UOB 의 수행과 관련된 시간 관계에 따라 다음과 같은 세 종류가 있다.

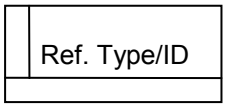
① Unconditional(Information, Go_to)



Information : 단순한 정보가 등록된 Referent 로 관련된 Process, Scenario, Object, OSTN, Note 등의 내용을 참조하는데 ID 로 관련 레이블을 표시한다.

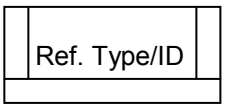
Go_to : 해당 UOB 에서 항상 Referent 에 표시된 Process, Scenario, OSTN, Junction 으로 분기되어짐을 표시한다.

② Asynchronous(Call & Continue)



Call & Continue : 해당 UOB 의 수행 시 Referent 에 표시된 Process, Scenario, OSTN 등을 호출하고 피호출 엔트리의 응답 여부와 상관 없이 계속 진행한다.

③ Synchronous(Call & Wait)



Call & Wait : 해당 UOB 수행 시 Referent 에 표시된 Process, Scenario, OSTN 등을 호출하고 피호출 엔트리의 응답에 따라 진행 여부를 판단한다.

그림 4-6 에서 우리는 세 가지의 Referent 를 발견하게 되는데 '페인트를 칠한다' UOB 에 부착된 Information referent 인 'Object / 소형차 도장라인'은 '소형차 도장라인'이 UOB 에 단순

히 Object 로 관련되어 있음(Unconditional Referent)을 나타낸다. 그러나 '도장상태를 검사한다' UOB 에 부착된 'Process Flow / 시료를 검사한다' Referent 는 UOB 수행 시 '시료를 검사한다'로 명명된 Scenario 를 호출 후 시료검사 결과와 관련된 별도의 응답이 있을 때 까지 대기상태로 유지됨(Synchronous / Call & wait)을 표현하고 있으며, '부품을 출고한다' UOB 에 부착된 'Process Flow / 도장완료 수량을 입력한다' Referent 는 '도장완료 수량을 입력하다'로 명명된 시나리오를 호출하고 결과에 상관없이 계속 수행된다는 것(Asynchronous / Call & Continue)을 나타내고 있다.

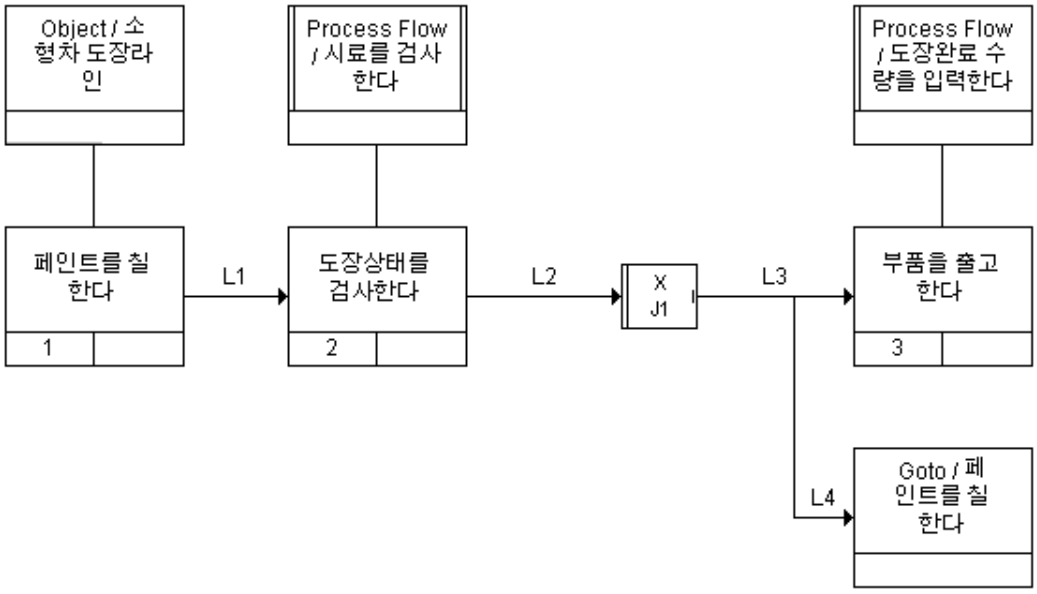
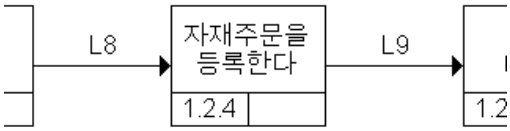


그림 4-6 : Referent 사용의 예

■ 4.2.1.5 상세화(Elaboration)

각 UOB 는, UOB 에 관련된 내용을 기술한 서술적 표현의 집합인 '상세 설명(Elaboration)'에 의하여 구체화 된다. 우리는 Elaboration 을 이용하여 프로세스에 관련된 추가적인 정보를 등록할 수 있는데 Elaboration 에서는 프로세스와 관련된 개체(object), 사실(fact), 제약사항(constraint), 노트(note), 자료의 출처를 알려주는 소스(source) 그리고 관련된 사항을 서술적으로 표현하는 설명을 추가할 수 있다. Elaboration 은 UOB 에 대한 자세한 설명으로서 다이어그램에서 표현하기 복잡한 서술적 내용을 별도의 양식에 표현하며 각각의 UOB 와 한 쌍을 이루는데 그 내용은 그림 4-7 과 같다.



Elaboration :

UOB Name :	자재주문을 등록한다
Objects :	등록요청서(Entity)
	등록담당자(Resource)
	컴퓨터 터미널(Resource)
	등록접수실(Location)
Facts :	수입자재의 경우 외자과의 확인을 받는다.
Constraints :	매일 15 시 이전에 전산등록을 완료한다.

그림 4-7 : Elaboration 의 예

① 개체(Obects) : UOB 의 수행과 관련된 Object 를 나타내며 Object 의 종류에 따라 Entity, Location, Resource, Queue, Transport, Logical 로 분류된다. 개체는 프로세스나 접속에 관련된 물리적 혹은 개념적인 요소로서 기본적으로 프로세스를 이루는 것들이다. 프로세스가 수행되는 동안에 개체는 만들어지고 변경되며, 소멸된다. 또한 우리는 프로세스와 연관된 많은 개체들(부품, 장소, 사람, 기계, 정보 등)에 관한 정보를 모델 판독자에게 전달하여야 하는데 주로 다음과 같은 목적을 달성하기 위하여 개체와 관련된 정보를 파악한다.

- 프로세스가 무엇을 생산하는가? (entity)
- 어디에서 프로세스가 수행되는가? (loaction)
- 어떻게(어떤 자원을 사용하여) 생산을 수행하는가? (resource)
- entity 는 어디에 저장되며 그 용량은 얼마인가? (Queue)
- entity 의 운반에 관련된 개체는 무엇인가? (Transport)
- 프로세스의 수행과 관련된 논리적 제약은 무엇인가? (Logical)

② facts & constraints : Fact 와 Constraint 는 모두 프로세스의 수행에서 관찰될 수 있는 상태를 설명하는데 Fact 는 프로세스의 특정 상태에서 관찰될 수 있는 상황을 말하며

constraint 는 특정상태가 아닌 프로세스의 수행에서 항상 일어나야 될 것을 표현한다. 그림 4-7 의 예에서 '수입자재의 경우 외자과의 확인을 받는다.'는 Fact 는 프로세스가 수입자재 주문을 등록하는 특정 상황에 관한 제약이며, Constraint 로 등록된 '매일 15 시 이전에 전산등록을 완료한다.'는 특정 조건이 아닌 일반적인 지침으로서의 제약사항을 나타내고 있다. 프로세스 모델의 구축에 있어서 다양한 Fact 의 추출 및 기록은 constraint 를 찾아내기 위한 좋은 방편이 된다.

③ 소스(source) : 프로세스와 관련된 기초자료나 정보의 출처를 나타낸다.

④ 노트(note) : 모델 검토의 결과인 주석을 나타낸다.

⑤ 설명(Description) : UOB 에 대한 일반적 설명을 기술하며 프로세스에 관한 전반적인 설명을 포함한다.

4.2.2 객체상태 전이 다이어그램 (Object State Transition Network Diagram)

IDEF3 에서 지원하는 또 하나의 다이어그램은 프로세스의 수행과 함께 변환되는 Object 의 전환 상태를 표시하는 객체상태 전환 다이어그램이다. 객체상태 전환 다이어그램은, 프로세스를 객체 중심적 관점에서 파악하기 위해 사용하는데 이는 프로세스 다이어그램 전반에 걸친 영역 내에서 객체의 허용 가능한 전환상태를 요약 표현하는 것이다.

객체상태의 전환 다이아그램은 두 가지로 구성되는데 첫째는 둥근 원으로 표시되는 객체의 상태와 둘째는 객체상태 간에 연결되어 그 전환관계를 표시하는 화살표(Transition Arc)이다. 그림 4-8 은 객체상태 1(Object State #1) 이 객체상태 2(Object state #2)로 전환되는 과정에서 Action2 라는 UOB(Referent)가 관여하고 있음을 보여 주고 있다.

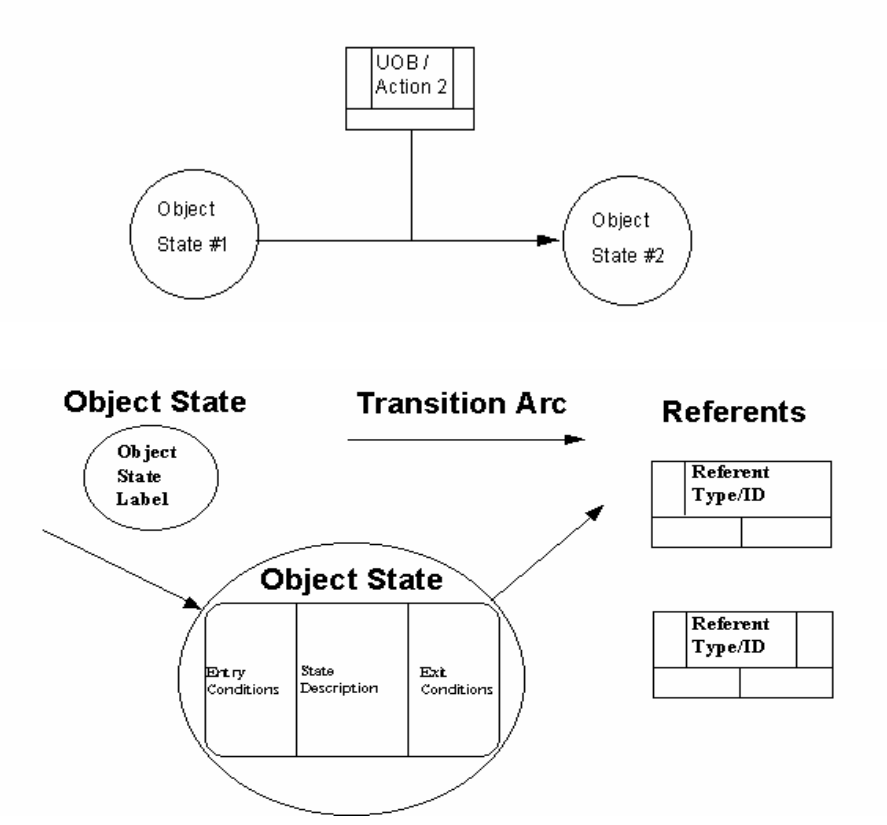


그림 4-8 : Object State Description

각 객체상태는 객체 상태를 유지하기 위한 제약조건과 그 값으로 정의되어진다. 정보 시스템 내에서 연속되는 객체상태의 값들은 IDEF1 모델에서의 속성과 같이 정의되어야 하며 이러한 값들은 객체상태 전환 다이어그램과 상호 연관되어 있다. 하나의 객체상태는 그 상태로 되기 위한 전환 전의 제약조건(entry condition, 입장조건)과 다음 상태로 전환되기 위

한 전환 후의 제약조건(exit condition, 퇴장조건)을 가지고 있다. 이 제약조건들은 상태전환이 시작될 수 있기 전의 상태와 상태전환이 완료될 수 있기 전의 상태를 규정한다. 제약조건은, 간단한 성질 과 값의 쌍(pair) 리스트 혹은 제약조건에 대한 설명에 의해 구체화 된다.(예를 들면 – 승인된 자재주문, 발주된 자재주문) 속성의 값은 그것이 되기 위하여 요구되는 특정한 값과 연결되어야 한다(예를 들면 – 승인된 자재주문:승인번호, 발주된 자재주문:발주번호).

상태를 정의하는데 필요한 세 가지 형태의 요구는 다음과 같다.

① 어떤 상태로 전환해 들어가는 객체에 존재해야 하는 '입장조건'
② 상태로부터 빠져 나오는 객체에 존재해야 하는 '퇴장조건',
③ 객체가 어떤 상태에 있는 동안에 존재하는 '상태설명'.

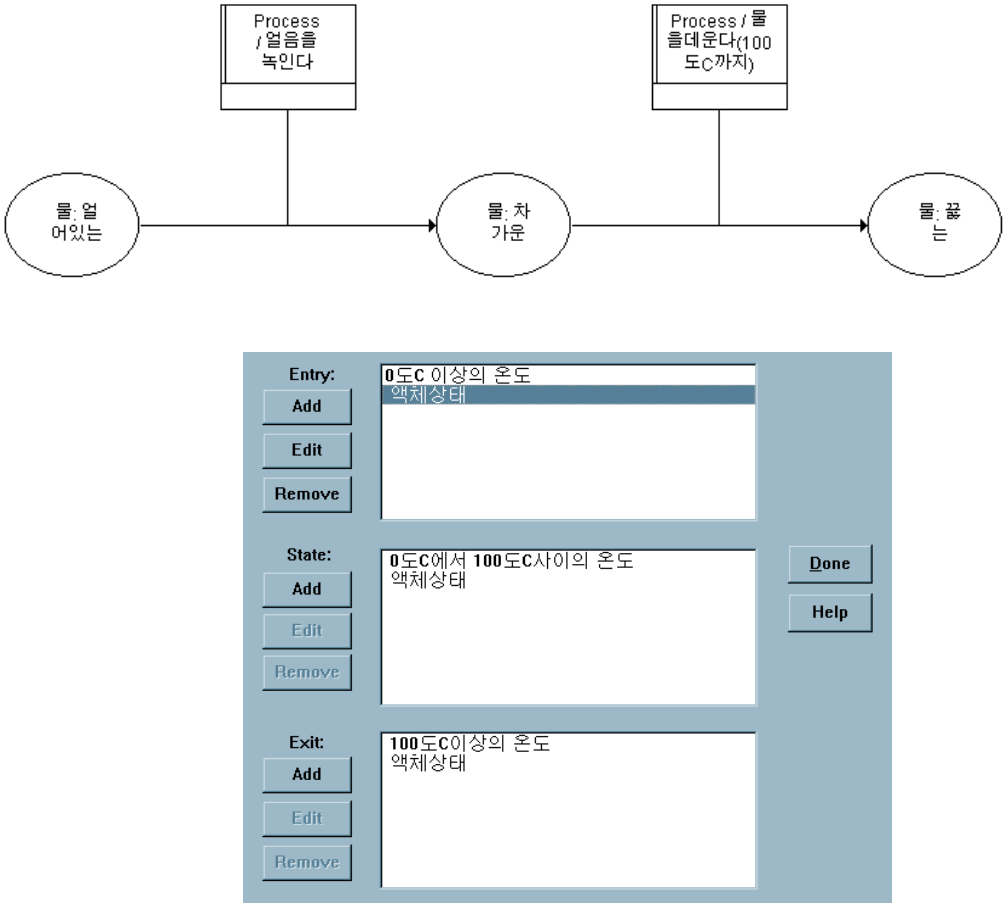


그림 4-9 : OSTN 과 상태정의

■ OSTN의 기능

① 개체 중심의 관점에서 모델의 구축을 가능케 한다.
② 한 분야에서 개체의 전환이 가능한 상태들을 요약한다.
③ 자료의 생성에서 소멸까지를 문서화 한다.
④ 프로세스 흐름 다이아그램을 관통한다.
⑤ 개체의 동적 형태를 특성화 한다.

그림 4-10 은 객체상태 전환 다이어그램의 예로서 이발소에서 머리를 깎는데 발생할 수 있는 상황을 고객의 상태 변환의 관점에서 기술한 것이다. 각 각의 상태가 전환되는 과정에서 어떠한 UOB 가 관련 되고 있는가를 우리는 표현 할 수 있으며 이는 각 UOB 의 기능을 명확히 하고 있다 .

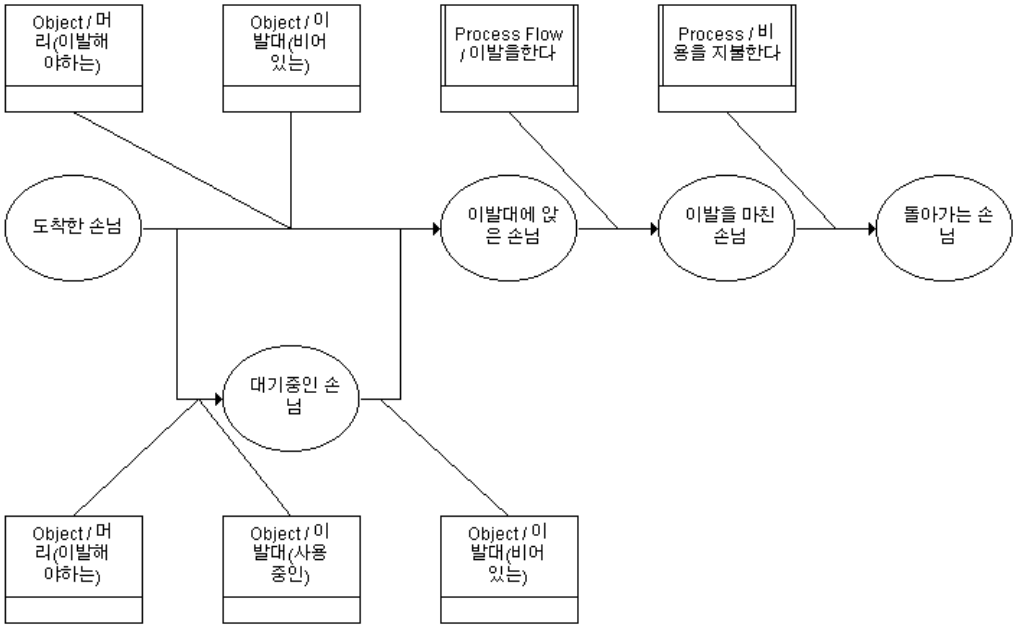


그림 4-10 : OSTN 과 UOB 의 관계

4.2.3 프로세스의 분해

■ 분해의 목적

① 다이아그램의 복잡성을 감소시킨다.
② 다양하게 추상화 된 레벨에서의 묘사를 포착하 수 있도록 지원한다.
③ 같은 프로세스를 다른 지식의 원천 및 다른 관점에서 모델화 할 수 있는 능력을 제공한다.

IDEF∅와 마찬가지로 IDEF3 프로세스 흐름 다이아그램은 다이아그램의 복잡성을 줄이고 다양하게 추상화된 레벨에서 다양한 관점의 설명을 포착할 수 있도록 지원하기 위하여 프로세스의 분해기법을 지원한다. 그런데 IDEF∅가 최상위 레벨에서 하나의 활동만을 표시 표현하고 이를 경계조건으로 선언하는 것과 달리 IDEF3 프로세스 다이아그램은 최상위 레벨의 다이아그램에서부터 우리가 파악하고자 하는 영역내의 프로세스의 흐름을 여러 개의 프로세스간의 관계로 자유롭게 나타낼 수 있으며 이들 프로세스의 분해에 있어서도 또한 하나의 UOB 에 대하여 한 개 이상의 분해 다이어그램을 허용하고 있다.

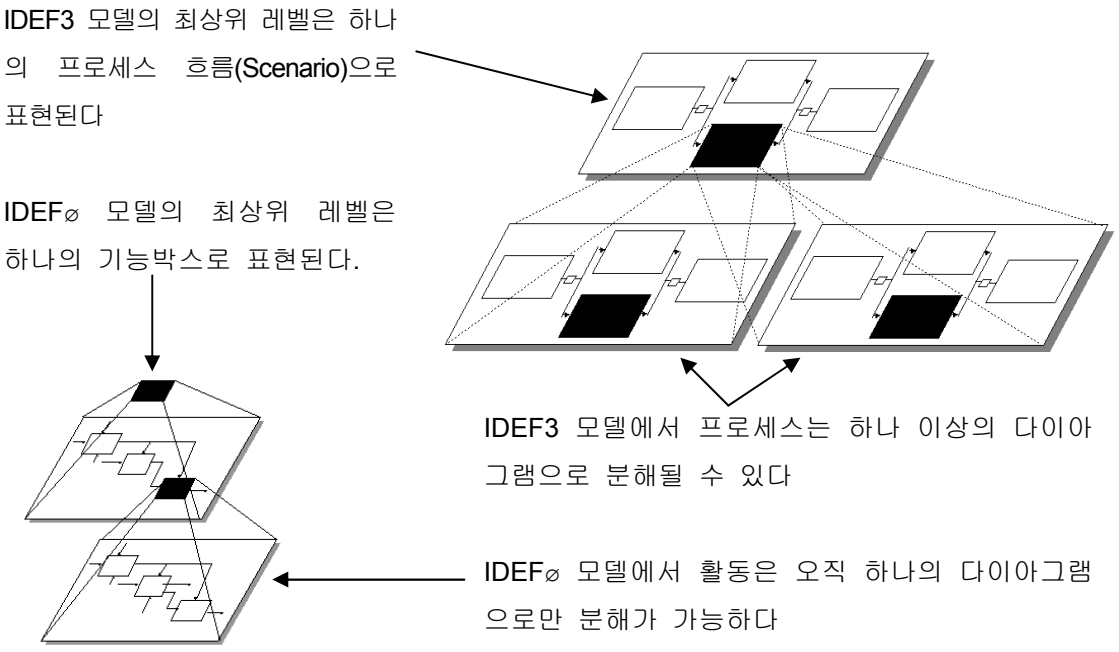


그림 4-11 : IDEF3 와 IDEF∅의 분해 구조

이는 IDEF3 의 기본적 개발 목표인 전문가의 설명 내용을 획득하여 표현함에 있어서 사실적, 객관적 관계의 확인 보다는 다양한 관점에서의 여러 전문가의 설명을 표현할 수 있도록 하기 위함이다. . IDEF3 에 있어서 분해는 추상화 된 단계의 설명을 포착하기 위한 중요한 방법 중의 하나로서 하나의 프로세스(UOB)에 관하여 다른 관점이나 지식 근거에서 모델을 구성 할 수 있도록 다양한 방법을 지원한다. 문법적으로 IDEF3 에 있어서 분해이란 하나의 프로세스를 구체화하는 세분화된 또 하나의 프로세스 흐름 다이어그램의 구축을 말한다.

여기서 우리는 왜 이러한 다양한 관점의 설명의 포착이 필요한지를 알아보자.

그림 4-12 는 차량정비소의 업무를 분석하기 위하여 모델작업자와 관련된 담당자와의 인터뷰를 바탕으로 만들어진 세가지 관점의 인터뷰 내용 및 다이아그램이다. 우선 모델작업자가 숙련 정비공과 인터뷰한 내용과 이를 중심으로 표현된 자동차 정비 시나리오의 '정비를 수행한다' 프로세스 분해 다이아그램은 그림 4-12 와 같다.

① 차량이 정비소에 들어오면 차량상태(엔진, 전기계통, 동력전달계통)를 점검한다.
② 문제가 되는 부분을 정비한다.
③ 차량이 정상적으로 수리되었는지 정비상태를 점검한다.
④ 정상적으로 수리가 종료되었으면 정비내역을 기재한다.

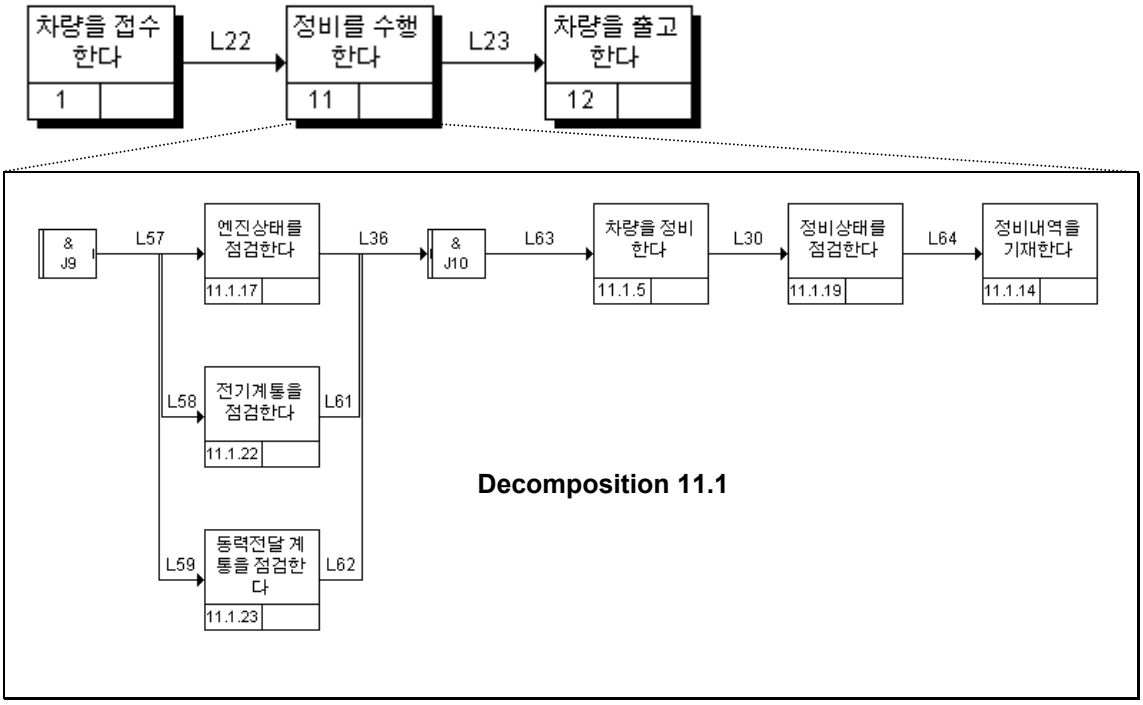


그림 4-12 : 분해의 예(숙련 정비공의 관점)

다음으로 모델작업자가 수습 정비공과 인터뷰한 내용과 이를 중심으로 표현된 '정비를 수행한다' 프로세스의 분해 다이아그램은 그림 4-13 과 같다.

① 차량이 정비소에 들어와 접수가 되면 접수증을 확인한 후 자동차키를 손님으로부터 받아 차량을 수리장소에 주차 대기시킨다.
② 차량의 정비는 숙련 정비공이 수행하며 정비전이나 정비가 끝난 차량의 폐기물을 처리하고 차량을 청소한다.
③ 정비가 끝난 차량은 손님이 출고증을 가져오면 출고증을 확인한 후 자동차 키를 인도한다.

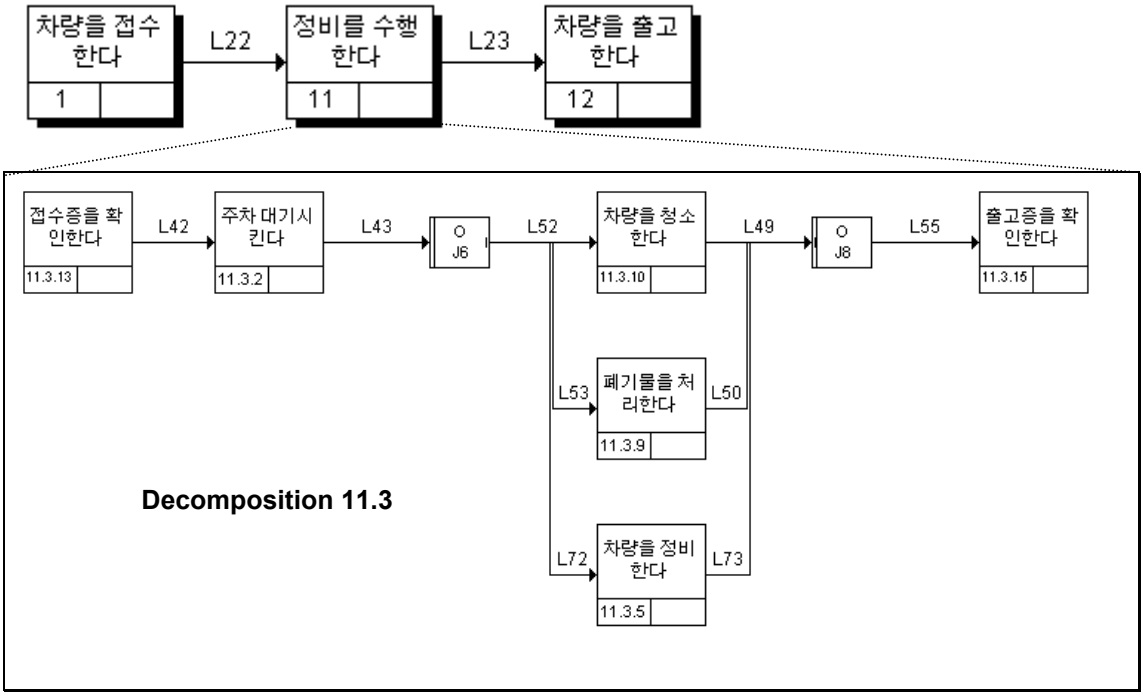


그림 4-13 : 분해의 예 (수습 정비공의 관점)

마지막으로 모델작업자가 관리자와 인터뷰한 내용과 이를 중심으로 표현된 '정비를 수행한다' 프로세스의 분해 다이아그램은 그림 4-14 와 같다.

① 접수된 차량에 대해서는 우선 정비팀을 배정한다.
② 배정된 차량은 해당 정비팀에서 수리된다.
③ 수리내역이 전달되면 해당 내역을 전산등록한다.
④ 등록된 내용을 바탕으로 수리비 청구를 발행하여 청구한다.
⑤ 수리비용이 접수되면 해당 차량의 출고에 필요한 출고증을 발행한다.

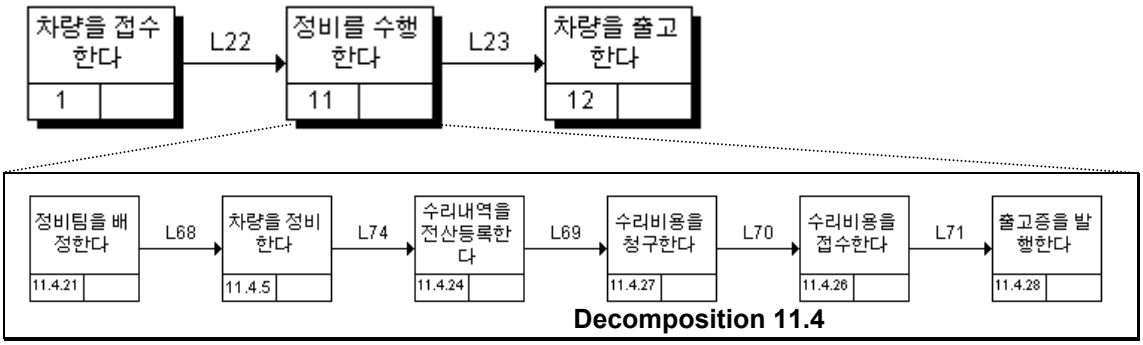


그림 4-14 : 분해의 예 (관리자의 관점)

우리는 위의 예에서 '정비를 수행한다' 프로세스에 대한 설명이 인터뷰 대상에 따라 각기 다른 관점에서 다르게 표현될 수 있고 이를 바탕으로 작성된 분해 다이아그램은 결국 각기 다르게 표현될 수 밖에 없음을 알아보았다.

■ 분해의 종류

① Objective view : 다양한 관점의 분해는 아마도 개체중심의 관점으로 통합될 것이다. – Objective view 는 객관적인 관찰자의 관점을 수용한다. 따라서 하나의 Objective view 가 있을 뿐이다.
② Role view : 이해하는 바에 따라, 혹은 각자의 관점에 따라, 각각의 역할과 타입에 따라, 조직의 기능에 따른 프로세스의 관점에 따라 프로세스에는 하나 이상의 role view 가 있을 수 있다.

IDEF3 에 있어서 분해는 두 가지로 분류 될 수 있는데 하나는 객체 관점의 다이어그램이고 다른 하나는 프로세스를 수행하는 역할(Role)의 관점에서 기술된 다이어그램이다. 자연적(중립적) 관찰자의 입장에서 기술되는 객체의 상태 변환에 관한 객체 관점은 오직 하나의 하위 단계만을 수용 할 뿐이나 프로세스를 수행하는 역할의 관점에서 기술되는 프로세스 흐름 다이어그램은 하나의 프로세스를 여러 가지의 다이어그램으로 분해 할 수 있다. 즉 하나의 프로세스에 대하여 우리는 관찰자의 설명에 따라 다양한 관점의 지식을 표현 할 수 있는 융통성을 확보할 수 있는 것이다. 그림 4-15 는 차량정비소의 업무를 객체 중심적 관점에서 OSTN 으로 표현하고 이를 바탕으로 관리자와 숙련 정비공, 수습 정비공의 설명을 통합한 프로세스 모델을 작성한 예이다.

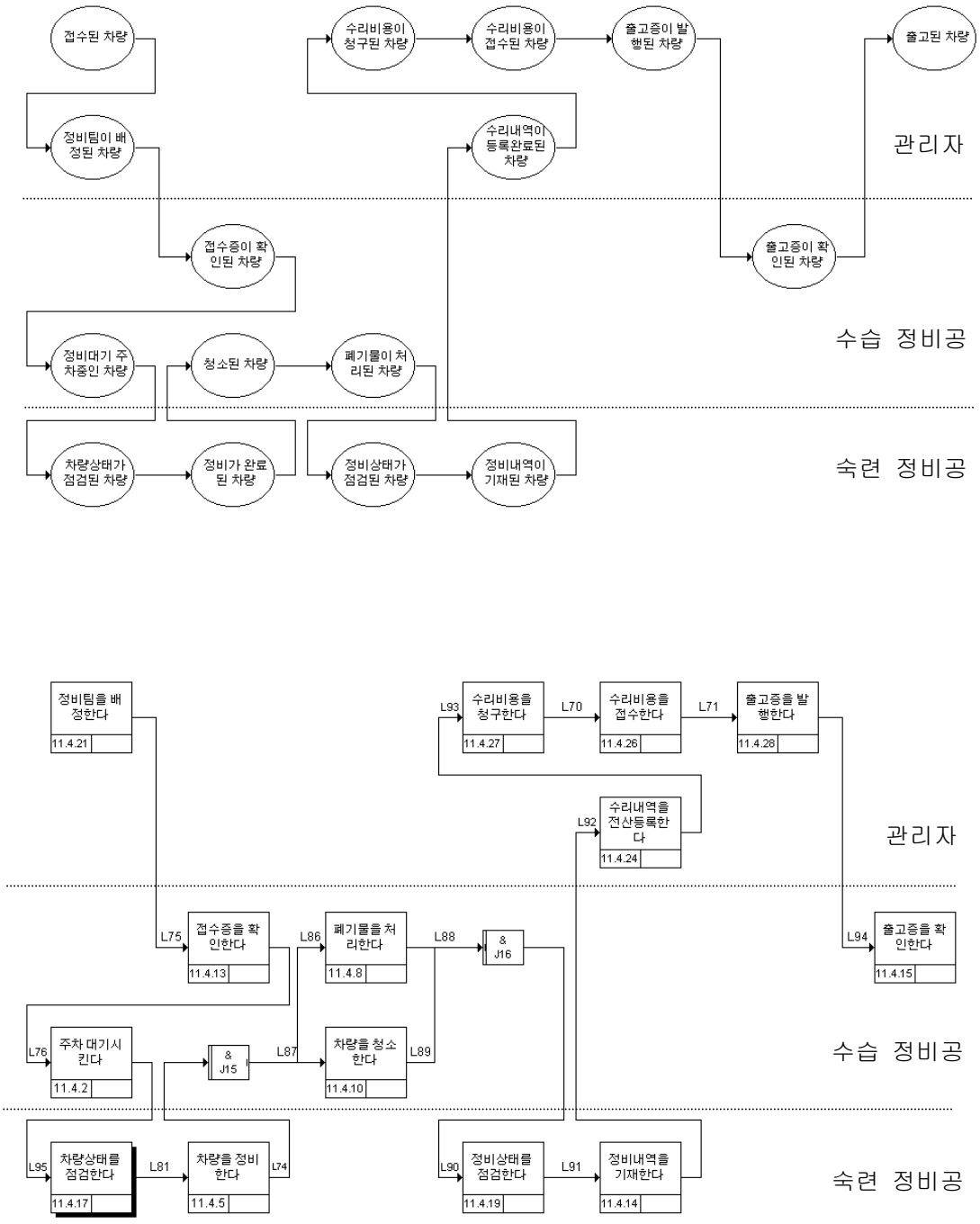


그림 4-15 : OSTN 과 프로세스 흐름 다이아그램

4.2.3 다이아그램 번호 부여

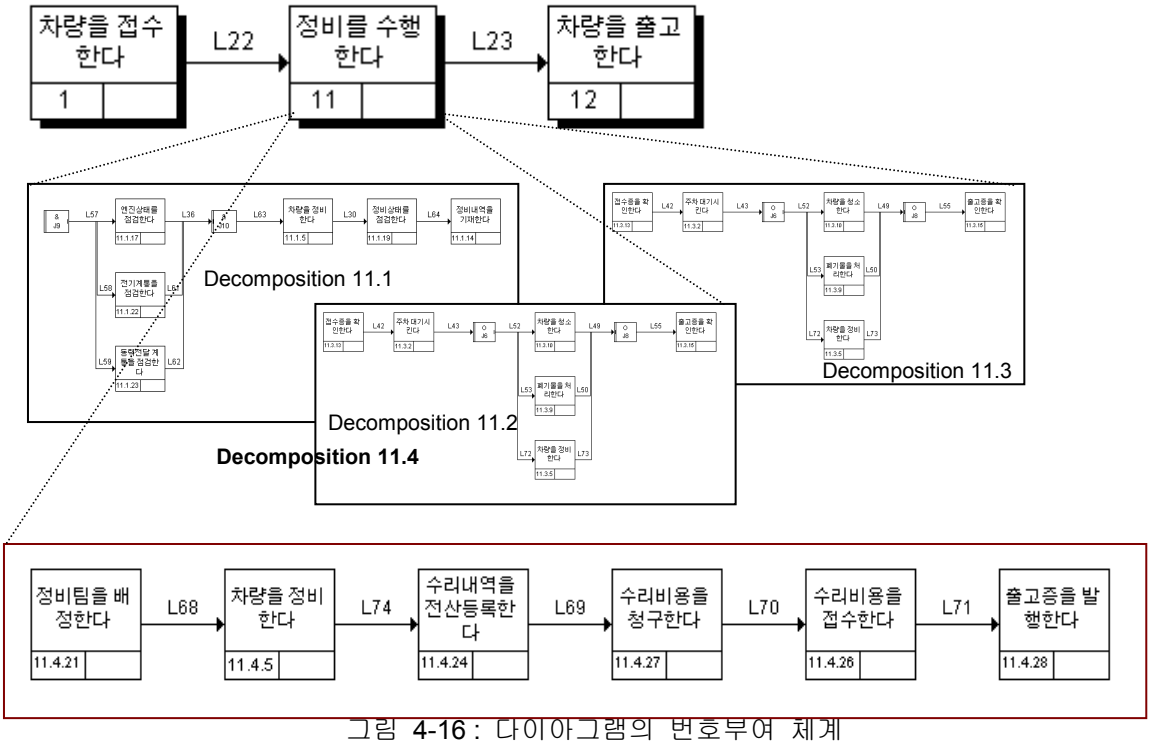


그림 4-16 : 다이아그램의 번호부여 체계

IDEF3 에서 UOB 번호는 프로세스의 분해 기법을 지원하기 위한 형식으로 표현되는데 그림 4-16 은 차량 정비업무 시나리오의 최상위 레벨에 표현된 세 개의 UOB 중 '정비를 수행한다' 프로세스가 네 가지의 다이아그램으로 분해된 내용을 보여주고 있다. 최상위 레벨에서의 UOB 번호는 모델에 등록된 UOB 순서에 따라 1 부터 순차적으로 부여된다. (1, 2, 3,... 등으로). 분해 다이아그램 다이아그램에서의 번호는 두 가지 부분으로 구분될 수 있는데, 그림 4-17 에서 '정비를 수행한다'의 네 번째 분해다이아그램(Decomposition 11.4)에 표시된 첫번째 UOB 인 '정비팀을 배정한다'에 나타난 11.4.21 이라는 UOB 번호는 분해다이아그램의 번호를 나타내는 11.4(상위 레벨의 11 번 UOB 의 네 번째 분해 다이아그램)와 모델 전체에서 '정비팀을 배정한다' UOB 가 등록된 유일한 순차적 번호인 21 이 결합한 상태를 보여주고 있다

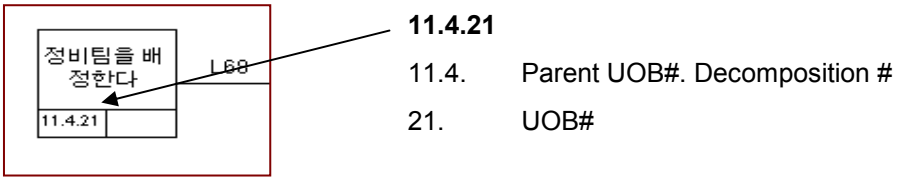


그림 4-17 : UOB 의 번호구조

그 밖에 IDEF3 다이아그램에서 표현되는 Link 의 경우는 L1, L2, L3 과 같이 링크를 나타내

는 L 과 등록된 순차적 번호인 1, 2, 3 등으로 결합되어 표시되며, 접속의 경우도 J1, J2, J3 과 같이 접속을 나타내는 J 와 등록된 순서를 나타내는 1, 2, 3 등이 결합되어 표현된다.

IDEF3 다이어그램은 종종 IDEF∅ 다이어그램과 같이 상호 보완적으로 이용되는데 그림 4-18 의 UOB 1 우측 하단에 표현된 Reference 번호는 이미 만들어진 기능모델을 IDEF3 프로세스 모델로 변환할 때, IDEF∅의 참조된 Activity 번호를 표현하기 위한 항목으로 해당 UOB 가 IDEF∅ 다이어그램의 어떤 활동(Function)으로부터 전환된 것인지를 나타낸다.

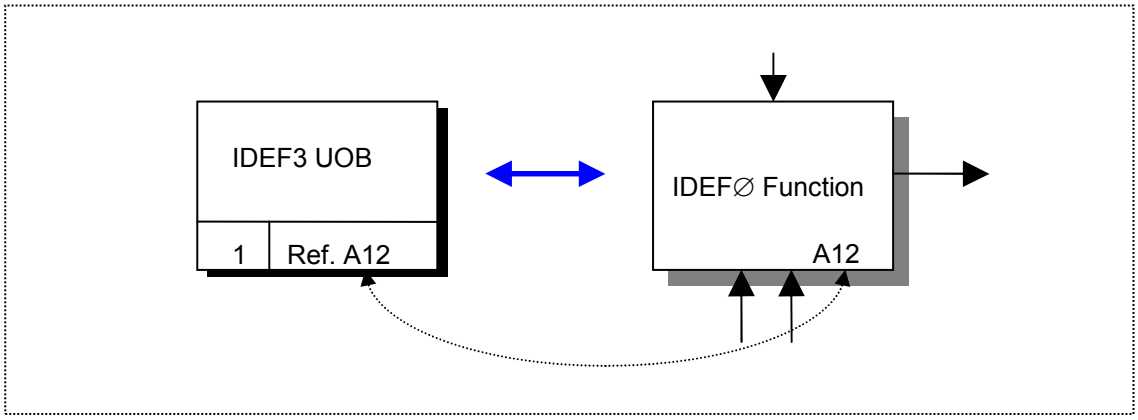


그림 4-18 : UOB 의 Reference

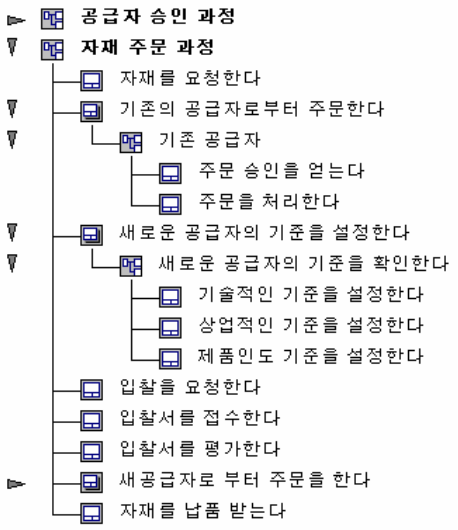
4.3 IDEF3 프로세스 모델의 개발

프로세스 모델링의 근본적인 목적은 시스템이 활동을 어떻게 수행하는가를 밝혀내는 것이다. 언뜻 보기에, 이것은 어느 정도 수월하게 곧장 진행 될 수 있는 일같이 보여진다. 만일 여러분이 어떤 사람으로부터 프로세스와 관련된 것들을 정의하여 제조시스템의 운영을 설명해 달라고 요청 받았다면, 여러분은 꽤 광범위한 자료(리스트)를 만들 수 있을 것이다. 여하여튼, 여러분은 복잡한 시스템이 무엇을 하는지에 대하여 설명하는 동안에 여러가지 다양한 요소들을 고려해야 할 것이다.

즉, 프로세스가 수행되기 위하여 무엇이 필요한가?
어떠한 object 가 프로세스에 관여되는가?
환경 안에서 프로세스와 이벤트 사이에 어떠한 선행조건과 원인관계가 존재하는가?

IDEF3 방법은 주어진 현실세계 시스템에 관한 지식의 포착과 추상화에 초점을 맞추는 데 시스템 안에서 일어나는 프로세스 사이에 시간적, 인과적, 논리적 관계를 포함한다. 또한, 이들 프로세스와 관련된 object 들, 그리고 이들 object 들의 상태 변화를 포함한다. 결과적으로, 이 방법은 시스템에서 어떤 특정시간에 무엇이 수행되는가를 설명하는 것이 아니라, 대신에 시스템 안에서 무엇이 기본적으로 어떻게 수행되는가를 표현한다. 즉, 시스템 안에서 반복적으로 수행되는 동적인 패턴을 말한다. 이러한 지식은 프로세스나 오브젝트의 상태를 순차적으로 정의한 스토리보드 접근 방법으로 결합된 모델 혹은 시나리오로 구성된다. 이미 기술한 바와 같이 IDEF3 설명은 두 가지의 다른 관점에서 개발될 수 있다. 프로세스 중심 혹은 오브젝트 중심이다.

프로세스 모델은 현실세계 시스템에 있어서 하나의 프로세스 집합과 관련된 시간적, 순차적, 인과적 관계를 모두 포착한다. PROSIM 에서는, 프로세스 모델을 시나리오로 간주하는데, 시나리오는 개별적인 프로세스 흐름 다이아그램으로 나누어 진다. 아래에 그려진 시나리오는 8 개로 분리된 프로세스를 포함하는 자재 주문 과정과 그 밑에 그려진 추가적인 프로세스 정보로 구성되어 있다.


공급자 승인 과정
자재 주문 과정
자재를 요청한다
기존의 공급자로부터 주문한다
기존 공급자
주문 승인을 얻는다
주문을 처리한다
새로운 공급자의 기준을 설정한다
새로운 공급자의 기준을 확인한다
기술적인 기준을 설정한다
상업적인 기준을 설정한다
제품인도 기준을 설정한다
입찰을 요청한다
입찰서를 접수한다
입찰서를 평가한다
새공급자로 부터 주문을 한다
자재를 납품 받는다

4.3.1 프로젝트의 생성과 시나리오 문서화

이번 절에서 여러분은 왜 문서화가 여러분이 프로젝트나 시나리오를 작성하는 동안에 중요한지에 대하여 배울 것이다. 또한 여러분은 새로운 프로젝트와 새로운 시나리오에 관련된 문서를 작성할 것이다

■ 배경(Context)의 확립:

박과장은 XYZ 회사에 다년간 근무하였고 자재수급 프로세스와 관련된 많은 지식을 가지고 있다. 그는 지금 구매부서의 관리자이기 때문에, 더 이상 직접 자재구매 오더를 발행하지는 않는다. 그런데 그는 각 직원들이 각자 개인적인 성향이나 고객에 따라 다양하게 다른 절차로 자재구매 오더를 발행한다는 사실을 알고있다. 박과장은 XYZ 회사의 다른 부서 직원이 구매요청을 할 경우 그의 직원이 어떻게 구매오더를 발행할 것인가에 대한 표준화를 하기 원하고 있다. 그는 XYZ 회사의 컨설턴트인 김병익에게 구매부서의 모든 사람들과 인터뷰를 하고 , 그의 직원들이 구매요청을 접수하는 절차를 표현하기 위한 프로세스 모델을 작성할 것을 지시했다. 박과장은 각각의 직원이 프로세스를 다르게 설명하리라는 것을 잘 알고 있었다. 그러나 그는 그의 부서 안에서 합의를 도출하기 위한 것 뿐이 아니라 그의 모든 직원들이 가장 효율적인 방법으로 자재오더 업무를 수행할 수 있도록 하기 위하여 모델을 완성 시키고자 하였다. 더욱이, 그는 모델을 새로운 직원이 그의 부서에 참가할 경우 그것을 교육용 자료로 활용하고자 하였다.

구매부서의 직원들과 인터뷰를 진행하는 동안, 김병익은 XYZ 회사에서는 부품별로 기존의 공급선가 있는가 아닌가에 따라 자재 오더발행과 관련된 두 가지 형태의 구별되는 프로세스가 있다는 것을 알게 되었다. 그러한 한 가지 경우는 프로세스 표준화가 아주 쉽게 될 수 있었는데 즉, 기존의 공급선가 있는 경우에는 구매부서의 요원은 오더발행을 위한 승인만 받아서 발송하면 되는 것이었다.

새로운 공급선으로 오더를 발행하는 경우는 조금 더 복잡하고 이를 수행하기 전에 몇 가지 단계를 더 수행해야 했다. 첫째, 새로운 공급선의 명세사항이 개발되어야 하는데 이러한 단계는 몇 가지의 하부단계로 이루어져 있었다. 구매부서는 예상 공급선으로부터 입찰을 요청하고 이를 검토해야 한다. 마지막으로 새로운 공급선에 주문을 내기 전에, 구매부서는 공급선 승인 프로세스를 완료해야 하며, 이와 같은 요청사항은 ‘Call and Wait referent’로 문서화 된다.

■ 프로젝트 생성

하나의 프로젝트는 하나의 시나리오 그룹, 이들의 다이아그램, 그리고 각 다이아그램을 구성하는 엘리먼트(element)들로 이루어진다. 프로젝트는 프로젝트의 시나리오, 다이아그램, 그리고 엘리먼트들의 그룹화를 지원한다.

여러분이 PROSIM 에서 파일을 오픈하거나 생성할 때마다, 여러분은 근본적으로 새로운 프로젝트 – 즉 하나의 파일은 하나의 프로젝트만을 포함한다.– 를 여는 것이다

여러분이 새로운 프로젝트를 오픈할 때, 활성화 되는 윈도우는 **Project Window** 이다. 새로운 프로세스의 생성, 프로젝트 풀(Pool)에 엔리먼트의 추가, 그리고 프로젝트의 문서화를 포함하여 특정한 프로젝트 레벨의 기능에 접근할 수 있는 다양한 메뉴가 지원된다.

File menu 에서 *New* 를 선택한다. 혹은 Toolbar 에서  를 클릭한다.

■ 프로젝트의 문서화

프로젝트의 문서화는 여러분이 프로젝트의 이름을 지정하고, 프로젝트의 작성자를 구분하고, 프로젝트의 일반적인 설명 그리고, 프로젝트에 연관된 파일을 첨부할 수 있도록 지원한다. 서술적인 문서는 프로젝트의 모델링 작업의 기준, 또한 프로젝트와 계속적으로 관련되어지는 모델작업자에게 시작점을 제공한다. 우리는 프로젝트 자체의 문서화에서부터 새로운 모델링 프로젝트를 시작하기로 한다. 프로젝트 문서화는 후에 여러분의 작업과 관련될 모델 작업자에게 전달되는 유용한 정보가 될 것이다.

① Project menu 에서 *Project Summary....*를 선택한다.
② Project Summery dialog 에서, Project Name, Creator, 그리고 Used At field(field 에서 field 로 움직이기 위하여 **<Tab>**을 사용한다)를 채운다
예를 들면 :

자재주문과정 문서화

김병익

XYZ 회사

③ 프로젝트의 목적과 범위를 설명하는 문장을 등록하기 위하여 **Description** 을 클릭한 후 설명을 등록한다.
예를 들면 :

이 모델은 XYZ 회사의 재고 부품에 관한 자재구매 절차를 문서화 하기 위한 것이다. 이 설명은 사람들에게 구매요청의 시작에서 오더가 발행되기까지의 프로세스 흐름의 절차에 관한 이해를 제공하기 위한 것이다. 문서는 어떻게 자재가 발주되는가와 문서가 어떻게 이용되는가에 대한 설명을 포함하고 있다.

④ 프로젝트와 관련된 여러 가지 파일을 첨부하기 위하여 **Attachments** 를 클릭한 후 PROSIM 안에서 이 파일들을 오픈한다.
⑤ Project Summary dialog 를 닫기 위하여 **OK** 를 클릭한 후 Project Window 로 돌아온다.

■ 시나리오의 작성

하나의 시나리오는 **scenario diagram** 이라 불리는 톱 레벨의 다이아그램과 여러 개의 하위 다이아그램으로 이루어졌다. 하나의 시나리오 안에서 각각의 분해(decomposition) 다이아그램은 이 시나리오 다이아그램으로부터 생성되어지며, 이러한 결과로, 시나리오 다이아그램은 전체 시나리오의 배경을 제공한다.

① Project menu 에서 *Open Scenario...* 를 선택하거나, Toolbar 에서  를 클릭한다.
② Name field 에 **자재 주문 과정** 을 등록한 후 새로운 시나리오를 작성하기 위하여 **Create New** 를 클릭한다.
③ New scenario 를 선택한 후, 새로운 시나리오를 Process Flow Diagram Window 에 display 하기 위하여 **OK** 를 클릭한다.

■ 시나리오의 문서화

① Diagram menu 에서 *Diagram Summary...*를 선택한다
② 시나리오가 무엇을 표현하기 위한 것인가에 대한 설명을 등록하기 위하여 **Purpose** 를 클릭한다. 예를 들면:

이 다이아그램의 목적은 자재요청 프로세스를 문서화 하기 위함이다.

③ 누구의 관점에서 당신이 모델을 작성하는가를 설명하는 문장을 등록하기 위하여 **Viewpoint** 를 클릭한다. 예를 들면 :

프로세스는 오더를 수령하고 수행 완료하는 사람의 관점에서 서술되었다.

④ **Description, Notes, Sources,** 그리고 **Context.** 를 클릭하여 시나리오에 관한 추가적인 정보를 등록한다.

⑤ 시나리오에 관련된 파일을 첨부하기 위하여 **Attachments** 를 클릭한다. 이들 파일은 PROSIM 에서 오픈할 수 있다.

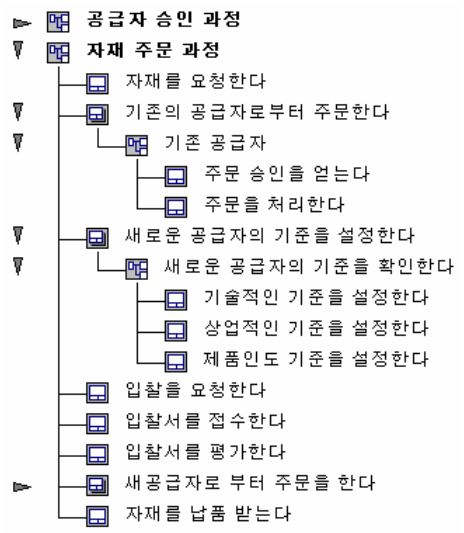
⑥ Process Flow Diagram Window 로 돌아가기 위하여 **OK** 를 클릭한다.

4.3.2 PROSIM이 지원하는 윈도우

PROSIM 은 프로세스 모델을 보기위한 세 개의 윈도우 타입과 오브젝트(object) 상태를 보기 위한 하나의 윈도우를 제공한다. 각 윈도우는 모델에 대한 다른 관점을 지원하는데, 이번 절에서는 우리가 후에 샘플 모델을 작성하는데 사용하게 될 기본적인 윈도우 타입에 대하여 알아보도록 한다.

■ 프로세스 흐름 노드리스트 윈도우(Process Flow Nodelist Window)

프로세스 흐름 노드리스트 윈도우는 현재 프로젝트 안에 있는 시나리오의 전체 계층구조를 보여주며, 또한 각 다이아그램에 있는 프로세스 및 각 시나리오에 있는 분해 다이아그램을 보여준다.

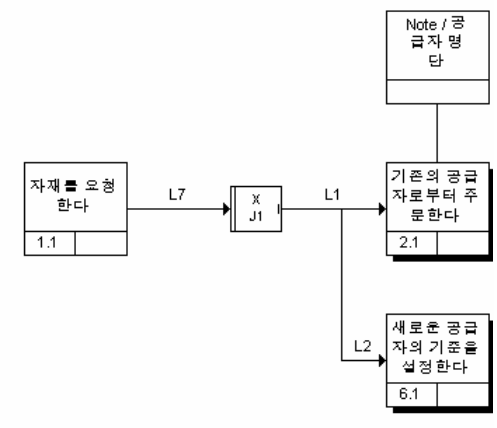


프로세스 흐름 노드리스트 윈도우에서 여러분은 엘리먼트를 드래그-앤-드롭(drag-and-drop)하여 다음과 같은 작업을 할 수 있다.

① 시나리오를 복사한다.
② 새로운 분해구조(decomposition)를 생성한다.
③ 하나의 다이아그램에 있는 내용을 다른 다이아그램으로 옮긴다.
④ 프로세스를 재정렬한다.

■ 프로세스 다이아그램 윈도우(Process Diagram Window)

프로세스 다이아그램 윈도우는 표준 IDEF3 그래픽 디스플레이를 사용하여 각각의 모델을 보여준다. 이는 모델화 된 시스템의 프로세스 중심적 관점을 제공한다.



■ 프로세스/오브젝트 매트릭스 윈도우(Process/Object Matrix Window)

P/O 매트릭스 윈도우는 모델 내의 모든 프로세스와 이들 프로세스와 관련된 모든 개체(object)를 보여준다.

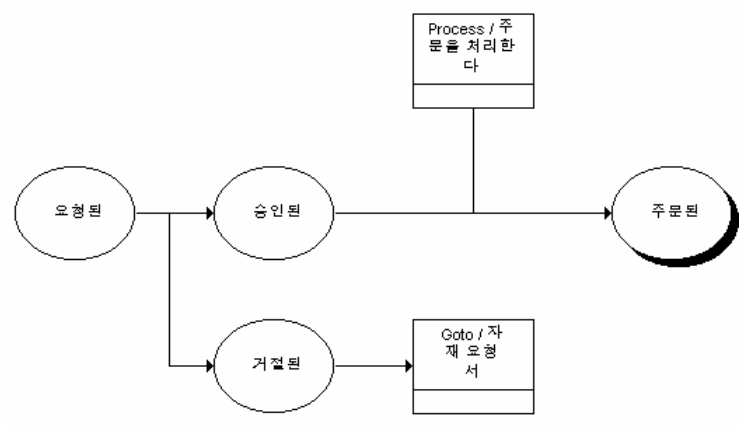
매트릭스 셀(matrix cell)은 프로세스와 개체간에 관계를 가리킨다. 셀은 또한 각 개체의 시뮬레이션 타입(simulation type)을 보여준다.

Legend:
E = Entity
L = Location
Q = Queue
S = Resource
T = Transport
I = Logical
? = Conflict
- = Default

Processes / Objects		자재 주문 이력	기존 공급자	반송처	사양서 작성자 접수처	새로운 공급자	승인처	견적서 작성자	입찰서	입찰참가 자격 부여	수취처	자재	자재 주문 작성자	자재 주문 처리자	자재 주문 창구	자재 품질 평가	자재 품질 미달	출력
		E	E	L	L	E	L	L	E	L	S	L	E	S	S	L	L	S	Q
기존의 공급자로부터 주문한다	L		X				X								X			X	
새공급자로 부터 주문을 한다	L								X						X	X		X	
새로운 공급자의 기준을 설정한다	L					X				X					X			X	
입찰서를 접수한다	L								X	X					X			X	
입찰서를 평가한다	L								X	X					X			X	
입찰을 요청한다	L					X				X					X			X	
자재를 납품 받는다	L											X	X		X			X	
자재를 요청한다	L		X			X		X						X	X			X	

■ 객체상태 전이 네트웍 윈도우(Object State Transition Network (OSTN) Window)

OSTN 윈도우는 프로세스 흐름이 수행되는 동안 객체의 변화를 도형적으로 설명한다.



여러분은 toolbar button  .을 이용하여 윈도우 타입을 빠르게 변경할 수 있다.

우리의 시나리오에 엘리먼트를 추가하여 보자.

4.3.3 시나리오에 엘리먼트 추가하기

우리는 프로세스를 추가하는 것으로부터 시나리오를 만드는 것을 시작하자. 다음 절에서 우리는 프로세스간에 링크를 작성할 것이다 . 각 프로세스는 프로세스 이름과 함께 디스플레이 되는 박스로 다이아그램에서 표현된다. 프로세스 구분 번호는 프로세스가 생성될 때 순차적으로 부여되며, 각 박스의 왼쪽 아래부분에 표시된다.

다이아그램에 프로세스를 추가하는 방법에는 몇 가지 옵션이 있다. PROSIM 의 'Quick Detailing'을 사용하여, 여러분은 새로운 프로세스나 접속(junction), 혹은 분해 다이아그램이 만들어질 때 마다 이름을 물을 것인지 아니면 기본값으로 이름을 부여 할 것인지에 대한 옵션을 지정할 수 있다. 다른 하나의 방법은 PROSIM 의 풀(Pool)을 이용하는 것이다. 풀은 여러분에게 여러분이 시나리오를 작성하는데 있어서 사용될 엘리먼트를 수집하는 것을 지원함으로써 여러분이 '톱-다운' 방식으로 여러분의 프로젝트를 구축하는 메커니즘을 지원한다.

■ 풀(Pool)은 무엇인가

풀은 프로젝트에서 특정한 엘리먼트 타입을 수집하는 저장고이다. – 프로세스, 객체(Object), 객체의 상태(Object Status), properties, fact, constraint, note, source 그리고 unknown – .
여러분이 프로젝트에 이들 엘리먼트를 추가할 때, PROSIM 은 자동적으로 엘리먼트들을 관련된 풀에 저장한다.

풀은 두 가지 중요한 장점을 지원한다.

① 전체 프로젝트에 걸쳐서 엘리먼트를 여러분의 모델로 복사해 갈 수 있는 중앙 저장고 역할 을 한다. 하나의 엘리먼트에 대한 각각의 복사는 *occurrence* 로 간주된다.
② 복사된 모든 엘리먼트를 수정하기 위하여 "마스터 카피(master copy)"로서의 중심점, 또한 마스터 카피에 반영된 수정사항은 모든 복사된 엘리먼트에 영향을 미친다.

■ 프로세스

프로세스는 현실세계 시스템에서 발생되는 절차나 기능에 대한 일반적인 표현이다. 모델 안에서 모든 프로세스는 이들과 관련된 분해 다이아그램들을 가질 수 있다. 분해 다이아그램은 시나리오 다이아그램과 같이 프로세스, 링크, 접속, 그리고 참조(referent)를 가질 수 있다. 그러나, 분해 다이아그램에서 이들 엘리먼트는 상위(Parent) 프로세스에 대한 좀더 자세한 관점을 지원한다.

■ 시나리오에 프로세스 추가

새로운 시나리오에 엘리먼트를 추가하기 위하여는 이장의 4.1 절에서 기술된 시나리오 배경의 원칙에 따른다. 프로세스 흐름 다이아그램 윈도우는 활성화 되었다.

① toolbar 에서  를 클릭한다. 나타나는 프로세스 다이아로그는 프로젝트(즉 프로세스 풀 안에 있는 프로세스) 에 있는 모든 프로세스들을 리스트 한다.
② Name text box 에서 **새공급자로 부터 주문을 한다** 를 등록한 후 **OK** 를 클릭한다. 다이아로그가 닫히고 프로세스 흐름 다이아그램 윈도우로 돌아온다.

■ 프로세스 풀에 프로세스 추가

① Pool menu 에서 *Process...*를 선택한다 혹은 키보드에서 **<Ctrl> + P** 를 친다. 여러분이 이전 절에서 추가한 프로세스 이름이 풀 다이아로그에 나타날 것이다.
② 아래의 각 타입의 프로세스 이름을 Name field 에 등록 한 후 이를 프로세스 리스트에 추가하기 위하여 **Create New** 혹은 **Enter** 를 친다.

자재를 요청한다	**입찰서를 접수한다**
입찰서를 평가한다	**자재를 납품 받는다**
기존의 공급자로부터 주문한다	**입찰을 요청한다**
새로운 공급자의 기준을 설정한다	

③ 프로젝트 윈도우로 돌아가기 위하여 **Done** 을 클릭한다.

■ 풀로부터 프로세스를 시나리오에 추가하기

① Quick Detailing 을 비활성화 시키기 위하여 toolbar 에서  토글(toggle)을 클릭한다.
② 프로세스 다이아로그를 오픈하기 위하여 toolbar 에서  를 클릭한다.
③ 프로세스의 전체 리스트를 선택한다.(시나리오에 이미 나타나있는 **새공급자로 부터 주문을 한다**를 제외하고) 그리고 풀에 들어 있는 모든 프로세스를 추가하기 위하여 **OK** 를 클릭한다.

■ 프로세스의 문서화

여러분이 시나리오의 배경과 목적을 정하기 위해 시나리오와 관련된 텍스트 문서를 연관시킨 것과 같이, 여러분은 시나리오 안에 있는 각각의 엘리먼트나 프로세스를 위하여 같은

작업을 할 수 있다. 여러분이 제공한 문서들은 여러분이나 여러분과 같이 작업하는 멤버들의 모델링 작업을 하기위한 기준이 된다. 여러분의 문서화 작업은 후에 여러분의 작업과 관련될 모델작업자에게 정보전달을 지원하는 중요한 사항이다. 여러분은 **Description, Notes,** 그리고 **Sources** 를 클릭함으로써 프로세스에 관한 추가적인 정보를 등록할 수 있다. 이들 각각의 버튼(button)들은 다이아로그를 오픈하는데 여기에서 여러분은 프로세스의 object, fact, 그리고 constraint 를 추가할 수 있다. ; 또한 프로세스에 관한 설명을 문서화 할 수 있고 프로세스에 note 나 source 를 추가할 수 있다. 여러분은 PROSIM 에서 관련된 파일을 오픈할 수 있는데 프로세스에 관련된 파일을 첨부하기 위하여 **Attachments** 를 클릭하면 된다.

① 프로세스 박스에서 **자재를 요청한다** 를 선택한 후 shortcut menu 를 디스플레이하기 위하여 right-click 을 한다.
② *Edit Process…* 를 선택한다.
③ **Description** 을 클릭한 후 다음과 같은 설명을 등록한다.

모든 작업자는 자재요청을 할 수 있다. 이는 모든 요청들이 시간적인 순서로 진행될 수 있도록 한다.

④ **Source** 를 클릭한 후 당신의 이름을 등록한다.

4.3.4 프로세스 흐름 다이아그램의 작성

이번 절에서는 프로세스의 흐름을 순차적으로 표현하기 위하여 링크(link)와 접속(junction)을 추가 할 것이다. 여러분은 PROSIM 이 드래그-앤-드롭, shift + 클릭, 그리고 Unconnected Element 다이아로그를 이용하여 엘리먼트를 링크하는 여러 가지 방법을 발견할 것이다.

■ 프로세스 흐름 다이아그램이란

프로세스 흐름 다이아그램은 시나리오의 특정 부분이나 독립된 프로세스, 링크, 접속, 그리고 referent 의 도형적인 디스플레이에 초점을 맞춘다. 프로세스 흐름 다이아그램 윈도우에서 프로세스 흐름 다이아그램들은 왼쪽에서 오른쪽으로 읽혀지는데, 기업의 모델화 된 부분에 대한 프로세스의 시간적인 흐름을 나타낸다. 하나의 프로젝트 안에는 두 가지 타입의 프로세스 흐름 다이아그램이 있다.

① **시나리오 다이아그램**은 각 시나리오의 톱 레벨을 나타내고, 다이아그램의 배경을 제공하며 시나리오에 있는 엘리먼트를 보여준다.
② **분해(Decomposition) 다이아그램**은 개별적인 프로세스와 관련되어있는데 각 모 프로세스의 자세한 설명을 다이아그램 형태로 지원한다. 각 프로세스는 이와 관련된 몇 개의 분해 다이아그램으로 나타날 수 있다.

■ 링크(Links)

여러분은 링크와 접속으로 프로세스의 흐름을 빠르게 모델화 할 수 있다. PROSIM 은 여러분의 모델이 프로세스의 흐름을 정확하게 반영할 수 있게 이들 두 가지를 작성할 수 있도록 하는 몇 가지 방법을 지원한다. 프로세스 흐름 다이아그램에서 엘리먼트 사이의 링크는 모델시스템 안에서의 엘리먼트들 사이의 관계를 나타낸다. 여러분은 3 가지 타입의 링크를 사용할 수 있다.

① 시간적 선행 링크(Precedence link)는 다이아그램 엘리먼트들 사이의 시간적인 순차 관계를 표현한다. 선행 엘리먼트(왼쪽에 있는)는 후행 엘리먼트(오른쪽에 있는)가 수행되기 전에 완료되어야 한다. 시간적 선행 링크는 링크의 한쪽 끝이 하나의 화살표를 가진 실선으로 표시된다.
② 개체흐름 링크(Object flow link)는 하나 이상의 객체가 링크를 가로질러 이동하는 것 만이 아니라 시간적 선행 관계를 나타낸다. 개체 흐름 링크는 링크가 끝나는 지점에 두개의 화살표를 가진 실선 (⟶⏩).으로 표시된다.
③ 관계적 링크(Relational link) 는 연결된 엘리먼트들 사이에 정의되지 않은 상태의 연결관

계가 성립함을 나타낸다. : 이 링크 타입에는 미리 연결관계(시간적 선행 링크와 같이) 가 정의되지 않는다. 관계적 링크는 한쪽 끝이 하나의 화살표와 점선(- - -▸)으로 표시된다.

■ 접속(Junctions)

접속은 모델화 된 하나의 프로세스 흐름을 논리적으로 규정할 수 있는 방법을 지원한다. 또한 이것은 다중 프로세스의 순차적, 시간적 특성을 표현한다. 접속은 분기와 결합의 논리적 구조와 시간적 순차적 특성들은 포함한다. 이들 특성은 각 접속 박스 안에서 도형적으로 표현된다.

① 프로세스의 분기와 결합구조를 지원하기 위해 접속은 다음의 두 가지로 구분된다.


하나의 UOB 가 두개 이상의 후속 UOB 로 분기되는 Fan Out(Divergence) Junction


두개 이상의 선행 UOB 가 하나의 UOB 로 결합되는 Fan In(Convergence) Junction

② 프로세스의 논리적 시간적 동시성 여부를 구분하기 위하여 다음과 같은 다섯 가지로 구분된다.

Asynchronous “AND” 접속은 흐름이 계속되기 위하여 이 접속의 앞이나 뒤에 연결된 모든 프로세스가 수행될 것임을 – 비록 모든 것이 동시에 수행될 필요는 없지만 – 나타낸다.

Synchronous “AND” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 모든 프로세스가 동시에 모두 수행되어야 함을 나타낸다.

Asynchronous “OR” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 하 나 이상의 프로세스가, 동시에 수행될 필요는 없지만, 수행되어야 함을 나타낸다.

Synchronous “OR” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 하나 이상의 프로세스가 동시에 수행되어야 함을 나타낸다.

“XOR” 접속은 흐름이 계속되기 위하여 접속의 앞이나 뒤에 연결된 오직 하나의 프로세스만이 수행될 수 있음을 나타낸다.

■ 링크의 추가

우리가 작성하는 자재오더 프로세스 시나리오에서, 자재 오더작업은 자재요청을 처음으로 시작된다. 이와 마찬가지로 자재요청 프로세스는 전체 프로세스 흐름을 시작하는 프로세스이다. 누군가가 자재요청을 하면, 두 가지의 수행 가능한 활동 중 하나가 수행된다. 자재를 기존의 공급자에게 주문하던지 아니면 자재요청을 하기위한 새로운 공급자를 찾아 정하는 프로세스를 수행하던지 하는 것이다.

이러한 논리적인 흐름을 표현하기 위하여, 우리는 우리의 시나리오에 접속**(junction)**을 추가하도록 하겠다. junction 은 '자재를 요청한다' 프로세스 다음에 논리적으로 흐름이 기존의 공급자에게 주문을 하느냐 혹은 새로운 공급자를 개발하느냐 하는 두 가지로 분기된다. 추가적으로 흐름의 분기는 상호 다른 프로세스간에 배타적으로 하나 만 선택해야 하기 때문에 우리는 배타적 OR 접속을 사용해야 한다.

여러분은 XOR 접속을 사용하여 PROSIM 에서 이 같은 흐름의 논리를 표현할 수 있다. 프로세스 흐름이 XOR 에 도달 하면, 흐름은 접속의 fan-out 방향에 연결된 프로세스중의 하나로 계속된다. 우리는 먼저 두 개의 프로세스 사이에 링크를 추가하고 새로운 접속을 이들 사이에 드래그-앤-드롭 하도록 하겠다.

① **자재를 요청한다** 프로세스를 선택한다. (이것은 링크의 시작점이다)
② 여러분은 키보드에서, **Shift** 를 누른 후 , 마우스 커서를 가지고, **기존의 공급자로부터 주문한다** (링크의 끝점)을 클릭한다. PROSIM 은 즉시 두 엘리먼트 사이에 시간적 선행 링크를 생성한다.

■ 접속(junction)의 추가

① toolbar 에서  를 토글링하여 Quick Detailing 을 비활성화 시킨다.
② Fan-out junction 을 다이아그램에 추가하기 위해  를 클릭한다. Add Junction 다이아로그는 다이아그램에 junction 을 추가할 수 있는 절차를 지원한다. 여러분은 각 junction 의 논리와 다이아로그에서 fan 타입을 명시할 수 있다.
③ Junction type group 박스에서 **XOR** 를 활성화 시키고 Fan Type group 박스에서 **Fan-Out** 을 활성화 시킨다.
④ junction 을 생성하기 위하여 **Create** 를 클릭하고 다이아그램에 추가한다. 여러분에게 junction 이 성공적으로 생성되었음을 알리기 위하여 Info dialog 가 오픈된다. Add Junction dialog 로 돌아가기 위하여 Info dialog 에서 **OK** 를 클릭한다.
⑤ 프로세스 흐름 다이아그램 윈도우로 돌아가기 위하여 Add Junction dialog 에서 **Done** 을

클릭한다.

⑥ junction 을 선택한 후 그것을 드래그하여 자재를 요청한다 와 기존의 공급자로부터 주문한다 사이의 링크 위로 움직인다.  커서가 나타날 때 junction 을 링크 위에 떨어뜨린다.

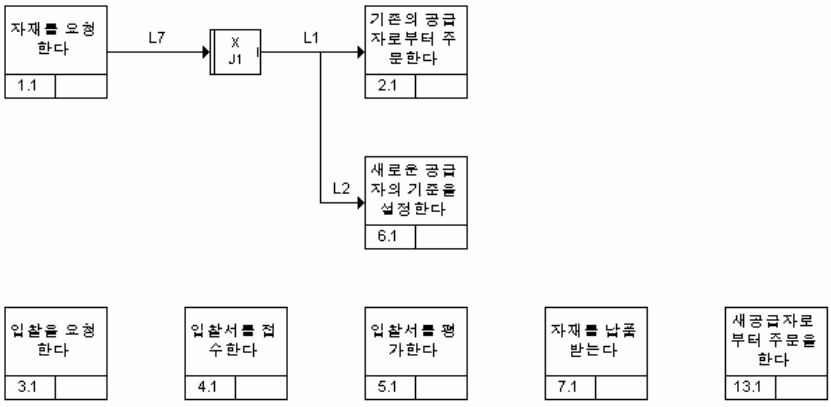
■ 드래그-앤-드롭으로 링크 추가

PROSIM 은 엘리먼트를 링크하기 위한 몇 가지 옵션을 지원한다. 우리는 다이아그램 엘리먼트의 링크를 드래그-앤-드롭이나 Unconnected Elements dialog 를 이용하거나, shift + 클릭을 사용하여 끝낼 것이다.

① 윈도우에서 **Develop New Supplier Specification** 를 선택한 후, 마우스 버튼을 누르고 **J1** 프로세스 위로 드래그한다.

②  커서가 나타날 때, 마우스 버튼을 해제하여 프로세스를 떨어뜨린다. 두 엘리먼트 사이에 precedence 링크가 나타날 것이다.

엘리먼트의 링크가 끝났을 때, 다이아그램은 아래 그림과 같은 모양을 나타낸다.



■ Unconnected Elements Dialog를 사용하여 링크 추가

두 개의 프로세스 중에서 하나로 흐름이 분기되면 - 기존의 공급자로부터 주문한다 그리고 새로운 공급자의 기준을 설정한다 – 자재오더 작업은 새로운 자재를 받는 지점으로 이어진다. 만일 우리의 모델화 된 시스템에서 우리가 기존의 공급선에 주문을 했다면, 다음 단계는 단순히 주문된 자재를 받는 것이다. 그런데 여러분이 새로운 공급선에 주문을 낸다면,

새로운 공급선을 개발하는 프로세스부터 시작하여 이와 관련된 일련의 프로세스를 수행해야 할 것이다.
우리가 새로운 거래선에 대한 명세서 개발을 완료한 후, 우리는 새로운 공급자에게 입찰서를 요청하고, 이를 받아서, 입찰서를 검토할 것이다. 우리의 자재구매 조건과 가장 적절한 입찰서를 결정하고, 우리는 그 공급자에게 주문을 할 것이다.

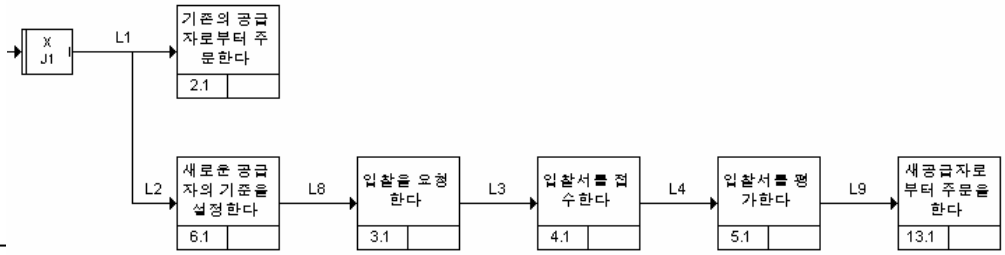
① toolbar 에서  을 클릭한다. 다른 엘리먼트와 연결되지 않았거나 혹은 outgoing link 만 가진 활성화된 다이아그램에 있는 Unconnected Elements dialog list 가 나타난다.
② 다이아로그에서 **입찰을 요청한다** 를 선택한 후 **Select** 를 클릭한다. 지정된 프로세스는 새로운 링크의 시작점이다.
③ 다이아로그에서 **입찰서를 접수한다** 를 선택한 후 링크를 클릭한다. PROSIM 은 기본 링크 타입으로 즉시 precedence 링크를 생성한다.
④ 여러분은 링크를 right-click 함으로써 나타나는 메뉴에서 적절한 타입을 선택함으로써 링크타입을 쉽게 바꿀 수 있다. 여하튼 우리의 연습문제 목적을 위하여 링크를 precedence 링크형태로 유지한다.
⑤ 다이아로그를 닫기 위하여  를 클릭한다.

■ Shift + Click을 이용한 링크 추가

우리의 시나리오 다이아그램에 링크를 추가하자. 나머지 링크에 대하여 우리는 PROSIM 의 Shift + click 기능을 이용하도록 한다.
① 아래에 있는 링크(시작점) 칸의 프로세스를 선택한다.
② **Shift** 키를 누른 상태에서 아래 (종단점) 칸에 있는 프로세스를 클릭한다.

링크(시작점) . . .	(종단점) . . .
새로운 공급자의 기준을 설정한다	**입찰을 요청한다**
입찰서를 접수한다	**입찰서를 평가한다**
입찰서를 평가한다	**새공급자로 부터 주문을 한다**

여러분이 링크하는 프로세스를 끝냈을 때, 다이아그램의 새로운 부분은 다음의 그림과 같은 형태일 것이다.



우리의 다이아그램에서 프로세스 흐름의 양쪽 끝은 주문한 자재를 받는 것으로 같은 결과를 가져오므로 우리는 이러한 흐름의 논리를 표현하는 junction 을 다이아그램에 추가한다. 이전의 XOR junction 의 제약에 따라 J1 에서 나온 두 개의 프로세스 분기 중 하나만 다이아그램의 마지막 프로세스인 자재를 납품 받는다 로 계속된다는 것을 알고있다. 우리는 이러한 조건을 표현하기 위하여 fan-in XOR junction 을 사용한다.

① Quick Detailing 이 비활성화 된 상태에서 toolbar 의  를 클릭한다.
② Add junction dialog 에서, junction Type group box 에서 **XOR** 을 활성화시키고, fan type 을 fan-in 으로 남겨두고 **Link** 를 클릭한다.
③ Info dialog 에서 **OK** 를 클릭한 후 add Junction dialog 에서 **Done** 을 클릭한다.

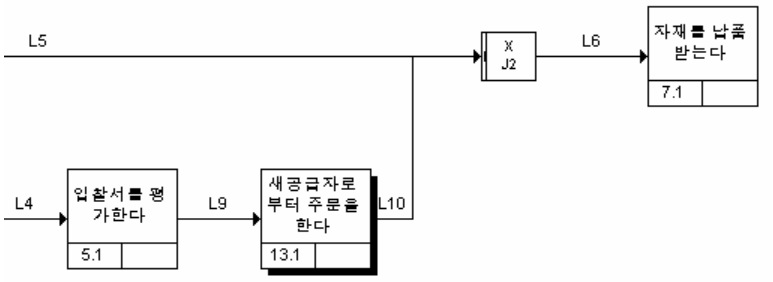
우리는 이제 우리는 junction 을 추가 했으며, 이제 두 개의 프로세스 흐름 가지에 있는 두 개의 마지막 프로세스에 junction 을 추가하자. 우리는 엘리먼트를 부착하기 위하여 Shift + click 방법을 이용한다.

④ **기존의 공급자로부터 주문한다** 프로세스를 선택한 후, **Shift** 를 누르고, 새로운 **J2** junction 을 클릭한다. 두 엘리먼트 사이에 하나의 precedence link 가 나타날 것이다.
⑤ 다음으로 **새공급자로 부터 주문을 한다** 프로세스를 선택한 후, **Shift** 를 누르고, **J2** junction 을 다시 한 번 클릭한다. 이들 두 개의 엘리먼트 사이에 역시 하나의 precedence link 가 나타날 것이다.

우리 작업의 다음단계는 자재를 납품 받는다 프로세스를 junction 에 추가하는 것이다. 우리의 다이아그램이 화면이 지원하는 범위를 초과하여 확장되었기 때문에 두 개의 엘리먼트를 연결하기 위하여 Unattached Elements dialog 를 사용한다.

① **J2** junction 을 선택한 후 toolbar 에서  를 클릭한다.
② Unconnected Elements dialog 에서 **자재를 납품 받는다** 을 선택한 후 **Link** 를 클릭한다.
③ dialog 를 닫기 위하여  를 다시 한 번 클릭하고 프로세스 흐름 다이아그램 윈도우로 돌아간다.

여러분이 엘리먼트의 링크작업을 끝내고 나면 다이아그램의 새로운 부분은 아래의 그림과 같은 형태를 띨 것이다.



우리는 이제 톱 레벨 다이아그램을 완료했다. 우리는 다음으로 프로세스에 분해 다이아그램을 추가함으로써 추가적인 다이아그램 레벨을 작성할 것이다.

4.3.5 다이아그램의 계층구조 생성

시나리오는 계층적 구조의 다이아그램으로 이루어져 있으며 이는 시나리오 다이아그램에서 프로세스의 자세한 설명을 제공하기 위한 것이라는 것을 기억하라.

■ 분해(Decomposition) 다이아그램이란

분해 다이아그램은 기본적으로 특정 프로세스에 대한 자세한 표현으로, 분해된 프로세스가 수행될 때 그 안에서 발생하는 프로세스의 흐름을 보여준다. 다이아그램 내의 각 프로세스는 그와 관련된 어떤 수의 분해 다이아그램도 가질 수 있으며, 하나의 프로세스에 대하여 다양한 관점에서 다양한 다이아그램을 작성할 수 있도록 지원한다. PROSIM 은 어떤 분해 다이아그램이 parent 프로세스의 기본적 분해 다이아그램(**objective view**)인지 그리고 어떤 것이 부차적 표현(**role views**)인지를 명시할 수 있도록 지원한다.

여러분이 기존의 공급자에게서 주문할 때, 프로세스는 실질적으로 두 가지 단계로 구성된다. 첫째로, 여러분은 주문을 위한 승인을 받아야 되며, 그리고 주문을 발송하는 것이다. 우리는 추가적인 정보에 따라 우리의 모델을 수정할 것이다.

■ 프로세스의 분해

① **기존의 공급자로부터 주문한다** 프로세스를 선택한 후 toolbar에서  를 클릭한다.
② Decomposition Name dialog 에서, **기존 공급자 상세화** 를 등록한 후 **OK** 를 클릭한다.

새로운 분해 다이아그램을 디스플레이 하기위해서 새로운 프로세스 흐름 다이아그램 윈도우가 나타날 것이다. 분해 다이아그램의 이름이 title bar 의 틀과 status bar 에 나타날 것이다. 다음으로 우리는 기존의 공급자로부터 주문한다를 구성하는 두개의 프로세스를 다이아그램에 추가할 것이다. 분해 다이아그램 안의 프로세스를 분해된 parent process 에 대하여 child process 라고 한다.

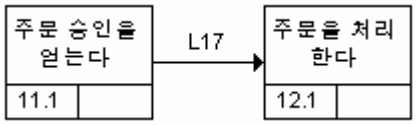
③ 분해 다이아그램 을 나타내기 위하여 toolbar 에서  를 클릭한다
④ Process dialog 에서, Name field 에 **주문 승인을 얻는다** 을 등록한 후 **Create New** 를 클릭한다.
⑤ 같은 절차를 이용하여, **주문을 처리한다** 프로세스를 추가한다.

⑥ 리스트에서 **주문 승인을 얻는다** 과 **주문을 처리한다** 를 선택한 후 **OK** 를 클릭한다. dialog 가 닫히고 프로세스 흐름 다이아그램 윈도우로 돌아간다. 두 개의 새로운 프로세스가

윈도우에 디스플레이 된다는 것을 주시하라.

자 이제 두개의 프로세스를 링크해 보자. 기존의 공급자로부터의 주문하는 프로세스에서 승인된 주문을 발행하기 전에 주문을 위해서 우리는 먼저 승인을 획득하는 부분을 모델링 해야 한다는 것을 추측할 수 있다. 이러한 결과로, 우리는 주문 승인을 얻는다를 시작으로 주문을 처리한다가 끝으로 링크되어지는 두 프로세스 사이의 precedence link 를 추가한다.

⑦ **주문 승인을 얻는다** 을 선택한 후 **주문을 처리한다** 위에서 **Shift + click** 을 한다. 프로세스는 이제 분해 다이아그램에서 링크되었다.



⑧ Parent diagram 으로 돌아가기 위하여 toolbar 에서  를 클릭한다. 기존의 공급자로부터 주문한다는 프로세스 박스 주변에 음영이 나타나는데 이는 프로세스가 분해 다이아그램을 가지고 있다는 것을 나타낸다.

⑨ 기존의 공급자로부터 주문한다의 objective view 로서 분해 다이아그램을 명시하기 위해 프로세스를 right-click 후 나타나는 메뉴에서 *Edit Process...* 를 선택한다.

⑩ Edit Process dialog 에서 **Decomps** 를 클릭한다.

⑪ Process Decompositions dialog 에서 **기존 공급자 상세화** 를 선택한 후 **ObjView <-> RoleView** 를 클릭한다. 하나의 "$"가 리스트 안의 분해 다이아그램 옆에 나타나는데 이는 여러분이 이를 프로세스의 objective view 로 명시하였음을 나타낸다.

⑫ Process Decomposition dialog 를 닫기 위하여 **Done** 를 클릭한 후 Edit Process dialog 에서 프로세스 다이아그램 윈도우로 돌아가기 위하여 **OK** 를 클릭한다.

때때로 여러분은 parent 프로세스에서 이들의 분해 다이아그램을 보고자 하는 경우가 있을 것이다. Parent 대신에 이들 분해 다이아그램을 보기위해 여러분은 옵션을 활성화 시킨 후 parent 프로세스 위에서 right-click 을 한다.

① Diagram Options dialog 를 열기 위하여 Option menu 에서 *Diagram...* 을 선택한다.

② **Replace Marked Elements With Their Decomps** 체크박스에서 모든 항복을 체크한 후 dialog 를 닫기 위하여 **OK** 를 클릭한다.

③ **기존의 공급자로부터 주문한다** 위에서 Right-click 를 하고 메뉴에서 **Show Decomp. In Place** 를 선택한다.

이제 우리는 우리의 프로세스 흐름 다이아그램을 완료했다. 우리의 다음 단계는 프로세스와 우리의 두 다이아그램에 있는 junction 에 object 를 추가하는 것이다.

4.3.6 프로세스 흐름의 문서화

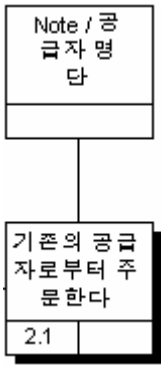
이번 절에서 우리는 시나리오에 referent 를 추가함으로써 프로세스 흐름을 문서화 할 것이다. 여러분은 또한 Process/Object 매트릭스 윈도우를 이용하여 여러분의 모델에서 프로세스와 object 가 어떻게 연관되는지를 배울 것이다.

■ 참조(Referents)

Referent 는 다이아그램 혹은 referent 가 부착된 다이아그램 엘리먼트와 다이아그램 혹은 referent 에 의하여 참조된 다이아그램 엘리먼트 사이에 연관관계를 보여준다.

■ Referent Types

PROSIM 은 여러분에게 **Informational, Call-and-Continue, Call-and-Wait** 그리고 **Goto**, 네 가지의 referent 를 생성할 수 있도록 지원한다. 이다.



① **Informational** referent 는 다른 엘리먼트가 프로세스 흐름에 중요한 정보를 포함하고 있음을 나타낸다. PROSIM 은 informational referent 를 엘리먼트와 관련된 박스와 박스 안에 쓰여진 엘리먼트의 타입을 보여준다.

② **Call-and-Continue** referent 는 다이아그램의 특정 지점에서 흐름 혹은 전이가 계속되기 위하여는 연관된 엘리먼트의 수행이 시작되어야 한다는 것을 지정한다. Call-and-Continue referent 는 박스의 왼쪽에 수직으로 한 개의 선이 그어진 박스로 표시되는데, 비 동기화를 나타낸다. 연관되어진 엘리먼트와 엘리먼트 타입은 박스 안에 리스트 된다.

③ **Call-and-Wait** referent 는 다이아그램의 특정 지점에서 흐름 혹은 전이가 계속되기 위하여는 연관된 엘리먼트의 수행이 시작되고 또한 완료되어야 한다는 것을 나타낸다. Call-and-Wait referent 는 박스의 왼쪽과 오른쪽에 수직의 선이 있는 박스로 표시되는데, 이는 동기화를 나타

낸다. 연관되어진 엘리먼트와 엘리먼트 타입은 박스 안에 리스트 된다.

④ **Goto** referent 는 프로세스 흐름이나 전이를 어떤 엘리먼트로 다시 향하게 한다. Goto referent 는 하나의 프로세스 흐름의 반복이나 순환(loop back)을 나타낸다.

프로세스나 링크, 그리고 junction 과 달리 referent 에는 유일한 번호가 붙지 않는다.
referent 는 프로세스 흐름 다이아그램에서 그들의 타입에 따라 사용되어진 횟수를 가진다.
referent 로 당신은 다음과 같은 것을 할 수 있다.

① 프로세스 흐름 다이아그램에 엘리먼트(중복 없이)를 삽입하거나 혹은 프로젝트 안에서 다른 프로세스 다이아그램 안에 있는 특정 부분을 강조한다.
② 하나의 엘리먼트에 대하여 추가적인 상세사항을 제공하거나,
③ 다른 프로세스 흐름이나 OSTN 다이아그램에 링크를 생성한다.

■ 노트 풀을 이용한 노트의 생성

우리의 톱 레벨 다이아그램에서, 우리는 기존의 공급자로부터 주문한다 프로세스에 informational referent 를 부착한다. 우리는 공급자 리스트를 노트로서 기존의 공급자로부터 주문하는 프로세스에 대한 추가적인 정보로 제공한다. 노트로 주어진 공급자 리스트는 월간 단위로 갱신된다. 우리는 첫번째로 referent 를 위한 노트를 만들고 referent 자체를 만든다.

① Pool 메뉴에서 *Note...* 를 선택하거나 키보드로 **<Ctrl> + N** 을 등록한다.
② Note Pool dialog 에서 Name field 에 **공급자 명단** 을 등록하고 Node list 에 이를 추가하기 위하여 **Create New** 혹은 **<Enter>** 를 누른다.
③ 새로운 note 를 선택한 후 이름을 바꾸고 note 의 작성자와 note 자체의 텍스트를 바꾸기 위하여 "**Edit**"을 클릭한다.
④ descriptive Text field 를 클릭 후 다음과 같은 문장을 등록한다.

공급자 리스트는 월간 단위로 갱신된다 .

⑤ Edit Note Dialog 를 나오기 위하여 **OK** 를 클릭한다.
⑥ Project Window 로 돌아가기 위하여 **Done** 을 클릭한다.

■ Referents의 생성과 부착

① toolbar 에서  를 클릭한다. Add Referent dialog 는 여러분이 새로운 referent 의 타입을 명시할 수 있고 관련된 엘리먼트를 선택할 수 있도록 해준다.

② Referent Type 의 그룹 박스에서 **Informational** 을 선택한 후 Referent Elmt Type 그룹 박스에서 **Note** 를 선택한다. Referenced Element 리스트는 이제 공급자 리스트를 디스플레이 하는데 – 이는 오직 프로젝트 안에서 note 일 뿐이다 – 이를 주목하라.

③ Referenced Element 리스트에서 **공급자 명단** 을 선택 한 후 **Create Ref**.를 클릭한다. 하나의 Info dialog 가 나타나는데, 이는 프로젝트에 referent 가 성공적으로 등록되었음을 나타낸다. Add Referent dialog 로 돌아가기 위하여 Info dialog 에서 **OK** 를 클릭한다.

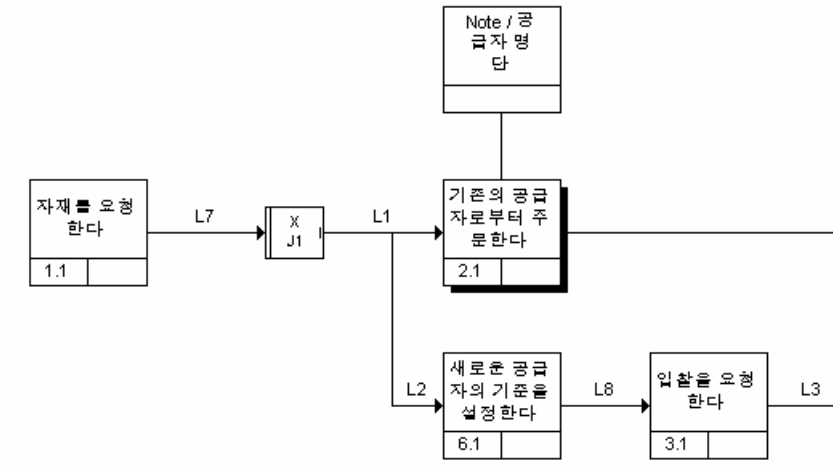
④ dialog 를 닫기 위하여 **Done** 를 클릭한 후 프로세스 흐름 다이아그램으로 돌아간다.

referent 가 프로세스 흐름 다이아그램의 첫번째 엘리먼트 밑에 나타난다는 것을 참고하라.

우리의 다음단계 작업은 기존의 공급자로부터 주문한다 프로세스에 referent 를 추가하는 것이다. referent 를 부착하기 위한 가장 빠른 방법은 PROSIM 의 드래그-앤-드롭 기능을 이용하는 것이다.

⑤ referent 를 선택하고 마우스 버튼을 누른 상태로 **기존의 공급자로부터 주문한다** 위로 referent 를 이동한다.

⑥ 커서가  로 바뀌었을 때, referent 를 부착하기 위해 마우스를 해제한다.



■ 객체(Objects)

Object 는 프로세스를 이루는 프로세스와 junction, 프로세스의 기초적인 구성 요소들인 물리적 혹은 추상적 엘리먼트들이다. 프로세스가 수행되는 동안에 object 는 생성, 수정, 소멸된다. 지금 우리가 엘리먼트에 추가하는 object 들은 우리가 우리의 시뮬레이션 모델을 좀더 자세하게 구성할 때 다시 사용할 것이다.

여러분은 프로세스 흐름 다이아그램이나 PROSIM 의 pool menu 혹은 Process/Object 매트릭스 윈도우 어디에서나 프로세스에 object 를 추가할 수 있다.

자재요청 프로세스에 있어서, 두 가지 종류의 entity 가 프로세스에 의하여 생산될 수 있는데: 기존의 공급자에 대한 주문과 새로운 공급자에 대한 주문이다. 또한 네개의 자원이 이들 entity 를 생산하기 위하여 사용되어지는데: 컴퓨터 단말기, 오더 접수자, 주문서 작성자, 그리고 구매 관리자이다. 이러한 작업은 몇 개의 장소에서 수행되는데, 요청서창구, 입찰서 desk, authorization 구역, 주문창구 그리고 접수처이다.

■ Object Pool에 Object 추가

① Pool menu 에서 *Object...* 를 선택하거나 키보드에서 **<Ctrl> + O** 를 입력한다.
② Object Pool dialog 의 Name field 에 **승인처** 를 등록한 후 **Create New** 를 클릭하거나 **Enter** 를 누른다. Object 가 Object list 에 추가될 것이다.
③ Object Pool dialog 가 열려진 상태에서 다음의 object 를 프로젝트에 추가적으로 추가한다.

입찰서	**새로운 주문요청서**	**주문서 처리자**
입찰창구 주문서	**자재부장**	**컴퓨터터미널**
주문창구	**접수처**	**기존 주문요청서**
주문서 작성자	**요청서창구**	

여러분이 추가하는 object 의 기본 시뮬레이션 타입 값은 location 으로 정해진다. 우리는 특정 object 에 대한 시뮬레이션 타입을 프로세스와 관련되어질 때 수정할 것이다.

④ 프로젝트 윈도우로 돌아가기 위하여 **Done** 을 클릭한다.

■ Matrix view에서 프로세스에 Object 추가

Process/Object 매트릭스 윈도우는 각각의 시나리오 다이아그램에서 프로세스와 object 의 자세한 관계를 나타낸다. 프로젝트의 모든 object 는 수평 축에 리스트 되며, 활성화 된 다이아그램에서 기술되는 프로세스는 수직 축에 리스트 된다. 각각의 object 바로 밑의 셀은 object 의 타입을 나타낸다. 각각의 object 타입은 매트릭스 범례(Legend)에 따라 해당되는 색깔과 문자로 표현된다. 프로세스에 object 를 추가하기 전에, 우리는 우선 몇몇 기존의 object 의 시뮬레이션 타입을 변경해 보도록 하자. 기억해야 할 것은 시뮬레이션 타입은 주어진 프로세스에서 프로세스들에 의하여 object 가 어떻게 이용되는가를 제공하는 것이다.

① toolbar 에서  를 클릭하여 Process/Object Matrix Window 를 open 한다.
② **입찰서** object 를 right-click 한다.
③ Right-click 후 나타나는 메뉴에서 ***Entity*** 를 선택한다. 입찰서는 입찰서를 평가한다에 의해 진행되는 entity 이다.

같은 절차로 다음 테이블에 있는 object 에 시뮬레이션 타입을 정한다.

Object	Simulation Type
컴퓨터터미널	**Resource**
기존 주문요청서	**Entity**
새로운 주문요청서	**Entity**
주문서	**Entity**
주문서 작성자	**Resource**
주문서 처리자	**Resource**
자재부장	**Resource**

자 이제 우리는 object 들에 시뮬레이션 타입을 설정 했으니, 톱 레벨 다이아그램에 있는 프로세스들에 object 를 추가하도록 하자.

■ Process/Object Matrix Window에서 프로세스에 Object 추가

① 프로세스인 **자재를 요청한다** 과 object 인 **컴퓨터터미널** 사이에 교차되는 셀을 클릭한다

"X'표시가 매트릭스 셀에 나타나게 되는데, 이는 object 가 프로세스와 관련되어졌음을 나타낸다. 같은 절차를 이용하여 다음 테이블에 따라서 남아있는 object 를 다이아그램의 프로세스에 추가한다.

Object	Destination Process
승인처	기존의 공급자로부터 주문한다
입찰서	입찰서를 평가한다
	새공급자로 부터 주문을 한다
	입찰서를 접수한다
입찰창구	새로운 공급자의 기준을 설정한다
	입찰서를 평가한다
	입찰서를 접수한다
	입찰을 요청한다
컴퓨터터미널	새로운 공급자의 기준을 설정한다
	입찰서를 평가한다
	기존의 공급자로부터 주문한다
	새공급자로 부터 주문을 한다
	입찰서를 접수한다
	자재를 납품 받는다
	입찰을 요청한다
	자재를 요청한다
기존 주문요청서	기존의 공급자로부터 주문한다
	자재를 요청한다
새로운 주문요청서	새로운 공급자의 기준을 설정한다
	입찰을 요청한다
	자재를 요청한다
주문서	자재를 납품 받는다
주문창구	새공급자로 부터 주문을 한다
주문서 작성자	자재를 요청한다
주문서 처리자	새로운 공급자의 기준을 설정한다
	입찰서를 평가한다
	기존의 공급자로부터 주문한다
	새공급자로 부터 주문을 한다
	입찰서를 접수한다
	자재를 납품 받는다
	입찰을 요청한다
	자재를 요청한다
접수처	자재를 납품 받는다
요청서창구	자재를 요청한다

여러분이 프로세스에 object 를 추가하는 작업을 마치고 나면 매트릭스는 아래의 그림과 같은 형태를 가질 것이다.

Legend:
E = Entity
L = Location
Q = Queue
S = Resource
T = Transport
1 = Logical
? = Conflict
- = Default

Processes	Objects	기존 주문 요청서	사양서 작성자 접수처	새로운 주문 요청서	승인처	요청서 창구	입찰서	입찰창구	자재부장	접수처	주문서	주문서 작성자	주문서 처리자	주문창구	컴퓨터 터미널	
		E	L	E	L	L	E	L	S	L	E	S	S	L	S	
기존의 공급자로부터 주문한다	L	X			X								X		X	
새공급자로 부터 주문을 한다	L						X						X	X	X	
새로운 공급자의 기준을 설정한다	L			X				X					X		X	
입찰서를 접수한다	L						X	X					X		X	
입찰서를 평가한다	L						X	X					X		X	
입찰을 요청한다	L			X				X					X		X	
자재를 납품 받는다	L									X	X		X		X	
자재를 요청한다	L	X		X		X						X	X		X	

4.3.7 객체상태 전이 네트웍(Object State Transition Networks)

이번 절에서 여러분은 객체상태 전이 네트웍(OSTN)의 톱 레벨 다이아그램을 작성한다. OSTN 은 하나의 object 에 관한 object 의 상태를 나타내는 다이아그램이다. OSTN 다이아그램 윈도우는 하나의 프로세스나 프로세스들이 수행될 때 object 상태의 변화가 일어나는 과정을 보여줌으로써 프로젝트에서 프로세스의 object 중심의 관점을 제공한다.
OSTN 은 시나리오와 같이 하나의 톱 레벨 다이아그램과 다이아그램 안의 object 상태와 관련된 여러 개의 분해 다이아그램으로 이루어져 있다.

■ OSTN의 생성

우리는 자재요청이 수행되는 동안 – 요청되는 상태에서부터 접수되는 상태까지- 의 상태 변화를 나타내는 OSTN 을 작성할 것이다.

① toolbar 에서  을 클릭한다.
② OSTN dialog 에서 Name field 에 **자재 요청서** 를 입력한 후 **Create New** 를 클릭하거나 **<Enter>**를 누른다. 새로운 OSTN 의 이름이 리스트에 나타날 것이다.
③ dialog 를 닫기 위해 **OK** 를 클릭한 후 새로운 OSTN 을 오픈한다.

여러분은 프로젝트와 시나리오를 문서화 했던 것처럼 같은 방법으로 OSTN 을 문서화 할 수 있다. Diagram menu 에서 *Diagram Summary*...를 선택한다. 우리는 이번 단계를 생략하고 object status 를 추가하는 작업으로 가보자.

■ 객체상태(Object States)

Object state 는 프로세스나 프로세스집합 안에서 하나의 예상할 수 있는 object 의 특정상태를 표현하는 것이다. 프로세스와 같이, object state 는 object 의 현재 상태에 대하여 제약사항을 설명하는 elaboration 에 의해 object status 가 특성화 될 수 있다.

① **Entry** 는 object 가 active state 로 변화되는데 요구되는 제약사항을 명시한다.
② **State** 는 object 가 active state 로 유지되는데 필요한 제약사항을 명시한다.
③ **Exit** 는 Object 가 active state 에서 퇴장하는데 필요한 제약사항을 명시한다.

■ OSTN에 객체상태 추가

① toolbar 에서  를 토글하여 비활성화 시킨다.
② Object Status dialog 를 열기 위하여 toolbar 에서  를 클릭한다.
③ 다음의 object state 이름들을 Name field 에 입력 후 **Create New** 를 클릭하거나 **<Enter>**

를 누른다. Object states 가 다이아로그 리스트에 나타날 것이다.

요청된 **승인된**

주문된 **접수된**

반려된

④ 리스트에 있는 모든 object status 를 선택한 후 **OK** 를 클릭한다.

OSTN 윈도우에 모든 object states 가 나타날 것이다.

■ 전이 원호선(Transition Arcs)

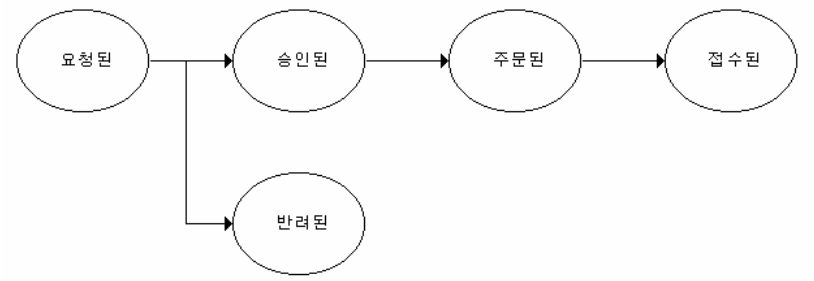
OSTN 은 프로세스 흐름 다이아그램과 같이 복잡한 흐름을 논리적으로 모델화하지 않는다. junction 을 사용하는 대신, OSTN 은 transition arc 로 흐름의 분기를 표시한다. Transition arc 는 연결되거나 "arc(원호)"된 object 사이에 전이 가능한 것을 표현한다. 각 arc 는 시간적, 순차적 전이를 표시한다.

■ Object States의 링크

자재 요청서 form 의 초기상태에서 object 는 두 개의 가능한 상태 중 하나로 전이될 수 있다. 승인되거나 반려된다. 따라서 우리의 OSTN 다이아그램은 두 개의 가능한 출구가 있도록 표현되어야 한다. 추가적으로, 한 번 object 가 승인되어지면 그것은 주문되어야 하며 그리고 접수되어야 한다.

① **요청된** object state 를 선택한다.
② Shift key 를 누르고 우선 **승인된** object state 를 클릭한 후 **반려된** obect state 를 클릭한다.
③ **승인된** object state 를 선택한다.
④ Shift key 를 누르고 **주문된** 를 클릭한다.
⑤ **주문된** object state 를 선택한다.
⑥ Shift key 를 누르고 **접수된** 를 클릭한다.

여러분이 object states 의 링크작업을 끝 냈을 때 다이아그램은 아래의 그림과 같은 형태를 띨 것이다.



■ OSTNs에서의 Referent

프로세스 흐름 다이아그램에서와 같이, OSTN 에서의 referent 는 특정 전이 지점에서 다른 프로젝트 엘리먼트와 관련되거나 혹은 프로젝트 안의 다른 다이아그램이나 혹은 엘리먼트와 연결된 OSTN 에 관련되어서 사용된다.

■ OSTN에 Referent 추가

우리의 다이아그램을 완료하기 위하여, 우리는 상태 전환에 관련된 프로세스를 명시할 필요가 있다. 자재 요청서의 state 변화는 receiving the material 의 프로세스에서 발생되는 주문에서 접수까지의 상태이다. 우리는 자재를 납품 받는다 프로세스에 informational referent 를 추가하고 주문된 와 접수된사이의 transition arc 에 부착 할 것이다.

추가적으로, 요청이 반려되는 경우, 그것은 검토 프로세스가 진행되는 동안에 승인되기 전에 요청된 state 로 되돌려지게 된다. 이러한 조건을 반영하여 우리는 네트웍의 흐름에서 반려된 상태에서 Request 상태로 되돌려지는 GoTo referent 를 추가할 것이다.

① toolbar 에서  를 클릭한다.
② Referent Type group box 에서 Informational 을 활성화 시키고 Referencde Elmt Type group box 에서 **Process** 를 활성화시킨다.
③ Referenced Element list 에서 자재를 납품 받는다을 선택하고 **Create Ref**.를 클릭한다. Info dialog 에서 **OK** 를 클릭한다.
④ Referent Type group box 에서 **Goto** 를 선택한 후 Referenced Elmt Type group box 에서 **OSTN** 을 선택한다.
⑤ Reference Element list 에서 **1 자재 요청서** 를 선택한 후 **Create Ref** 를 클릭한다. Inof dialog 에서 **OK** 를 클릭한다.
⑥ dialog 를 닫기 위해 **Done** 을 클릭한 후 OSTN Window 로 돌아간다.

■ Referents의 부착과 링크

① informational referent 인 **Process / 자재를 납품 받는다** 를 선택한 후 **주문된** 와 **접수된** 사이에 있는 transition arc 위로 drag 한다.

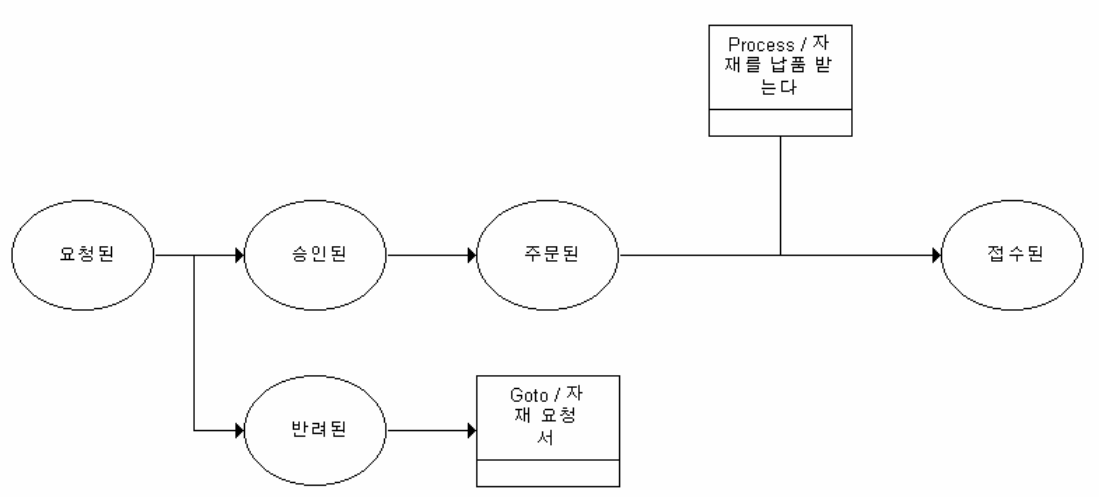
② 커서가 로 바뀌어졌을 때 마우스 버튼을 해제한다. Referent 가 transition arc 에 부착될 것이다.

Goto referent 는 직접적으로 네트웍 흐름의 논리에 관여되기 때문에 object state 에 부착되기 보다는 직접적으로 링크되어야 한다.

③ Object state 인 **반려된** 을 선택한다.

④ **Shift** 를 누른 상태에서 **Goto / 자재 요청서** referent 를 클릭한다. Object state 가 referent 에 링크될 것이다.

여러분이 이러한 단계를 마치고 나면 여러분의 OSTN 다이아그램은 아래와 같은 형태를 나타낼 것이다.



4.3.8 Model의 프리젠테이션

PROSIM 은 당신의 모델을 디스플레이 하고 프리젠테이션 하는데 있어서 최적화 를 지원하기 위한 몇 가지 특성을 지원하다. 이번 절에서는 어떻게 Custom Display 와 프리젠테이션 모드를 활성화 시키는가 그리고 필요한 자료만을 출력하기 위하여 프린트 옵션을 어떻게 이용할 것인가를 보여줄 것이다.

■ 프리젠테이션 특성

PROSIM 은 여러분의 모델을 디스플레이하고 프리젠테이션 하는데 있어서 최적화를 위한 몇 가지 특성을 가지고 있다. Custom Display 는 프로세스 흐름 다이아그램이나 OSTN 다이아그램 윈도우에서 엘리먼트나 문장의 디스플레이를 최적화 할 수 있도록 지원한다. **Custom Display** 모드에서 여러분은 다음과 같은 작업을 할 수 있다.

① 평범한 IDEF3 의 프로세스, junction, object status, 그리고 referent 의 그림을 사용자의 bitmap 으로 대치한다.
② 엘리먼트 제목의 글자타입이나 크기를 customize 한다. 그리고
③ 선의 두께나 타입을 바꾸고, 링크나 transition arc 의 색상을 바꾼다.

■ Custom Display 절차

Presentation Mode 는 여러분이 **활성화된 프로세스 흐름**이나 OSTN 다이아그램에서 프로세스나 transition flow 를 깨뜨리지 않고 여러분이 선택한 다이아그램의 엘리먼트를 재 배치할 수 있도록 지원한다. 또한 이 모드는 여러분의 다이아그램에 swim lane 을 추가할 수 있도록 지원하는데 이러한 기능은 여러분의 다이아그램을 구성하거나 활성화 된 다이아그램을 복사하거나 출력하기 전에 엘리먼트의 위치를 정할 때 유용하게 사용된다.
PROSIM 의 프린트 옵션 다이아로그를 이용한 출력은 출력될 파일에 포함 할 field 를 선택할 수 있는 몇 가지 옵션을 제공함으로써 *.rtf 나 HTML 호환 문서를 구성 할 수 있도록 지원한다. 각 프린트 옵션 다이아로그에 있는 각 그룹 박스는 다른 것들과 조합되어 작동되는데 예를 들면 여러분은 특정 엘리먼트에 관련되어진 정보나 혹은 모든 엘리먼트에서 발생되는 정보를 출력할 수 있다.

① Custom display mode 를 활성화 하기 위하여 toolbar 에서  를 클릭한다.

② **자재를 납품 받는다** 프로세스를 right-click 한 right-click menu 에서 *Customize Display* 를 선택한다.

③ Box Display Option dialog 에서 여러분은 프로세스의 제목을 위한 새로운 글자 크기와 타입을 선택할 수 있으며; 프로세스 박스 라인의 스타일, 폭, 색상을 조절하고; 혹은 프로세스 박스 전체를 사용자의 bitmap 으로 대체할 수도 있다.

④ 다이아로그에서 가능한 모든 옵션을 활성화 시키기 위하여 **Custom Display This Element** 를 체크 한다.

- 프로세스 제목의 font type, 스타일, 크기 그리고 색상을 선택할 수 있도록 Font dialog 를 열기 위해 **Font** 를 선택한다.
- Line Drawing Option box 에서 line 스타일, 폭(선택된 line 이 깨지지 않은 line 인 경우), 그리고 색상을 지정하기 위하여 옵션을 선택한다.
- text field 와 어울리게 평범한 도형을 사용자의 bitmap 으로 변환하기 위하여 **Display Custom Bitmap** 을 체크한다.
- bitmap 을 찾고 프로세스에 할당하기 위하여 **Browse** 를 클릭한다. PROSIM 소프트웨어에서는 이용 가능한 bitmap file 들을 지원한다.
- 사용자의 bitmap 에 관계에 있어서 어디에 엘리먼트의 제목이 나타나게 할 것인지를 지정하기 위하여 Position of Element Name group box 에서 하나의 옵션을 선택한다.

⑤ 변경사항을 저장하고 dialog 를 닫기 위하여 **OK** 를 클릭한다.

■ 링크의 사용자 디스플레이

① custom display mode 가 아직 활성화 된 상태에서, link **L1** 을 right-click 하고 right-click 메뉴에서 *Customize Display* 를 선택한다.
이때 나타나는 Link Display Option dialog 는 link 의 새로운 중단점을 선택하고 link line 의 스타일, 폭, 그리고 색상을 조절할 수 있도록 지원한다.

② dialog 에 있는 모든 display option 을 활성화 하기 위하여 **Custom Display** 를 체크한다.

- Link Terminator drop-down list 에서 link 가 끝나는 지점에서 나타나게 할 도형 item 을 선택한다.
- line 스타일, 폭(선택된 line 스타일이 깨지지 않은 line 인 경우), 그리고 색상을 지정하기 위하여 Line Drawing Option box 에서 옵션을 선택한다.

③ 변경사항을 저장하기 위하여 **OK** 를 클릭하고 dialog 를 닫는다.

■ 프리젠테이션 모드에서 엘리먼트의 정렬

① 프리젠테이션 모드를 활성화 하기 위하여 toolbar 에서  를 클릭한다.
② 하나의 엘리먼트를 right-click 한 후, 마우스 버튼을 누른 상태에서 화면의 다른 곳으로 이동한다.
③ 커서가  형태로 나타날 때, 마우스를 해제하여 떨어뜨리거나 커서가 있는 곳으로 엘리먼트의 위치를 바꾼다.
④ 프리젠테이션 모드를 비활성화 하기 위하여  를 클릭한다. 다이아그램은 원래의 자동 경로 연결 상태로 돌아 갈 것이다.

■ 스윔레인의 추가

프리젠테이션 모드가 활성화 되었을 때, 여러분은 다이아그램에 swim lane 을 추가할 수 있다.

① Presentation Mode 를 활성화 하기 위하여  를 누른다.
② Option menu 에서 *Diagram*...을 선택한다.
③ General Option group box 에서 **Enable User Placement** 옵션을 선택한다. **Enable Swim Lanes** 옵션이 자동적으로 선택 될 것이다.
④ Diagram Option dialog 을 나가기 위하여 **OK** 를 클릭한다.
⑤ Diagram menu 에서 *Edit Swim Lanes* 를 선택한다.
⑥ 두개의 swim lane 를 생성한 하기위해 Name field 에 다음과 같은 이름을 입력한 후, **Create New** 나 **<Enter>**를 누른다.

타 부서 **구매부서**

⑦ Swim Lanes dialog 에서 나오기 위하여 **Done** 을 클릭한다.

이제 여러분은 여러분이 원하는 바에 따라 프로세스와 junction 을 재정렬 한다.

■ 프린팅

Print Option dialog 는 다이아그램이나 특정 프로세스 문서에 있어서 여러분이 원하는 정보만 출력할 수 있도록 지원한다. 이번 예제에서는 세가지 형식의 출력물을 발견할 수 있을 것이다.

① File menu 에서 *Print*...를 선택한다.

다이아그램의 프린팅

② Print Option dialog 에서 Out To box 에 있는 **Printer** 를 선택한다.
③ Scenario group box 에 있는 **Graphics** 를 선택한다.
④ Print Scope box 에서 프린트 작업을 위한 범위를 선택한다.
⑤ Format Option field 에서 출력물의 형태인 **Diagram** 을 클릭한다.
⑥ **OK** 를 클릭한다.

프로젝트 레벨 정보의 프린팅

⑦ Print Option dialog 에서, Elements to Print box 에 있는 여러분이 출력하고자 하는 pool 을 선택한다.
⑧ Fields to Print list 에서 어떤 추가적인 정보라도 선택한다.
⑨ Print Scope box 에서 **Entire Project** 를 선택한다.
⑩ **OK** 를 클릭한다.

활동 되는 특정 정보의 프린팅

⑪ Elements to Print group box 에서 **Process Occurrence Info.** 혹은 **Obj. State Occurrence Info.** 를 선택한다.
⑫ Print Scope box 에서 출력작업을 위한 범위를 정한다.
⑬ **OK** 를 클릭한다.

4.3.9 Importing and Exporting Models

PROSIM 의 import 와 export 기능은 다른 분야의 모델로 변화와 교환을 가능토록 해준다. 이번 과에서는 KBSI 의 tool, 특히 AIØ WIN 이나 SMARTER 로 import 와 export 하는 것을 소개하도록 하겠다. 여러분은 쉽게 PROSIM 으로 activity 모델을 import 하거나 SmartER 로 프로세스 모델을 export 할 수 있다. 추가적으로, 여러분은 이미 여러분의 프로세스 모델에서 만들어진 text file 을 import 할 수 있고; 전체 프로젝트나 하나의 모델을 export 할 수 있다.

■ 활동모델의 Importing

PROSIM 이 활동모델을 import 할 때, PROSIM 은 부분적으로 import 된 모델을 재구성 한다. 엘리먼트들은 풀로 들어가서 여러분이 프로젝트에서 이것들을 모델에 사용할 수 있도록 한다.

① Activities – 프로세스로 변환되며 프로세스 풀에 위치한다.
② Concepts – object 로 변환되며 Object 풀에 위치한다.

■ 프로세스 모델의 Exporting

여러분이 PROSIM 으로부터 정보를 export 할 때, 여러분은 몇 가지 옵션을 가질 수 있다. 여러분은 전체 프로젝트나, 독립된 다이아그램, 선택된 풀, 혹은 시뮬레이션 모델을 export 할 수 있다. 우리는 시뮬레이션 모델과 시뮬레이션 모델 생성에 관해서는 다음 절에서 토의할 것이다. 여러분이 프로세스 모델을 export 하고 이것을 SMARTER 에 import 할 때, 엘리먼트는 여러분이 프로젝트에서 이것들을 모델로 분배할 수 있도록 풀에 들어간다.

① Processes – Unknown Pool 에 들어간다.
② Objects – entities 로 변환되어 Entity Pool 에 들어간다.
③ Properties – attribute 로 변환되어 Attribute Pool 로 들어간다.

4.3.10 PROSIM을 이용한 시뮬레이션 모델링

여러분이 모델에 시뮬레이션 정보를 추가하면 여러분의 관점은 프로세스 중심에서 object 중심으로 이동하게 된다. 이번 과에서는 PROSIM 의 시뮬레이션 기능에 대하여 소개하도록 한다.

PROSIM 은 모델화 된 시스템의 프로세스 중심적 관점을 나타낸다. WITNESS®은 Lanner Group 의 시뮬레이션 엔진으로서 여러분의 모델을 시스템의 object 중심의 관점으로 변환한다. PROSIM 은 프로세스, junction, 그리고 object 에 대하여 여러분이 시뮬레이션 파라메터를 빠르게 정의할 수 있도록 지원하는데, 정의된 시뮬레이션 리포트는 WITNESS 에서 볼 수 있으며; 시뮬레이션 을 위해 모델을 검증할 수 있으며, 그리고 WITNESS 가 읽을 수 있는 시뮬레이션 모델을 생성한다.

■ 시뮬레이션이란

프로세스 모델은 모델화 된 시스템의 프로세스 중심적 관점을 표현하며 시스템이 어떻게 수행되는지에 대한 논리적 추측으로 구성된다.
시뮬레이션 모델에 있어서는, 이러한 프로세스 중심의 관점이 object 중심의 관점으로 되는데, 모델 작업자에게 이산시점에서 모델의 수행성을 정량적으로 검증할 수 있도록 지원한다.

■ 프로세스와 시뮬레이션

프로세스가 완료되기 위하여 소요되는 시간이나 한 번 수행되는데 자원이 어떻게 사용되는지를 결정하는 것과 같이 프로세스 시간과 자원의 룰은 하나의 프로세스 기능이다. 이러한 자료의 설정은 WITNESS 에 있어서 cycle time 과 labor 이다.
이번 예제는 PROSIM 에서 사용하는 다양한 simulation-specific dialog 로서 여러분을 친숙하게 지원할 것이다. 우리는 다시 프로세스 흐름 다이아그램 윈도우로 돌아간다.

① Windows menu 에서 **Scenario: 자재 주문 과정** 를 선택한다.
② **자재를 요청한다** 을 클릭한 후 menu 를 Right-click 하여 *Edit Simulation Info…*를 선택한다.

Sim. For dialog 는 프로세스와 관련된 entity 들을 리스트 한다. 여러분은 각 entity 에 대한 프로세스 시간과 자원 룰을 지정할 수 있다. 여러분은 또한 프로세스와 연관된 object 를 수정하고 , 각 자원의 비용을 정의하며, 프로세스에 있는 entity 들을 위한 어떤 종류의 수정도 할 수 있다.

■ 프로세스 시간의 할당과 자원 룰의 정의

여러분은 연관된 각 entity 와 자원에 대하여 프로세스가 그 작업을 완료하는데 어느 정도 시간을 소모하는지를 정의할 수 있다. Sim. Info For dialog 는 여러분에게 프로세스 시간을 위한 함수를 작성하고 정확한 시간을 정할 수 있도록 지원한다. 여러분은 text field 에서 쉽게 resource rule 을 만들거나 직접 입력할 수 있다.

■ 자원을 위한 비용 정의

Resource Costs dialog 를 access 하기 위하여 **Costing** 을 클릭한다. 여러분은 특정 cost unit 이나 미리 정의된 함수를 사용하여 어떤 자원에도 비용을 할당할 수 있다.

■ Changing Entity Types

여러분은 어떻게 하나의 entity 가 들오고 나가는 과정에서 선택된 프로세스가 entity 를 변화시키는지를 표현할 수 있다. 자재를 요청한다 프로세스에서는 아무런 타입의 변화가 없는 동안 기존의 공급자로부터 주문한다에서는 요청된 주문이 실질적인 주문이 될 때 타입의 변화가 발생한다. 변화가 발생되는 것을 표현하기 위하여, 기존의 공급자로부터 주문한다 프로세스를 right-click 한 후 *Edit Simulation Info..*를 선택한다.

① Sim. Info dialog 에서 **Entity Type Change** 를 클릭한다. Entity Type Change dialog 는 프로세스에 연관된 각 entity 를 리스트 하는데, 이들 각 input entity 들에 output entity 를 관련짓게 할 수 있도록 지원한다.
② **기존 주문요청서** 를 선택한 후, Available Entities drop-down 리스트에서 **주문서** 를 선택한다.
③ **Change Entity** 를 클릭한다. Output Entity 열서 주문서가 나타날 것이다.
④ Entity Type Change Logic dialog 에서 **OK** 를 클릭한다.
⑤ 프로세스 흐름 다이아그램으로 돌아가기 위하여 Sim. Info For dialog 에서 **OK** 를 클릭한다.

■ Objects 와 시뮬레이션

object 에 대한 시뮬레이션 정보는 프로세스 안에서 object 가 어떻게 관여하는지를 표현한다. 예를 들면, entity 들을 위한 시뮬레이션 정보는 다음 내용을 포함한다.

① 프로세스 안에서 어떻게 entity 가 도착하는가
② 도착하는 entity 의 숫자
③ entity 의 도착시간 간격

■ Object를 위한 시뮬레이션 정보의 정의

object 와 관련된 시뮬레이션 정보는 object 타입이나 프로세스 안에서 object 의 역할에 따라서 다양하다. 우리는 자재를 요청한다 프로세스와 관련된 object 타입을 검토해 보자.

① **자재를 요청한다** 을 right-click 을 한 후 *Object…*를 선택한다.
② Sim. Info dialog 에서, 프로세스와 관련된 object 를 그들의 시뮬레이션 타입과 별명(alias)에 따라서 리스트를 디스플레이 하기 위하여 **Objects** 를 클릭한다. 여러분은 프로세스에 object 를 추가할 수도 있고 제거할 수도 있다.; 선택된 object 를 수정하고; 그리고 object 와 관련된 시뮬레이션 정보를 수정한다.
③ 하나의 entity 를 선택하고 **Detail Sim Info**.를 클릭한다. Edit Entity dialog 에서 여러분은 entity 의 시뮬레이션 정보를 정의하고, entity 를 표현하는 WITNESS 아이콘을 정하고, entity 의 별칭을 변경한다.

■ Junctions 과 시뮬레이션

시뮬레이션에서, junction은 일반적으로 기계(machine)로 표현된다 그리고 어떻게 entity를 생산 혹은 결합하는지를 표현한다. 예를 들면, fan-out AND junction은 몇 개의 entity를 생산하는데 따라서 production machine으로 WITNESS에 매핑된다.

■ Junction을 위한 시뮬레이션 정보의 정의

PROSIM 은 여러분에게 junction 이 어떻게 들어오는 entity 를 생산하고 결합하는지 혹은 다른 프로세스에 어떤 경로로 entity 를 내보내는지를 결정할 수 있도록 한다.

J1 XOR junction 을 right-click 한다. 그리고 *Edit Simulation Info…*를 선택한다. Junction type(OR, AND, 혹은 XOR)에 따라서 시뮬레이션 정보를 추가할 수 있다.

이 XOR junction 을 위해서 여러분은 다음과 같은 작업을 할 수 있다.

① Current Logic Type group box 에서 옵션을 선택하여 junction 의 logic type 를 변경한다.
② **Detail Logic** 을 클릭하여 junction logic 을 상세화 한다.
③ **Sim. Icon** 을 클릭하여 WITNESS 아이콘을 할당한다.

■ 시뮬레이션 모델의 생성

여러분이 여러분의 모델에 시뮬레이션 정보를 추가하고 난 다음에, 여러분은 WITNESS 에서 생성하고자 하는 리포트와 그래프를 정할 수 있다. PROSIM 의 Verification Option 은 여러분이 WITNESS 파일을 생성하기 전에 모델을 다시 한 번 검토할 수 있도록 지원한다.

■ 시뮬레이션 리포트의 정의

PROSIM 은 활성화 된 다이아그램에서 entity 의 용량이나 location 의 비용과 활용에 관한 자세한 리포트를 WITNESS 에서 생성할 수 있도록 지원한다. 여러분은 또한 다양한 타입의 비용, 활용성, 그리고 용량 인자들을 WITNESS 그래프로 나타낼 수 있다.

여러분은 최종적으로 생성되는 시뮬레이션과 상세화 된 이들 각각의 리포트에서 모델 안의 object 에 관한 비용, 활용성, 그리고 용량 리포트를 포함할 것인가 아닌가를 명시할 수 있다. 여러분은 또한 WITNESS 의 시뮬레이션 리포트의 그래프 타입과 그래프 아이템, 그리고 그래프 생성을 위한 인자를 상세화 할 수 있다.

WITNESS 가 생성하는 리포트는 시뮬레이션 모델이 PROSIM 에서 생성 되었을 때 다이아그램을 생생하게 상세화 한다.

■ 모델의 검증

PROSIM 이 시뮬레이션 모델을 검증할 때, 행위나 WITNESS 와 관련되어 시뮬레이션이 성공적으로 수행되기 위하여 꼭 있어야 되는 사항에 대한 아이템화 된 체크리스트를 통하여 진행된다. 검증은 오직 활성화 된 다이아그램에 대해서만 이루어진다. 전체 시나리오을 검증하기 위하여는 여러분은 각각의 다이아그램에 대한 모델 검증을 수행하여야 한다.

■ 시뮬레이션 모델의 생성

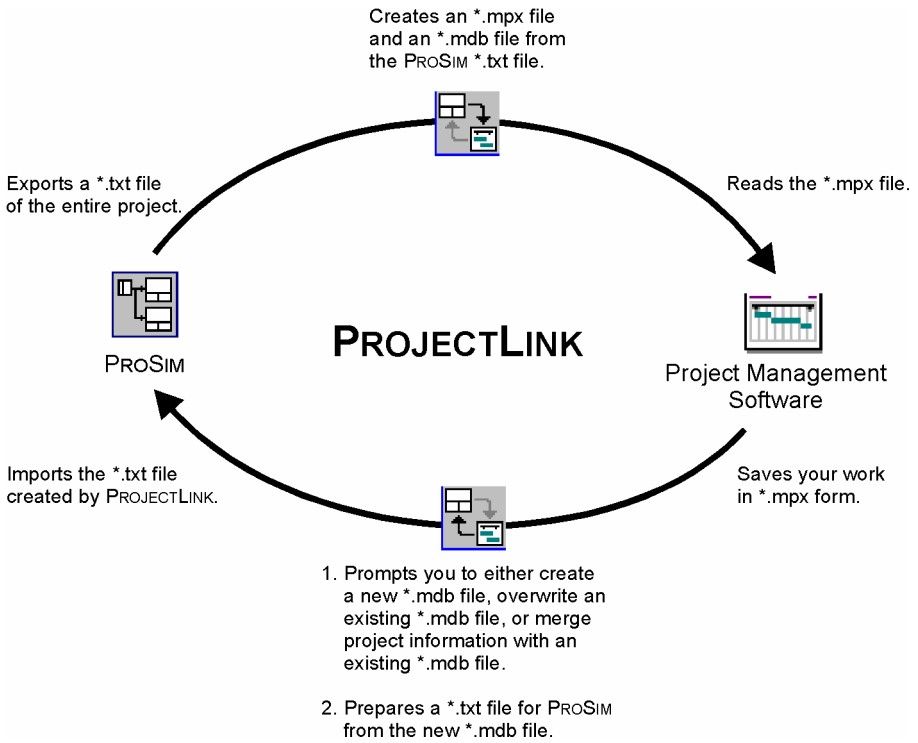
시뮬레이션 모델 exporting 은 시뮬레이션 모델 생성과 비슷한 뜻이다. PROSIM 이 시뮬레이션 모델을 export 혹은 생성할 때, export 되는 다이아그램의 엘리먼트에 관련된 시뮬레이션

정보를 만들고 지정된 다이아그램을 위한 명령어 문장을 생성하는 WITNESS command file(*.wcl)을 생성한다. 여러분은 *.wcl file 을 문장 편집기에서 open 할 수 있다.

4.3.11 프로세스 모델에의 프로젝트 플래닝

WITNESS 의 시뮬레이션이 여러분의 프로세스 모델의 동적인 분석을 환경을 지원하는 동안, 여러분은 또한 여러분의 모델을 적용시켜 보기를 원할 것이다. 여러분이 수행 일정편성을 시작하고자 할 때 여러분은 KBSI 의 PROJECTLINK™를 이용하여 여러분의 프로세스 모델을 프로젝트 관리 프로그램으로 쉽게 옮길 수 있다.

PROJECTLINK 는 KBSI 의 프로세스 모델링 툴인 PROSIM 과 프로젝트 관리 소프트웨어의 선두주자인 Microsoft® Project™간의 양방향 교환 프로그램이다. PROJECTLINK 는 여러분의 프로세스 모델을 일정표 형식으로 변환토록 지원하고 이러한 일정으로부터 프로세스 모델을 만들 수 있도록 지원한다. PROJECTLINK 에 의하여 변환되는 모든 정보는 분리된 Access database(*.mdb) file 에 저장되는데, 여러분이 새로운 모델을 작성할 것인지 기존의 모델에 겹쳐 쓸 것인지를 선택할 수 있도록 한다. 프로젝트 관리 툴에서 PROSIM 으로 import 할 때, 여러분은 기존의 프로세스 모델에 들어오는 프로젝트 정보를 합치는데 필요한 추가적인 옵션을 가진다. ProjectLink 는 모든 MPX 호환 프로젝트 관리 소프트웨어를 지원한다.



■ PROSIM에서 ProjectLink로의 전환

여러분이 PROSIM 텍스트파일을 export 하고 ProjectLink 를 연결시키면, 활성화된 ProjectLink 는 두개의 파일 - Project 가 읽을 수 있는 text 파일과 PROSIM project 파일의 모든 정보가 저장되어있는 데이터베이스 파일 - 을 생성한다. 이 데이터베이스는 원래 PROSIM 모델에 Project 에서 변경된 내용을 병합할 때 사용될 수 있다.
ProjectLink 는 PROSIM text(*.txt)파일로부터 Project text(*.txt)파일을 생성한다. 모든 정보는 MS-Access 데이터베이스 파일에 포함된 생성된 모델과 연관되어 있다. 당신은 PROSIM 에서 Project 로 이동 시킬 때 마다 새로운 데이터베이스를 생성하거나 기존 데이터베이스를 엎어 쓴다.

PROSIM 프로세스모델에서 Project task 스케줄로 변환하는데 있어서, PROSIM 엘리먼트 형식은 그들과 연관되어 있는 Project 엘리먼트 형식으로 변환된다. 마찬가지로 링크와 여러 junction 형식에 의해 모델링된 PROSIM 프로세스 로직은 연관된 Project task relationship 으로 변환된다. 아래 단락에서는 PROSIM 과 Project 엘리먼트들 사이의 차이점을 상세하게 설명하고 있다.

PROSIM 에서 Projectlink 로 엘리먼트 매핑

엘리먼트 매핑을 위한 기본적인 정보는 아래 테이블에 요약되어 있다.

<table>
<tr><th colspan="2">PROSIM Element</th><th>Project Element</th></tr>
<tr><td colspan="2">Process(UOB)</td><td>Task</td></tr>
<tr><td colspan="2">Process description</td><td>Task notes</td></tr>
<tr><td colspan="2">Entity Process Time</td><td>Task duration</td></tr>
<tr><td colspan="2">Resource rule</td><td>Resource allocation</td></tr>
<tr><td rowspan="3">Object types</td><td>Location</td><td rowspan="3">Resource</td></tr>
<tr><td>Resource</td></tr>
<tr><td>Transport</td></tr>
<tr><td colspan="2">Link between processes</td><td>FS task relationship</td></tr>
</table>

프로세스의 매핑

우선, PROSIM 의 프로세스(UOB) 는 Project task 가 된다. PROSIM 다이어그램 계층구조는 PROSIM 에서 Project 로 이동할 때 유지된다. 하나의 모 프로세스의 분해로 나타나는 자 프로세스는 그들의 가장 상위단계에 있는 톱니모양의 summary task 아래의 subtask 가 된다.

프로세스 설명은 task note 로 매핑된다.

PROSIM	Project
Process	Project Task
Description (각 process 의 description)	Task Note
Process Time(기본 단위 : 단위시간)	Duration(기본 단위 : Day)

오브젝트

Location, resource, transport 는 Project 에서 resource 가 된다. PROSIM 에 프로세스와 연관되어 있다면, 그 object 는 Project 의 task 와 연관될 것이다. 그렇지 안다면, 이것은 쉽게 수정하거나 할당할 수 있도록 다양한 resource view 로 나타날 것이다. Project 내에 resource 의 총 수량은 PROSIM 프로젝트에서의 그 수와 일치한다.
PROSIM 에서 resource 와 location 의 총 수량을 정의할 수 있으나, transport 는 그렇게 할 수 없다. 기본 사양 중 하나의 transport 는 Project resource 로 변경된다.

추가적으로 각각의 task 에 할당된 resource 는 각 프로세스에 할당한 시뮬레이션 정보를 반영할 것이다. 예를 들어 resource rule 에 두개의 주문서 처리자와 하나의 컴퓨터터미널이 하나의 프로세스를 완벽하게 수행하는데 필요하다고 정의 되었다면, 그것과 일치하는 task 는 그것에 할당될 두개의 주문서 처리자와 하나의 컴퓨터터미널을 가질 것이다.

PROSIM	Project
Entity	Project_Entity
Location	Resource
Resource	
Transport	
Queue	-
Logical	-

링크

PROSIM 다이어그램에서 프로세스를 연결하는 precedence link 는 Project 에서 task relationship 으로 나타날 것이다. 이 link 는 연결된 두 task 사이에 FS(Finish-to-start) relationship 으로 나타날 것이다. 그래서 Process/Task A 는 Process/Task B 가 시작하기 전에 끝나야 한다.

PROSIM	Project
Precedence Link	FS(Finish to Start)
Object Flow Link	
Relational Link	

PROSIM 에서 Project 로 논리적인 흐름의 매핑

FS : Finish-to-Start

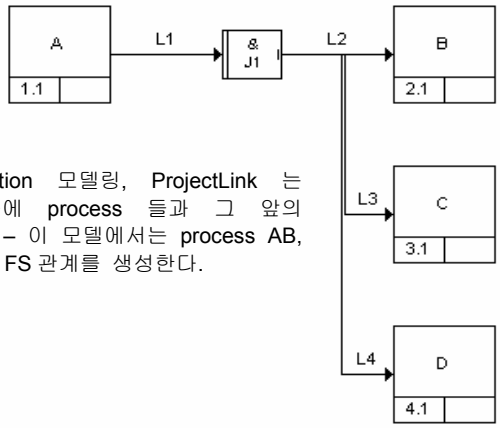
SS : Start-to-Start

FF : Finish-to-Finish

Junction 의 매핑

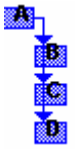
PROSIM Junction		Project Task Relationship(s)
Asynchronous AND; OR; XOR	Fan-Out Fan-In	FS task relationship 은 junction 으로 나뉘어진 process 들에 해당된다.
Synchronous AND	Fan-Out	FS task relationship 은 junction 으로 나뉘어진 process 들에 해당된다. SS task relationship 은 junction 에 따르는 process 들과 일치한다.
	Fan-In	FS task relationship 은 junction 으로 나뉘어진 process 들에 해당된다. FF task relationship 은 junction 에 선행되는 process 들과 일치한다.

Asynchronous fan-out AND junction

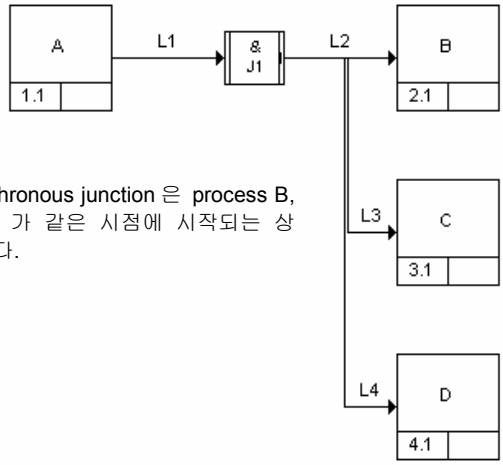


Fan-out junction 모델링, ProjectLink 는 junction 다음에 process 들과 그 앞의 process 간에 – 이 모델에서는 process AB, AC, AD 사이 - FS 관계를 생성한다.

ProjectLink



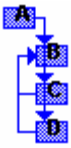
Synchronous fan-out AND junction



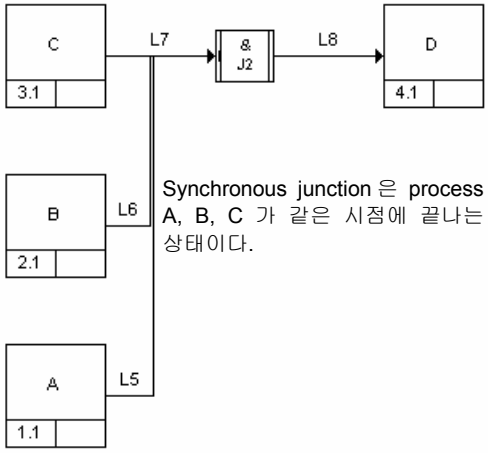
Synchronous junction 은 process B, C, D 가 같은 시점에 시작되는 상태이다.

Project 는 따라서 task 들 사이에 SS(Start-to-start)관계를 생성한다.

ProjectLink



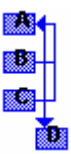
Synchronous fan-in AND junction



Synchronous junction 은 process A, B, C 가 같은 시점에 끝나는 상태이다.

Project 는 따라서 task 들 사이에 FF(Start-to-start)관계를 생성한다.

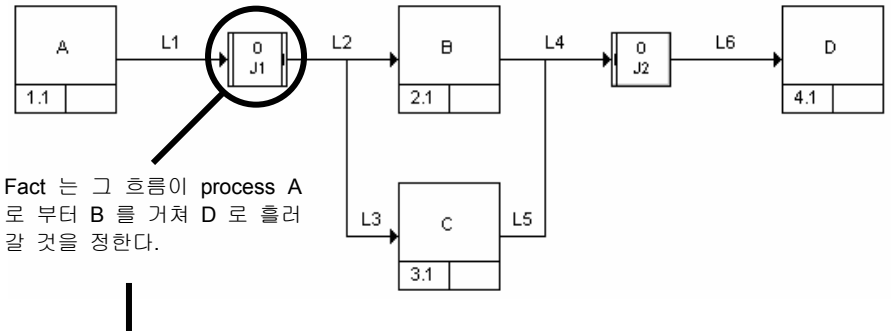
ProjectLink



OR and XOR Junctions

OR 나 XOR junction 의 경우에 PROSIM 에서는 하나의 다이어그램에 각각의 가능한 경로의 흐름을 모두 표현 가능하나, Project 는 하나의 프로젝트 내에 동시에 표현하는 활동의 경로에 제약을 둔다. AND junction 은 동시에 발생하는 여러 경우를 모두 포함 하므로, Project 에서 그 관계를 task 들간의 다중 순차관계로 표현이 가능하다.
그러므로 OR 나 XOR junction 은 여러 가능성과 유일한 경로흐름을 나타낸다. 이제 Project 는 여러분이 지정한 OR 나 XOR junction 의 유일한 경로를 표현할 것이다.

J1 junction 에 다음과 같은 fact 를 추가함으로써 그 경로를 지정한다.



Choose L2 to J2
1 2 3 4

1, 3 : 이 fact 문장은 첫 번째와 세 번째 위치에서 placeholder 로 사용된다.
2 : 두 번째 위치에는 원하는 경로의 처음에 위치한 link 의 위치를 지정한다.
4 : 네 번째 위치에는 그 다음에 오는 fan-in junction 의 label 을 정한다.

OR 그리고 XOR junction 에서 그 경로를 정하는 방법 :

하나의 OR 혹은 XOR junction(**J1**)을 통해 여러 개의 UOB(B, C)로 그 활동이 분할 될 경우 (위 그림 참조) **J1** junction 에서는 특정 경로를 지정할 수 있으며, 그 지정 방법은 아래와 같다. 여기서는 앞에서 정의한 자재 구매 절차 모델을 사용한다.

① **J1** junction 을 오른쪽 클릭한다.
② Pop-up 메뉴가 나오면 *Edit Elaboration* 을 선택한다.
③ Fact 목록 상자에서 Add 버튼을 클릭한다.
④ Name 란에 **PathAtJ1** 을 입력하고 Create New 를 클릭한다.

이제 경로를 지정할 준비가 되었으며, 다음 단계로 직접적인 경로를 지정한다.

⑤ **PathAtJ1** 이 선택된 상태에서 *Edit* 를 클릭한다.
⑥ Relation Description 란에 **Choose L2 to J2** 을 입력하고 **OK** 를 클릭한다.
⑦ 다이어그램 창으로 돌아가기 위해 **OK** 를 클릭한다.

이제 모든 경로가 지정되었으며, Project 로 이 정보를 넘기면 **기존의 공급자로부터 주문한다** 와 **자재를 납품 받는다** 사이에 FS relationship 이 설정될 것이다.

지금까지의 내용을 요약하면 아래와 같다.

OR/XOR junction(Right-Click) -> *Edit Elaboration*...(Click) -> Fact section *Add*(Click) -> **PathAtJ1**(Type) -> *Create New* -> |*OK*| -> *Edit*(Click) -> **Choose L2 to J2**(Type) -> |*OK*|

■ PROSIM에서 프로젝트의 Exporting

PROSIM 에서 프로젝트.txt 파일을 export 하기 위해서는 다음과 같은 절차를 수행한다.

① File 메뉴 -> *Export* 선택
② Export Text Option dialog 에서 **|Entire|** 를 선택한다.
③ Save As dialog 에서 원하는 이름(*.txt)을 입력하고 **OK** 를 Click 한다. PROJECTLINK 로 database(*.mdb)와 Project text(*.mpx)를 생성할 때 이 들 파일명을 *.txt 이름과 같게 지정해 준다.

■ PROSIM에서 PROJECTLINK의 Launching

PROSIM 에서 Projectlink 를 launch 하기 위해서는 다음과 같은 절차를 수행한다.

① File 메뉴에서 *Launch Program* 을 선택한다.
② PROJECTLINK Bridge 실행유무와 목록상에 *.txt 파일을 변환할 것인지 확인 및 실행여부를 묻는 Confirm Program Launch dialog 가 나타난다. 이 dialog 는 두 가지 선택사항을 제공한다.

YES : PROJECTLINK 를 실행한다. 활성화된 project 를 저장하지 않았다면, PROSIM 은 PROSIM 을 종료하기 전에 변경사항을 저장할 것인지를 묻고 PROJECTLINK 를 실행할 것이다.

NO : 실행을 취소하고 PROSIM 의 활성 diagram 으로 돌아온다.

③ Export 된 *.txt 파일과 이름이 같은 *.mdb 파일이 이미 있다면 Project Translator dialog 가 나타날 것이다. 다음 두 가지 선택사항이 있다.

|Create New Database| : 다른 이름의 새로운 database 를 생성한다. Enter Database Name dialog 가 나타나면, database 의 새로운 이름을 입력하고 **OK** 를 클릭한다.

|Overwrite Existing Database| : 기존 database 에 엎어 쓰기를 한다.

④ PROJECTLINK 는 프로젝트내에 여러 시나리오가 있는지 검사할 것이다.

PROSIM 프로젝트에 시나리오가 하나만 발견되었다면, □단계로 계속될 것이다.

PROSIM 프로젝트에 시나리오가 하나이상 발견되었다면, Project 로 모델링하고자 하는 시나리오의 선택을 묻는 Select Scenario dialog 가 나타날 것이다.

이 목록 중에 하나의 시나리오를 선택하고, **|OK|**를 클릭한다. 그 변환 작업은 5 단계로 계속된다.

⑤ ④에서 선택한 시나리오에서, Project 는 다중 decomposition 을 검색할 것이다.

PROJECTLINK 가 시나리오 안에서 다중 decomposition 을 찾지 못했다면, PROJECTLINK 는 Project 를 실행하고 *.mpx 로 부터 diagram 정보를 읽어올 것이다.

PROJECTLINK 가 시나리오 안에서 다중 decomposition 을 찾았다면, 다중 decomposition 을 포함하는 각 프로세스를 나타내고 task schedule 에 포함 시킬 decomposition 의 선택을 묻는 Select Decomposition dialog 가 나타난다. 그들 항목 중에 하나의 decomposition 을 선택하고, **OK** 를 클릭한다.

이 변환작업이 완료되면 Project 가 실행되고, 새롭게 들어온 task schedule 을 기본 Gantt Chart 형식으로 보여준다.

■ 새로운 Task Schedule 보기

PROSIM 디이아그램은 아래 변환규칙에 따라 Project 에 표현될 것이다.

① 프로세스는 원본 다이아그램의 계층적 구조를 지니고 task 로 표현된다. parent process 는 summary task 로 child process 는 subtask 로 보여진다.
② Precedence link 는 FS(Finish-to-start)task relationship 으로 변환될 것이다.
③ Process 의 Sim Info.에서 입력한 entity process time 은 원본 process 와 연관된 task 에 duration time 으로 표현될 것이다. 만약 entity process time 을 입력하지 않았다면 기본 duration (1day)가 적용될 것이다.
④ Process description 은 Task ID 번호 옆에 page symbol.에 의해서 알 수 있는 task note 로 표현될 것이다.
⑤ Process 와 연관된 location, resource, transport 는 지정된 task 에 resource 로 할당될 것이다.

4.4 IDEF3 모델링 지침

4.4.1 IDEF3 모델의 검토

■ IDEF3 모델의 판독

① 모델의 범위를 이해하기 위하여 배경, 목적, 관점을 연구한다.
② 프로세스 흐름 다이아그램은 왼쪽에서 오른쪽으로 맨 왼쪽에 있는 프로세스에서부터 시작한다. 이러한 다이아그램 판독 방법을 “Workthrough 의 수행”이라고 한다.
③ 각 엘리먼트의 설명과 상세화(elaboration)를 검토한다.

■ IDEF3 모델 작성의 힌트

① XOR fan-out 접속 다음에 AND fan-in 접속이 와서는 안 된다.
② 다이아그램의 맨 왼쪽에 여러 개의 프로세스가 있으면 안 된다. 이들은 해석에 혼란을 주므로, 프로세스의 흐름을 명쾌하게 하기 위하여 여러 개의 프로세스 전에 fan-out 접속을 쓴다.
③ 명료한 다이아그램을 위해 한 곳에서 분기되는 fan-out 접속을 피한다.
④ Fan-out 접속이 fan-in 접속으로 바로 이어지는 것은 다이아그램에서 생략된 프로세스가 있다는 것을 나타낸다.

4.4.2 프로세스 설계의 8가지 원칙

■ 원칙 1. 프로세스 설계는 디자인 활동이다.

① 당연히 우선 창조적 이여야 한다. 여러분은 최선의 수행방법을 적용하고, 찾고, 따라 하여야 한다.
② 수행에 있어서는 우선 반복적이어야 한다.
③ 비용/수행성/이익/위험회피 등이 요구되며, 특히 시뮬레이션과 ABC 분석에 있어서 그렇다.
④ 하나의 해결책만을 제시해서는 안 된다.
⑤ 세부적인 사항들이 작성될 때 까지는 완료되지 않아야 한다.

■ 원칙 2. 프로세스 디자인 전문기술은 여러 가지 기술의 조합과 이들을 어떻게 적절히 구사하느냐 하는 것으로 이루어진다.

① 프로세스 디자인은 당신이 제약사항을 조정하고 만족을 성취하는 것을 도와준다.
② 요구사항과 디자인 목표는 다르다.
③ 프로세스 모델은 플로우 차트가 아니다.
④ 유행에 따라서 발전할 필요는 없다.
⑤ 모델링은 트레이드오프 분석을 위한 주제로서 여러 가지 대안을 제시해야 한다.

■ 원칙 3. “Object design”은 프로세스 디자인에서 중심적인 역할을 제시한다.

① 입력과 출력들
② 자원
③ 중간에 존재하는 개체
④ 인터페이스 개체
⑤ 개체 상태 변이
⑥ 개체의 “품질” 측정

■ 원칙 4. 프로세스는 실행되는 환경에서 특정 자원의 분배가 될 수 있도록 지원하는 하나의 레벨에서 표현되어야 한다.

① 하위 프로세스로 분해한다.
② 프로세스 디자인의 종단 조건을 사용한다.
③ 기술의 변화나 수용능력의 변화등과 같이 증가되는 프로세스

■ 원칙 5. 물리적 그리고 논리적 입력/출력들은 계속적으로 관리되어야 한다(보호법칙).

① 각 프로세스 단위의 특정 입력/출력 그리고 프로세스 흐름에서 어떤 프로세스의 위치에서 요구되는 출력 그리고 일치되는 입력 가능한 것들.
② 분해를 계속한다.
③ 개체 설계와 높은 의존을 보이는 것들.

■ 원칙 6. 당신이 검토해야 할 오류는 항상 있는 법이다.

① 모델 정의의 오류
② 모델 분석의 오류
③ 하위 프로세스 디자인 검출의 오류
④ 하위 프로세스 디자인 작업의 오류
⑤ 오류에 관련된 인내.

■ 원칙 7. 프로세스 디자인은 제품 관리에 의한 프로세스 단계들의 디자인을 포함한다.

① 쓰레기나 못쓰게 된 것들
② 정의
③ 수집
④ 배치

■ 원칙 8. 프로세스 디자인은 협력의 수행과 관리를 위한 프로세스 스텝의 디자인을 포함한다.

① 동시공학 적 프로세스
② 자원 배분
③ 작업 아이템 우선순위
④ 상태, 수행, 추적, 자료수집
⑤ 의사소통 관리

■ 결론

IDEF3 은 시스템 작동 방식에 관하여 유력한 전문가가 알려 준 조정되지 않은 설명을 파악하는 매체를 제공하도록 디자인되었다. 그러나 이것은 반대로 객관적 사실을 조직.전달하기

위한 풍부한 메커니즘을 제공하지는 못한다. 많은 시나리오와 관점을 하나의 다이어그램으로 결합하는 데서, 지나치게 복잡한 IDEF3 다이어그램이 나올 수 있다. 그러나 다이어그램은 다음과 같은 간단한 발견에 의해 단순화 될 수 있다. 시나리오와 관련된 모든 장면으로부터 어떤 UOB 가 보이지 않을 경우, 그것은 분리된 UOB 일 것이다. 또한 수집된 데이터의 '간격을 메우려는' 경향을 조심해야 한다. 설명 파악 방법론으로서 IDEF3 은 '부분적인' 심지어는 모순된 설명에 관하여 관대하도록 디자인 되었다. 그것들을 지원하는 정보 시스템과 조직 분석 가운데, 특별한 문제의 근거가 되는 것은 바로 이러한 모순된 혹은 불완전한 부분이다. 생략에 의해 혹은 의도적으로 이들 영역을 설명에서 제외 한다면, 문제는 무시될 수 있다. 그러나 이는 '어떤 것도 가리키지 않는 이들 모델은 잘못되었다' 는 비난을 유발하게 될 것이다. 시스템의 문제점을 발견하지 못하는 것은, IDEF3 모델화 방법 자체의 부족에서 나오는 것이 아니고 모델 작업자의 개방적 관대 함에서 나오는 것이다.

5. IDEF1X 정보/데이터 모델링 방법(Information/Data Modeling Method)

5.1 IDEF1X 개요

정보/데이터모델링의 중요한 목적은 특정 환경이나 시스템 내에서 정보를 구분하고 문서화하기 위해서이다. 이러한 두 가지 목적은 유사하면서도 중요한 점에서 매우 다른데 그 하나는 실세계의 시스템 내에서 정보의 의미와 구조를 표현하는 정보모델을 제공하기 위해서 사용되는데 비하여 다른 하나는 정보시스템 구축에 있어서 이러한 정보를 효과적으로 저장하고 활용하기 위한 데이터베이스 디자인을 위한 데이터모델을 제공하기 위하여여 사용된다.

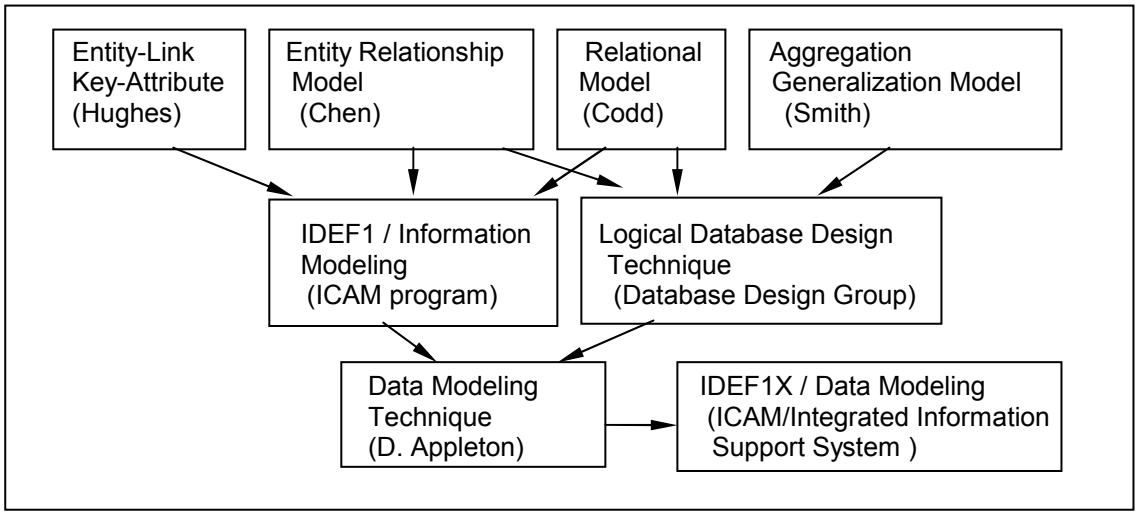


그림 5-1 : IDEF1/IDEF1X 의 기원

IDEF1X 정보/데이터모델링 방법의 개발은 그림 5-1 과 같이 IDEF1 정보(Information)모델링 방법과 데이터베이스 디자인 그룹의 LDDT(Logical Database Design Technique)로부터 많은 영향을 받았다. IDEF1 정보모델링 방법은 원래 ICAM 프로그램 안에서 Codd(Dr. E.F. Ted Codd)에 의해 개발된 관계형 이론과 Chen(Dr. P. P. S. Peter Chen)의 엔티티-관계 모델링 개념을 바탕으로 Hughes Aircraft 사와 D. Appleton 사(DACOM)의 주도하에 개발되었다. 또한 ER 모델, Relational 모델, 집합화/일반화(Aggregation/Generalization) 모델은 '데이터베이스 디자인 그룹'의 LDDT(Logical Database Modeling Technique) 개발을 유도하였는데 LDDT 는 뒤에 D. Appleton 사에 의해 Data modeling Technique 라는 상업용 제품으로 발표되었다.
1983 년에 미 공군은 ICAM 프로그램 내에서 IISS(Integrated Information Support System)프로젝트를 시작하였는데, 이 프로젝트의 목적은 다양한 컴퓨터 하드웨어와 소프트웨어 네트웍하에서 논리적 물리적 통합이 가능한 정보기술의 구현이었다. IISS 는 개념적 스키마

(Conceptual Schema)라고 불리우는 하나의 의미론적적 정의에 의한 정보자원의 포착과 관리 그리고 이용을 통합하는데 초점을 맞추게 되었는데 D. 애플턴을 비롯한 General Electric, SDRC(Systems Design Research Corporation), CDC(Control Data Corporation) 등의 프로젝트 참여기업들은 IDEF1 의 개념적 스키마의 확장과 D. 애플턴의 Data Modeling Technique 을 결합한 새로운 모델링 방법을 그 해결책으로 미 공군에 제안하였으며 이러한 제안이 받아들여져 미 공군은 IDEF1X(IDEF1 Extended)를 정보/데이터 모델링을 위한 표준 방법으로 채택하였으며 이를 일반에게 공개하였다. IDEF1X 는 1986 년 미 국방부에 의해 정보/데이터 모델링을 위한 표준 방법(Department of Defense 8020.1-M)으로 채택되었고 1993 년 12 월 NIST(National Institute of Standards and Technology)는 미연방 정보처리 표준(Federal Information Processing Standard Publication 184-Integrated Definition for Data Modeling)으로 IDEF1X 를 채택하였다.

(우리는 이장에서 IDEF1X 를 IDEF1 정보모델링 방법을 통합한 확장된 개념의 정보/데이터 모델링 방법으로 사용하며 IDEF1X 의 용어를 기준으로 설명한다. 한다. 특별히 IDEF1 정보모델링 표현 방법에 관심을 가지고 있는 독자는 이 방법에 관한 구체적 사항에 대하여 IDEF1 Information Modeling Method / Technical Report Developed Under Air Force Contract #AF33615-80-5155 [Richard J. Mayor, Knowledge Based Systems, Inc. 1994]를 참고하기 바란다.)

5.1.1 IDEF1X 정보모델링 방법

IDEF1X 정보모델링 방법은 시스템 분석에 있어서 '요구사항 정의'에 대한 효과적 분석 및 커뮤니케이션 메커니즘이 되도록 디자인된 방법이다. IDEF1X 정보모델링 방법은 분석대상 시스템에서 어떤 정보가 존재하는지 혹은 기업이 어떤 정보를 관리해야 하는지에 관한 사항을 파악하는 방법론이다. 정보모델링 방법은 일반적으로 조직 내부에 현재 어떤 정보가 관리되는지 확인하고, 요구사항(needs)을 분석하는 동안에 판별된 문제들이 어떠한 정보의 적절한 관리 부족에 의해 야기되는지를 확인하며, 'TO-BE' 모델 속에서 어떤 정보가 관리되어야하는 지를 규정하기 위해서 사용되는데 IDEF1X 정보모델은 다음과 같은 것들에 초점을 맞춘다.

① 조직에 의해 다루어지고, 저장되고, 수집되어지는 정보
② 그러한 정보에 대한 조직내의 규칙(제약조건)
③ 정보에서 반영된 조직내의 논리적 관계성
④ 정보시스템 설계를 위한 기초
⑤ 문제식별
⑥ 요구조건의 정의

IDEF1X 정보모델링 방법을 개발한 최초의 의도는 기업 내에 존재하거나 혹은 관리되어야 하는 정보를 파악하는 것이었다. 정보모델일의 관점은 자동화, 정보시스템화 시켜야 할 정보 요소 뿐 아니라 기업 내부에 존재하는 모든 대상(예를 들면 사람, 서류함, 전화, 가구 등)을 포함 한다. IDEF1X 정보모델링 방법은 '데이터베이스 설계 방법'이 되지 않도록 특별히 디자인 되었는데 데이터베이스 설계를 위한 방법은 데이터모델링 방법에서 지원된다. 정보모델링 방법은 데이터베이스 설계보다는 정보자원의 관리에 대한 요구사항 분석을 지원하는 방법으로서 다음과 같은 사항을 확인하는데 적절한 분석 방법론이다.

① 기업이 수집, 저장, 관리하는 정보는 무엇인가
② 정보 관리를 통제하는 규칙은 무엇인가
③ 정보에 반영되는 기업 내부의 논리적 관계는 무엇인가
④ 정보관리의 부실로 발생되는 문제는 무엇인가

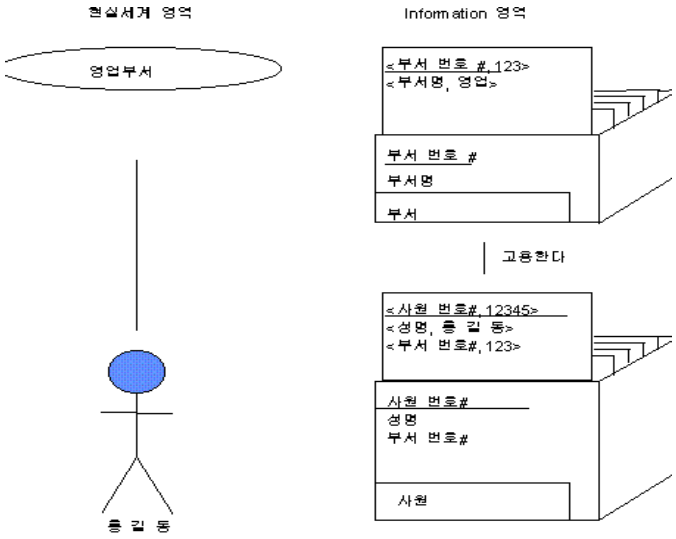


그림 5-2 : 정보모델의 관점

정보분석의 결과는 기업 내부의 전략, 전술적계획에 이용 할 수 있으며 정보자산을 이용하여 상대적으로 경쟁적 이익을 확보하도록 지원한다. 기업은 이러한 정보의 파악을 바탕으로 유용한 정보를 효과적으로 이용하는데 필요한 자동화된 정보시스템의 구축을, 설계, 시행할 수 있는 것이다.

그런데 정보모델은 이러한 디자인을 하는데 있어서 기초적 자료를 제공한다. 따라서 정보모델은 데이터베이스 디자인의 전 단계로서 훌륭한 정보관리 정책을 수립하는데 필요한 지

식과 통찰력을 관리자에게 제공하는 것이다. IDEF1X 정보모델은 간단한 그래픽 약속을 활용하여 모델 작업자가 다음사항을 구분하는데 필요한 일련의 규칙을 구체적으로 제시한다.

① 현실세계의 객체 (Real-world objects)
현재 실세계에서 다루어지는 정보가 무엇인지를 구분한다. IDEF1X 정보모델은 자동화된 시스템 요소들과 사람, 서류 철, 전화와 같은 자동화되지 않은 요소들을 포함한다.
② 현실세계의 객체 사이에 유지되는 물리적 혹은 추상적 연관 관계
정보를 다루기 위한 제약조건을 정의한다. 정보모델에서 사용된 간단한 그래픽적인 부호규약(convention)은 모델작업자가 실세계 오브젝트간에 유지되는 물리적, 추상적 관련과 실세계 오브젝트를 구별할 수 있도록 해준다.
③ 현실세계의 객체에 관하여 관리되는 정보
현실세계 정보관리의 문제점을 정의한다. IDEF1X 정보모델은 그러한 정보를 다루고, 적용하고, 획득하기 위한 데이터 구조의 원리를 포착한다.
④ 현실세계에 관한 정보를 표현하는데 필요한 데이터의 구조(정보를 효율적으로 수집하고, 응용, 관리하기 위하여)
TO-BE 모델 구현에서 다루어질 정보를 분류한다.

IDEF1X 정보모델은 현존하는 정보, 혹은 기업이 수집, 관리, 통제해야 할 정보를 제시하도록 디자인 되었다. IDEF1X 정보모델링 방법의 규칙은 현실적 대상이나 현실적 대상 사이의 물리적 추상적 관계를 모델링하지 않도록 함과 동시에 모델 작업자의 관심을 데이터베이스 디자인(이것은 일반적으로 소프트웨어 엔지니어의 영역으로 간주된다)에서 멀어지게 한다. 정보수집에 있어서 모델 작업자에게는 두 가지 영역을 구분하는 것이 중요하다.

① 첫째는 조직 내부의 사람이 인지하는 현실 세계이다. 이 영역에서는 물리적. 추상적 객체(물리적 그리고 추상적 객체, 즉 사람, 장소, 물건, 아이디어 등)와 그 대상의 특성, 그리고 그들 간에 연관된 관계성 등을 포함한다.
② 둘째는 정보영역이다. 정보영역에서는 현실 세계에서 발견되는 이들 대상의 정보 이미지를 포함한다. 정보 이미지는 현실세계의 객체가 아니라 현실세계에 관하여 수집, 저장, 관리되는 정보일 따름이다. IDEF1X 정보모델링 방법은 정보 이미지를 발견하여 조직, 문서화하는 것을 돕도록 디자인 되었다.

5.1.2 IDEF1X 데이터모델링 방법

IDEF1X 데이터모델링 방법은 기본적으로 관계형 데이터베이스를 설계하기 위한 방법이다. 이 방법은 데이터베이스 시스템의 개념적인 스키마(scheme) 개발을 위해 설계된 구문이다.

이러한 이유로 인해, IDEF1X 데이터모델링 방법은 정보의 요구조건이 파악되고, 관계형 데이터베이스를 구현하기 위한 결정이 이루어진 이후에 논리적 데이터베이스를 설계를 위해서 가장 유용하다. 따라서, 데이터모델링 방법의 관점은 관계형 데이터베이스 내에서의 실제 엘리먼트에 집중되어진다

■ 왜 IDEF1x 데이터모델을 개발하는가 ?
① 정보자원 관리의 필요성 및 요구조건을 결정한다.
② 수집되고, 저장되고, 처리되는 정보가 무엇인지를 명시한다.
③ 정보관리를 지배하는 규칙(제약조건)을 명시한다.
④ 연결된 IDEF0 모델의 개념(concept)에 대한 타당성을 확인한다.
⑤ 경쟁적 우위를 득하기 위하여 정보를 활용.

IDEF1X 데이터모델링 방법은 '시스템 설계' 활동을 수행하기 위한 방법으로 작성된 것이다. 디자인 방법이기 때문에 IDEF1X 데이터 모델링 방법은 종종 IDEF1X 정보모델링 방법의 대안으로 제시되기도 하지만 특별히 'AS-IS'분석 방법으로는 적합하지 않다. IDEF1X 데이터 모델링 방법은 정보의 요구가 알려지고, 관계형 데이터베이스를 적용하기로 결정한 후에 논리적 데이터베이스 디자인에 대하여 적용 될 때 아주 유용하다. 따라서 정보시스템의 IDEF1X 데이터모델링 관점은 관계형 데이터베이스 속에서 실행되는 논리적 데이터 구조의 디자인에 초점을 맞춘다. 만일 목표 시스템이 관계형 데이터베이스 시스템이 아니라면 IDEF1X 데이터모델링 방법은 최상의 방법이 아니다.

일단 요구되는 정보의 분석이 완료되면 그 정보를 관리하는 방식에 관한 의사결정이 더 효과적으로 될 수 있다. 한 가지 가능한 결정은, 이들 요구를 만족 시킬 자동화된 정보시스템을 구축하는 것인데 이러한 결정은 적절한 디자인 방법의 선택을 필요로 한다.
훌륭한 디자인 방법의 선택과 올바른 적용은 시스템 전체의 라이프사이클에 걸쳐 예정된 비용 안에서 강고한 고품질의 시스템을 구축할 수 있도록 지원할 것이다. 따라서 훌륭한 디자인 방법은 특정 기술과 연관된 최상의 응용경험을 바탕으로 구현해야 한다.

■ IDEF1X 데이터모델은 어디에 사용되나 ?
① 데이터베이스의 논리적 설계
② 응용프로그램의 논리적 설계
③ 데이터베이스 구현의 물리적 설계

선택된 기술이 관계형 데이터베이스(Relational Database)라면 IDEF1X 데이터모델링 방법은 논리적 데이터베이스의 설계와 개발에 효과적으로 적용될 수 있다. 선택된 기술이 객체 지향적 데이터베이스 패러다임이라면, IDEF4 객체지향적 설계 방법(뒤에 논의 된다)이 더 적

절할 것이다.

IDEF1X 데이터모델링 방법은 비 관계형 시스템에는 잘 맞지 않는다. 이에는 몇 가지 이유가 있다. IDEF1X 데이터모델링 방법은 모델 작업자가 하나의 실체를 다른 것과 구별하는 키 클래스를 지정하도록 요구한다. 반면 객체 지향적 시스템은, 하나의 객체를 다른 것과 분리시키는 키(key)를 요구하지 않는다.

예를 들어, 객체지향 시스템과는 반대로, IDEF1X 데이터모델링 방법은 모델작업자가 하나의 엔티티 인스턴스와 다른 엔티티 인스턴스를 구분하기 위하여 키 애트리뷰트를 명시할 것을 요구한다. 또한 IDEF1X 데이터 모델은 하나 이상의 애트리뷰트나 혹은 애트리뷰트 셑이 하나의 엔티티 인스턴스를 구별하기 위하여 동일하게 사용될 수 있는 경우에도 이들 중 하나만을 Primary 키로 설정하고 다른 모든 것들을 Alternate 키로 명시할 것을 요구한다. 또한 명확한 Foreign 키 라벨도 요구된다. IDEF1X 데이터모델링 프로젝트의 결과는 정보시스템 구축을 위한 '청사진'으로서 시스템 구축에 참여하는 프로그래머에 의해 활용된다. (여기에 사용된 용어는 다음절에서 상세하게 설명될 것이다.)

공장(Real World) Semantic Model Physical Data Stores(Tape..)

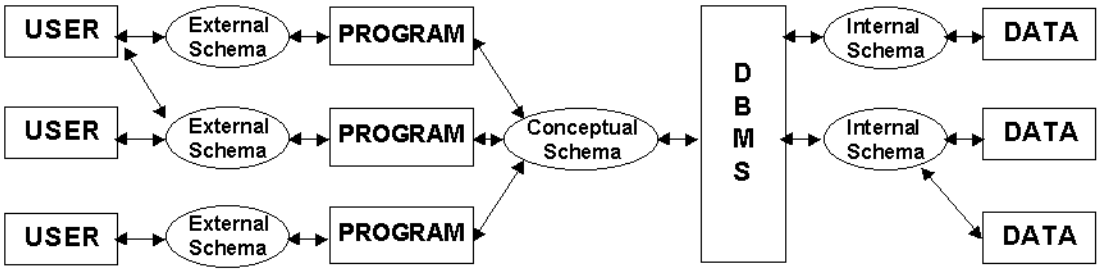


그림 5-3 : Three Schema Approach

5.1.3 IDEF1X의 예

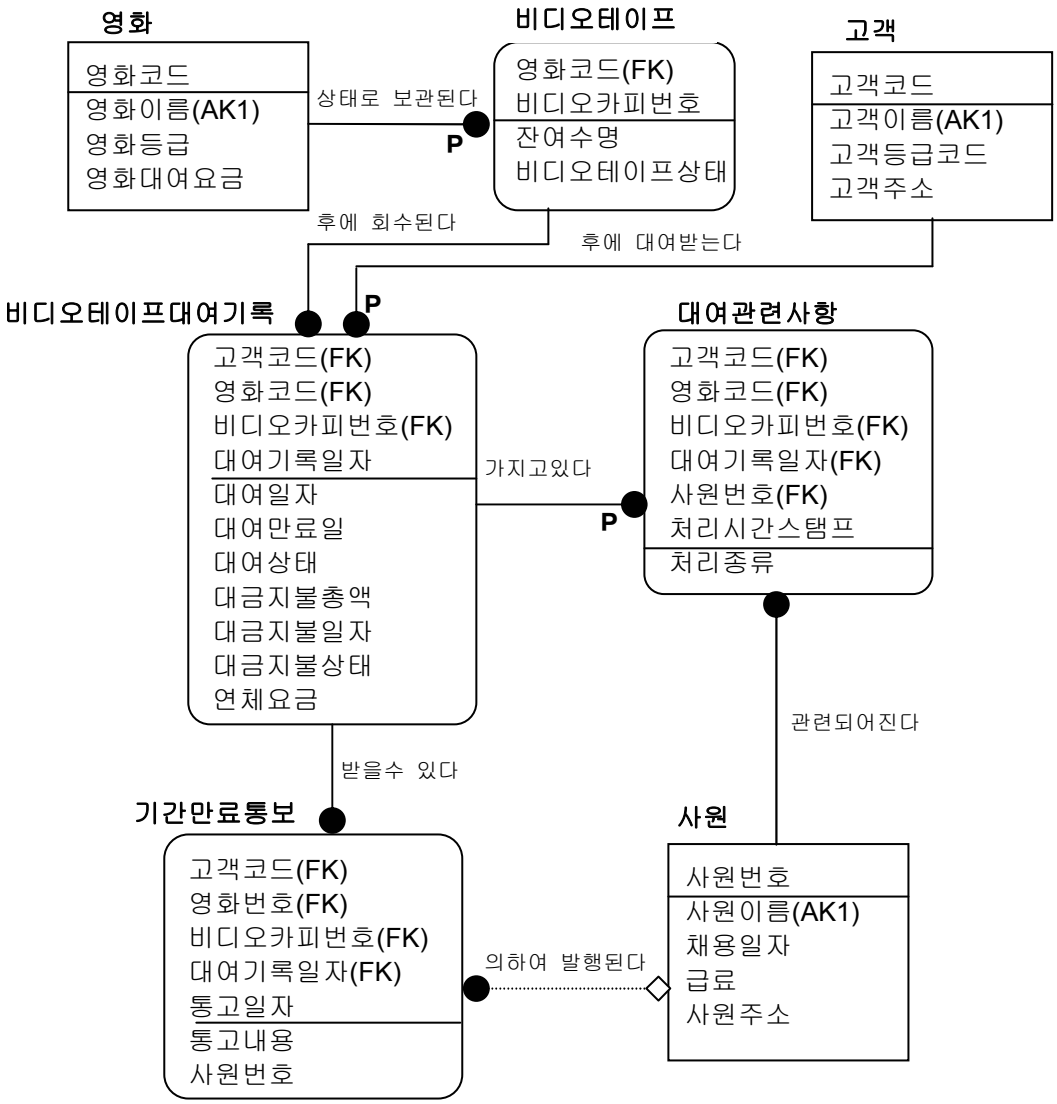


그림 5-4 : 정보모델의 예

그림 5-4 는 IDEF1X 를 이용하여 작성한 정보모델의 예이다. 이것은 일반적으로 우리가 쉽게 접할 수 있는 비디오테이프를 고객들에게 대여하는 비디오대여점의 업무에 관한 키 기반의 모델(Key based model)이다. 위의 그림에서 각 개체에 대한 정의는 다음과 같은 선언을 하고 있는데 우선 선언되어진 내용을 구성된 정보를 기준으로 살펴보면 다음과 같다.

① 하나의 영화는 한 개이상의 비디오테이프 상태로 보관된다. 각 영화에 관하여 이름,

등급, 대여요금이 기록된다. 각 비디오테이프에 관하여는 비디오테이프의 상태와 잔여 수명이 관리된다.

② 고객은 비디오테이프를 대여받는다. 비디오테이프 대여점에서는 각 고객에 대한 이름과 주소를 기록관리한다. 또한 각 고객에 대한 등급코드를 부여하는데 이러한 등급은 이전의 거래실적에 의한 것으로서 외상이나 신용카드 결제, 혹은 오직 현금거래만이 가능한지의 정도를 구분한다.

③ 비디오대여기록은 고객에 의해서 대여된 비디오테이프에 관한 사항을 기록한다. 같은 비디오테이프는 여러 번에 걸쳐서 많은 고객에게 대여된다. 각 비디오테이프대여기록은 대여일자, 대여만료일, 그리고 대여기간이 만료됐는가를 나타내는 대여상티를 포함한다.

④ 비디오대여점의 각 사원들이 수행한 여러 비디오테이프대여와 관련된 사항은 대여관련 사항을 통하여 기록된다. 여기서 수행한 업무의 종류는 처리종류로서 구분된다. 각 비디오테이프대여기록과는 최소한 한명의 사원이 관련된다. 또한 같은 사원이 동일 대여기록과 관련하여 하루에도 여러 번 관련될 수있기 때문에 처리시간스탬프로 이를 구별한다.

⑤ 고객의 대금지불사항은 비디오테이프대여기록에 등록된다. 연체요금이 가끔씩 발생하기도 한다.

⑥ 고객들에게 비디오테이프가 반환되어야 한다는 것을 알리기 위하여 종종 기간만료통보가 보내진다. 하나의 기간만료통보는 한사원에 의해서 발행되고 이를 발행한 사원은 그 결과를 관리한다.

⑦ 비디오테이프 대여점에서는 각각의 사원에 대한 채용일자, 급료, 주소를 기록한다. 고객이나, 사원, 영화에 관한 사항은 고유코드나 번호보다 이름으로 찾기도 한다.

위에 그려진 작은 모델은 비디오테이프 대여점에 관한 많은 정보를 우리에게 제공하고 있다. 우리는 여기서 이러한 업무를 위하여 데이터베이스가 어떤 내용을 포함하는가 뿐 아니라 업무가 어떻게 이루어지고 있는가에 대하여 알기 위하여 이들 업무 룰(business rule)에 관하여 알아보자.

① 영화에 대한 기록업무는 영화가 만들어지는 시점이 아닌 비디오테이프 형태로 복사되어 고객에게 대여할 수 있는 상태에서부터 시작된다.

② 모든 비디오테이프의 임대와 관련하여 최소한 한명의 사원이 관여된다.

③ 어떤 사원은 어떠한 비디오테이프대여기록과도 관련되지 않을 수도 있다. 또한 모든 비디오테이프대여기록에 관여할 수 도 있다.

④ 어떤 사원은 어떠한 기간만료통보와도 관련되지 않을 수도 있다. 또한 모든 기간만료통보도 처리할 수 있다.

⑤ 비디오테이프 대여점은 최소한 한 개 이상의 비디오테이프를 대여한 고객에 대한 사항만 기록관리한다.

⑥ 하나의 비디오테이프대여기록은 오직 하나의 비디오테이프에 대한 정보만을 포함한다. 만일 고객이 동시에 여러 개의 비디오테이프를 대여하고자 하는 경우에는 각각의 비디오테이프에 관한 기록이 필요하다.
⑦ 비디오대여점은 사원들에 의해서 대여되는 사항에 관한 기록은 별도로 가지고 있지 않다. 만일 그러한 상황이 발생하면 사원은 고객으로 분류되며 어떤 고객이 사원인지를 알 필요는 없다.
⑧ 이 모델은 하나의 비디오테이프 대여점만을 위한 것으로서 이 점포에서는 추가적인 대여점의 확장이나 그를 위한 데이터베이스에 관한 계획을 가지고 있지 않다.

그림 5-5 에서 모델은 여러가지 다른 종류의 도형과 개체들로 구성되어 있는데, 이들은 각각 엔티티, 애트리뷰트, 엔티티간의 관계등을 표현하고 있으며 이들에 관한 자세한 사항은 다음절에서 논하도록 한다.

5.2 IDEF1X 모델링 표현 방법

5.2.1 엔티티(Entity)

■ 엔티티란 무엇인가

엔티티는 모델의 범위 내에서 사람, 장소, 사물, 이벤트, 개념 등과 같은 대상에 대하여 유일한 식별자(이름, identifier)로 구분 가능한 관련된 정보의 집합이다. 좀더 정확하게 표현하자면 엔티티는 엔티티 인스턴스라고 불리우는 것들의 집합니다.

즉, 엔티티의 인스턴스(instance)는 엔티티를 이루는 고유한 각각의 엔티티 멤버인데 예를 들면 사원이라는 엔티티에서, 홍길동이나 김철수와 같은 각각의 사원은 사원 엔티티의 엔티티 인스턴스가 된다.

현실세계의 영역에서 '현실세계의 엔티티(Entity)'란 용어는 현실세계의 사람, 장소, 물건, 뿐만 아니라 개념등을 설명하는데 사용된다. 이러한 맥락에서 기업에 속해있는 부서는 사원과 마찬가지로 판매부서, 인사부서 등의 엔티티 인스턴스로 구성된 엔티티로서 나타낼 수 있다. 이러한 엔티티들은 속성명(Attribute name, 예를 들면 사원정보 엔티티의 경우 성명, 나이, 담당업무)이나 그것의 값(Attribute value, 예를 들면 홍길동, 29 세, 프로그래머)으로 표현되는 특성과 속성값들을 가지고 있다. 더욱이 현실적으로 하나의 엔티티는 다른 엔티티들과 여러가지 의미있는 '연관관계(relationships)' 들을 가질 수 있는데, 예를 들어 사원들은 부서를 위해 '일'한다고 할 수도 있고 부서는 한명 이상의 사원들로 '구성'된다고 할 수 있는 것이다.

이제 정보영역에 초점을 맞추자. IDEF1X 의 엔티티 인스턴스는 물리적, 개념적 개체(즉 사람, 장소, 물건, 아이디어)에 관하여 특정 조직에서 관리되는 정보를 표현한다. 예를 들어

한 조직에서 판매부서에 관한 정보를 관리 할 때 정보시스템 내에 그 개체(판매부서)의 정보가 존재하게 됨으로 판매부서는 정보영역에서 부서라는 엔티티의 엔티티 인스턴스로 존재하는 것이다. 현실세계의 개체로부터 수집된 추상적 정보의 집합(예를 들면 부서, 직원), 혹은 IDEF1X 에 의해 도출된 엔티티 인스턴스의 집합을 칭하여 엔티티라고 한다.
엔티티는 직원의 신상카드를 담도록 된 빈 파일박스로 생각될 수 있다. 각각의 카드는 IDEF1X 의 엔티티 인스턴스를 표시하는 것이다. 박스의 전면에는 엔티티의 이름과 , 내부에 있는 개별카드에는 카드에 등록된 정보에 대한 템플릿 레이블이 표시된다. 그림 5-5 는 사원번호가 부여된 사원들에 대한 정보의 집합을 보여 준다.

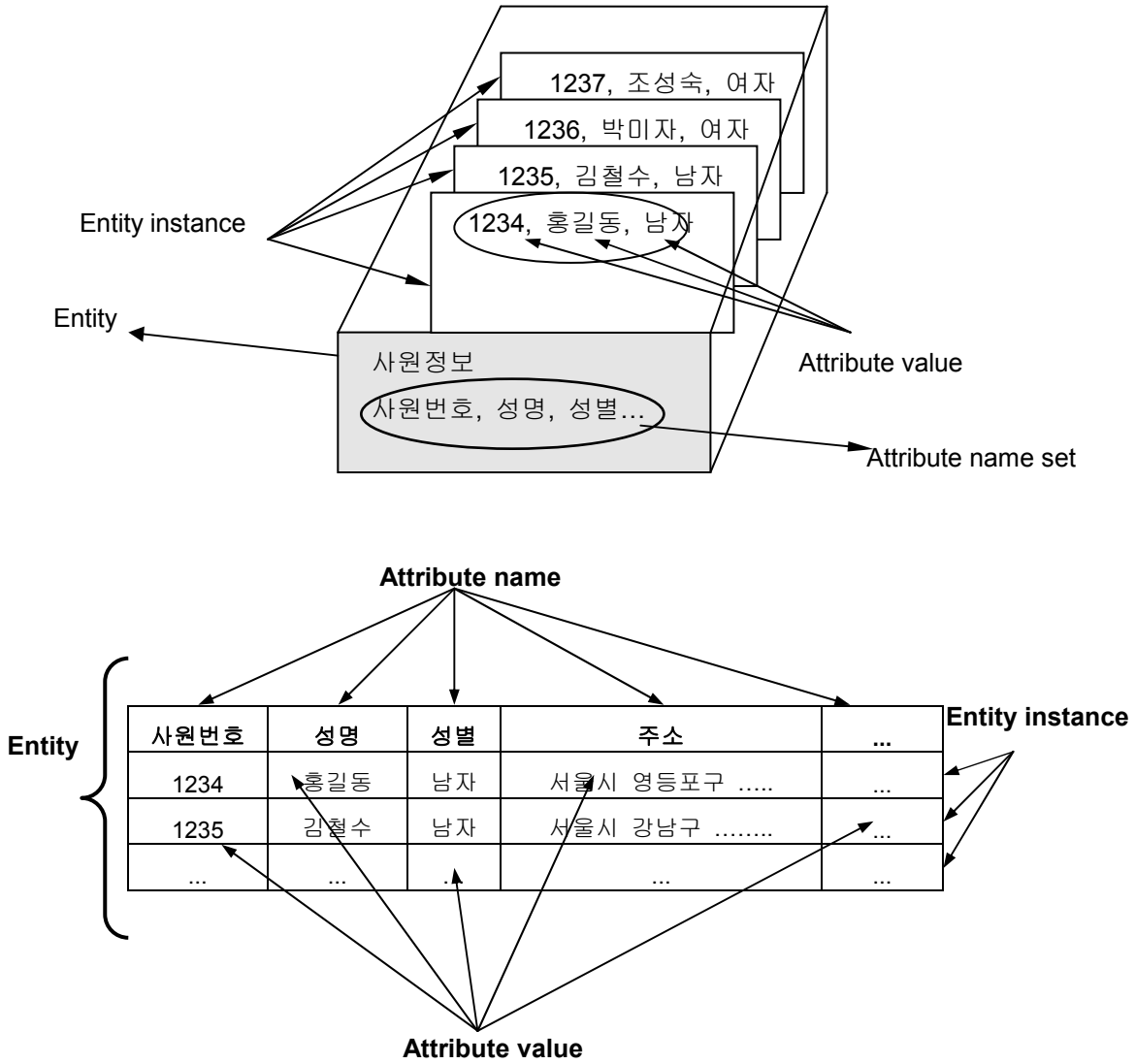
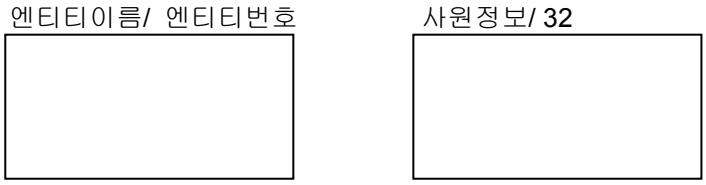


그림 5-5 : 엔티티 / 애트리뷰트

■ 엔티티 표현형식

IDEF1X 에서 각각의 엔티티는 그림 5-6 과 같이 박스형태로 표시하는데, 독립적(Independent) 엔티티와 비독립적(Dependent) 엔티티로 구분된다. independent 엔티티는 여타의 엔티티의 존재와 상관 없이 독립적으로 엔티티 인스턴스를 추출(Identifying)할 수 있는 엔티티를 말하며 dependet 엔티티는 independent 엔티티로 이미 존재하는 엔티티와의 관련에 의해서만 추출이 가능한 엔티티를 말한다. 그림 5-4 에서 우리는 이러한 예를 찾을 수 있는데, 고객엔티티의 경우 고객의 이름을 알기 위하여 다른 엔티티의 존재와 상관 없이 바로 이를 추출할 수 있으므로 이를 independent 엔티티라고 하고 IDEF1X 에서는 이를 네모난 박스로 표현한다. 비디오테이프 대여기록 엔티티의 경우는 비디오테이프를 대여받은 고객의 이름을 기준으로 엔티티 인스턴스를 추출하기 위하여는 추가적으로 고객엔티티의 고객이름 항목을 참조해야만 가능하므로 이를 dependent 엔티티라고 하며 independent 엔티티에 비하여 네 꼭지점을 둥글게 표시한다. 그림에서와 같이 엔티티의 이름은 엔티티박스위에 표현되며 보통명사의 단수형태로 표현된다. 엔티번호는 모델에 등록된 엔티티의 고유한 일련번호를 나타낸다.

Independent 엔티티



Dependent 엔티티

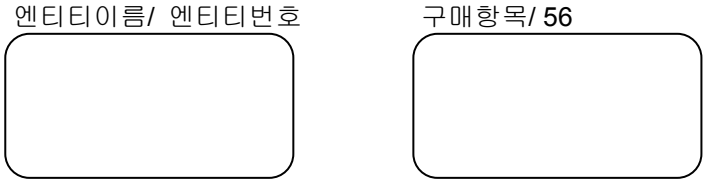


그림 5-6 : 엔티티 표현형식

■ 엔티티 규칙

① 각각의 엔티티는 유일한 이름을 가져야 하며 중복 사용되어서는 안된다. 또한 같은 이름은 항상 같은 의미로 사용되어야 한다. 역시 같은 의미에 대하여 여러가지 이름으로 사용해서도 안된다.
② 하나의 엔티티는 자신의 것이든 혹은 다른 엔티티와의 관계에 의하여 상속된 것이든 하나 이상의 애트리뷰트를 가지고 있어야 된다. (상속된 애트리뷰트에 대하여는 다음절에서 설명한다)
③ 하나의 엔티티는 각각의 엔티티 인스턴스를 유일하게 추출할 수 있는 하나 이상의 애트리뷰트를 가지고 있어야 한다. (다음절의 Primary Key 와 Alternate Key 를 참조)
④ 어떠한 엔티티이든 수적인 제한 없이 많은 엔티티들과 관계를 가질수 있다.
⑤ 만일 foreign key 가 엔티티 primary 키의 일부나 전체로 사용될 경우 이 엔티티는 독립적으로 인스턴스를 추출할 수 없는(identifier-dependent) 엔티티라하고 역으로 엔티티 primary key 로서 전혀 foreign key 가 사용되지 않는 경우를 독립적으로 인스턴스를 추출할 수 있는(identifier-independent) 엔티티라 한다.

5.2.2 애트리뷰트(속성, Attribute)

■ 애트리뷰트란 ?

애트리뷰트는 사람, 사물, 사건, 상태, 아이디어 등과 같은 현실적 혹은 추상적인 것들과 관련되어 하나의 특성이나 값을 표현한다. IDEF1X 에서 애트리뷰트는 애트리뷰트 이름(Attribute Nam)과 애트리뷰트 값(Attribute Value)으로 이루어진 이름과 값의 쎝(Set)이다.
예를 들어 그림 5-5 에서 서류박스 앞에 사원번호, 성명, 성별로 표시된 애트리뷰트 이름의 집합은 박스 안의 개별적 카드에 1234, 홍길동, 남자로 표시된 애트리뷰트 값과 하나의 애트리뷰트 쎝을 형성하며, 애트리뷰트 이름의 집합은 개별카드의 정보내용에 대한 템플릿으로 작용한다. 여기서 우리는 이러한 애트리뷰트 이름의 집합을 구성하는 각각의 애트리뷰트 이름, 즉 사원번호, 성명, 성별을 애트리뷰트 인스턴스라고 표현한다. 그림 5-5 에서 사원번호 애트리뷰트는 하나의 카드와 다른 카드를 구분하기 위아여 사용될 수 있는데 이와 같이 엔티티 안에서 하나의 엔티티와 다른 엔티티를 구분하기 위하여 사용되는 하나(혹은 여러 개)의 애트리뷰트를 Primary Key 로 부른다. IDEF1X 에서 모든 엔티티는 적어도 하나의 Primary Key 를 가져야 된다.

■ 애트리뷰트 표현형식

각각의 애트리뷰트는 명사(혹은 전치사나 형용사와 결합된 명사)나 명사구 형태로 표현되는 유일한 이름을 가져야 하며 이들 이름은 복수형태가 아닌 단수형태로 표현되어야 한다.
애트리뷰트의 이름으로 약어나 약자가 사용될 수도 있으나 모델 전체에 걸쳐서 같은 뜻으로 일관되게 사용되어야 하며 애트리뷰트에 대한 정의와 동의어나 유사어에 대한 목록도 모델 용어해설에서 정의되어야 한다.
애트리뷰트는 관련된 엔티티 박스 안에서 애트리뷰트 이름으로 표현되는데 엔티티 박스를 가로지르는 선 위에는 primary 키를 밑에는 primary 키가 아닌 애트리뷰트를 기술한다.

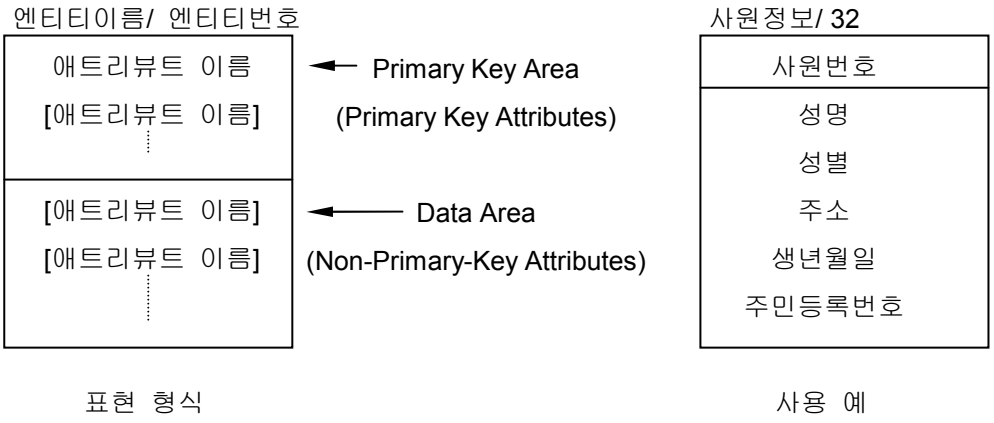


그림 5-7 : Attribute 표현 형식

■ 애트리뷰트 규칙

① 각각의 애트리뷰트는 유일한 이름을 가져야 하며 같은 의미는 항상 같은 이름으로 사용되어야 한다. 또한 동일한 이름이 다른 여러가지 뜻으로 사용되어서는 안된다.
② 하나의 엔티티가 가질 수 있는 애트리뷰트의 수에는 제한이 없다. 모든 애트리뷰트는 오직 하나의 엔티티에 의해서만 소유된다.(이를 Single-Owner Rule 이라고 하며, 소유와 관련 하여서는 다음 절에서 설명된다.)
③ 하나의 엔티티가 가질 수 있는 상속된 애트리뷰트의 수에는 제한이 없다. 그러나 상속된 애트리뷰트는 관계된 parent 엔티티나 generic 엔티티의 primary 키를 구성하는 애트리뷰트여야 한다.
④ 하나의 엔티티에서 각각의 엔티티 인스턴스는 모든 애트리뷰트에 대하여 값은 가져야 한다.(이를 널 값을 허용하지 않는 No-Null Rule 이라 부른다)
⑤ 하나의 엔티티에서 각각의 엔티티 인스턴스는 그 엔티티와 관련된 하나의 애트리뷰트에 대하여 하나 이상의 값을 가질 수 없다.(이를 No-Repeat Rule 이라 한다.)

2.4.1 Primary 키와 Alternate키

데이터 모델링에서 attribute 의 종류는 키 애트리뷰트와 그렇지 않은 애트리뷰트의 두 종류로 나눌 수있으며 키 애트리뷰트는 다시 다음과 같은 타입은 세가지로 나뉘어 진다.

① Primary 키 : 엔티티의 모든 인스턴스를 유일하게 구분하는 애트리뷰트 집합. 하나의 엔티티에서 Primary 키는 오직 하나만이 존재할 수 있으며 하나의 Primary 키는 하나 이상의 애트리뷰트로 구성된다.
② Alternate 키 : Attribute 가 그 값으로 엔티티를 유일하게 구분하지만 primary 키가 아닌 것, 이는 하나 이상 될 수 있다.
③ Foreign 키 : 상위 엔티티의 primary 키에서 전이되어 하위 엔티티에 나타나는 Attribute.

우리는 이 절에서 Primary 키와 Alternate 키에 대해서만 논하기로 하고 Foreign 키에 관해서는 다음 절의 Relationship 에 관해서 살펴 본 뒤 논하기로 한다.

■ Primary 키

위의 그림 5-7 의 사원정보 엔티티에서 특정 사원을 다른 사원과 구분하기 위하여 사원번호를 이용한 것과 같이 IDEF1X 모델에서 특정 엔티티 인스턴스는 다른 엔티티 인스턴스와 특정 애트리뷰트 혹은 애트리뷰트의 조합을 통하여 구분되어야 한다. 이와 같이 특정 엔티티의 모든 인스턴스를 유일하게 구분하는 애트리뷰트 혹은 애트리뷰트의 조합을 키라고 하는데 모든 엔티티는 이와 같이 최소한 하나의 키가 될 수 있는 애트리뷰트를 가져야 되며 이를 우리는 Candidate 키라고 한다.
사실 하나의 엔티티를 구성하는 애트리뷰트들 중에서는 키로서 사용될 수 있는 애트리뷰트나 애트리뷰트의 집합이 여러 개 존재할 수 있는데 그림 5-8 에서 XYZ 회사의 사원정보 Primary 키 설정 작업에서와 이를 확인하여 보자. 그림에서 우리는 주민등록번호 애트리뷰트나 성명 애트리뷰트와 생년월일 애트리뷰트를 결합한 형태(성명과 생년월일이 같은 사람이 한 회사에 있을 경우는 거의 없다고 생각되므로)의 Candidate 키를 추출 할 수 있으며 이들 또한 Primary 키가 될 수 있다고 생각 할 수 있다. 그러나 단지 성명이나 생년월일 하나만을 Candidate 키로 추출하는 것은 경험적으로 볼 때 같은 이름의 사원이나 생년월일이 같은 사원이 종종 발견되므로 배제된다. 그림 5-8 에서 형식 A, B 는 각각 적당한 Primary 키의 선정으로 보여지나 XYZ 회사의 모델 작업자는 주민등록번호나 성명과 생년월일을 결합한 현태로 엔티티의 Primary 키를 설정하는 것이 자료의 등록이나 검색과정에서 상당히 불편하다고 생각되어 형식 C 와 같이 사원번호 애트리뷰트를 추가하여 Primary 키로 정했다.

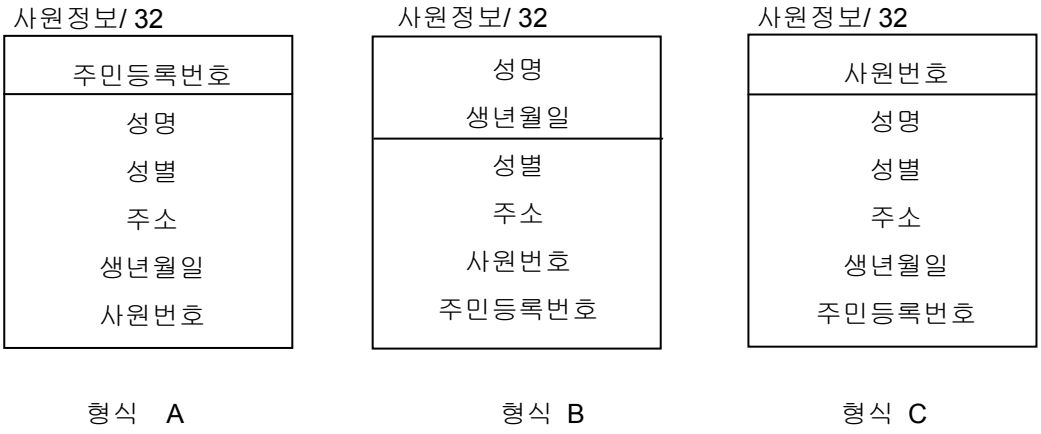


그림 5-8 : XYZ 회사의 사원정보 Primary 키의 설정 예

■ Primary 키 설정 지침

Primary 키를 정하기 위한 많은 원칙들이 있으나 다음은 Primary 키를 쉽게 결정하기 위한 지침이다.

① 가장 중요한 것은 각각의 엔티티 인스턴스 수명에 있어서 그 값이 바뀌지 않는 것을 찾아야 한다.

② 가능하면 짧은 것을 선택해야 한다. 두개의 애트리뷰트로 구성된 Primary 키(위의 그림에서 성명+생년월일과 같이) 보다는 한나의 애트리뷰트로서 Primary 키가되는 것이 효과적이다. 하나 두개 이상의 애트리뷰트로 결합된 Primary 키를 선정하는 경우에도 Primary 키를 구성하는 모든 애트리뷰트는 항상 값을 가지고 있어야 하며 만일 이를 만족하지 못하는 경우는 다른 Candidate 키를 찾아야 한다.

③ 특정 그룹이나 위치, 분류, 날짜 등과 같은 의미를 가진 애트리뷰트를 Primary 키로 선정해서는 안된다. 만일 키가 아무런 뜻을 가지고 있지 않다면 그 값이 바뀌는 경우는 그만큼 줄어드는 것이다.

④ 날짜 등의 애트리뷰트를 꼭 Primary 키로 사용해야 할 경우 엔티티 인스턴스가 유효한 값으로 존재될 수 있도록 제한되어야 한다. 즉 현실세계에서 존재할 수 없는 2000 년 2 월 30 일과 같은 값이 20000230 과 같은 형태로 등록되는 것은 통제되어야 한다.

⑤ 복잡하고 길게 구성된 Primary 키는 짧고 간단한 하나의 애트리뷰트로 대체되어야 한다. 위의 그림에서 만일 '주소'가 중복되는 상황이 발생되지 않고 따라서 이를 Candiate 키로 추출하였다고 가정하여도 주소와 같이 길고 복잡한 애트리뷰트를 Primary 키로 설정하는 것은 효과적인 것이 될 수 없다.

■ Alternate 키

모든 엔티티는 반드시 하나 이상의 애트리뷰트로 이루어진 유일한 키를 가져야 하는데 만약 유일한 키가 하나 이상 존재한다면 단지 하나만이 Primary 키가 될 수 있고 나머지는 Alternate 키가 된다. 엔티티 내에 각각의 엔티티 인스턴스를 구분할 수 있는 키는 전술한 Primary 키 이외에 Alternate 키와 Foreign 키가 있는데 Alternate 키는 Primary 키로 정하여진 키 이외에 각 엔티티에 의해 소유된 애트리뷰트중에서 각각의 엔티티 인스턴스를 구분할 수 있는 애트리뷰트나 애트리뷰트의 집합을 말하며 애트리뷰트 이름 뒤에 애트리뷰트 키임을 나타내는 AK 와 순번을 나타내는 정수의 조합으로 AK1, AK2, AK3..등으로 표시되며 엔티티 박스를 가로지르는 가로 선 아래 데이터 영역(Data Area)에 기술한다. 각각의 애트리뷰트는 하나 이상의 Alternate 키의 부분으로 정의될 수 있으며, Primary 키 애트리뷰트 또한 Alternate 키의 부분으로서 정의될 수 있다. 만약 어떤 애트리뷰트가 키가 아닌 경우(즉, 그 엔티티와 관련이 있지만 인스턴스를 구분하는데 있어서 어떠한 역할도 하지 않는), 이를 Descriptive 애트리뷰트라 한다.
그림 5-9 는 Alternate 키의 표현방법을 설명하고 있다.

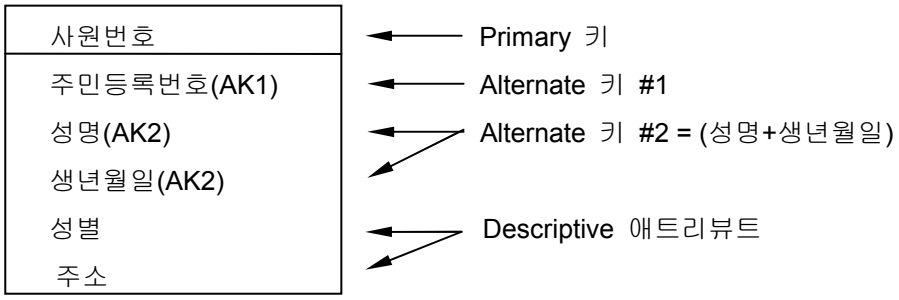


그림 5-9 : Alternate 키의 표현방법

■ Primary 키와 Alternate 키 규칙

① 모든 엔티티는 반드시 Primary 키를 가져야 한다. 또한 각 엔티티는 하나의 엔티티만을 가질 수 있다.
② 어떤 엔티티도 가질 수 있는 Alternate 키의 수에는 제한이 없다.
③ 하나의 Primary 키나 Alternate 키는 하나의 애트리뷰트나 애트리뷰트의 조합으로 구성된다.
④ 하나의 엔티티에서 Primary 키나 Alternate 키를 이루는 애트리뷰트들은 그 엔티티에 의해 소유되었거나 Relationship 을 통하여 상속된 것이다.

⑤ Primary 키와 Alternate 키는 엔티티 인스턴스를 유일하게 구분하기 위한 것들 만으로 이루어져야 한다. 즉 키를 구성하는 애트리뷰트 중 하나라도 생략하면 그 엔티티에서 유일한 값을 갖는 인스턴스를 추출할 수 없는 것들만으로 이루어져야 하는데 이를 Smallest-Key Rule 이라 한다.

⑥ 만일 Primary 키가 하나 이상의 애트리뷰트로 구성된 경우, 모든 키가 아닌 애트리뷰트(Non-key attribute)의 값은 Primary 키에 기능적으로 의존적(Functionally dependent)이어야 한다. 즉 Primary 키를 알면 모든 키가 아닌 애트리뷰트의 값을 알 수 있어야 하며 모든 키가 아닌 애트리뷰트의 값은 오직 Primary 키에 의해서 결정되어야 하는데 이를 Full-Function-Dependency Rule 이라 한다.

⑦ 모든 키가 아닌 애트리뷰트들은 오직 Primary 키와 Alternate 키에 의하여 기능적으로 의존적이어야 한다. 즉 키가 아닌 애트리뷰트의 값이 다른 키가 아닌 애트리뷰트의 값에 의하여 결정될 수 있어서는 안된다. 이를 No-Transitive-Dependency Rule 이라 한다.

IDEF1X 모델에서 모든 애트리뷰트는 반드시 하나의 엔티티의 구성원으로 속해져야 하며 엔티티의 모든 엔티티 인스턴스는 엔티티와 관련된 모든 애트리뷰트에 대한 값을 가져야 한다.

5.2.4 연결 관계(Connection Relationship)

■ Relationship의 의미

IDEF1X 에서 Relationship 은 엔티티와 엔티티 간의 관계를 말하는데 기업의 정보시스템에 의해 관리되는 현실적 개체 사이의 의미 있는 관계를 표현한다. Relationship 의 표현방법은 우리가 포착한 중요사항을 기록하는데 있어서 업무 룰이나 자연어보다는 단순한 형태로 표시된다. IDEF1X 모델은 기업의 정보체계 구조를 설명하기 때문에 기업의 정보관리 정책을 통하여 '업무 룰'이 정보시스템 속에 반영된 경우에만 IDEF1X 에 의해서 그 것이 명백하게 표현되어질 것이다. 달리 말하면 현실세계에 있어서 두 개 이상의 개체 사이에 정보가 유지되지 않으면, IDEF1X 관점에서 볼 때도 관계가 존재하지 않는다. 따라서 정보시스템에 의해 반영되지 않는 업무 룰은 정밀한 IDEF1X 모델 속에도 나타나지 않을 것이다. 엔티티간의 관계를 설정하기 위하여는 엔티티를 선정하기 위한 자료의 수집 및 검증 작업이 선행되어야 하며 다음은 이들 엔티티간의 Relationship 을 설정하는 것인데 IDEF1X 모델링에서는 Relationship 을 크게 특정한 의미를 가지는 관계(Specific Relation)와 그렇지 않은 관계(Non-Specific Relationship)로 나눌 수 있다.

■ Specific Relationship

Specific relationship 은 두 엔티티 간에 모자관계(parent-child) 내지는 종속관계(existency-dependency)를 가진 경우를 말하는데 Specific Relationship 은 다시 엔티티간의 상호 의존관계에 따라 Identifying relationship 과 Non-identifying relationship 으로 나눌 수 있으며 Identifying relationship 이란 자(child) 엔티티의 유일한 인스턴스의 구분을 위해서는 관계된 모(parent) 인스턴스를 알아야 되는 경우로서 자(Child) 엔티티의 유일한 인스턴스는 항상 모(Parent) 엔티티 인스턴스와 연관되어야만 추출이 가능하다. Identifying Relationship 에 있어서 자(child) 엔티티는 항상 종속된 엔티티로서 전술한 바와 같이 비독립적(Dependent) 엔티티로 불리워 지며 네 꼭지점을 둥글게 표현한다. Non-identifying relationship 은 자(child) 엔티티의 유일한 인스턴스의 구분을 위해서 관계된 모(parent) 인스턴스를 알아야 될 필요가 없는 경우인데 모(parent) 엔티티와 관계는 가지나 모(Parent) 엔티티와의 연관에 상관없이 독립적으로 자신의 유일한 엔티티 인스턴스를 추출할 수 있다. Non-identifying relationship 에서 자(Child) 엔티티는 다른 엔티티와 identifying relationship 관계가 아닌 이상 identifyier-independent 엔티티로 표현되며 독립적(Independent) 엔티티로서 네모난 박스 형태로 표현한다. IDEF1X 다이아그램에서 Identifying relationship 의 연결관계는 실선으로 Non-identifying relationship 은 점선으로 각각 문자로 표시되는 Relationship 이름과 함께 표현한다.

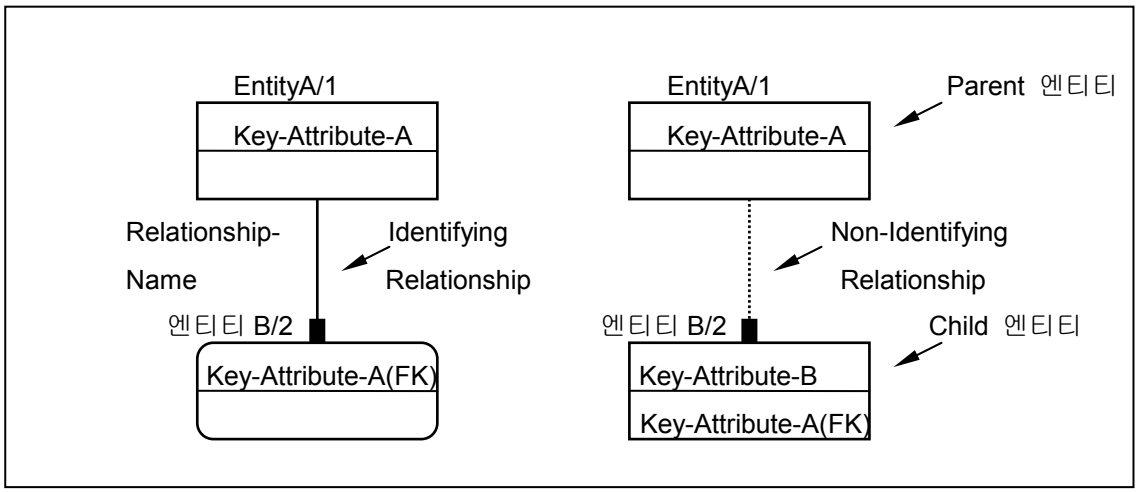


그림 5-10 : Identifying/Non-Identifying Relationship

■ 카디날리티(Cardinality)

Relationship 은 각 엔티티가 다른 엔티티와 연관될 때 카디날리티(Cardinality)라고 불리어지는 값을 갖는데 Relationship 으로 관련된 엔티티의 인스턴스들 간에 수적으로 어떠한 상

관관계가 존재하는지를 표현한다. 예를 들어 그림 5-11 은 XYZ 회사에서 사원들의 월별 급료를 계산하기 위한 모델인데 '부서' 엔티티와 '사원' 엔티티를 연결하는 Relationship 에서 Relationship 이름을 나타내는 '고용한다'와 사원 엔티티쪽에 표시된 'P'를 발견할 수 있는데 이는 각각 Relationship 이름과 카디날리티를 나타내는 것으로 'P'는 하나이상의 즉 양수(Positive)를 의미한다. Relationship 이름은 동사구(혹은 필요에 따라서 전치사나 부사에 의해 수식되는) 형태로 표기되며 Relationship 을 나타내는 선 옆에 표시된다. 같은 두 엔티티 사이에 존재하는 Relationship 이름은 항상 유일해야 하나 모델 안에서 하나의 Relationship 이름이 유일할 필요는 없다. 또한 Relationship 이름은 항상 모(Parent) 엔티티에서 자(Child) 엔티티 방향으로 표현되는데 모(Parent) 엔티티, 카디날리티, 자(Child) 엔티디 그리고 카디날리티의 순으로 표현되는 문장을 만들 수 있다 이러한 원칙을 적용하면 '고용한다' Relationship 은 '하나의 부서에는 적어도 한명 이상의 사원을 고용한다'는 의미를 나타낸다. 마찬가지로 급호 엔티티와 급료 엔티티를 연결하는 Relationship 에서는 'M'을 발견 할 수 있는데 이는 '하나의 급호는 한번도 사용되지 않을 수도 있지만 한번이나 여러 번 사원들의 급료계산에 사용될 수도 있다(만일 XYZ 회사에 3 급 30 호봉이라는 급호는 존재하지만 그런 급호를 가진 사원이 없을 수도 있으므로)'는 것을 의미한다. 그런데 IDEF1X 모델에서 'M'이라는 문자는 보통 생략된다. 같은 방법으로 'Z'로 표시된 Relationship 은 '각각의 사원에게 해당년월에 급료가 지급되지 않거나(상당기간의 무급휴가나 정직등에 의해서) 지급되더라도 오직 한 번만 지급된다'는 업무 룰을 나타낸다. 다음 절에서 우리는 이러한 Cardinality 의 종류에 대하여 자세히 알아본다.

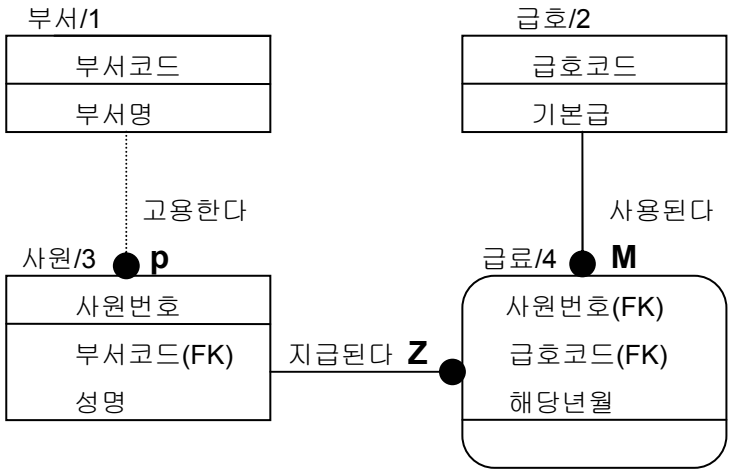


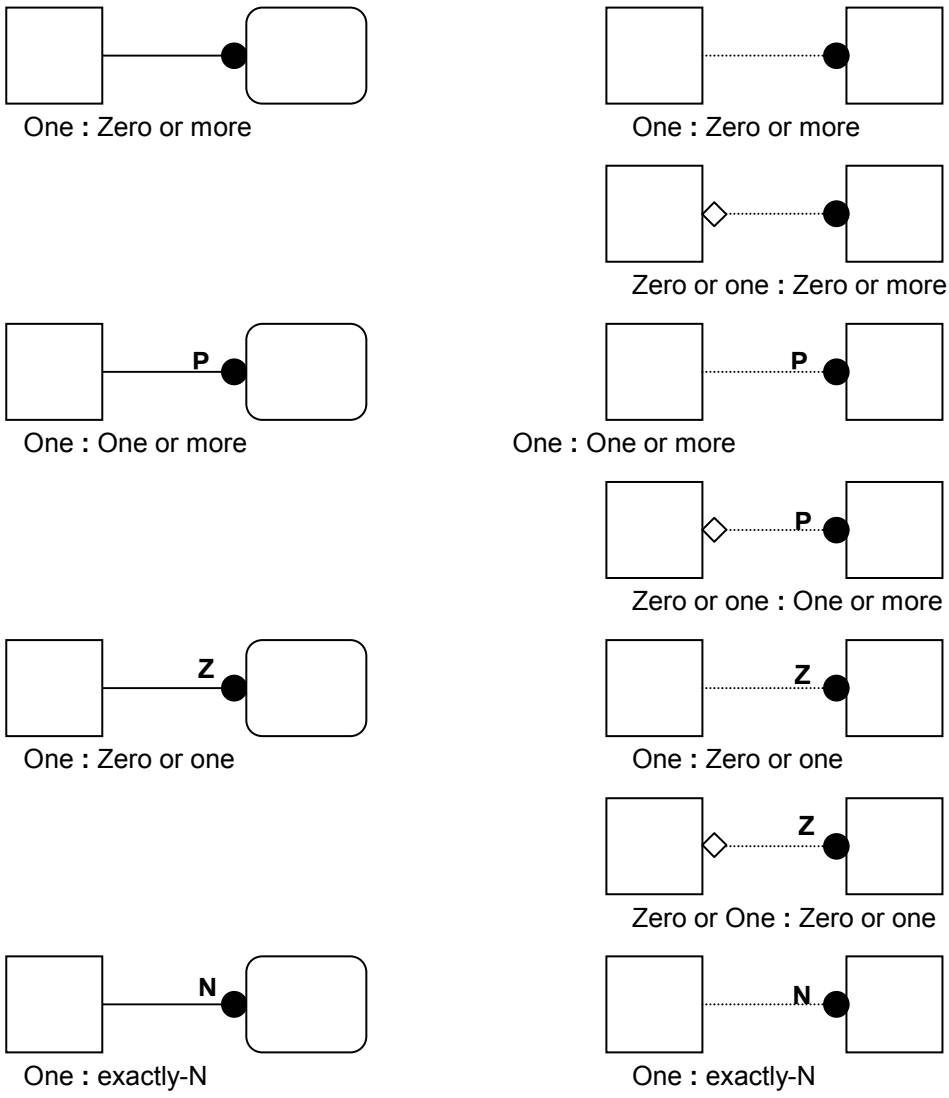
그림 5-11 카디날리티 예

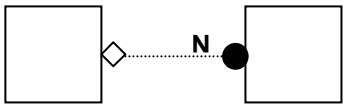
■ 카디날리티의 종류

전술한 바와 같이 Specific Relationship 은 identifying 혹은 non-identifying 으로 구분 될 수

있는데 이러한 구분은 데이터 모델 안에서 실선(identifying)이나 점선(non-identifying)으로 나타내어 진다. 또한 이들 각각의 Relationshp 은 추가적으로 다음과 같은 네 가지 종류의 Relationship 으로 분류할 수 있으며 그 표현 형식은 그림 5-12 와 같다.

① 각각의 모(Parent) 엔티티 인스턴스는 0, 1, 혹은 그이상의 관련된 자(Child) 엔티티 인스턴스를 가질 수 있다.
② 각각의 모(Parent) 엔티티 인스턴스는 적어도 하나 혹은 그 이상의 관련된 자(Child) 엔티 인스턴스를 가져야 한다.
③ 각각의 모(Parent) 엔티티 인스턴스는 관련된 자(Child) 엔티티 인스터스를 하나도 가지지 않거나 최대 하나만을 가질 수 있다.
④ 각각의 모(parent) 엔티티 인스턴스는 정확하게 특정 숫자의 관련된 자(Child) 엔티티 인스턴스를 가진다.





Zero or one : exactly –N

Identifying Relationship **Nonidentifying Relationship**

그림 5-12 Cardinality 표현 방법

■ Specific Relationship의 규칙

① 하나의 Specific Relationship 은 항상 정확하게 두 개의 엔티티 사이에 있어야 하며 이때 하나는 모(Parent) 엔티티로 다른 하나는 자(Child) 엔티티로 구성된다.
② 자(Child) 엔티티의 하나의 인스턴스는 항상 정확하게 모(Parent) 엔티티의 하나의 인스턴스와 관련되어져야 한다.
③ 모(Parent)의 하나의 인스턴스는 표현된 카디날리티에 따라 자(Child) 엔티티의 인스턴스들과 0, 1, 혹은 그이상 관련될 수 있다.
④ Identifying-relationshp 에서 자(child) 엔티티는 항상 Indentifier-dependent 엔티티이다.
⑤ 하나의 엔티티는 모(Parent) 엔티티나 자(Child) 엔티티로서 숫자에 제한 없이 다른 엔티티와 연관되어질 수 있다.

■ Categorization Relationship

종종 현실세계에서는 같은 '사원'이라는 동일한 특성을 가진 엔티티의 인스턴스들을 '남자'와 '여자' 혹은 '정규직'과 '계약직'등 우리가 필요로 하는 바에 따라 분류(범주화)하는데 IDEF1X 에서는 이러한 현실세계의 특성을 정보모델 안에서 지원하기 위하여 Identifying-relationship 의 특수한 형태인 Categorization Relationship 으로 포괄적인 엔티티의 분류관계를 표현한다. 그림 5-13 은 XYZ 신용카드사의 회원관리를 위한 포괄적(Generic) 엔티티인 회원 엔티티와 이를 필요에 따라 특정 등급의 회원들에게만 관련 자료를 별도로 발송하기 위하여 회원등급을 기준(Discriminator)으로 분류한 카테고리(Category) 엔티티들과의 관계를 나타낸 것이다.

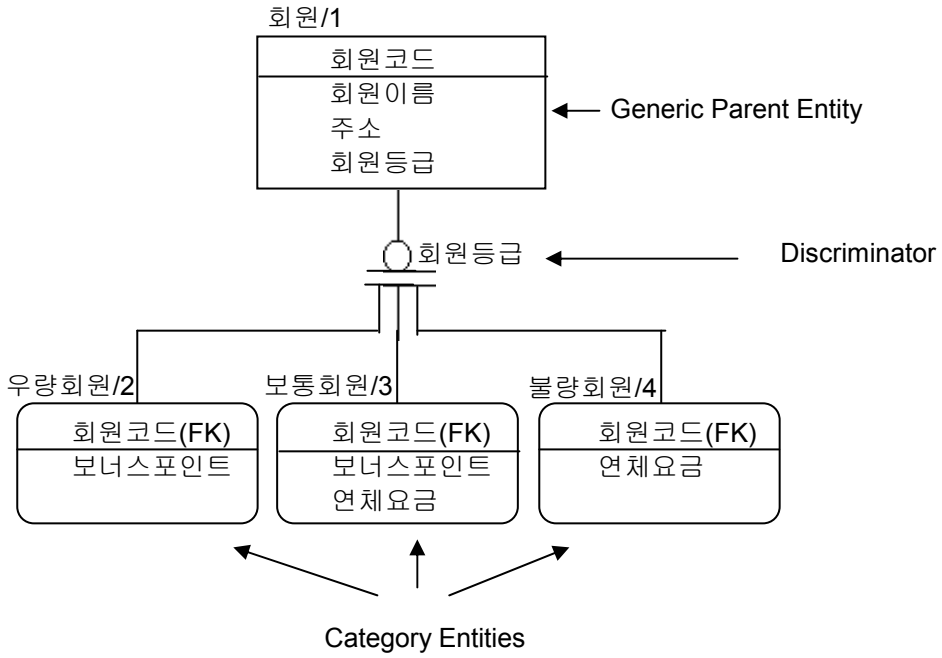


그림 5-13 : Categorization Relationship

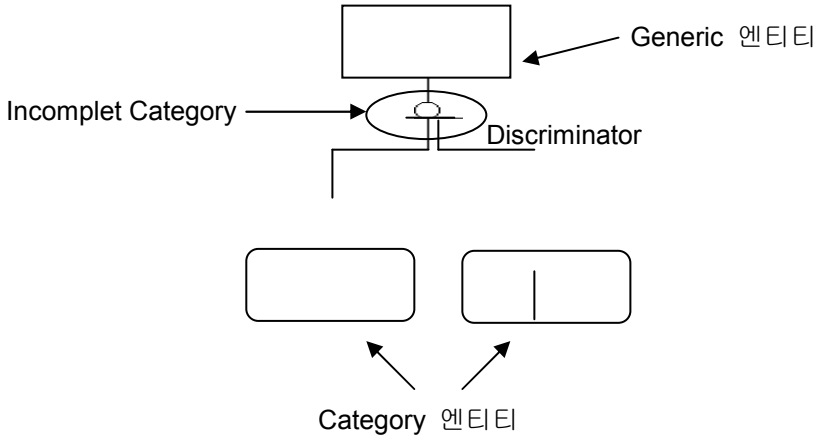
이들 '범주 관계(Categorization Relationship)'는 포괄적(Generic Parent, Generalization) 엔티티의 배타적 하위세트(Sub set, Category Entities)를 상호 표현한다. 달리 말하자면, 상위 엔티티에서 분리된 하위 엔티티는 공통된 경우를 지닐 수 없다. 예를 들어 Generic Parent 엔티티가 '회원'이라고 할 때, 회원등급(Discriminator)을 기준으로 우량회원, 보통회원, 불량회원이라는 세개의 하위 Category 엔티티로 분리될 수 있는데 이것은 모든 분류의 회원등급을 포함한 것으로 우량회원이면서 보통회원이 될 수 없으며 또 그 반대도 마찬가지이다. Category 엔티티의 Primary 키는 항상 Generic 엔티티의 Primary 키를 상속 받으며 '회원'

모두에게 적용되는 일반적 애트리뷰트는 Generic Parent 엔티티에 표현된고 분류에 따른 나타낼 필요가 있는 애트리뷰트(위의 경우에는 보너스포인트 등)는 Category 엔티티에 표시된다.

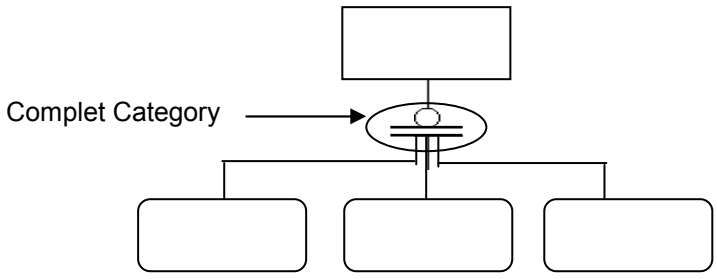
■ Categorization 표현형식

Categorization Relationship 은 아래의 그림에서와 같이 Generic 엔티티에서 나간 선이 밑줄이 그어진 둥근원으로 나타나는데, 이 밑줄에서부터 갈라진 선이 각각의 Category 엔티티로 이어진다. 여기서 Cardinality 의 종류는 표시되지 않는데 Category 엔티티들은 항상 Identifier-dependent 엔티티이고 Z 으로 표현할 수 있는 0 혹은 1 의 값을 가지기 때문이다. Categorization Relationship 은 그 분류의 완료정도에 따라 다음과 같은 두가지로 분류 된다.

① Incomplete category 관계는 categorization 이 아직 완벽하게 완료되지 않았음을 나타낸다(이는 후에 다른 엔티티가 추가될 수도 있음을 나타낸다). Incomplete category 관계는 아래 그림과 같이 하나의 수평선으로 나타낸다.



② Complete category 관계는 categorization 이 완벽하게 구축되었음을 나타낸다. Complete category 관계는 아래 그림과 같이 두개의 수평선으로 나타낸다.



Discriminator 로 사용되는 Generic 엔티티 애트리뷰트의 이름은 둥근원 옆에 표시된다. 또한 Relationship 자체의 이름은 생략되는데 Generic 엔티티를 주어로 Category 엔티티를 목

적어로 해서 '될 수 있다' 혹은 '되어야 한다'로 읽는다. 그림 5-13 의 경우는 회원은 '우량', '보통', '불량'외원이 될 수 있다. 혹은 모든 회원은 '우량', '보통', '불량'회원중 하나로 되어야 한다로 읽을 수 있다.

■ Categorization Relationship 규칙

① 하나의 카테고리 엔티티는 오직 하나의 Generic 엔티티만을 가질 수 있다. Generic 엔티티는 하나의 카테고리 Relationship 을 위한 카테고리들의 집합에서 하나의 구성요소로서 존재한다.
② 하나의 카테고리 Relationship 안에 있는 카테고리 엔티티는 다른 카테고리 Relationship 의 Generic 엔티티가 될 수 있다.
③ 하나의 엔티티는 Generic 엔티티로서 가질 수 있는 카테고리 Relationship 의 숫자에는 제약이 없다.
④ 하나의 카테고리 엔티티는 Identifying Connection relationship 의 자(Child) 엔티티가 될 수 없다.
⑤ 카테고리 엔티티의 Primary 키는 Generic 엔티티의 Primary 키와 같아야 한다.
⑥ 하나의 카테고리 엔티티의 모든 인스턴스는 같은 discriminator 값을 가져야 하며 서로 다른 카테고리 엔티티에 속한 인스턴스는 서로 다른 discriminator 값을 가져양 한다.

IDEF1X 는 연관된 데이터의 구조가 아니라 개체 그 자체만을 모델화하는 방법으로 가장 적당한데 이런 형태의 모델을 종종 '개념모델(Conceptual model)'이라고 부른다. 그런데 이러한 방법은 일반화/특수화 구조에서 어려움이 발생한다. IDEF1X 는 공통적 개체를 하나의 종류로 모델화 하려는 일반화 구조와 하나의 개체를 종류별로 범주화 하려는 특수화 구조로서 이를 지원하는데 현실세계에서 하나의 개체는 일반적으로 다른 개체와 중복되지도 않고 또 하나의 개체가 여러 개체로 분리되지도 않기 때문이다. IDEF1X 가 카테고리 엔티티에 대하여 상호 성원을 허용하지 않기 때문에, IDEF1X 는 개념 모델링 언어로서 기능하는 데에는 제약이 따른다. 예를 들어 기업에서 근무하는 사원의 종류에 관한 모델은 관리자,엔지니어,디자이너, 비서 등을 포함 할 수 있다. 이 경우에 일반화 된 실체는 '사원'이 될 것이다. 또 특수화 실체는 앞에 열거한 것들이 될 것이다. 이 구조의 의미는, 엔지니어는 디자이너가 될 수 없고 또 그들은 관리자가 될 수 없다는 것이다.

■ Non-Specific Relationship Senmantics

Parent-child 연결이나 Categorization relationship 모두는 하나의 엔티티의 인스턴스가 연관된 다른 인스턴스와 정확하게 어떤 관계를 가지는지를 정의하기 때문에 Specific 엔티티로 분류된다. 완성된 IDEF1X 모델에서 엔티티간의 모든 연관괸계는 Specific Relationship 으로

표현된다. 그러나 모델의 초기 개발단계에서는 두 엔티티 사이의 연관관계를 Non-specific 으로 정의하는 것이 종종 도움이 되는데 이들 Non-specific relationship 은 다음의 개발단계에서 정제된다. 이들 non-specific relationship 을 해결하기 위한 절차는 다음 절에서 논한다. 하나의 Non-specific relationship 은 또한 다대 다(many to many) relationship 으로도 불리워지는데, 연관된 두 엔티티에서 첫번째 엔티티의 각각의 인스턴스가 0, 1, 혹은 그이상의 두번째 엔티티 인스턴스와 연관될 수 있고 그 역으로도 가능한다는 것을 나타낸다. 만일 하나의 사원이 여러 개의 프로젝트에 동시에 투입되고 또한 하나의 프로젝트가 많은 사원들에게 할당되는 경우라면 사원 엔티티와 프로젝트 엔티티의 관계는 Non-specific relationship 으로 표현될 것이다.

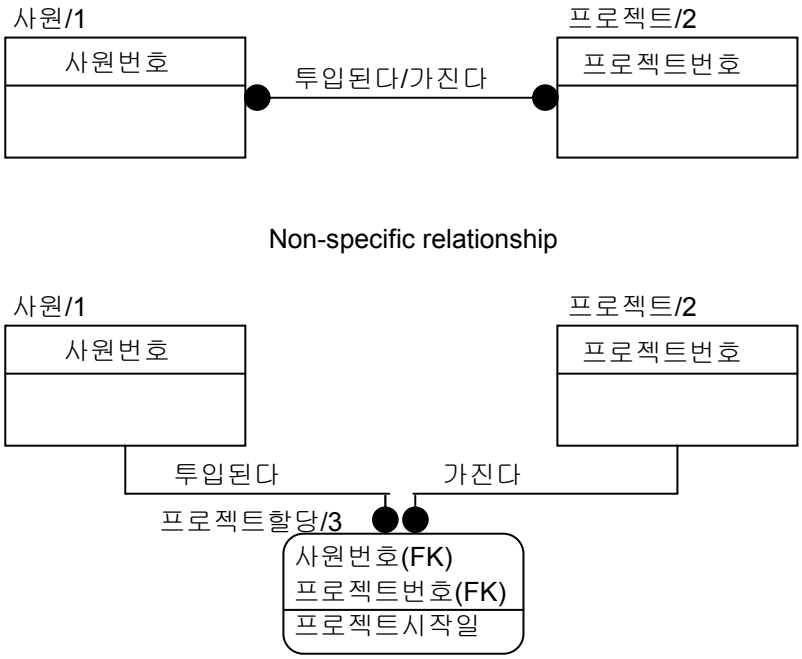


그림 5-14 : Non-specific relationship 과 Specific relationship

이 Non-specific relationship 은 후에 개발단계에서 프로젝트할당과 같은 별도의 엔티티를 추가함으로서 Specific Relationship 으로 전환되는데, 새롭게 추가되는 프로젝트할당 엔티티는 사원엔티티와 프로젝트엔티티의 공통의 자(Child) 엔티티가 된다. 새로운 relationship 들은 하나의 사원은 0, 1, 또는 그 이상의 프로젝트할당에 투입된다 그리고 하나의 프로젝트는 0, 1, 혹은 그 이상의 프로젝트할당을 가진다고 상세회된다. 각각의 프로젝트할당 엔티티 인스턴스는 정확하게 하나의 사원과 하나의 프로젝트와 연관된다. 이와 같이 non-specific relationship 문제를 해결하기 위하여 추가된 엔티티를 intersection 혹은 associative 엔티티

라고 한다.

■ Non-Specific Relationship 표현형식

하나의 non-specific relationship 은 두 엔티티 사이에 양쪽 끝에 점이 붙은 선으로 표현된다. 또한 그림 5-12 에서처럼 양쪽 끝에 카디날리티를 나타낼 수 있다. 예를 들어 한 쪽의 점 옆에 표시된 'P'라는 표시는 다른 쪽 엔티티의 하나의 인스턴스가 관련된 엔티티의 하나 혹은 그이상의 엔티티 인스턴스와 연관될 수 있음을 나타낸다. 또한 'Z'나 'N' 등의 카디날리티도 표현될 수 있으며 이와 비슷하게 최소값이나 최대값 혹은 특정 숫자(양수)를 카디날리티로 표현할 수 도있다. 이때 카디날리티의 기본값은 0,1, 혹은 그이상의 값을 가지며 카디날리티의 표현은 생략된다.

Relationship 이름은 동사구(전치사나 부사를 동반하는 동사)의 형태로 Relationship 선 옆에 표기되는데 Non-specific relationshp 의 이름은 관련된 두 엔티티 모두를 기준으로 각각에 대하여 만들어지며 이들 두 이름은 슬래시('/')에 의해 구분된다. Relationship 이름은 엔티티들의 위치에 따라 달라지는데 첫번 째 이름은 연관된 엔티티들이 수직으로 정렬된 경우는 위에서 아래로 수평으로 정렬된 경우는 좌에서 우로 이어지는 엔티티간의 관계를 나타낸다. 두 번째 이름의 경우는 그 역방향으로의 관계를 나타낸다. 그림 5-14 에서 Non-Specific Relationshp 의 이름을 나타내는 '투입된다/가진다'의 경우 첫번째 이름인 투입된다는 '사원은 0,1, 혹은 그 이상의 프로젝트에 투입되다' 는 문장으로 두번째 이름인 가진다는 '프로젝트는 0,1, 혹은 그이상의 사원을 갖는다'는 문장으로 표현될 수 있다.

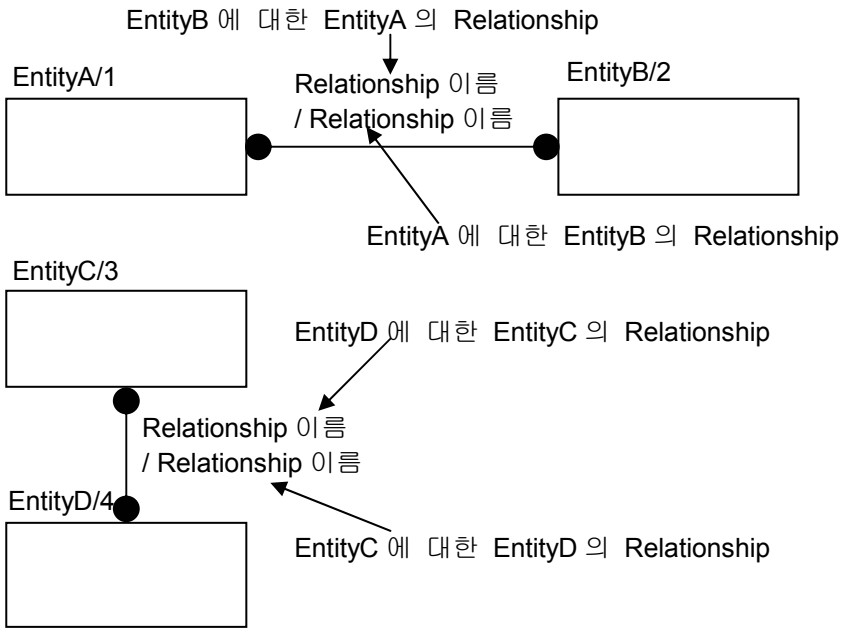


그림 5-15 : Non-Specific Relationship 표현방법

■ Non-Specific Relationship 규칙

① 하나의 non-specific relationship 은 정확하게 언제나 두개의 엔티티들 사이에만 있어야 한다.
② 양쪽 엔티티 모두의 하나의 엔티티 인스턴스는 다른 엔티티 쪽에 표시된 카디날리티에 따라 0, 1, 혹은 그 이상의 엔티티 인스턴스와 관련된다.
③ Non-specific relationship 은 완벽한 모델이 되기 위하여 필히 specific relationship 으로 변환되어야 한다.

■ 키의 상속과 Foreign 키

만일 두 엔티티 사이에 specific relationship 이나 카테고리 relationship 이 존재하면 모(parent)엔티티나 generic 엔티티의 키를 이루는 애트리뷰트들은 자(Child)엔티티나 카테고리 엔티티에 상속된다. 이와 같이 상속된 엔티티들을 Foreign 키라고 한다.

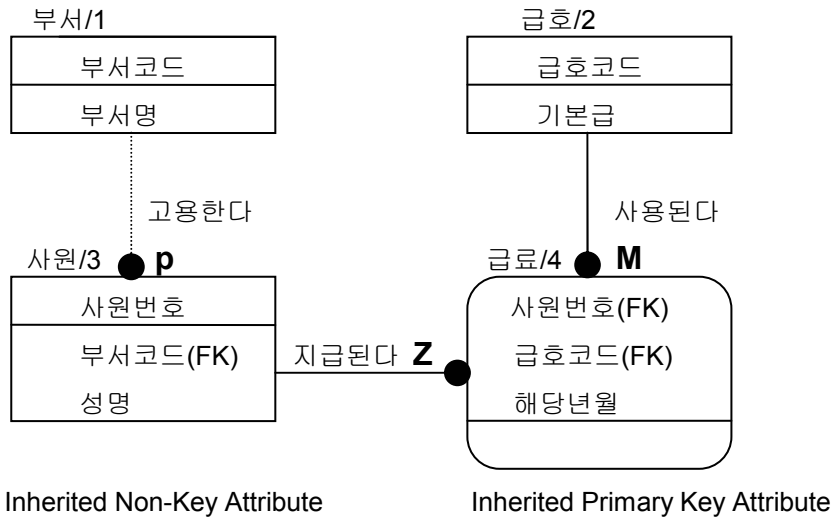


그림 5-16 : 키의 Inheritence 와 Foreign 키

그림 5-16 에서와 같이 Specific Relationship 으로 연관된 두 엔티티 사이에서 모(Parent) 엔티티의 Primary 키는 자(Child) 엔티티의 애트리뷰트나 Foreign 키로 상속되는데, 상속되는 애트리뷰트는 자(Child) 엔티티의 Primary 키 혹은 Alternate 키의 부분이나 전체를 이룰 수

도 있고 또는 Non-Key 애트리뷰트어느 것으로도 될 수 있다. 사원엔티티와 급료엔티티의 관계와 같이 만일 모(Parent) 엔티티의 Primary 키를 이루는 애트리뷰트들이 모두 자(Child) 엔티티의 Primary 키의 부분으로 상속되는 경우, 이를 Identifying Relationship 이라고 하며 연결관계는 실선으로 표시된다. 다른 한편으로 위의 그림에서 부서엔티티와 사원엔티티의 관계와 같이 모(parent) 엔티티의 Primary 키를 이루는 애트리뷰트들이 자(Child)엔티티의 Primary 키의 부분으로 상속되지 않는 경우를 Non-identifying Relationship 이라고 하며 연결관계를 나타내는 선을 점선으로 표시한다.

Categorization Relationship 의 경우에는 Generic 엔티티와 카테고리 엔티티들이 모두 같은 현실세계의 개체를 표현하고 있으므로 모든 카테고리 엔티티들의 Primary 키는 generic 엔티티의 Primary 키를 Categorization Relationship 을 통하여 항상 상속받는다. 그림 5-13 에서 우리는 모든 키테고리 엔티티들의 Primary 키는 Generic 엔티티인 회원엔티티의 Primary 키가 상속된 것임을 확인할 수 있다.

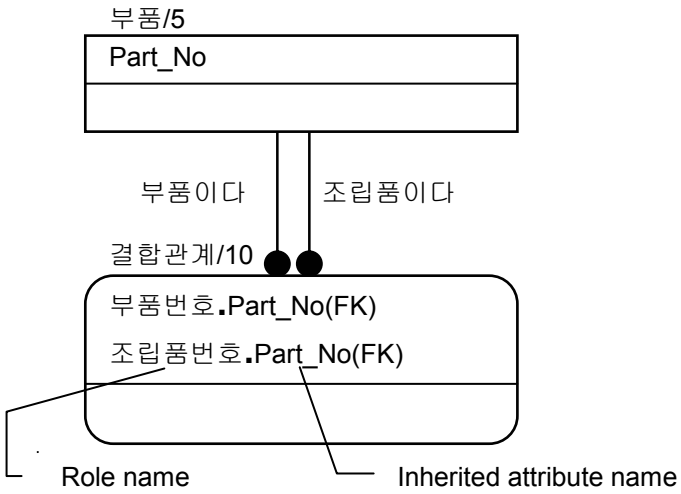


그림 5-17 : Role name 표현 형식

특정 경우에는 자(Child) 엔티티는 동일 모(Parent) 엔티티와 여러 개의 Relationship 을 가질 수도 있는데 모(Parent)엔티티의 Primary 키는 자(Child)엘티티의 각각의 Relationship 에 상속된 애트리뷰트로 나타난다. 자(Child) 엔티티에 주어진 인스턴스는 각각의 Relationship 을 위한 상속된 애트리뷰트의 값들이 다를 수도 있는다. 예를 들어 모(Parent) 엔티티의 두 인스턴스 값들이 연관될 수도 있다. 하나의 BOM(Bill Of Material) 구조는 부품 과 결합관계(하위레벨의 부품으로 조립된) 두 엔티티들로 표현된다. 여기서 부품 엔티티는 결합관계 엔티티에 대하여 모(Parent) 엔티티로서 두개의 Relationship 을 갖는다. 동일한 부품은 어떤 경우에 조립되어질 조립품의 부품으로 작용하기도 하고 어떤 경우에는 하나나 그이상의 부품이 조립된 조립품으로 작용하기도 한다. 부품 엔티티의 Primary 키는 Part-No 인데

Part_no 는 상속된 애트리뷰트로서 결합관계 엔티티에서 두번 나타난다.
하나의 애트리뷰트가 한 번 이상 상속되면 각각에 대하여 하나의 'Role name'을 할당해야 한다. 그림 5-17 에서는 부품번호와 조립품번호라는 Role name 을 Part_No 를 구분하기 위하여 사용하였다. 그러나 상속되는 애트리뷰트가 하나일 경우에는 상속된 엔티티의 애트리뷰트를 구분하기 위하여 Role name 을 굳이 표현하지 않아도 된다.

■ Foreign 키의 표현형식

Foreign 키는 엔티티 박스안에 상속된 애트리뷰트 뒤에 '(FK)' 라는 표시로 나타내어진다. 만일 상속된 애트뷰트가 Primary 키에 속하면 Primary 키 영역에 그렇지 않은 경우는 Parimary 키 영역의 경계를 나타내는 가로선 아래에 나타낸다. Role name 은 애트리뷰트와 마찬가지로 명사구로 나타내는데, 상속된 엔티티에서 애트리뷰트의 구분을 나타내기 위한 Role name 은 상속된 애트리뷰트 앞에 점으로 분리되어 표시된다. (그림 5-17 참조)

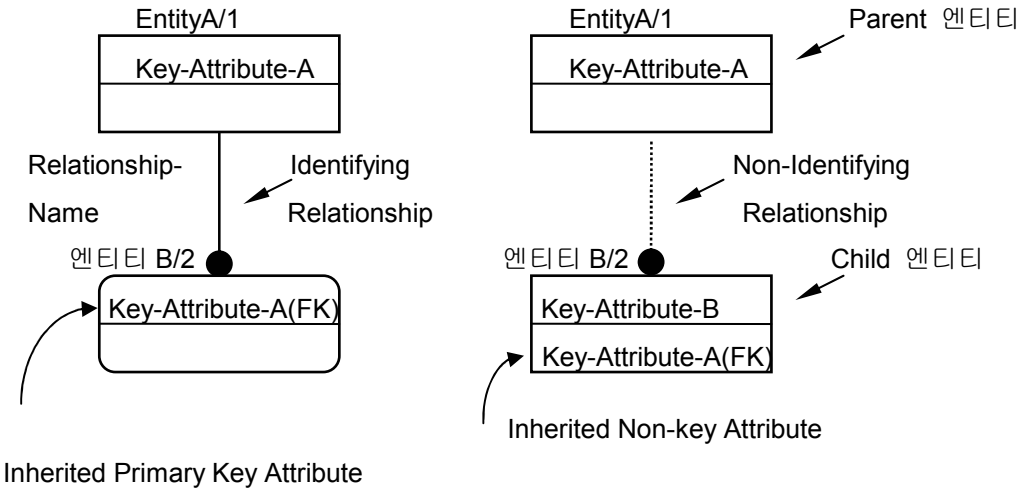


그림 5-18 : Foreign 키의 표현 형식

■ Foreign 키 규칙

① 모든 엔티티는 자(Child) 엔티티나 카테고리 엔티티일 경우 각각의 Specific Relationship 이나 Categorization Relationship 에 대하여 별도의 Foreign 키를 가져야 된다.
② 카테고리 엔티티에 대한 Primary 키와 마찬가지로 Generic 엘티티의 Primary 키도 상속되어야 한다.
③ 하나의 자(Child) 엔티티나 카테고리 엔티티의 각각의 인스턴스는 한 개 이상의 모(Parent) 엔티티나 Generic 엔티티의 Foreign 키로 적용될 수는 없다.

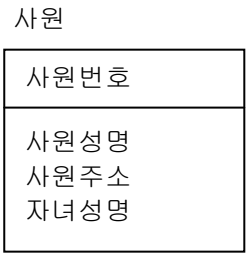
④ 모든 상속된 자(Child) 엔티티나 카테고리 엔티티의 애트리뷰트는 관련된 모(Parent) 엔티티나 Generic 엘티티에서 Primary 키로 사용되어야 한다.

⑤ 상속된 애트리뷰트에 적용되는 각각의 Role name 은 유일해야 하며 같은 의미는 항상 같은 이름으로 사용되어야 한다. 또한 같은 이름은 한상 같은 의미로 사용되어야 한다.

⑥ 모(parent) 엔티티나 카테고리 엔티티의 애트리뷰트는 하나 이상의 엔티티에 Foreign 키로 상속되어질 수 있는데 관련된 엔티티의 인스턴스에 항상 같은 값으로 제공되어져야 한다.

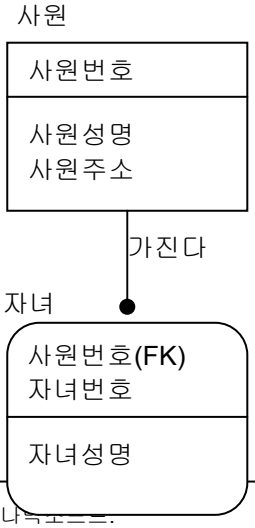
■ Normalization

① 반복 애트리뷰트 제거



사원

사원번호	사원성명	사원주소	자녀성명
1234	홍길동	서울특별시 영등포구 …	홍수철
1235	변강쇠	서울특별시 강남구 …	변인수, 변인자, 변인중
1236	박미자	경기도 성남시 분당구 ….	-
1237	조성숙	서울특별시 영등포구 …	홍수철



사원

사원번호	사원성명	사원주소
1234	홍길동	서울특별시 영등포구 …
1235	변강쇠	서울특별시 강남구 …
1236	박미자	경기도 성남시 분당구 ….
1237	조성숙	서울특별시 영등포구 …

자녀

사원번호	자녀번호	자녀성명
1234	1	홍수철
1235	1	변인수
1235	2	변인자
1235	3	변인중

② 동일 애트리뷰트의 중복사용

first normal form : 각각의 애트리뷰트는 각 인스턴스에서 정확하게 하나의 값만을 가져야 한다.

사원

사원번호
사원성명 사원주소 입사일-퇴사일

사원

사원번호
사원성명 사원주소 입사일-퇴사일 입퇴사 구분

사원

사원번호	사원성명	사원주소	입사일-퇴사일
1234	홍길동	서울특별시 영등포구 …	1992 년 3 월 2 일
1235	변강쇠	서울특별시 강남구 …	2000 년 6 월 5 일
1236	박미자	경기도 성남시 분당구 ….	1998 년 1 월 4 일
1237	조성숙	서울특별시 영등포구 …	1999 년 7 월 21 일

사원

사원번호
사원성명 사원주소 입사일자 퇴사일자

사원

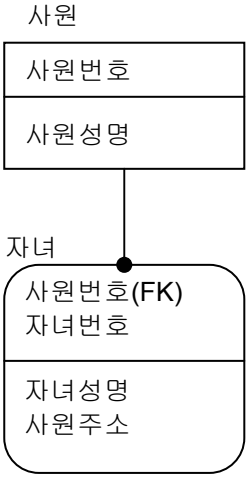
사원번호	사원성명	사원주소	입사일자	퇴사일자
1234	홍길동	서울특별시 …	1992 년 3 월 2 일	
1235	변강쇠	서울특별시 …	1986 년 1 월 4 일	2000 년 6 월 5 일
1236	박미자	경기도 …	1998 년 1 월 4 일	
1237	조성숙	서울특별시 …	1999 년 7 월 21 일	

③ 동일한 사실을 여러가지로 표현.

second normal form : 각각의 non-key 애트리뷰트는 엔티티의 전체 키에 대하여 종속적이어야 한다.

Third normal form : 엔티티의 모든 non-key 애트리뷰트는 키가 아닌 다른 애트리뷰트에 종속적이어서는 안된다.

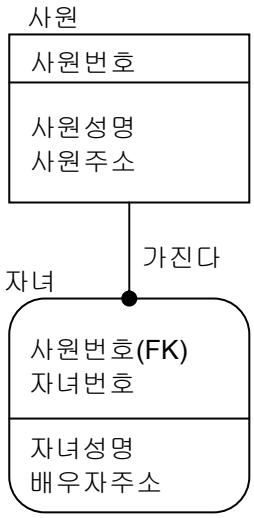
.



사원

사원번호
사원성명 사원주소 생년월일 나이

④ 사실의 상치

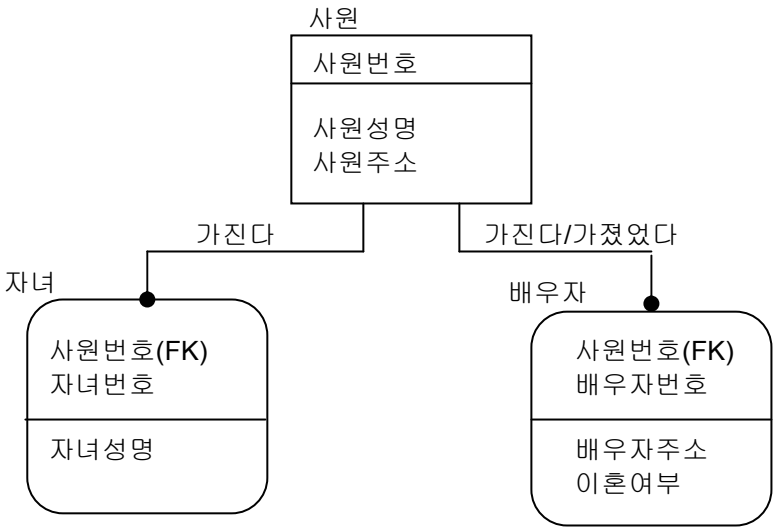


사원

사원번호	사원성명	사원주소
1234	홍길동	서울특별시 …
1235	변강쇠	서울특별시…
1236	박미자	경기도 ….
1237	조성숙	서울특별시 …

. 자녀

사원번호	자녀번호	자녀성명	배우자주소
1234	1	홍수철	서울특별시 …
1235	1	변인수	서울특별시 …
1235	2	변인자	서울특별시 …
1235	3	변인중	충청북도 …



사원

사원번호	사원성명	사원주소
1234	홍길동	서울특별시 …
1235	변강쇠	서울특별시…
1236	박미자	경기도 ….
1237	조성숙	서울특별시 …

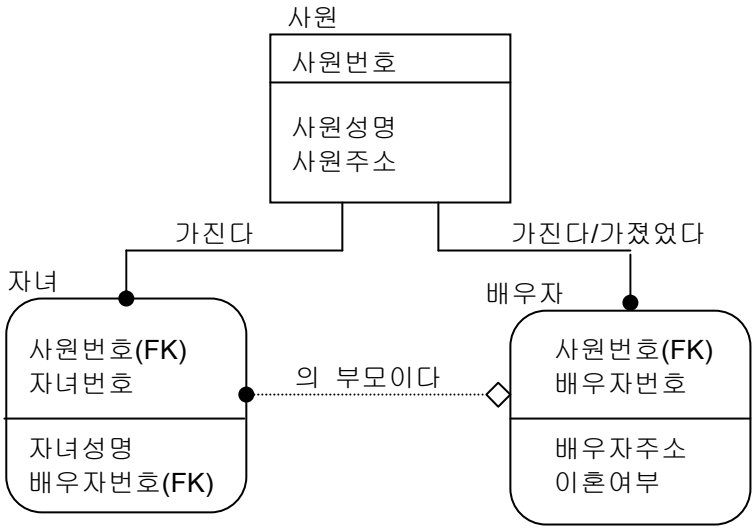
. 자녀

사원번호	자녀번호	자녀성명
1234	1	홍수철
1235	1	변인수
1235	2	변인자
1235	3	변인중

. 배우자

사원번호	배우자번호	배우자주소	이혼여부
1234	1	서울특별시 …	N
1235	1	서울특별시 …	Y
1235	2	충청북도 …	N
1236	1	경기도 …	N

⑤ 정보의 생략



사원

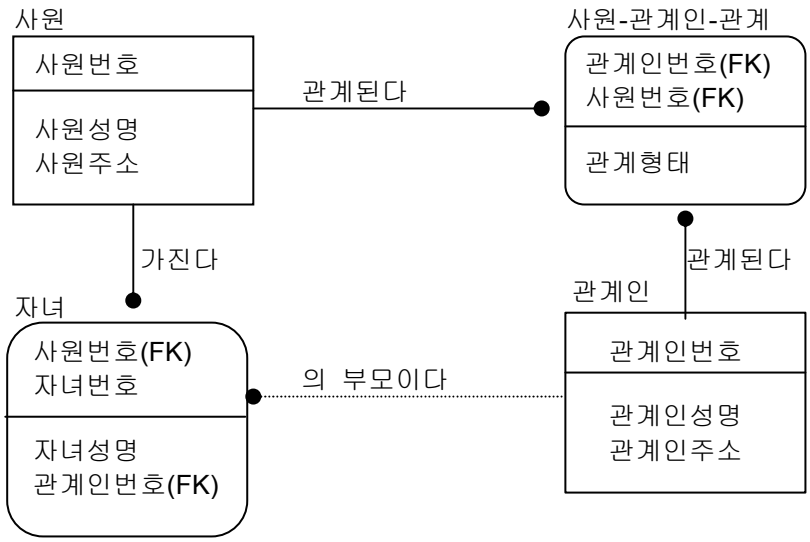
사원번호	사원성명	사원주소
1234	홍길동	서울특별시 …
1235	변강쇠	서울특별시…
1236	박미자	경기도 ….
1237	조성숙	서울특별시 …

. 자녀

사원번호	자녀번호	자녀성명	배우자번호
1234	1	홍수철	-
1235	1	변인수	1
1235	2	변인자	1
1235	3	변인중	2

. 배우자

사원번호	배우자번호	배우자주소	이혼여부
1234	1	서울특별시 …	N
1235	1	서울특별시 …	Y
1235	2	충청북도 …	N
1236	1	경기도 …	N



사원

사원번호	사원성명	사원주소
1234	홍길동	서울특별시 …
1235	변강쇠	서울특별시…
1236	박미자	경기도 ….
1237	조성숙	서울특별시 …

. 자녀

사원번호	자녀번호	자녀성명	관계인번호
1234	1	홍수철	1
1235	1	변인수	2
1235	2	변인자	2
1235	3	변인중	3

. 사원-관계인-관계

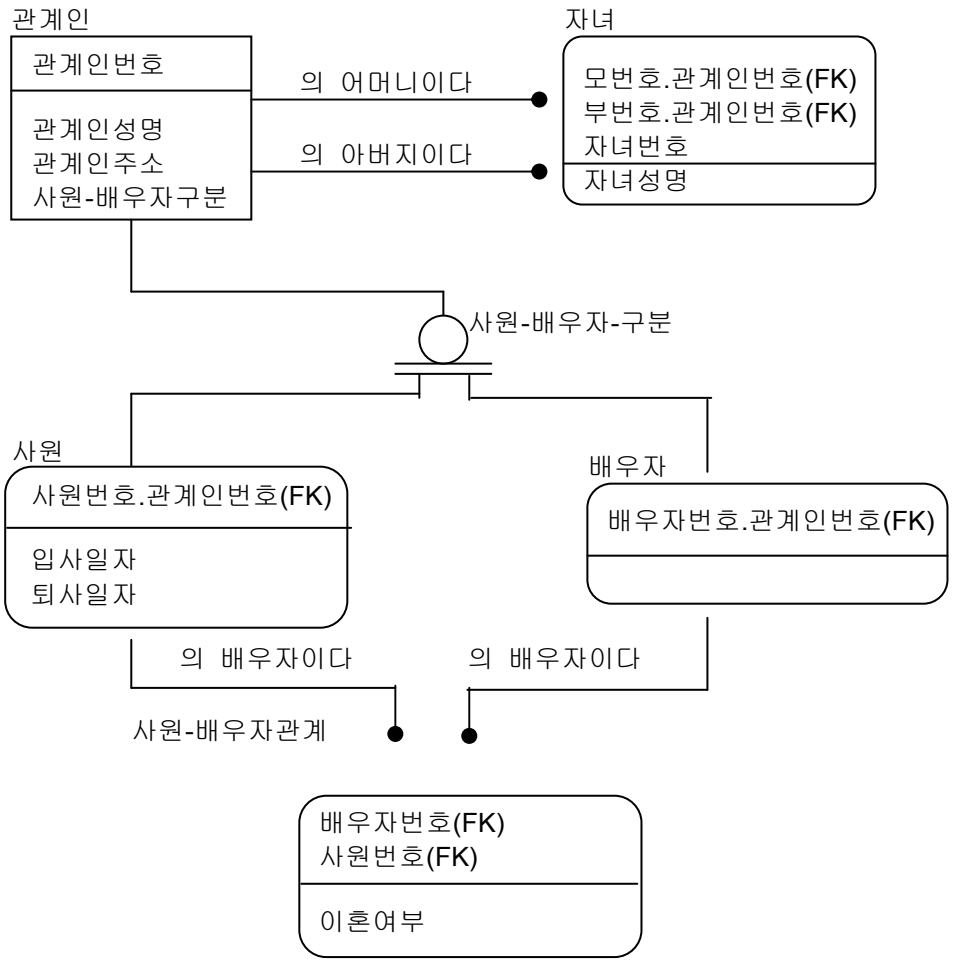
사원번호	관계인번호	관계형태
1234	1	배우자
1235	2	이전 배우자
1235	3	배우자
1236	4	배우자

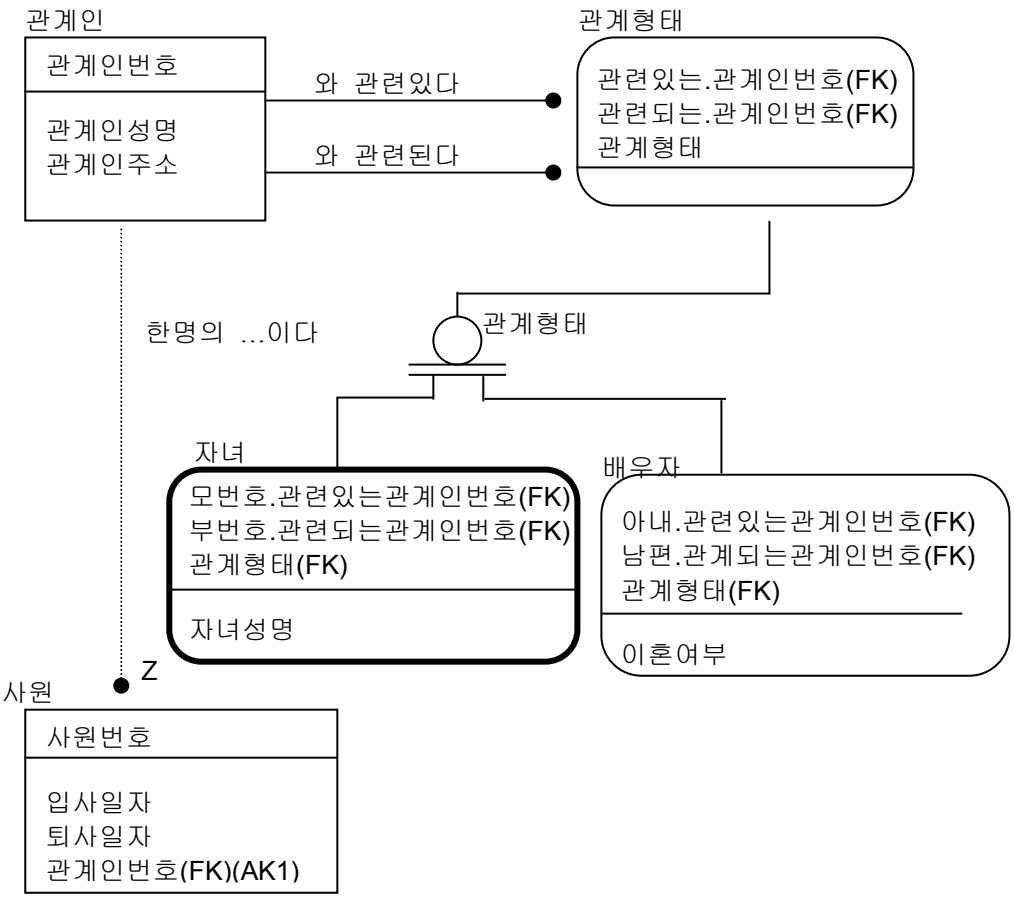
1237	5	배우자

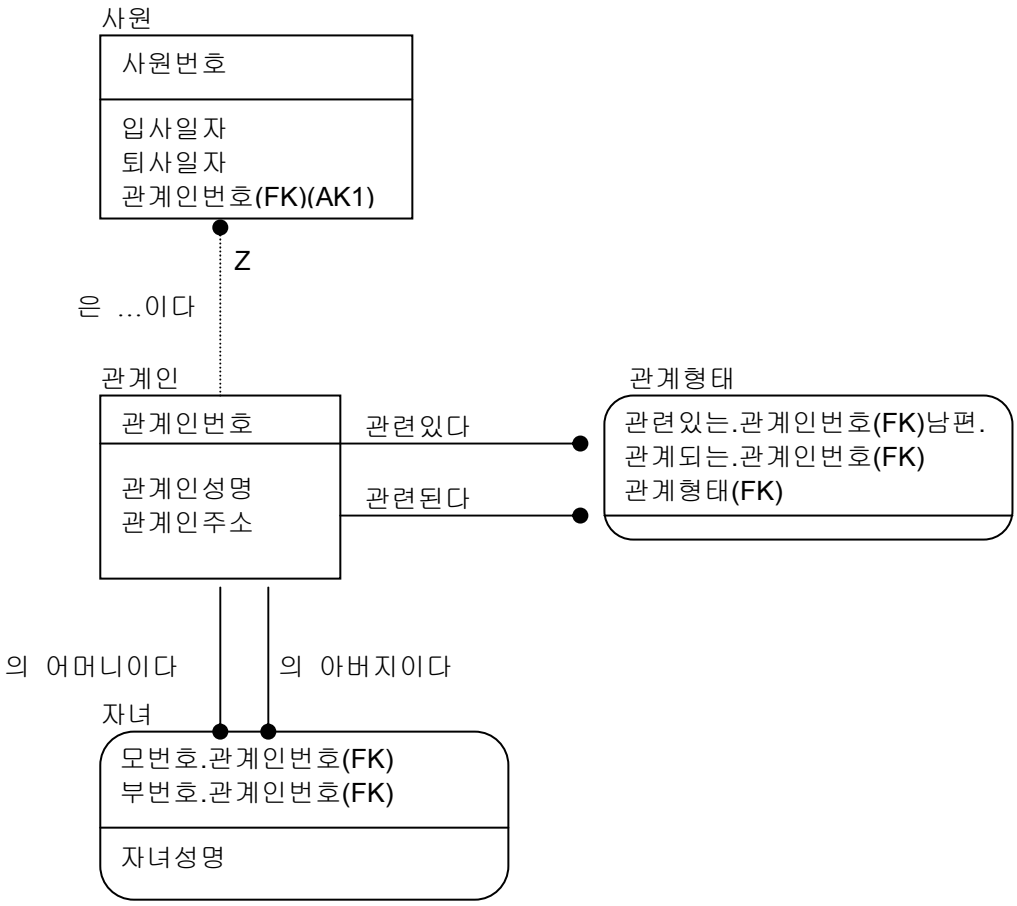
관계인

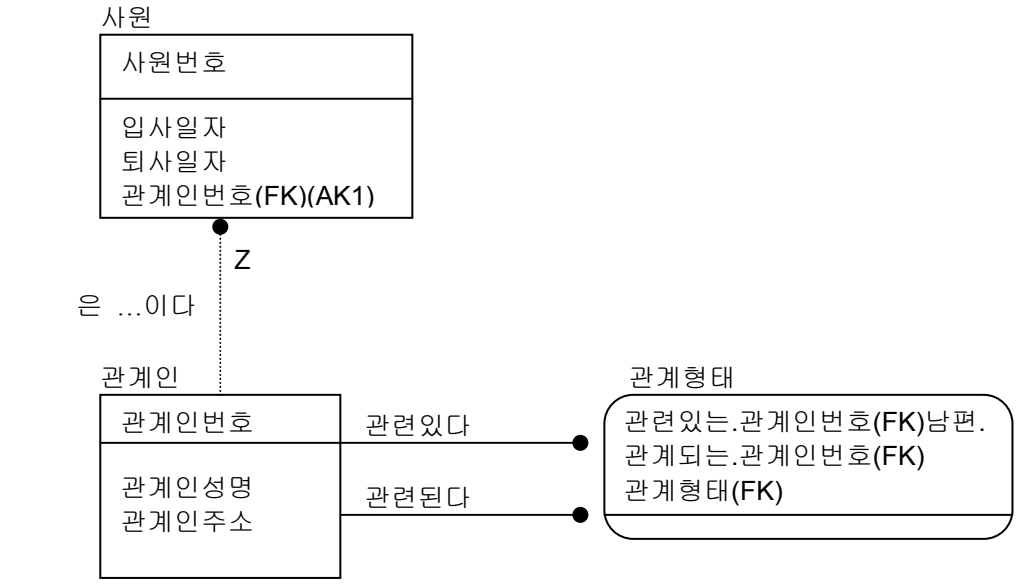
관계인번호	관계인성명	관계인주소
1	조성숙	서울특별시 …
2	옹녀	서울특별시 …
3	새옹녀	충청북도 …
4	김택수	경기도 …
5	홍길동	서울특별시 …

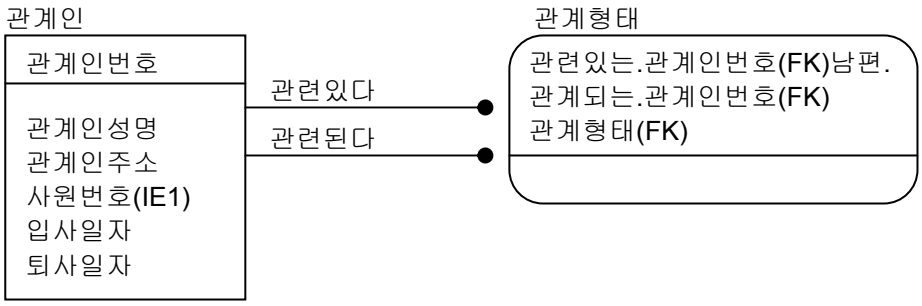
⑥ 옳지 않은 업무룰의 적용




관계인
관계인번호
관계인성명
관계인주소
와 관련있다
와 관련된다
관계형태
관련있는.관계인번호(FK)
관련되는.관계인번호(FK)
관계형태
한명의 ...이다
관계형태
자녀
모번호.관련있는관계인번호(FK)
부번호.관련되는관계인번호(FK)
관계형태(FK)
자녀성명
배우자
아내.관련있는관계인번호(FK)
남편.관계되는관계인번호(FK)
관계형태(FK)
이혼여부
Z
사원
사원번호
입사일자
퇴사일자
관계인번호(FK)(AK1)


사원
사원번호
입사일자
퇴사일자
관계인번호(FK)(AK1)
Z
은 ...이다
관계인
관계인번호
관계인성명
관계인주소
관련있다
관련된다
관계형태
관련있는.관계인번호(FK)남편.
관계되는.관계인번호(FK)
관계형태(FK)
의 어머니이다
의 아버지이다
자녀
모번호.관계인번호(FK)
부번호.관계인번호(FK)
자녀성명


사원
사원번호
입사일자
퇴사일자
관계인번호(FK)(AK1)
Z
은 ...이다
관계인
관계인번호
관계인성명
관계인주소
관련있다
관련된다
관계형태
관련있는.관계인번호(FK)남편.
관계되는.관계인번호(FK)
관계형태(FK)



5.3 IDEF1X 모델의 개발

데이터 모델링의 다섯 단계 과정은 다음과 같은 다섯 단계로 구성되어 있다.

① Context 를 정의한다.
② 엔티티 추출한다.
③ Relationship 정의한다.
④ 엔티티 키를 추출한다.
⑤ Non-key 애트리뷰트를 정의한다.

위의 모든 단계들이 반드시 순차적일 필요는 없다. 그러나 이러한 과정은 경험적으로나 기술적으로 모두 오류가 발생되는 것을 최소한으로 방지하도록 설계되었다. 이번 절에서는 이러한 각 단계에 대한 설명을 통하여 실제 프로젝트 수행에 필요한 절차를 알아보도록 하자.

5.3.1 0단계 : 배경(context)을 정의한다.

모든 모델링 프로제트에서 범위의 설정은 모델개발에 있어서 가장 큰 영향을 미친다. 우선 모델작업자와 프로젝트 관리자는 목적을 달성하기 위한 계획을 세워야 하는데 이러한 목적에는 다음과 같은 것들이 포함된다.

① 프로젝트 정의 – 무엇이, 왜, 어떻게 수행되어야 하는지에 대한 일반적인 설명.
② 원시자료 – 원시자료를 수집하기 위한 계획, 그리고 원시자료에 대한 색인과 파일링.
③ 모델링 작성규약 – 모델을 작성하고 관리하기 위하여 모델링 작업자가 선택한 규약에 관한 기본적인 선언

이들 산출물들은 다른 설명적이고 서술적인 정보들과 함께 0 단계 작업의 산출물이 될것이다.

■ 모델링 목표를 확립한다.

모델링의 목표는 다음과 같은 두가지 문장으로 구성된다.

① 모델의 사용과 관련된 목적에 관한 문장.
② 모델의 기능적 경계를 표현하는 범위에 관한 문장.

모델의 목표수립과 관련된 것들 중 가장 기본적인 것은 모델의 개발과 관련된 시간적 관계일 것이다. 즉 현재의 수행되고 있는 것(AS-IS)에 관한 것인가 혹은 향후 변화가 완료된 후(TO-BE)의 상태를 나타내기 위한 모델인가를 구분하는 것이다. IDEF1X 모델링 프로젝트를 위한 문제영역에 대한 일반적인 설명은 IDEF0 모델의 상세화 작업을 통하여 검토, 구성, 수정되는 작업을 포함할 것이다. 이러한 연유로 모델작업자와 프로젝트 관리자는 IDEF0 모델의 작성과 사용에 익숙해야 할 것이다. 일반적으로, IDEF0 모델이 이미 존재한다면, 이는 문제영역을 규명하는 기초가 제공된 것이다.
마찬가지로 데이터 모델링에 내포된 의도는 기업 전체를 지원하는 데이터 구조와 관련된 객관적인 관점을 확립하는 것인데, 중요한 특정 데이터를 밝혀내기 위하여는 각각의 모델이 확정된 범위를 갖는 것이 중요하다. 이러한 범위는 사용자의 종류(예를 들면 구배담당자 혹은 설계담당자)나 업무기능(작업일정 작성 혹은 설계도면 배포), 혹은 데이터의 종류(예를 들면 회계정보 혹은 제품형상정보)에 따라 달라질 것이다. 이러한 범위를 표현하는 문장은 목적을 나타내는 문장과 함께 모델링 목표를 구성한다. 다음은 모델링 목표를 기술한 예이다.

“이 모델의 목적은 제조라인의 감독자가 생산과 제품시험을 하기위하여 사용하는 현재의 (AS-IS) 데이터를 정의하기 위한것이다.”

또한 이러한 범위는 한 종류의 사용자만을 위한 것으로 제한될 수 있는데, 객관적인 모델을 개발하기 위하여 다른 사용자들의 관점도 모델링 작업에 반영되어야 한다.

■ 모델링 계획을 수립한다.

모델링 계획은 수행되어야 할 작업의 순서와 이러한 작업의 결과로 획득되어야 할 것들을 밝히는 것이다. 모델링 작업은 일반적으로 다음과 같은 절차를 포함한다.

① 프로젝트 계획
② 데이터 수집
③ 엔티티 정의
④ Relationship 정의
⑤ 키 애트리뷰트 정의
⑥ Non-key 애트리뷰트 추출
⑦ 모델 검증
⑧ 승인 검토

모델링 계획은 업무의 할당과 일정과 관련된 지침, 그리고 모델링 작업을 위한 예상 소요비용을 포함한다.

■ 팀을 구성한다.

모델의 가치는 특정의 절대적인 기준에 의해서 평가된다기 보다는 그 것을 가지고 의사소통 하는데 있어서 전문가와 초보자 모두에게 받아들여 질 수 있는 것인가에 달린 것이다. 이것은 두가지 메커니즘에 이해서 획득될 수 있는데, 첫 째는 모델을 개발한 전문가의 지속적인 검토가 각 전문분야의 특정 환경하에서 그 모델의 측정을 통하여 유효성을 지원한다. 둘 째는 모델에 대한 전문가 위원회와 일반인들의 정기적인 검토가 모델에 대한 상호 협력적인 합의를 지원한다. 모델링 과정에서 다양한 부서의 업무활동 방식에 있어서 불일치성을 발견하는 것은 어렵지 않은 것이다. 이러한 불일치성은 하나의 수용 가능하고 통합된 형태로서 기업 전체를 지원하는 데이터 모델로서 작성되어야 한다.
가능하다면, 모델 작성자는 모델이 무엇을 말하는지에 대하여 모델 자체로서 설명할 수 있어야 할 것이다. 모델 판독자가 임의로 상상할 수 있는 부분이 남아있어서는 안된다. 이러한 사항은 모델 작업자가 모델에 각각의 데이터 단위를 추가함에 있어서 심사숙고 함으로

써 모델의 판독에 있어서 불필요하게 추측될 수 있는 부분을 없앨 수 있다.
팀의 조직은 이와 같은 기본적인 원칙을 지원하고 프로젝트를 운용을 지원하기 위하여 구성되는데, IDEF1X 팀의 조직 구성원은 다음과 같은 다섯가지 역할로서 분류될 수 있다.

① 프로젝트 관리자
② 모델작업자
③ 원시자료
④ 각 분야별 전문가
⑤ 승인 및 검토 위원회

이와 같이 역할을 설정한 이유는 각각의 역할에 대한 책임을 명확히 하기 위한 것이다. 한 사람이 팀에서 한가지 이상의 역할을 수행할 수 있는 능력을 가지고 있을 수도 있다. 그러나 이러한 상황은 모델 구축과정에 있어서 다양한 관점을 반영하기 어렵고 매우 일방적인 관점만을 표현하는 빈약한 상태가 되어 결국 모델링 프로젝트의 목표를 부분적으로 밖에 수용하지 못하는 상태가 될 수 있다.
프로젝트 관리자와 모델작업자의 경우에는 원칙적으로 각각의 역할이 나뉘어져야 한다. 또한 모델작업자의 최종 목표는 검토위원회로부터 모델의 승인을 받는 것이지만 모델작업자는 프로젝트 관리자에게 보고서를 제출할 뿐 직접 검토위원회로부터 승인을 받는 것은 아니다. 이러한 방법에 있어서 모델작업자, 프로젝트관리자, 검토위원회 간의 의견 충돌이 있기 마련인데 프로젝트 관리자는 항상 이를 조정할 수 있는 위치에 있어야 한다. 그러나 여러가지 기술적인 사항에 대한 토의와 승인은 검증된 구성원에게 자동적으로 위임된다.
그림 5-19는 프로젝트에 있어서 프로젝트 관리자가 중심에 있는 프로젝트 구성원의 역할을 나타낸 것이다.

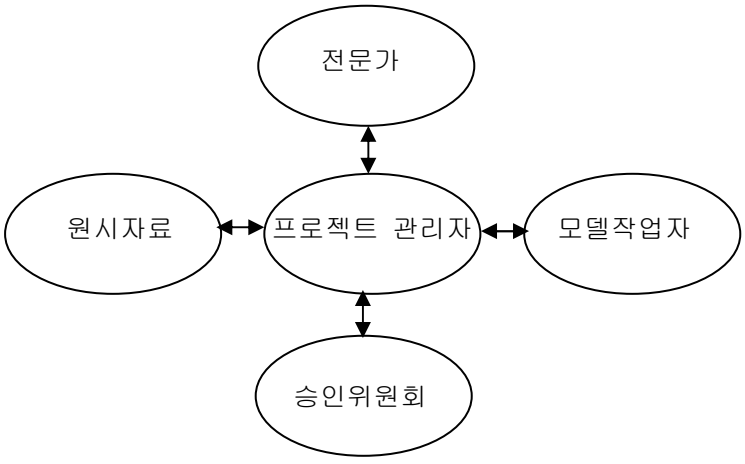


그림 5-19 : 팀의 구성

■ 원시자료를 수집한다

모델작업자들이 마주치는 가장 첫번째 문제는 어떠한 자료들을 수집해야 하고 또한 그 것들을 어디에서 수집할 수 있는가 하는 것이다. 우리는 종종 IDEF1X 모델의 범위와 상황을 IDEF0 기능모델의 분석에 의해서 얻을 수 있다. 기업 내부에서 대상이 되는 기능과 이에 관련된 개념들(ICOM)을 분석 함으로써 우리는 문제영역에서 사용되는 정보들을 파악할 수 있다. 대상이 되는 기능이 정의되고 관심있는 기본적인 정보의 범위가 선택되면, 기능에 관련된 각각의 정보들은 데이터 수집 과정을 통해서 수집될 수 있다. 이러한 데이터 수집은 관련된 지식을 가지고 있는 각각의 대상과의 인터뷰나 활동이나 문서, 정책, 절차 그리고 특정 정보모델의 검토 및 분석 등 여러가지 방법으로 수 가능하다. 이러한 대상 기능과 관련된 사항들이 파악되면 프로젝트 관리자는 모델 구축을 위한 원시자료로 사용될 각각의 혹은 특정 분야의 자료들을 정의할 수 있다. 원사자료는 조직 전체에 걸쳐서 다양한 형식으로 나타날 수 있는데 다음과 같은 것들을 포함하고 있다.

① 인터뷰 결과
② 관찰 결과
③ 정책이나 절차
④ 기존 시스템의 출력물(보고서 그리고 화면)
⑤ 기존 시스템의 입력물(데이터 입력 폼 그리고 화면)
⑥ 기존 시스템의 데이터베이스/파일 명세

■ 모델링 작성규약을 채택한다.

모델 작성규약은 모델 개발 및 모델 검토철 및 기타 표현에 있어서 지켜져야 될 것들과 모델 작업자에게 허용되는 정도를 나타내는 것들이다. 모델 작성규약의 목적은 자료 작성에 있어서 표현방법의 통일을 위한 것인데 이를 통하여 좀더 이해하기 쉽고 판독이 용이한 모델을 개발할 수 있다. 예를 들면, 엔티티나 애트리뷰트의 명칭의 부여에 관한 표준규약을 들수 있다.
작성규약은 다양한 부문에서 다양한 형태로 이용될 수 있는데 모든 모델링 작성규약이 취하지 말아야 할 중요한 사항은 다음과 같다.

① 모델링 작성규약은 기술적으로 공식적인 사항이 아니다.
② 모델링 작성규약은 기술적인 사항의 위반이 아니다.

모델링 작성규약은 각각의 필요사항을 지원하기 위하여 개발될 수 있다. 각각의 규약은 작성되는 즉시 문서화 되어 0 단계의 문서로서 포함되며 검토를 위하여 배포된다.

5.3.2 1단계 엔티티 정의

이 단계의 목적은 프로젝트의 다른 면을 보완하고 균형을 이루는 것이다
팀을 구성한다.(모델작업자, 전문가, 이해당사자)
사용할 데이터 수집기술을 결정하고 데이터 수집계획을 개발한다.
원시 데이터 리스트(Source Data List)를 정의하고 유지한다.
모델 규약을 만든다. Establish model conventions

5.3.2 Phase I: 엔티티 Class 를 정의한다.

예상되는 엔티티 클래스를 정의한다.
IDEF0 활동모델이나 IDEF3 프로세스모델과 관련된 문서들과 인터뷰 결과를 검토한다.
엔티티 클래스 ID 번호를 할당한다.
엔티티 클래스를 위한 용어해설을 정의한다.
엩티티 클래스를 정의하고 엔티티 클래스 동의어을 정의한다.
프로젝트 목적과 관련 있는 모델 엔티티에서 대상 엔티티를 분리한다.
엔티티를 정의한다.

다음은 엔티티 클래스 정의 리스트의 한 예이다.

엔티티 클래스	정의
예상 앤티티 리스트	사원, 컴퓨터, 정부기관, 부서, 제품, 훈령
사원	정부기관의 한 부서에서 근무하는 사람 동의어: 사람, 감독자, 작업자, 관리자
컴퓨터	타자기와 계산기로 이루어진 기계로 가능한 모든 기능을 다 사용하지 못함. 동의어 : PC, 바보상자, 멍청한 기계
정부기관	정부의 한 단위
부서	정부기관의 하위단위
제품	의회에 의해 필요한 종이서류 그러나 아무도 요청하는 이는 없음
지시	제품을 생산하기 위한 요구서

5.3.3 Phase II: Relation 클래스정의

엔티티 클래스간의 relation 을 정의한다.
다이아그램에 초점을 맞춘다.
Views
Entities/Relations Matrix.

Entity Relations Matrix 는 다음과 같이 나타난다.

	사원	컴퓨터	정부기관	부서	제품	지시
사원		할당된다	가진다	할당된다	생산한다	받는다
컴퓨터	x		소유한다	가진다	프린트한다	
정부기관	x	x		가진다	책임이 있다	받는다
부서	x	x	x		할당된다	의 결과이다
제품	x	x	x	x		반응한다
지시	x	x	x	x	x	

용어해설에서 각각의 relation 을 정의한다.
정보모델의 첫번째 다이어그램을 생성한다.
매트릭스에서 각 “행”을 위한 다이어그램을 생성한다. 혹은
중요한 엔티티 클래스를 선택하고 그 엔티티 클래스 주위에 관련된 엔티티 클래스를 그룹화한다.
엔티티 클래스간에 종속관계를 정의한다.

5.3.4 Phase III: 키 클래스를 정의한다

어떻게 하나의 엔티티 클래스 멤버가 키 클래스의 사용을 통하여 정의되는 지를 정의한다.
필요하다면 복합 키 클래스를 만든다..
키 클래스가 다른 엔티티에서 사용되기는 하지만 키 클래스로 사용되지 않는다면 키 클래스를 다른 엔티티에 migration 한다.
키 클래스 Migration
“기능 의존성”을 유지한다.
모델이 좀더 상세화 된 다음 단계로 전환되기 위해서
6 가지 기본 규칙
모델의 모든 사항이 사실인지 추적한다.
No Nulls 규칙을 검토한다.
No repeats 규칙을 검토한다.
하나의 애트리뷰트 클래스의 소유자는 오직 하나인가?
만일 키 클래스가 migration 되었다면 relation 은 오직 하나인가 ?
non-decreasing 사이클은 아니어야 한다.
Phase IV: 애트리뷰트 클래스의 일반화
애트리뷰트 클래스를 정의한다.
용어해설에서 애트리뷰트 클래스를 정의한다.
애트리뷰트 클래스의 “소유자”를 결정한다. 이 때 “소유자”인 엔티티 클래스는 애트리뷰트 클래스의 생성을 통제한다.

엔티티 클래스간의 연관관계 생성을 통하여 non-specific relation 을 specific relation 으로 변환한다.

품질의 조정 및 검토

엔티티 레이블들이 하나인지 검토한다.

Non-key attribute class 없이 엔티티를 검토한 후 상세 attribute 리스트와 함께 검토한다.

한 두개의 relation 에 관해 엔티티를 검토하고 나서 다른 많은 realtion 을 검토한다.

IDEF1X Model 을 완료하고 확정한다.

각각의 엔티티 클래스는 정확하게 하나의 의미를 가진다.

어떠한 두개의 엔티티 클래스도 동일한 의미를 가지지 않는다.

하나의 엔티티 클래스의 모든 멤버들은 정확하게 동일한 애트리뷰트 클래스를 항상 포함한다. (No Null Rule)

각각의 애트리뷰트 클래스는 오직 하나의 사실만을 표현한다. (No Repeat Rule)

두개의 애트리뷰트 클래스가 같은 의미를 가져서는 안 된다.

각각의 애트리뷰트 클래스는 정확하게 오직 하나의 엔티티 클래스에 의해 소유된다. (속한다) .

엔티티 클래스의 멤버들은 오직 전체(완벽한) 엔티티 키에 의해서만 구별되고 키가 없이는 아무런 의미도 가지지 못한다.

A. non-key attribute class 가 없는 엔티티 클래스는 전체 엔티티 클래스 키보다 적은 어떤 것에 의해서 정의가 가능하다는 것을 말한다. No Non-Key Attribute Class within an Entity Class represents a fact that is identifiable by anything less than the Entire Entity Class Key.

B. non-key attribute 가 없는 엔티티 클래스는 다른 non-key attribute 가 없는 엔티티에 관한 사실을 표현한다. No Non-Key Attribute Class within an Entity Class represents a fact about (is a Descriptor of) another Non-Key fact.

하나의 클래스와 짝을 이루는 엔티티 클래스의 연관 관계를 표현하는 업무 룰은 항상 완벽하다. Each business rule describing the Relationship Class between any pair of Entity Classes is always completely true.

하나의 짝을 이루는 엔티티 클래스들간에 있어서, 하나는 항상 Independent 엔티티이고 하나는 항상 dependent 엔티티이다.

Independent 엔티티 클래스를 이루는 모든 애트리뷰트 클래스들은 연관된 Dependent 엔티티 클래스에 각각의 짝을 이루는 relationship class 를 위해 정확하게 한번 씩 상속된다.

하나의 엔티티 클래스에 Non-Key Attribute Class 가 없는 경우, 그것이 상속되었건 엔티티 클래스 자체의 소유이건 간에, 다른 엔티티 클래스에 migration 된다.

하나의 엔티티 클래스 안에(key 혹은 non-key 이던) 애트리뷰트 클래스가 없는 것은 엔티티 클래스의 생성시 여러 값은 갖는다. None of the Attribute Classes in an Entity Class (Key or Non-Key) will ever by multi-valued (have multiple values) in an Entity Class occurrence.

하나의 엔티티 클래스 안에(key 혹은 non-key 이던) 애트립트 클래스가 없는 것은 엔티티

타입에 있어서 논리적으로 null 로 처리된다. None of the Attribute Classes in an Entity Class (Key or Non-Key) will ever be logically “Null” in an Entity Type occurrence.

Key Class 의 이전

Key class migration 은 독립적인 엔티티에서 종속 엔티티에 이르기까지 key class attribute 의 “상속”을 참조한다. attribute 의 migration 은 “functional dependency”를 유지시키며 모델을 다음단계의 레벨로 상세화 시킨다. Attribute migration 은 :

“Non-specific” Relation Class 문법이 확실한지가 우선 필요하다.

“Independent” 엔티티 Class 에서 “Dependent” 엔티티 Class 로의 전이를 보여준다.

“엔티티 Class”짝에 의하여 각 Frelation Class 는 공유가 발생한다.

어떤 것이 정의되지 않은(non-specific) relationship 인가 ?

non-specific 관계는 두 개의 엔티티간에 다대 다의 관계를 가리킨다.

카디날리티(cardinality)는 양방향에서 표현된다.

Relationship 은 두 방향에서 이름 붙여진다.

Specific relationship 이란 무엇인가 ?

Specific relationship 은 모자관계(parent-child) 내지는 종속관계(existency-dependency)를 가리킨다.

하나의 parent 는 child 의 인스턴스를 가지지 않거나(0), 하나(1) 내지는 여러 개(n)를 가진다.

Child 엔티티의 각 인스턴스는 관계를 가지는 parent 엔티티의 인스턴스가 존재하는 경우에만 존재할 수 있다.

Specific relationship 은 identifying 혹은 non-identifying 이 될 수 있다.

Identifying relationship 이란 child 인스턴스의 유일한 구분을 위해서는 관계된 parent 의 인스

턴스를 알아야 된다는 것이다. relationship 을 정의하는데 있어서 child 엔티티는 항상 종속된 엔티티이다.
Non-identifying relationship 은 child 인스턴스의 유일한 구분을 위해서 관계된 parent 인스턴스를 알아야 될 필요가 없다. Child 엔티티는 다른 엔티티와 identifying relationship 관계가 아닌 이상 identifyier-dependent 엔티티로 존재한다.

카테고리 Relation 이란 무엇인가?
카테고리 relation 은 여러분에게 그룹 특성이나 카테고리 엔티티를 하나의 통합된 엔티티로 구성한다. 여러분은 incomplete category 관계나 complete category 관계 양쪽 모두를 정의할 수 있다

5.4 SmartER 을 이용한 데이터 모델링

5.4.1 SmartER의 개요

이번 장에서는 SmartER 을 이용한 데이터 모델의 개발에 관하여 소개한다.
SMARTER 은 데이터 모델링과 정보 모델링 방법을 제공하는 소프트웨어 툴이다. SMARTER 은 모델링을 지원하기 위한 편리한 구조와 다양한 윈도우 형식으로 신속하게 엔티티, 애트리뷰트를 정의하고 그러한 엔티티를 사용하여 다이아그램을 생성할 수 있도록 지원한다.

■ 데이터 모델링을 목적은 무엇인가 ?

① 기업에서 관리해야 할 데이터를 확인한다.
② 기업이 관리하는 데이터 중에서 생략 가능한 부분을 확인한다.
③ 데이터베이스 구축하는데 사용되어질 설계모델을 개발한다.
④ 데이터베이스 구축을 위한 SQL 코드 생성한다.

■ 데이터 모델은 무엇인가 ?

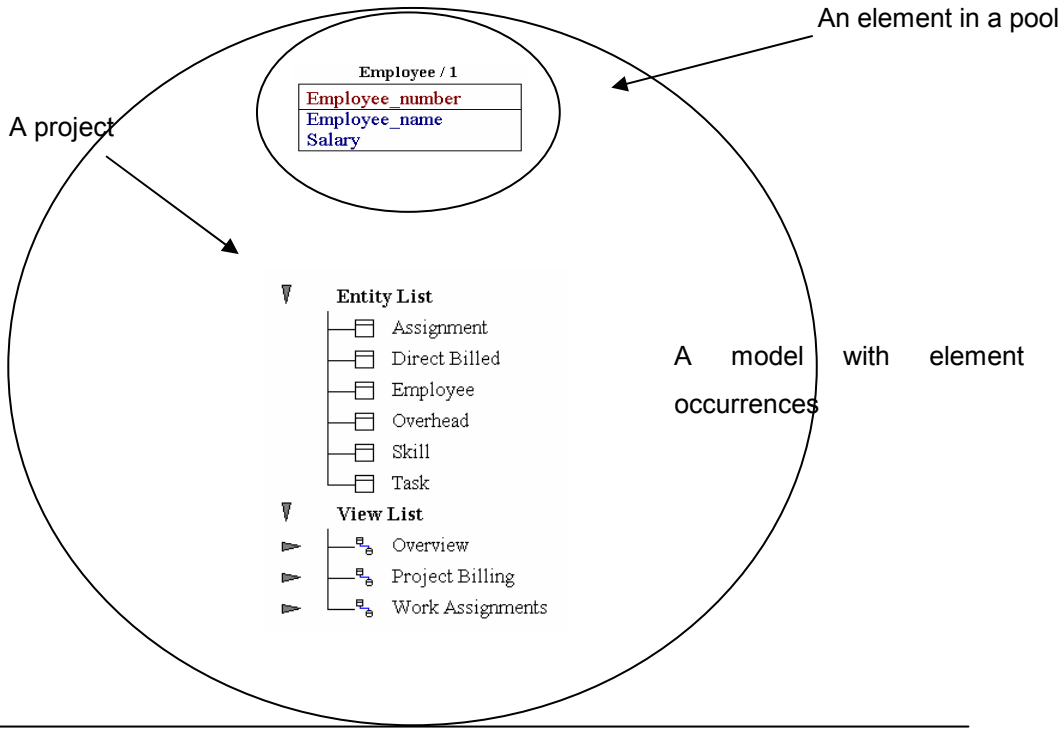
데이터 모델은 기존의 시스템이나 계획된 시스템에서 엔티티와 엔티티 간의 관계(relationship)의 표현이다. 데이터의 모델은 데이터베이스를 구축하기 위한 언어와 절차를 제공한다.

5.4.2 프로젝트, 모델, 그리고 다이아그램

이번 과정은 SmartER 에서 간단한 데이터 모델을 구성하기 위한 절차를 기술한다. 여러분은 제공되는 예제를 통해 모델을 구축하는 작업을 진행함으로써, SmartER 의 전반적인 특성과 그러한 특성들을 사용하기 위한 방법을 배울 것이다. 이번 절에서는 SmartER 에서 데이터 모델링 작업을 위한 기본 골격을 제공할 것이다. 여러분들은 프로젝트와 모델을 생성하고 문서화하는 과정을 수행할 것이다.

■ 프로젝트, 모델, 다이아그램이란 ?

SMARTER 로 데이터 모델링 작업을 수행하기 위하여는 우선 SMARTER 의 구조와 SmartER 에서 사용되는 용어에 익숙해야 한다. SmartER 에서 하나의 파일은 하나의 project 만을 포함한다. 각각의 프로젝트에는 하나의 풀(Pool)이 있고, 우리는 풀에다가 프로젝트를 수행하는데 있어서



사용되게 될 엔티티, 애트리뷰트, 노트(note), 소스(source), Unknown 등과 같은 정보를 등록할 수 있다. 즉 풀에 한번 등록된 각각의 엘리먼트들은 임의의 모델에서 참조되거나 사용되어질 수 있다. 하나의 프로젝트는 여러 개의 데이터 모델이나 정보모델을 포함할 수 있는데 SmartER 에서는 기본적으로 프로젝트와 관련된 작업이 각각의 모델 단위의 레벨에서 수행된다. 예를 들어 모델 노드리스트, 엔티티/엔티티 매트릭스, 그리고 엔티티 /애트리뷰트 매트릭스 등의 윈도우들은 모두 모델 레벨의 작업을 지원하는 기능들이다.

SmartER 에서 View 는 모델을 좀더 상세하게 볼 수 있는 기능을 제공한다. 프로젝트를 여러 개의 모델로 분할할 수 있는 것처럼, 하나의 모델은 여러 개의 view 를 포함할 수 있다. 모델의 특정 부분을 표현하는 활성화된 view 를 물리적 모델(physical model)이라고 한다. 여러분은 또한 이러한 여러 개의 물리적 모델들이 통합된 개념적 모델(conceptual model) 을 만들 수 있는데 이는 부분적인 물리적 모델에서 만든 모든 엔티티, 애트리뷰트, Relation 을 포함하는 통합 view(overview)이다.

SmartER 구조의 중요한 특징은 occurrence 개념을 적용한 것이다. Occurrence 란 풀에 등록된 각각의 엘리먼트를 각 모델에서 필요에 따라 복사하여 사용하는 개념으로 우리는 이를 통하여 프로젝트 전반에 걸쳐 한 번 풀에 등록된 엘리먼트들을 각각의 모델에서 여러 번 반복하여 사용할 수 있다. 예를 들면 모델에 새로운 엔티티를 추가할 때 풀에 등록되어 있는 엔티티 중에서 그 모델에서 필요한 엔티티의 Occurrence(복사본)를 만드는 것으로 새로운 모델에 필요한 엔티티를 생성할 수 있는 것이다. 일단 모델에 등록된 엔티티는 그 모델의 경계(context)내에서 유일해야 하며 중복되어서는 안된다. 만약 프로젝트 레벨(풀)에서 특정 엔티티를 제거한다면, 그 프로젝트에 안에 있는 모델에 복사된 각 엔티티는 동시에 제거된다.

■ 새로운 Project의 생성

모델구성 예제의 첫번째 단계는 예제모델을 구성하기 위해서 새로운 프로젝트 파일을 오픈하는 것이다. 여러분은 단지 새로운 파일을 오픈함으로써 새로운 프로젝트를 만들 수 있다.

toolbar 에서  클릭하거나 File 메뉴에서 New 를 선택한다.

새 프로젝트를 처음 오픈했을 때, 활성화된 윈도우는 Project 윈도우이다. 메뉴에서는 새로운 모델을 생성하고 프로젝트 풀에 엘리먼트를 추가하고 그 프로젝트를 문서화하는 등의 프로젝트 레벨과 과련된 기능을 수행할 수 있도록 지원한다. 새 프로젝트의 이름은 기본적으로 NONAME 으로 할당되는데 우리는 프로젝트를 저장하는 단계에서 할당된 이름을 바꿀 수 있다.

File 메뉴에서 save 혹은 saveAs 를 선택하거나 toolbar 에서  를 클릭함으로써 그 프로젝트의 이름을 저장한다. 이름을 바꾸기 위하여는 SaveAs 대화상자에 Name 필드에 새로운 이름을 입력하고 OK 를 클릭한다. 새로운 프로젝트의 이름이 status bar 에 나타날 것이다.

■ 프로젝트 문서화

모델 엘리먼트들을 추가하기 전에, 프로젝트를 문서화 함으로써 작업의 범위와 경계를 설정해야 한다. SmartER 의 문서화 기능은 여러분이 프로젝트, 각 모델, 그리고 각각의 모델 엘리먼트와 관계된 설명들을 문자로 기록할 수 있게 해 줌으로써 프로젝트에 표기된 도형적 엘리먼트를 보완할 수 있도록 지원한다. 이러한 기록들은 여러분이 수행한 작업을 나중에 참조하게되는 모델작업자에게 유용하게 사용된다.
우리는 프로젝트 문서화를 프로젝트의 범위와 목적을 설명하는 구문을 입력함으로써 시작한다.

① Project 메뉴에서 *Project Summary* ...을 선택한다.
Project Summary 대화상자는 여러분이 프로젝트에 이름을 할당하고 그 프로젝트를 생성한 담당자와 프로젝트가 사용될 곳을 기록한다. 또한 다이아로그박스는 여러분이 프로젝트에 대한 일반적인 설명을 포착하고 관련된 파일들을 첨부할 수 있도록 지원한다.

② Project Summary 대화상자에서 프로젝트 이름, 작성자, 사용될 분야를 입력한다. 각 필드로의 이동을 위해 <Tab>을 누른다..
Proeject Summery 에 입력되는 작성자 이름은 그 프로젝트에 포함된 모든 모델에서 생성되는 모든 엘리먼트로 상속된다. 그러나 여러분이 모델레벨에서 작성자를 편집 혹은 수정하는 경우 이는 Project Summary 에 영향을 주지 않는다.

③ Description 을 클릭한다. Project Description 대화상자는 여러분이 프로젝트의 범위와 목적관한 설명내용를 입력할 수 있게 해준다. 예를들어 :
이 프로젝트의 목적은 프로젝트 관리를 수행하는데 있어서 프로젝트 관리자에의해 사용되는 정보를 정의하기 위한 모델들을 구축하는 것이다.

④ Project Description 대화상자에서 OK 를 클릭하면 Project 윈도우로 돌아간다.

아직까지는 비록 프로젝트에 이름이 부여하고 이를 문서화 했지만, 프로젝트는 완전히 빈 상태이다.

■ 모델의 생성

이 시점에서 프로젝트 레벨의 윈도우는 아직 비어있는 상태이다. 이 윈도우에서 menu 와 toolbar option 은 모델을 추가하고 풀에 자료를 등록하는 것과 같은 프로젝트 레벨의 기능으로 한정된다. 우리는 모델을 생성하는 것부터 진행하도록 한다.

① Project menu 에서 *Open Model …*을 선택한다.
나타난 Models dialog 는 프로젝트에 있는 모든 모델을 리스트하고 새로운 모델을 생성하거나 기존의 모델을 오픈할 수 있게 해준다.

② Name 필드에 **Project Management** 를 입력한다. Window Type group box 에서 **Node** 를 활성화하고 Model Type drop-down 리스트에서 Database 를 선택한다. Create New 를 클릭한다. 대화상자에 새로운 모델이 리스트될 것이다.
모델 이름 옆의 FEO 는 그 모델이 For Exposition Only 이라는 것을 가리킨다.

③ 선택된 새로운 모델에서, **OK** 를 클릭하여 대화상자를 닫는다. 새로운 모델을 디스플레이하는 Model Nodelist 윈도우가 오픈될 것이다.

Open window 의 맨 윗부분의 title bar 와 모델 이름을 디스플레이하는 바닥부분의 status bar 를 참고한다. 또한 두개의 리스트 즉, Entity List 와 View List 가 윈도우에 나타나는데 모델의 문서화를 위한 텍스트 필드에서 아래와 같이 모델에 대한 viewpoint 와 purpose 를 정의한다.

■ 모델의 문서화

프로젝트에서와 마찬가지로 모델과 관련된 문서의 텍스트는 여러분이 모델의 내용을 설정하는데 도움이 되고, 여러분과 모델링 작업과 다른 팀 구성원들의 작업을 안내할 기준을 제공한다.

① Project 메뉴에서 *Model Summary …*를 선택한다.
Model Summary 대화상자에서 모델의 이름은 Name filed 에 나타난다. Revision Number 필드와 Review Status 드롭다운 리스트는 그 모델이 첫번째로 작성된 것이고 현재 작업중이라는 것을 가리킨다.

② Description 을 클릭한다. Model Description 대화상자에서 그 모델의 범위와 목적을 묘사하기 위한 text 를 입력한다.

이 모델의 목적은 사원들에게 할당된 업무와 투여 시간에 따른 급료를 지급하기 위하여 프로젝트 관리자가 사용하는 현재(AS-IS)의 데이터를 정의하기 위한 것이다.

③ 설명적인 텍스트를 저장하기 위해서 **OK** 를 클릭하고 Model Summary 대화상자로 되돌아간다.
Notes 와 Sources 를 클릭함으로써 모델에 대한 더 많은 정보를 추가할 수 있다. 또한 **Attachment** 를 클릭하여 모델과 관련된 다른 파일을 첨부할 수도 있다.

④ Model Summary 대화상자에서 **OK** 를 클릭한다.

이 시점에서도 비록 이름은 부여되었지만 모델은 거의 비어있는 상태이다. toolbar 를 사용하여 모델에다 view 를 추가한다.

■ View의 생성

① toolbar 에서  를 클릭한다. View 대화상자는 활성화된 모델에 새로운 view 를 생성할 수 있도록 해준다.

② Name 필드에 Work Assignment 를 입력하고 **Create New** 를 클릭하거나 <Enter>를 누른다. Name 필드의 Work Assignment 는 대화상자의 Viewlist 에 보여진다.

③ View List 에 view 를 추가하기 위해서 **Cancel** 을 클릭하고 Nodelist 윈도우에 남는다.

④ View 의 왼쪽  를 클릭함으로써 View List 를 확장한다.

모델작성을 계속하기 전에 SmartER 의 윈도우에 대해 배우기로 한다. 그리고 우리는 모델에 엘리먼트들을 추가할 것이다.

5.4.3 SMARTER의 윈도우

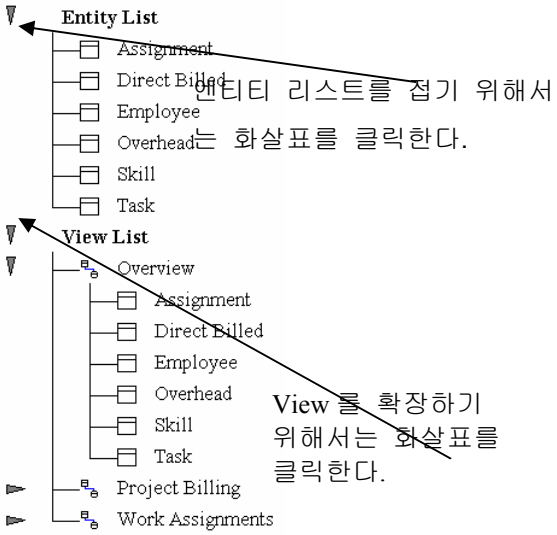
SmartER 은 데이터와 정보 모델링에서 엔티티, 애트리뷰트, Relation 을 빠르고 효과적으로 생성하기 위해서 네개의 윈도우를 지원한다. 이번 과정은 그러한 윈도우들과 그러한 윈도우들이 여러분의 모델 작성에 제공하는 기능들을 소개한다.

여러분은 모델 노드리스트 윈도우, 엔티티/엔티티 매트릭스 윈도우, 엔티티/애트리뷰트윈도우에서 하나의 전체 모델을 볼수 있다. View 윈도우는 모델에 등록된 엔티티간의 부분적 혹

은 전체적인 연관관계를 도형적으로 표현할 수 있도록 지원한다. 비록 각 윈도우가 각각의 특성과 구현능력를 상이하게 가지고 있다 할 지라도, 임의의 윈도우에서 수행된 수정이나 추가, 삭제사항은 모델 전체에 걸쳐 자동적으로 반영되고 각각의 다른 윈도우에도 반영된다.

■ 노드리스트 윈도우(Model Nodelist Window)

노드리스트 윈도우는 계층적 구조를 가지는 트리형식으로 이러한 형식내에 각각의 엔티티와 View 그룹을 리스트한다. 이 윈도우가 처음 오픈되었을 때에는 단지 노드리스트의 제목만이 디스플레이된다. 그 이름의 왼쪽에 있는 화살표를 클릭하여 트리를 확장할 수 있다. 확장된 노드리스트는 아래와 같이 보여진다.



Entity List 는 활성화된 모델에서 아이콘으로 표현되는 엔티티들을 보여준다. Nodelist Window 에서 엔티티들은 모델 내에서 임의의 view 로 드래그(drag) 혹은 드롭(drop)되어 질 수 있다. View List 는 모델에서 아이콘으로 표현되는 view 와 그 view 에 추가된 엔티티들을 보여준다.

■ 엔티티/엔티티 매트릭스 윈도우(Entity/Entity Matrix Window)

노드리스트 윈도우와 같이 엔티티/엔티티 매트릭스 윈도우는 프로젝트 레벨에서 모델을 표현한다. 그러나 그 초점은 특정 엔티티들 간의 관계들로 변한다. 매트릭스는 그 프로젝트 내에 있는 엔티티들간에 할당된 Relation 을 디스플레이하고 그러한 Relation 을 바꾸거나 추가할 수 있는 빠른 방법을 제공한다. 여러분은 두 엔티티간의 어떠한 Relation 도 생성할 수 있다.

Legend: 1:M = 0, 1, or More Rel. 1:P = 1 or More Rel. 1:Z = 0 or 1 Rel. 1:n = Exactly N Rel. M:N = Non-specific Rel. z:1 = Optional Rel. CC, IC = Categorization Independent Entities Dependent Entities Detached Entities **Entities**	**Entities**	Assignment	Direct Billed	Employee	Overhead	Skill	Task
Assignment						M:1	M:1
Direct Billed							CC
Employee						1:P	
Overhead							CC
Skill		1:M		P:1			
Task		1:M	CC		CC		

그러나 그 매트릭스 셀(matrix cell)은 한번에 하나의 선택된 Relation 만을 디스플레이할 것이다.

엔티티/엔티티 매트릭스는 엔티티와 엔티티간의 Relation 을 편집하고 생성하고 검사할 또다른 방법을 제공한다. Toolbar 버튼과 오른쪽 마우스 클릭의 조합은 Relation 을 편집하거나 지우거나 생성을 빠르고 용이하게 해준다. 매트릭스의 좌측 상단에 컬러로 표시된 글자는 엔티티와 Relation 을 빠르게 구분할 수 있도록 도와준다.

■ 엔티티/애트리뷰트 매트릭스 윈도우(Entity/Attribute Matrix Window)

엔티티/엔티티 매트릭스 윈도우처럼 엔티티/애트리뷰트 매트릭스 윈도우는 프로젝트 레벨에서 엔티티에 할당된 애트리뷰트를 편집하고 볼 수 있도록 해준다. 여러분은 각 엔티티의 애트리뷰트들을 demote(Primary 키 -> Alternate 키 -> Non-key 애트리뷰트의 순으로), promote(Non-key 애트리뷰트 -> Alternate 키 -> Primary 키의 순으로), 혹은 순환(cycle) 시킬 수 있고, 모델에서 각 엔티티를 위한 모든 애트리뷰트 타입을 전반적으로 파악할 수 이다. 엔티티/애트리뷰트 매트릭스에서 애트리뷰트는 화면의 윗편에 가로로 리스트되고, 엔티티는 화면의 왼쪽의 세로로 리스트된다. 애트리뷰트의 아래에 리스트된 숫자와 엔티티의 오른쪽에 리스트된 숫자는 SmartER 에 의해 할당된 애트리뷰트와 엔티티의 식별번호이다.

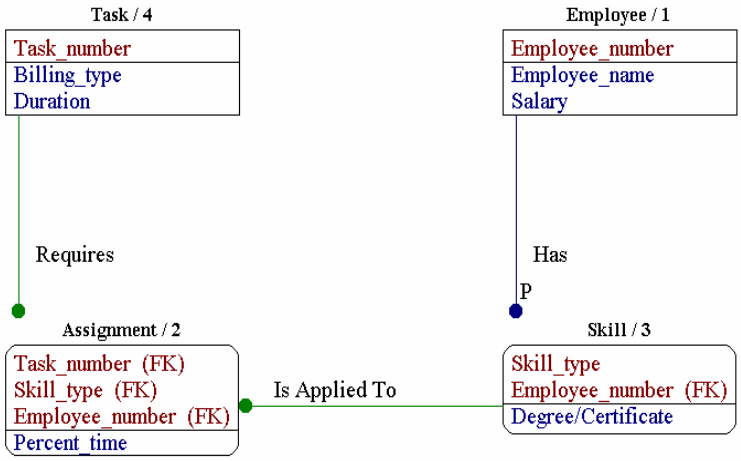
Legend:
P = Primary Key
A = Alternate Key
Desc. Attributes
O = Owned
M = Migrated
Independent Entities
Dependent Entities
Detached Entities

Entities	Attributes	Billing_type	Degree/Certificate	Duration	Employee_name	Employee_number	Percent_time	Salary	Skill_type	Task_number
		7	5	9	2	1	6	3	4	8
Assignment	2					MP	O		MP	MP
Direct Billed	6									MP
Employee	1				O	OP		O		
Overhead	5									MP
Skill	3		O			MP			OP	
Task	4	O		O						OP

엔티티/엔티티 매트릭스처럼 엔티티/애트리뷰트 매트릭스 윈도우는 toolbar 버튼과 오른쪽 마우스 클릭 메뉴의 조합을 제공한다. 그리고 엔티티와 애트리뷰트의 구분을 위해 컬러로 기호를 제공한다.

■ 뷰 윈도우(View Window)

뷰 윈도우는 일반적으로 물리적 모델(**physical model**)이라는 모델의 특정부분을 디스플레이 하기 위해 사용된다. 뷰 윈도우에서 릴레이션, 애트리뷰트, 엔티티들을 편집하거나 생성할 수 있다. 뷰 윈도우가 하나의 모델에서 동일한 엔티티에 대하여 서로 다른 여러가지 뷰를 제공하는 것을 기억하는 것은 아주 중요하다.

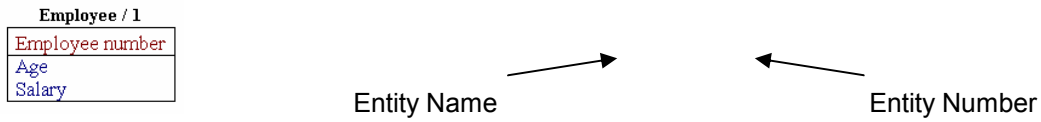


다음 과정은 IDEF1X 모델에서 첫번째로 고려해야 할 엘리먼트로서 엔티티에 대해 논의할

것이다. 계속해서 풀에 엔티티를 추가함으로써 모델을 구성해 나갈 것이다.

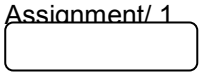
5.4.4 엔티티

IDEF1X 에서, 엔티티는 연관되어지는 데이터를 가지고 있는 어떤 것들의 집합을 표현한다. 이번 과정에서 새로이 생성한 프로젝트에 엔티티들을 추가할 것이다. 뷰 윈도우에서 박스는 모델의 타입에 따라 독립적인 엔티티(Independent 엔티티)와 종속적인 엔티티(Dependent 엔티티)를 다르게 표현한다. 엔티티의 이름, ID 번호는 데이터 모델박스 상단이나 정보 모델박스 안의 하단에 나타난다.



엔티티 ID 번호는 엔티티가 프로젝트에 추가될 때 SmartER 에 의해서 할당된다. 일단 ID 번호가 사용되면, 그 엔티티가 프로젝트에서 삭제될 지라도 ID 번호를 다시 사용할 수 없다.

네 모서리가 각진 사각박스는 데이터 모델에서 독립적인 엔티티를 표현한 것이고, 정보모델에서는 독립적인 엔티티나 종속적인 엔티티 중의 하나를 표현한 것이다. 참고로 정보모델에서는 독립적 엔티티와 종속적인 엔티티의 차이점이 그래픽적으로는 표현되지 않는다. 왜냐하면, 정보모델에서는 엔티티간에 primary 키나 alternate 키의 정의를 위한 키 애트리뷰트의 전이가 표현되지 않기 때문이다. 그림에서 보여지는 엔티티는 Task 라는 이름을 가지며 ID 번호는 4 이다.



데이터 모델에서, 라운드된 박스는 종속적인 엔티티이다. 데이터 모델에서 종속적인 엔티티는 자신의 Primary 키를 정의하기 위하여 전이된 다른 엔티티의 Primary 키의 일부분 혹은 전체를 참조해야 하는 종속적인 관계를 말한다. 즉 종속적 엔티티 인스턴스의 유일한 구분을 위해서는 최소한 일부분이라도 다른 엔티티의 Primary 키가 전이되어야 한다. 여기에 표시된 종속적 엔티티의 이름은 Assignment 이고 ID 번호는 1 이다.

SmartER 의 풀은 프로젝트 전반에 걸쳐 엘리먼트들(엔티티, 애트리뷰트, 소스, 노트)을 분배하기 위하여 저장하는 프로젝트 레벨의 저장소이다. 풀은 모델을 구성하는데 사용할 엘

리먼트를 미리 수집할 수 있게 해 줌으로써 "톱-다운" 방식으로 프로젝트를 구성하기 위한 메커니즘을 제공한다. 또한, 풀은 모델에 등록된 엘리먼트 전체를 수정하기 위해서 사용될 수 있다. 만약에 프로젝트에서 하나의 엘리먼트를 여러 모델에서 사용한 경우, 풀에서 해당 엘리먼트를 편집함으로써 각각의 이러한 여러모델을 한꺼번에 수정하는 것이 가능하다.

■ 엔티티 풀에 엔티티를 추가하기

① Pool 메뉴에서 *Entity...*를 선택하거나 키보드에서 **<Ctrl>+E** 를 타이핑한다.
Entity Pool 대화상자는 프로젝트에서 각각의 엔티티들을 리스트하고 프로젝트에서 엔티티를 사용하기 위한 관련된 여러가지 기능을 지원한다. 여러분이 모델에 직접 추가한 엔티티들도 또한 Entity Pool 에서 나타난다. 프로젝트에 임의의 엘리먼트를 추가했을 때, 관련된 풀에 자동적으로 엘리먼트가 저장될 것이다.

② Entity Pool 의 Name 필드에 **Assignment** 를 타이핑하고 **Create New** 를 클릭하거나 **<Enter>**를 누른다. 엔티티는 대화상자에 있는 엔티티 리스트에 추가된다.

③ Entity Pool 대화상자가 여전히 활성화된 채로, 프로젝트에 같은 절차로 다음과 같은 엔티티를 추가한다.

Employee **Skill** **Task**

④ Project Window 로 되돌아가기 위해 Done 을 클릭한다.

■ 풀에 등록된 엔티티를 모델에 추가

모델에 엔티티를 추가하기 위한 몇가지 옵션을 있지만 우리는 우선 Entity Pool 에서 엔티티를 신속하게 추가하기 위하여 SmartER 의 toolbar 명령을 사용할 것이다. SmartER 의 Quick Detailing 버튼은 다이어그램에 엘리먼트를 추가하기 위한 옵션을 제공한다. Toolbar 에서  토글 함으로써 Quick Detailing 이 활성화되었을 때, 다이어그램에서 직접 엘리먼트 이름을 할당하거나 SmartER 이 자동적으로 할당하는 이름을 가지고 엔티티, 애트리뷰트, 릴레이션을 새로이 추가할 수 있다. 이러한 옵션은 Quick Detailing 옵션 대화상자의 기능이다. Entity Pool 에서 엔티티를 추가한 다음에 수행할 첫번째 단계는 Quick Detailing 을 비활성화하는 것이다. Quick Detailing 을 비 활성화 함으로써 Entity Pool 에 있는 엔티티 리스트를 조회하기 위해서 toolbar 를 사용할 수 있다.

① toolbar 에서  토글 함으로써 Quick Detailing 을 비활성화 한다. 버튼이 인접한 버튼의 회색음영과 같을 때가 비활성화 된 것이다.

② 이제, 엔티티를 추가해보자
toolbar 에서  를 클릭한다. 엔티티 대화상자는 프로젝트에 있는 모든 엔티티들을 리스트한다. (즉, Entity Pool 에 있는 엔티티들)

③ 엔티티의 리스트를 선택하고 **OK** 를 클릭한다. 여러분은 첫번째 항목을 선택하고 <Shift>를 누른채 마지막 항목을 선택함으로써 여러 개를 한번에 선택할 수 있다. 모든 엔티티가 선택되었을 때 **OK** 를 클릭한다. 대화상자는 닫히고 노드리스트 윈도우로 되돌아간다.

노드리스트 윈도우에 있는 Entity List 의 왼쪽에 ▶가 있다는 점을 주목한다. 이 것은 엔티티가 리스트에 추가되었다는 것을 가리킨다. Entity List 에서 등록된 엔티티를 보기위해서 ▶를 클릭하거나 Entity List 를 선택하고 리스트를 확장하기 위해서 toolbar 에서  를 클릭한다.

자 이제 우리가 필요로 하는 엔티티를 가진 뷰를 만들어 보자. 뷰에 엔티티를 추가하기 위해서 우리는 드래그-앤-드롭(drag-and-drop)을 사용할 것이다.

■ 뷰에 엔티티를 추가하기

① Entity List 를 아직 확장하지 않았다면, 리스트 제목의 왼쪽에 있는 ▶ 클릭한다.
② <Shift>를 누른채 클릭함으로써 Entity List 의 모든 엔티티를 선택한다.
③ 선택된 그룹을 클릭하고 마우스 버튼을 누른 채로 View List 의 Work Assignment 아이콘으로 커서를 드래그한다.
④ 커서가  로 보일 때, 마우스 버튼을 놓는다.

지금까지 우리는 모델에 엔티티를 추가하고 뷰를 생성하고, 뷰에다 엔티티를 추가했다.

5.5.5 릴레이션(Relations)

엔티티나 애트리뷰트처럼 릴레이션도 뷰 윈도우에서 도형적으로 표현된다. Identifying, non-identifying, connection, categorization 과 같은 릴레이션의 다양한 타입이 이번 절에서 논의될 것이다. 또한 Work Assignment 뷰 간에 릴레이션을 정의하기 위해 엔티티/엔티티 매트릭스를 사용할 것이다.

■ Identifying 과 Non-Identifying 릴레이션

뷰 윈도우에서 엔티티와 엔티티를 연결하는 선들은 릴레이션이 존재한다는 것을 가리킨다. 선의 형태는 릴레이션이 identifying 인지 non-identifying 인지를 가리킨다.

① 실선으로 표시되는 릴레이션은 엔티티간의 연관관계가 identifying 릴레이션임을 나타낸다. Identifying 릴레이션에서 애트리뷰트가 전이된 엔티티는 dependent 엔티티이다. Primary 키가 전이될 때, 이들 Primary 키들은 dependent 엔티티에서 foreign key 가 된다.
② 점선으로 표시된 릴레잇ㄴ은 엔티티간의 연관관게가 non-identifying 릴레이션임을 표현한다. Non-identifying 릴레이션에서, 애트리뷰트가 전이되는 엔티티는 independent entity 이다. Primary 키가 전이될 때, 그 Primary 키는 관련된 엔티티에서 descriptive attribute 가 될 것이다.

■ Connection 릴레이션

하나의 Origin 엔티티 인스턴스는 0, 1 혹은 그 이상의 destination 엔티티의 인스턴스와 릴레이션을 가지거나 어떤 특정 수 만큼의 destination 엔티티 인스턴스와 릴레이션을 가질 수 있다. SmartER 에서는 적절한 릴레이션을 선택함으로써 릴레이션의 카디날리티(cardinality)를 정의 할 수 있다. 여기서 카디날리티는 각각의 origin 엔티티 인스턴스가 몇 개의 destination 엔티티 인스턴스를 가지는 가를 나타낸다.

여하튼, 데이터 모델을 구성할 때, 여러분은 다음과 같은 세가지 타입을 이용함으로써 카디날리티를 추출할 수 있다.

① 1:n 릴레이션은 숫자나 숫자의 범위중 하나로써 정수 “n”을 정의할 수 있게 해준다.
② M:N(non-specific) 릴레이션은 엔티티와 과련된 상세한 릴레이션을 지정할 때 까지 임시적으로 릴레이션을 유지할 수 있도록 해준다
③ Categorization 릴레이션은 generic, 혹은 origin 엔티티로부터 엔티티들의 그룹을 분류할 수 있게 해준다.

선의 색깔과 형태로 릴레이션을 표현하는 것에 더하여 카디날리티와 타입을 나타내기 위해서 숫자나 문자가 사용된다. 데이터 모델과 정보 모델을 지원하는 SmartER 의 각 릴레이션은 아래와 같다.

① 1 : M - 1 : 0, 1, 혹은 그 이상, 릴레이션에서 origin 엔티티의 하나의 인스턴스는 0, 1, 혹은 그이상의 destination 엔티티의 인스턴스를 갖는다.
② 1:P – 1 : 1, 혹은 그 이상 , 릴레이션에서 origin 엔티티의 하나의 인스턴스는 하나 혹은

그 이상의 destination 엔티티의 인스턴스를 갖는다.

③ 1:Z - 1 : 0, 혹은 1, 릴레이션에서 origin 엔티티의 하나의 인스턴스는 0 혹은 하나의 destination 엔티티 인스턴스를 갖는다.

④ 1:N – 1 : 특정의 양의 정수, 데이터 모델링에서 1:N 릴레이션은 origin entity 의 하나의 인스턴스가 정확히 N 개의 destination 엔티티 인스턴스를 가질 때를 나타낸다. 여러분은 하나의 숫자 값이나 숫자들의 범위 중 하나로써 "N"을 정의할 수 있다.

■ 임시적인 Connection 릴레이션

여러분은 요구되는 릴레이션의 타입이 무엇인지 결정될 때까지 모델에서 임시적인 Connection 릴레이션을 사용할 것이다. 정보모델 내지는 문서화만을 위한 목적으로 모델을 생성한다면, 임시적인 릴레이션을 재정의할 필요는 없다. 그러나, SQL 구문을 생성하기 위한 모델을 생성한다면, 데이터 모델이 SQL 구문으로 변환되는 것을 지원하기 위하여 connection 내지는 categorization relation 으로 temporary relation 을 재정의해야 한다.

M:N 릴레이션은 non-specific relation(data models) 내지는 다대다 릴레이션 relation(information models) 중의 하나로 불리어진다. M:N 릴레이션은 항상 non-identifying 일 것이다.

M:N 릴레이션에서, origin 엔티티의 M 개 인스턴스는 destination 엔티티의 N 개 인스턴스와 relation 을 가진다. 역으로의 릴레이션도 마찬가지이다. 여러분은 Edit Relation 대화상자에서, origin(M)과 destination(N) 카디날리티 둘다를 정의하 수 있다. 그러한 카디날리티를 정의하기 위해서 임의의 숫자나 숫자의 범위를 사용할 수 있다.

주의 : IDEF1 과 IDEF1X 모델링 룰은 하나의 origin 엔티티의 인스턴스는 destination 엔티티의 인스턴스와 0, 1, 혹은 그 이상의 릴레이션을 가진다는 것을 규정한다. 유효한 모델을 구성하기 위해서는, 모델이 포함하는 임의의 M:N 릴레이션을 재정의할 필요가 있다.

■ 카테고리 릴레이션(Categoty Relations)

카테고리 릴레이션은 specific 혹은 카테고리 엔티티를 하나의 generic 엔티티로 그룹핑할 수 있게 해준다. Origin(Generic) 엔티티에 있는 primary 키와 alternate 키는 destination 엔티티로 전이된다. 카테고리 엔티티가는 그 엔티티의 generic 엔티티로부터 primary 키를 상속하기 때문에, category 엔티티에 추가할 릴레이션은 반드시 non-identifying relation 으로 정의되어야 한다.

① IC – Incomplete Category(Data), 하나의 가로선으로 나타나는 incomplete category relation 은 categorization 이 아직 완성되지 않았다는 것을 가리킨다.

② CC – Complete Category(Data), 이중의 가로선으로 나타나는 complete category 는 categorization 이 완전히 완료되었다는 것을 가리킨다. 비록, 완전한 categorization 에 추가적으로 엔티티를 추가할 수는 있지만, categorization 이 완전히 개발되었다는 것을 확인할 때까지 incomplete categorization 상태로 표기하는 것이 바람직하다.

■ 릴레이션의 생성

Work Assignment 뷰에서 엔티티/엔티티 매트릭스 윈도우를 사용하여 릴레이션을 생성하도록 한다.

① 노드리스트 윈도우에서  를 클릭한다. 엔티티/엔티티 매트릭스 윈도우가 오픈될 것이다. 각 skill 이 하나 내지는 그 이상의 assignment 와 관련될 수 있다는 것을 나타내기 위해서 Skill 과 Assignment 간에 0 내지는 1 내지는 그 이상의 릴레이션(1:M)을 생성하고 시작할 것이다. Skill 은 origin 엔티티가 될 것이고, Assignment 는 destination 엔티티가 될 것이다. 왜냐 하면 여러분이 identifying 엔티티를 생성하기 때문에 destination 엔티티인 assignment 는 dependent 엔티티가 될 것이다(즉, 엔티티는는 independent 엔티티에서 전이되는 애트리뷰트에 의존한다.)
② 릴레이션 타입을 디자인하기 위해서 toolbar 에서  활성화하고 identifying 으로 표기하기 위해서  활성화한다.
③ Skill 을 표시하는 칸의 수평 축과 Assignment 를 표시하는 칸의 수직 축이 교차하는 지점의 셀(cell)을 클릭한다. 다시 말 해서, 릴레이션에서 특정 엔티티를 origin 엔티티로 지정하기 위해서는 클릭한 셀의 수평 축 왼쪽에 origin 엔티티가 있어야 한다. “1:M”라는 릴레이션 관계가 셀에 나타날 것이다. 또한, “M:1”은 수평축에 있는 Assignment 와 수직축에 있는 Skill 간에 교차지점을 표시하는 셀에 나타날 것이다.
④ 동일한 절차를 따라서, 다음 같이 테이블에 나타난 origin 과 destination 엔티티들을 사용하여 나머지 엔티티간의 릴레이션을 생성한다.

ORIGIN OR INDEPENDENT ENTITY	DESTINATION OR DEPENDENT ENTITY	Relation
Task	Assignment	1:M
Employee	Skill	1:P

릴레이션을 할당한 후에, 매트릭스는 아래 그림과 같은 형태로 나타날 것이다.

Legend: 1:M = 0, 1, or More Rel. 1:P = 1 or More Rel. 1:Z = 0 or 1 Rel. 1:n = Exactly N Rel. M:N = Non-specific Rel. z:1 = Optional Rel. CC, IC = Categorization Independent Entities Dependent Entities Detached Entities **Entities**	**Entities**	Assignment	Employee	Skill	Task
Assignment				M:1	M:1
Employee				1:P	
Skill		1:M	P:1		
Task		1:M			

매트릭스는 왼쪽에서 오른쪽으로 또는 윈도우 왼쪽에 수평축에 리스트 된 엔티티부터 수직축으로 리스트된 엔티티 순으로 "해석(read)"된다. 따라서, skill 과 assignment 의 1 : M 릴레이션은 "하나의 skill 이 0, 1 내지는 여러 개의 assignment 와 관련되어진다"라고 해석한다. 릴레이션 구문을 좀더 상세하게 설명하기 위해서, 우리는 정의할 각 릴레이션에 forward name 을 할당할 필요가 있다.

■ 릴레이션에 Forward Name 추가

릴레이션에 forward name 을 추가할 때, 뷰에서 forward name 은 관련된 릴레이션 옆에 나타난다. Forward name 은 origin 엔티티로부터 destination 엔티티의 순으로 읽혀지는데, 여러분은 이를 뷰에서 확인할 수 있다.

주의 : 여러분은 또한 모델의 판독에 도움이 되지 않더라도 Backward name 을 추가할 수 있다. Backward name 은 릴레이션을 반대방향으로 해독할 수 있도록 지원한다.

이제 forward name 에 대하여 알아 보도록 하자. 구성하고자 하는 모델에 forward name 을 추가한다. 우리는 이 작업을 뷰 윈도우에서 수행할 것이다.

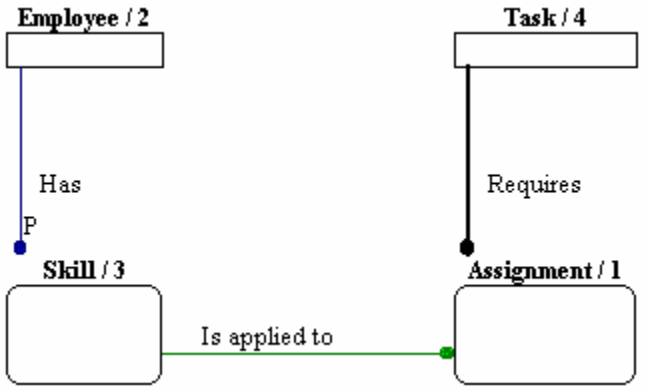
① 뷰 윈도우에서 Work Assignment 뷰를 오픈하기 위해서 toolbar 에서  를 클릭한다.
② Employee 와 Skill 간에 있는 1:P 릴레이션을 right-click 한다. Popup menu 에서 *Edit Relation ...*를 선택한다.
이 때 나타나는 Edit Relation 대화상자는 forward, backward, role name 을 지정할 수 있고, relation type 을 바꾸고, relation cardinality 를 지정할 수 있게 해준다(즉, 1:M relation 에서 정확한 M 의 수).

③ Forward Name 의 text field 에서 **has** 를 타입핑 한다. 이 릴레이션에서 정의된 룰은 **각 각의 종업원은 하나 내지는 그 이상의 기술을 가진다(has)**는 것이다. **OK** 를 클릭한다. "has"는 뷰 윈도우에서 릴레이션의 오른쪽에 나타난다.

④ 동일한 절차로 다음의 forward name 을 Work Assignment 뷰에 추가한다.

Relation	Forward Name	Read as:
1:M between Skill and Assignment	Is applied to	A skill is applied to 0, 1, or more assignments.
1:M between Task and Assignment	Requires	A task requires 0, 1, or more assignments.

릴레이션에 forward name 을 할당한 후에 뷰는 다음과 같이 보여진다.



다음 과정은 애트리뷰트와 하나의 애트리뷰트 전이에 대해 논의한다. 우리는 Working Assignment 뷰에 애트리뷰트를 추가하는 작업부터 시작할 것이다.

5.5.6 애트리뷰트와 애트리뷰트의 전이

애트리뷰트에는 전술한 바와 같이 primary 키 애트리뷰트, alternate 키 애트리뷰트, descriptive 애트리뷰트의 세가지 타입이 있다. "foreign 키 애트리뷰트"라는 용어는 다른 엔티티로부터 전이된 애트리뷰트말한다. 또한 original 엔티티에서 하나의 키로 나타나는 애트리뷰트를 말한다. 오직 키 애트리뷰트만이 다른 엔티티로부터 전이될 수 있기 때문에, 전이된 모든 애트리뷰트는 foreign 키 애트리뷰트이다. 여러분은 애트리뷰트를 디스플레이 하위한 옵션으로 display toolbar button ()들을 사용할 수 있다. 이를 토글을 함으

로써 애트리뷰트를 나타내지 않거나, 일부 애트리뷰트만을 나타내거나 혹은 전체 애트리뷰트 모두를 디스플레이 할 수 있으며 엔티티 박스가 모델에서 디스플레이 하고자 하는 애트리뷰트 정보를 조절할 수 있다.

이번 절에서는 Work Assignment 뷰에 애트리뷰트를 추가할 것이다.

■ Primary 키 애트리뷰트

Primary 키는 그 키가 엔티티에 등록된 모든 인스턴스를 유일하게 구분하는 애트리뷰트 이다. primary 키는 하나의 애트리뷰트나 혹은 여러 개의 애트리뷰트의 조합으로 이루어질 수도 있는데 Primary 키는 등록된 엔티티의 인스턴스를 구분하기 위해 필요한 애트리뷰트의 가장 작은 집합이다. 뷰 윈도우에서, primary 키 애트리뷰트는 박스 안에서 박스를 가로지르는 선의 상단에 나타난다.

■ Alternate 키 애트리뷰트

하나 혹은 여러 개의 집합으로 이루어진 Alternate 키는 엔티티에 등록된 모든 인스턴스를 Primary 키와 같이 유일하게 구분한다. 만일 하나의 엔티티에 하나 이상의 Primary 키가 있는 경우 그 중 하난를 Primary 키로 정의하고 나머지는 Alternate 키로 정의한다. 뷰 윈도우에서 Alternate 키는 박스를 가로지르는 선의 하단에 나타내는 데 Primary 키와 다른 점은 애트리뷰트 옆에 “(AK)”로 표기된다는 것이다.

■ Foreign 키 애트리뷰트

Destination 엔티티에서 foreign 키로 표시된 애트리뷰트는 다른 엔티티에서 전이된 키이다. 다시말해서, foreign 키로 표시된 애트리뷰트는 그 애트리뷰트의 origin 엔티티에서는 하나의 키로 나타난다. 또한 foreign 키 애트리뷰트는 애트리뷰트가 원래 속해진 Origin 엔티티와 destination 엔티티의 관계에 따라 destination 엔티티에서 다양한 역할의 엔티티로서 존재한다.
뷰 윈도우에서, 애트리뷰트 옆의 “(FK)”는 그 애트리뷰트가 foreign 키 애트리뷰트로서 다른 엔티티에서 전이되었다는 것을 가리킨다.

■ Descriptive 애트리뷰트

Descriptive 애트리뷰트는 키가 아닌 애트리뷰트 이다. Descriptive 애트리뷰트는 엔티티와 관련된 정보를 기술한다. 그러나 엔티티의 인스턴스를 구분하는데에는 어떤한 역할도 하지

않는다. 또한 descriptive 애트리뷰트는 엔티티간의 연관관계에 따라 전이되지도 않는다. 뷰 윈도우에서, descriptive 애트리뷰트는 반으로 나뉜 엔티티 박스의 하단에 나타난다.

■ 애트리뷰트 풀에 애트리뷰트 추가

① Pool menu 에서 애트리뷰트를 선택하거나 키모드로 **<Ctrl>+A** 를 누른다.
② 애트리뷰트 풀 대화상자의 name field 에 **employee_name** 을 등록하고 **Create New** 나 **<Enter>** 키를 누른다. 다이아로그의 애트리뷰트 리스트에 애트리뷰트가 추가될 것이다.
③ 애트리뷰트 풀 대화상자가 여전히 오픈된 채로, 동일한 절차로 프로젝트에 다음과 같은 애트리뷰트를 추가한다.

employee_number	task_number	Duration
skill_type	Salary	billing_type
degree_or_certificate	percent_time	

④ 뷰 윈도우로 되돌아가기 위해서 **Done** 을 클릭한다.

■ 엔티티에 애트리뷰트 추가하기

우리는 계속해서 엔티티에 키 애트리뷰트를 추가하기 위해 뷰 윈도우를 이용할 것이다. 하나의 Primary 키 – 하나 혹은 여러 개의 애트리뷰트로 이루어진 - 는 엔티티의 각 인스턴스를 유일하게 구분하는 집합이다. 우리는 이미 애트리뷰트 풀에 애트립트를 등록하였기 때문에, right-click menu 를 이용하여 이미 등록된 애트리뷰트를 모델의 엔티티에 추가할 것이다.

① Employee 를 right-click 한다. Popup menu 에서 *Add Attribute*…를 선택한다.
이 때 나타나는 애트리뷰트 대화상자는 엔티티에다 기존의 애트리뷰트를 추가하거나 새로운 애트리뷰트를 추가할 수 있게 해준다. 또한 애트리뷰트를 단지 그 모델안에 있는 엔티티에만 추가할 것인지 엔티티 풀에 등록된 엔티티에도 이를 반영할 것인지를 정의한다.

여러분은 모델에 있는 엔티티에만 애트리뷰트를 추가할 수도 있고 혹은 이와 함께 프로젝트 레벨의 풀에 등록된 엔티티에도 애트리뷰트를 동시에 추가할 수도 있다. 풀에 등록된 엔티티에 같이 애트리뷰트가 추가되는 경우 추가된 애트리뷰트는 모델에서 사용되는 모든 엔티티에 반영될 것이다.
② List filed 에서 employee_number 를 선택한다.

③ **Add to the Entity in the Pool** radio button 을 활성화해서 그 애트리뷰트가 프로젝트 안의 어디에서 사용되던 상관 없이 사용되는 모든 엔티티에 반영될 수 있도록 한다.
④ 대화상자를 닫기 위해서 **OK** 를 클릭하고 뷰 윈도우로 되돌아 간다.

엔티티에 애트리뷰트를 추가한 후에, 지금까지 추가해온 애트리뷰트의 종류(primary 키, alternate 키, descriptive 키)를 분류할 필요가 있다.

■ 애트리뷰트 타입의 표기

① Employee 를 right-click 한 후 popup menu 에서 *Edit Entity...*를 선택한다.
이 때 나타나는 모델 대화상자의 Edit Entity 는 한번에 dekdmarhk 같은 여러가지를 수행할 수 있도록 지원한다. 엔티티의 name, alias, 작성자를 변경하기; 관련된 애트리뷰트를 promoting 하거나 demoting 하기; 엔티티로부터 애트리뷰트를 추가, 편집, 삭제하기; 엔티티에 설명적인 텍스트와 노트, 소스의 추가 혹은 다양한 alias 를 엔티니에 추가하기와 같은 여러가지를 수행할 수 있도록 지원한다.

② 애트리뷰트 리스트에서 employee_number 를 선택하고 employee_number 가 PKA 즉 primary 키 애트리뷰트로 promote 될 때까지 Promote 를 클릭한다.

Promote 나 Demote 를 클릭하면 애트리뷰트의 타입이 Des(descriptive 애트리뷰트), AKA(alternate 키 애트리뷰트), PKA(primary 키 애트리뷰트)를 순환한다.

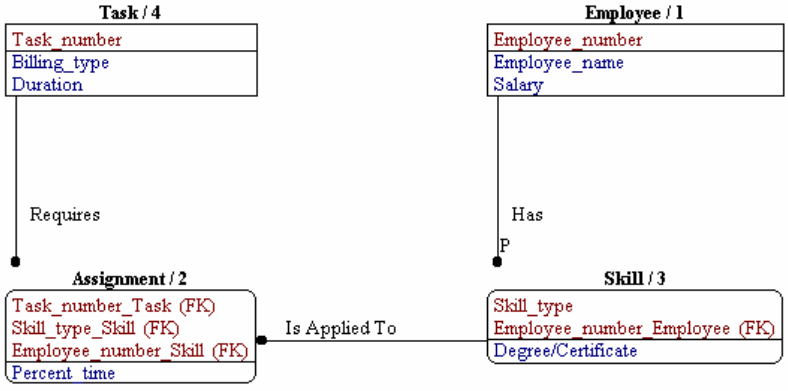
③ OK 를 클릭하여 대화상자를 닫고 뷰 윈도우로 되돌아간다.

자 이제 남아있는 엔티티에 descriptive 애트리뷰트와 primary 키를 추가해 보도록 하자. 각 애트리뷰트를 추가할 때 애트리뷰트 대화상자에서 **Add to the Entity in the Pool** 을 활성화한 상태로 해야한다는 것을 주의해야 한다. 위에서 수행한 절차를 따라서 아래에 열거된 엔티티에 다음과 같은 애트리뷰트를 추가한다. 필요할 경우 Promote 를 이용한다.

Entity	Attribute to Add	Attribute Type
Employee	Employee_name	Descriptive
	Salary	Descriptive
Skill	skill_type	primary key
	degree or certificate	Descriptive
Assignment	percent_time	Descriptive
Task	task_number	primary key

Entity	Attribute to Add	Attribute Type
	duration	Descriptive
	billing_type	Descriptive

여러분이 추가한 Descriptve 애트리뷰트를 보기 위해서는 뷰 윈도우의 toolbar 에 있는  를 클릭한다. 일단 primary 와 descriptive 애트리뷰트를 추가하면 Work Assignment 뷰는 다음과 같이 나타나야 한다.



Employee 에서 Skill 과 Assignment 로 전이된 employee_nmber 을 주목하자. 모델에서 보여진 것 처럼 Skill 과 Assignment 에서 employee_number 가 primary 키 애트리뷰트로 작용한다. **FK** 는 foreign 키 애트리뷰트를 의미한다. 이것은 Employee_number 가 origin 엔티티에서 키 애트리뷰트라는 것을 의미한다. Descriptive 애트리뷰트는 그 엔티티의 인스턴스를 identify 하는데 어떠한 역할도 하지 않으며 또한 dependent 엔티티로 전이되지 않는다는 것을 상기하자.

우리의 모델을 한 번 살펴보면, 프로젝트 비용 계산과 관련된 데이터를 저장하고 이를 상세하게 살펴보기 위하여 또 다른 뷰가 필요하다는 것을 알 수 있다. 따라서 우리는 다음 단계에서 Project Billing 이라는 뷰를 만들 것이다.

여러분이 모델링하는 실세계 시스템에서, 용역 프로젝트 수행과 관련된 각각의 Task 에 대한 비용청구는 는 직접비용과 간접비용으로 나뉘어져 계산된다. 직접비용과 간접비용은 Task 의 비용계산(billing) 카테고리를 나타낸다. 이러한 릴레이션을 표현하기 위해서, 우리는 이와 같은 상황을 반영하기 위하여 Task 는 generic(origin) 엔티티로 Overhead 와 Direct Billed 는 category(destination) 엔티티로 표현되는 category relation 을 만들 것이다. 이 두가지 카테고리는 기업에서 일반적으로 프로젝트 비용을 계산하는 두 가지 경우를 모두 포함하기 때문에 Complete Categorization 이 될 것이다.

■ 또 다른 뷰의 추가

노드리스트로 돌아기기 위하여 toobar 에서  를 클릭한다. 노드리스트 윈도우에서 프로젝트의 추가적인 요소들을 구성하도록 하자.

① 뷰를 추가하기 위해서 toolbar 에서  클릭한다.
② 이 때 나타나는 뷰 대화상자에서, name field 에 **Project Billing** 을 등록하고 **Create New** 를 클릭한다.
③ 새로운 뷰를 선택하고, 뷰 윈도우에서 Project Billing 을 오픈하기 위해서 **OK** 를 클릭한다.

일단 Project Billing 뷰 윈도우에서, 프로젝트에 새로운 엔티티를 추가하기로 한다.

■ 뷰에 엔티티를 추가하기

① toolbar 에서  클릭한다.
뷰 대화상자에서 Add Entities 를 수행하면 뷰 대화상자는 이미 모델에 존재하면서 해당 뷰에 “소속된” 엔티티들을 보여준다. 또한 뷰 대화상자는 새로운 뷰를 위한 엔티티를 생성하고 기존의 엔티티를 수정할 수 있게 해준다.

우리는 우선, 두개의 엔티티를 생성하고 뷰에다가 두개의 엔티티를 추가할 것이다.

② **Add to Model** 을 클릭한다.
③ Entities 대화상자에서, name field 에 **Overhead** 를 타입핑하고, **Create New** 를 클릭하거나 **<Enter>** 키를 누른다. 동일한 절차에 따라서, 모델에 **Direct Billed** 를 추가한다.
④ 엔티티들을 선택하고 **Ok** 를 클릭한다. 두 엔티티는 Already Owned filed 에 나타날 것이다. Already Owned field 는 이들이 이미 통합 뷰에 추가되어졌다는 것을 가리킨다.

이미 모델에 등록되어 있는 엔티티를 뷰에 추가하자.

⑤ Model 리스트에 있는 엔티티들 중에서 Task 를 선택하고  클릭함으로써 뷰에 엔티티를 추가한다. 뷰 대화상자에서 Add Entities 를 종료하기 위해서 **Done** 을 클릭한다. 그리고 뷰 윈도우로 되돌아간다

이제 우리는 Project Billing 뷰에 Task, Overhead, Direct Billed 를 추가했다. 우리는 이들을 연결하는 릴레이션을 생성하도록 한다.

■ Categorization 생성

① Project Billing 뷰 윈도우의 toolbar 에서  를 클릭한다.

Add Relation 대화상자는 모델 안에 있는 엔티티들을 리스트한다. origin 과 destination 을 선택하고 릴레이션 카디날리티를 설정함으로써 엔티티들 간의 relation 을 만들 수 있다.

② Cardinality group box 에서 **Complete Categorization** radio buttton 을 활성화한다.

③ Origin list 에서 Task 를 선택한다. 엔티티들을 선택하는 동안 <Ctrl> 누른 채로 Destination list 에서 Direct Billed 와 Overhead 를 다중 선택한다.

④ **Create Relation** 을 클릭한다. Info 대화상자가 나타나고, relation 이 성공적으로 생성되었다는 것을 말해준다.

⑤ 대화상자를 닫기 위해서 **Done** 을 클릭한다.

다음은, 데이터의 구성을 좀 더 잘 반영하기 위해서 categorization 에 discriminating 애트리뷰트를 지정한다.

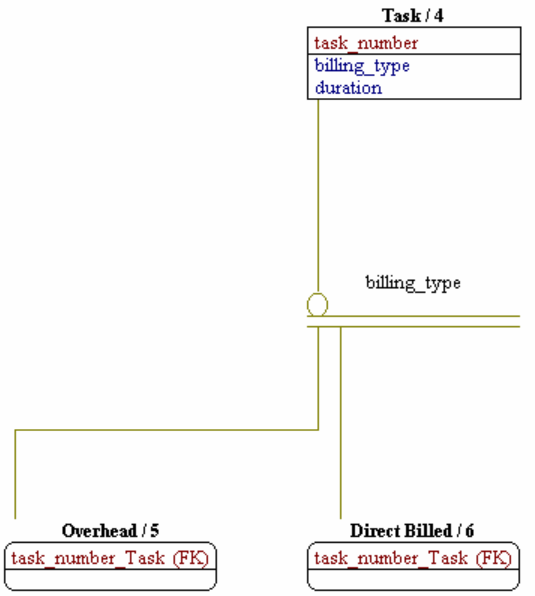
■ Assigning a Discriminating Attribute to a Categorization

① New categorization relation 을 right-click 한 후 Popup menu 에서 Edit Categorization…을 선택한다.

Edit Categorization 대화상자는 전이되는 애트리뷰트에 role name 할당하거나, discriminating 애트리뷰트 할당, categorization type 을 변경, destination 엔티티를 추가하거나 삭제할 수 있도록 지원하며, categorization 에 관한 텍스트로 이루어진 설명을 포함할 수 있게 해준다.

② Discriminating 애트리뷰트에서 billing_type 을 선택하고 **OK** 를 클릭한다.

relation 을 생성하고 discriminating 애트리뷰트를 할당하면, Project Billing 뷰는 아래처럼 보여질 것이다.



일단 데이터 모델을 구성하고 나면, IDEF1X 모델링 룰에 따라서 데이터 모델을 검증하는 것이 좋다. SmartER 은 모델의 유효성을 쉽게 검사할 수 있도록 해준다. 또한 SmartER 은 모델을 체크하는 방법을 또한 변경할 수 있는 옵션을 가지고 있다. 계속해서 작업을 진행하기 전에 모델의 유효성을 검사한다.

■ 모델의 유효성 검토

모델의 유효성 검토는 IDEF1 과 IDEF1X 모델링 규칙을 위반하지 않았다는 것을 보장하기 SmartER 에 의해 수행되는 자동 검토 기능이다.

① Projetc menu 에서 Validate Meodel ...을 선택한다.
② Validate Option 대화상자는 model validation check 를 사용할 것이니 혹은 생략할 것인지를 선택할 수 있도록 한다. 기본 값으로 SmartER 은 아래 사항들을 검토할 것이다.

- 연결되지 않은 엔티티들
- Primary 키가 없는 엔티티들
- Non-specific Relations
- Forward Name 이 없는 Relation 들.
- Discriminator 가 없는 Categorization 들.
- 유효하지 않은 Discriminator 를 가진 Categorization 들.
- FIPS-호환되는 이름과 Aliases

① 모든 Option 을 active 하게 남겨두고 **OK** 를 클릭한다.

Validation check 가 완료되었을 때 두개의 대화상자중 하나가 나타날 것이다.

- 만약에 모델이 유효하면, Info 대화상자는 유효하다는 메시지를 오픈할 것이다.
- 만약에 모델이 유효하지 않으면, Validation Results 대화상자가 오픈될 것이다. 이 대화상자는 타입이나 이름에의해 모델에서 유효하지 않은 엘리먼트를 리스트할 뿐만 아니라, 그 문제를 기술한 메시지를 제공한다.

  여러분은 대화상자내에서 오류(error)를 해결할 수 있다. 즉, 오류를 기술한 메시지를 선택하고 **Edit** 대화상자를 오픈하기 위해서 Edit 를 클릭한다. 선택한 오류와 관련있는 엘리먼트에 따라서 대화상자가 오픈된다.
  또한 **Print** 를 클릭함으로써 메시지의 리스트를 출력할 수 있다. 리스트는 대화상자에서 보여진 것 처럼 출력될 것이다.

다음 과정은 SmartER 의 FIPS 호환에 대해 논의한다.

5.5.7 FIPS 호환(Compliance)

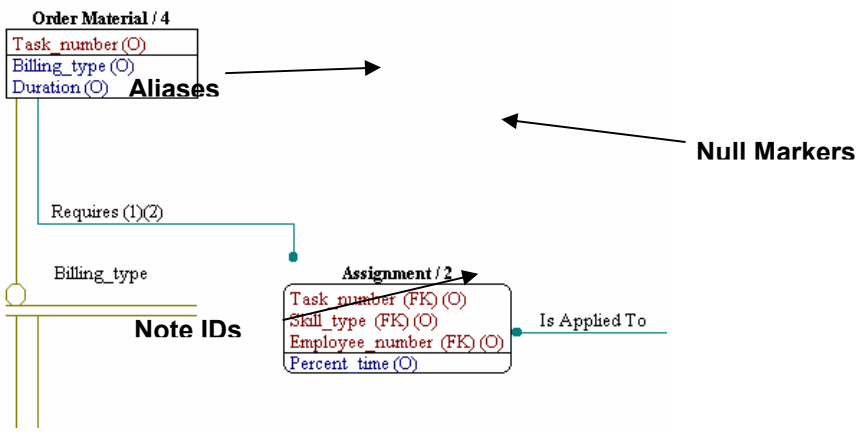
KBSI 는 FIPS 표준을 수용하는 개발을 완벽하게 지원하며 SmartER 은 기는적으로나 표현방법에 있어서 편리하고도 자동화된 FIPS 호환성을 제공한다.

SMARTER 은 여러분이 FIPS 호환 자료를 자동적으로 만들 수 있는 다음과 같은 사항들을 제공한다.

- 호환되지 않는 사항을 백그라운드에서 자동 체크
- Kit forms
- 엔티티와 애트리뷰트에 대한 Aliases
- 애트리뷰트를 위한 No-repeat Rule
- Primary 키에 한정된 Attribute inheritance
- Primary 키 애트리뷰트를 위한 Null 값의 디스플레이
- 중복된 이름이나, 중복된 Aliases, discriminating 애트리뷰트 등을 체트하는 자동화된 유효성 검토 기능

SMARTER 은 모델링을 용이하게 하기 위하여 FIPS 디스플레이 옵션을 사용하거나 사용하지 않도록 하는 옵션을 지원한다. FIPS 옵션의 디스플레이 방법을 변경하기 위해서는 Option menu 에서 FIPS 를 선택한후 다음 사항을 체크한다.

- Display Aliases
- Mark models FEO
- Show Note IDs
- Show Null Markers



5.5.8 뷰의 분할과 병합

이번 절에서는 커다란 뷰를 작은 뷰들로 분할하거나 두 개(혹은 그 이상)의 뷰를 하나로 병합하는 것과 같은 뷰를 빠르게 수정하는 방법에 대해 논의한다. 여러분의 모델에서 Overview of the Work Assignment 뷰와 Project Billing 뷰의 생성을 통하여 이를 연습할 것이다.

일반적으로, 뷰는 작게 구성하는 것이 바람직하다. 모델의 일부분을 표현하는 뷰를 **physical model** 이라 부른다. 여러 개의 작은 뷰를 가지고 작업하는 것이 모델을 개발하는데 훨씬 다루기 쉽다. 그리고 나서 각각의 뷰에서 모델을 완료했을 때, 이를 합쳐서 볼 수 있는 통합모델은 쉽게 만들 수 있다. 이와 같은 통합뷰(overview)를 **conceptual model** 이라 한다.

SMARTER 은 엔티티 위치에 따라 자동적으로 경로를 설정하고 필요로 할 때 마다 릴레이션을 자동적으로 다시 그린다.

모델 전체를 한번에 보기위한 하나의 뷰를 작성하기에 앞서 우리가 만든 두개의 뷰를 병합할 것이다. 노드리스트 윈도우는 이와 같은 통합뷰를 쉽게 만들 수 있게 지원한다.

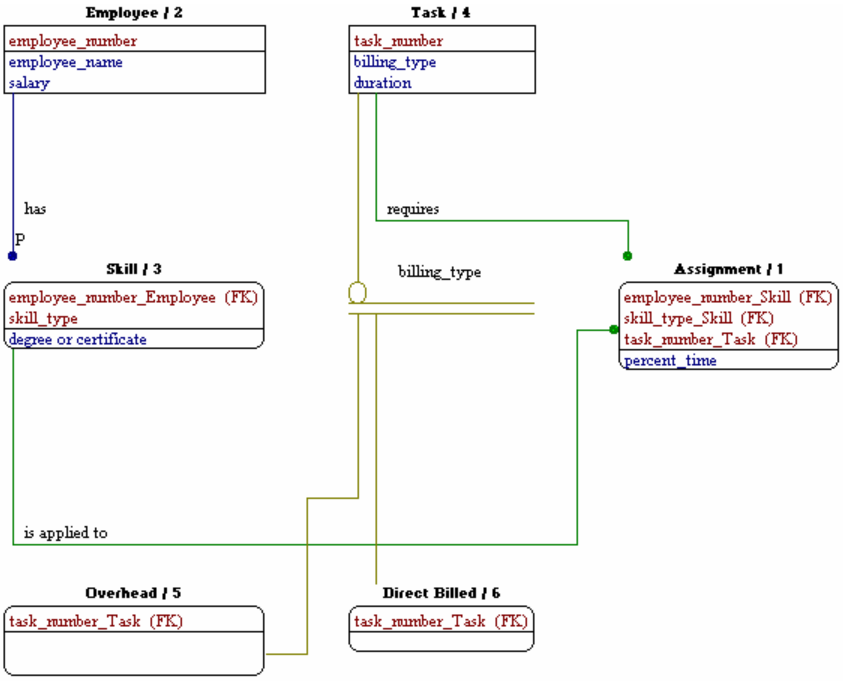
■ 뷰의 병합, 통합뷰의 생성

① 노드리스트 윈도우로 되돌아가기 위해서  클릭한다.
② toolbar 에서  를 클릭한다. 뷰 대화상자의 Name field 에서 **Overview** 를 타입핑하고 **Create New** 를 클릭한다.
③ 대화상자를 닫기 위해서 **Cancel** 을 클릭한다. 하지만 여전히 노드리스트 윈도우 상태유지한다.

Project Billing 과 Work Assignment 를 Overview 로 결합하기 위해서, SmartER 의 drag-and-drop 기능을 사용할 것이다.

④ View List 에서 name 의 왼쪽을 클릭하여 Project Billing 과 Work Assignment 의 계층구조를 숨긴다.
⑤ <Ctrl>를 누른채로 Project Billing 과 Work Assignment 둘다를 선택한다.
⑥ <Ctrl>를 누르고 있는동안, Overview icon 에다 Project Billing 과 Work Assignment 를 드래그한다. 만약 뷰를 드래그 할 때 <Ctrl>를 누르지 않으면 두개의 엔티티를 복사하는 대신에 Overview 로 이동시키는 결과가 될 것이다.
⑦ 커서가  로 바뀔 때, 마우스 버튼을 release 하고 Overview 내로 두개의 뷰의 내용을 drop 한다

완성된 overview 를 보기 위해서, Overview 에서 오른쪽 마우스를 클릭하고, popup menu 에서 *Open View Window* 를 선택한다. 그 뷰는 아래에서 처럼 보여진다.



다음 과정은 SmartER 의 데이터베이스 생성에 대해 논의한다.

5.5.9 데이터베이스의 생성

이번 절에서는 데이터베이스를 생성하기 위하여 SmartER 이 제공하는 다양한 옵션에 대하여 말한다.

SmartER 은 데이터베이스 생성을 위하여 두가지 옵션을 제공한다.

- 원시 SQL 코드 생성 – SmartER 은 ORACLE®, Microsoft® Access™, Microsoft® SQL Server™, and ANSI SQL/92-호환 데이터베이스의 SQL 을 지원한다.
- ODBC 연결 – SmartER 은 Microsoft® Access™ 에 ODBC protocol 을 지원한다.

여러분은 프로젝트 메뉴에서 *Generate SQL Code* 나 *Generate ODBC Database* 를 선택함으로써 위의 데이터베이스를 생성한다.

■ SQL 생성

SQL 텍스트 파일을 생성할 때, SmartER 은 각각의 엔티티를 위한 테이블과 데이터 타입, 제약사항, primary 키, foreign 키를 포함하는 각각의 엔티티 애트리뷰트를 위한 테이블을 생성한다. 또한, SmartER 은 키 detailing 과 index 생성과 같은 몇가지 코드형식을 위한 옵션을 제공한다.

■ ODBC 지원

ODBC protocol 을 이용하여 Microsoft Access 데이터베이스에 연결할 때, 여러분은 데이터 소스, ODBC 드라이버를 추가할 수 있고, 또한 ODBC interface 를 셋업할 수 있다. SQL Data Source 대화상자는 data source 를 구성하거나 선택할 수 있게 해준다.

■ SQL 코드 생성하기

① Project menu 에서 *Generate SQL Code...* 를 선택한다. 첫번째 보여지는 다이아로그는 Save As 인데, 여러분이 생성할 파일의 이름과 저장하고자 하는 디렉토리를 명시할 수 있도록 지원한다. 파일의 이름과 디렉토리를 지정한 후 **OK** 를 클릭한다.

② 현재 활성화된 모델의 SQL 코드 생성을 위해서 원하는 옵션을 선택한다. 여러분은 Detail, Detail Keys In, Create Indexes For, Sort Indexes 그룹 상자들에서 래디오버튼이나 체크박스 중에 적당한 것을 활성화하여 인덱스 생성이나 primary 키, foreign 키의 상세 사항을 사용자가 지정할 수 있다.

③ 만약에 애트리뷰트에 할당된 임의의 데이터 타입을 변경하거나 보기를 원한다면 **Data Types** 를 클릭한다..

Attribute Data Types 대화상자는 프로젝트에 있는 각각의 애트리뷰트에 특정한 데이터 타입을 할당할 수 있게 해준다. 데이터 타입을 할당하기 위해서는, 애트리뷰트를 선택하고, 그 애트리뷰트(정밀도, 스케일, 그리고 Null 을 허용할 것인가를 포함하여)에 할당하기를 원하는 데이터 타입을 선택한다. 그리고 **Assign** 을 클릭한다.

① 각 애트리뷰트를 위한 데이터 타입을 할당하고 난 후, **Done** 을 클릭한다. SQL File Generation 대화상자에서, 적당한 옵션을 활성화했는지를 확인하고 **OK** 를 클릭한다.

SMARTER 은 SQL 파일을 생성하고 그것을 선택된 디렉토리에 위치시킨다. 만약, SQL 파일을 편집하거나 보기를 원하면, 텍스트 편집기나 워드 프로세싱 도구로 그 파일을 오픈한다.

이번 마지막 과정은 프로젝트와 모델의 importing 과 exporting 에 대해 논의한다.

5.5.10 모델의 Importing 과 Exporting

SMARTER 의 import 와 export 기능은 서로 다른 분야의 모델들로 변환하거나 교환할 수 있게 해준다. 이번 절에서는 SmartER 의 다양한 기능에 대해 논의하고 export 처리과정을 단계별로 진행한다.

① Importing – 여러분이 작성한 프로세스 모델로부터 데이터모델을 구축한다. AI0Win 의

기능모델이나 ProSim/Cap 의 프로세스 모델을 import 한다.

예를들어, SmartER 이 프로세스 모델을 import 할때, import 된 엘리먼트는 자동적으로 각각에 상응하는 타입으로 변환된다. Import 된 엘리먼트들은 프로젝트에서 모델에 엘리먼트를 분배할 수 있는 풀에 등록된다.

② Exporting – ProSim/Cap 에 완성된 데이터 모델을 export 하고 여러분의 데이터 구조가 새로운 프로세스 설계에 반영될 수 있도록 한다.

SmartER 이 데이터 모델을 ProSim/Cap 에 export 할 때, 예를 들면 export 된 엔티티들과 애트리뷰트트들은 여러분이 각각에 알 맞는 프로세스 모델 타입에 할당할 수 있도록 하나의 다이아로그에 나타내어진다.

여러분은 SmartER 에서 하나의 프로젝트를 텍스트 파일(*.txt)로 export 하고 다른 KBSI 툴에서 그 파일을 import 할 수 있다. SmartER 은 프로젝트의 어떤 부분을 export 할 지를 지정할 수 있는 몇가지 옵션을 제공한다. 이러한 export 옵션은 Dump Text Options 대화상자에서 사용할 수 있다.

■ 프로젝트의 Exporting

① 모델 export 하기 위해서, File menu 에서 *Export...*을 선택한다. Fields to Dump group 박스에서 선택 가능한 옵션들은 Elements to Dump 리스트에서 선택한 엘리먼트에 따라 다르다. export 옵션의 기본 값은 전체 프로젝트와 모든 엘리먼트 정보를 export 하는 것이다.
② Elements to Dump 와 Fields to Dump 에서 export 하기를 원하는 텍스트 정보와 엘리먼트를 선택한 후 **OK** 를 클릭한다.
③ Save As 대화상자에서, 텍스트 파일의 디렉토리와 이름을 지정하고 **OK** 를 클릭한다.

export 의 진행상황을 나타내기 위하여 Job Status 대화상자가 나타날 것이다. 만약 export 를 취소하려면, 대화상자에서 **Stop** 을 클릭한다. Export 절차가 완료되고 나면, Info 대화상자가 export 가 성공적으로 완료되었는지를 알려줄 것이다.

5.6 IDEF1/1X 모델링의 지침

5.6.1 IDEF1 모델링 지침

IDEF1 모델을 신속하게 검사하고, 모델 작업자가 관련된 환경을 이해한 상태에서 필요한 사항을 표현할 수 있었는지, 현재 혹은 미래의 정보관리 요구를 성공적으로 정의하였는지를 결정할 수 있는 세 가지 간단한 방법이 있다.
이들 검사 테크닉은, IDEF1 모델 작업자가 현실영역의 관점에서 혹은 정보영역의 관점에서 모델작업을 수행하였는지를 거의 즉시 확인할 수 있게 한다.

첫번 째 기술은 엔티티 이름을 점검하는 것이다. 이름이 단수형태로 표시되지 않고 복수형태로 표시되었다면, 모델이 정보영역의 관점에서 작성되지 않은 수가 많다. 예를 들어 '사원'이라고 이름이 붙여진 박스가, '사원들'이라고 이름이 붙여진 것보다 엔티티는 현실세계(사원)에 관한 중요한 정보의 세트를 표현한다는 약속에 비추어 볼 때 더 적절하게 일치할 것이다. '직원들' 이라는 이름은 몇 가지 동일한 특성을 지닌 그룹(즉 회사에서 어떤 일을 하고 있는 각각의 개인들)을 의미할 수 있다. 다시 말해 IDEF1 모델 작업자는 현실적 개체들 각각에 관하여 관심을 집중할 것이 아니라 조직에 의해 실질적으로 관리되는 사원들의 공통적 정보에 관한 사항에 관심을 가져야 된다.
정보영역의 이미지가 아니라 현실적 개체들을 묘사하는 모델을 구분하는 두 번째 지표는, 종종 Non-key 애트리뷰트가 전혀 없는 박스나 혹은 리스트 형태로 모든 애트리뷰트를 열거하고 있는 박스를 찾는 것이다. Non-key 애트리뷰트가 전혀 없는 것들은 전형적으로 조직단위(부서)와 같은 개체들을 표현하는 경우가 많다. 또한 많은 경우 하나의 박스 안에 많은 애트리뷰트를 열거하고 있는 경우를 살펴 보면 서류 양식(Form)을 엔티티로 그 양식에 포함된 항목을 애트리뷰트로 열거하는 경우이다.
세 번째 방법은, 대부분의 엔티티 박스는 한 개나 두 개의 릴레이션을 가지고 있고 단지 몇 개의 엔티티 박스만 여러 개의 릴레이션을 가지고 있는 경우를 찾는 것이다. 이 때 많은 릴레이션을 가지고 있는 박스는 '사람'과 같은 너무도 추상화된 이름을 가진 박스일 가능성이 크다. 이러한 경우 이 엔티티 박스에 연결된 많은 릴레이션은 '사람'이 취할 수 있는 역할을 가리키는데, 예를 들자면 '사람이 부품을 검사한다', '사람이 자료를 등록한다', '사람이 자재를 주문한다', '사람이 .....'. 등으로 모델이 읽혀질 것이다.

사실 IDEF1 를 처음 사용하는 초보자들은 대부분은 현실세계에서 그들이 알고 있고 관찰할 수 있는 모든 것을 모델화 하려는 경향이 있으며 이로 인하여 별로 쓸모도 없는 굉장히 크고 복잡한 모델을 만드는 경우가 많다.
이들 모델들은 작업이 계속되면 될수록 더욱더 복잡하고 관리하기 힘든 모델이 되며 결국 모델화 작업의 본래 목적을 상실하게 된다. 정보영역의 관점이 아니라 현실세계 자체를 모

델화하려는 노력은 사실상 어떤 정보가 필요하고 어떤 효과적인 정보관리 정책이 경쟁상태를 개선시킬 수 있는지에 관한 통찰력을 제공하는데 있어서 아무런 도움도 되지 않는다.

정보시스템 개발자들이 빠지기 쉬운 함정은 '하나의 정보 엔티티 인스턴스가 다른 것들과 유일하게 구분된다는 것은 현실세계에서도 하나의 개체가 다른 것들과 유일하게 구분될 수 있다'고 생각하는 것이다. 하나의 정보 모델은, 현실세계의 개체에 관하여 실질적으로 관리되는 정보만을 표현하고 있으며 다른 어떤 지식도 정보시스템에 유용하지 않다는 사실을 우리는 기억해야 한다. 각각의 현실세계의 개체에 관한 정보가 유지될 때, 정보모델은 특정 현실세계의 개체를 판별하는데 사용될 수 있다. 예를 들면 일련번호가 하나의 엔진을 다른 엔진과 구분하는데 사용되는 것과 같이 엔진이라고 하는 하나의 엔티티와 다른 엔티티를 구분하는데 사용될 수 있다. 그러나 현실세계의 개체의 타입에 관해서만 정보가 유지되는 경우에는 위와는 다른 상황이 되는데, 한가지 예를 들자면 표준화 된 부품인 볼트의 경우를 들 수 있다. 일반적으로 표준화 된 부품인 볼트에 대하여는 이를 구분하기 위한 고유번호가 항상 각 볼트에 할당되지 않는다. 그러나 정보모델에서는 특정 볼트류들에 관한 정보를 유일하게 구분하기 위하여 부품 번호를 사용할 수 있다.

마찬가지로 많은 IDEF1 모델작업자와 정보모델 검토자들이 빠지기 쉬운 미묘하면서도 매우 전형적인 함정은 엔티티 사이의 연결관계와 우리가 사용하는 언어에 있어서의 사실(혹은 업무 룰)을 혼동하는 것이다.
예를 들어 [그림 3-2]에 나타난 모델을 직관적으로 해석하면 "한 '부서'는 하나 혹은 그 이상의 '사원'을 고용한다." 인데 이러한 해석은 현실적 개체 사이의 관계를 반영하고 있다. IDEF1 모델은 현실적 개체들과 그들의 관계에 관한 정보 이미지만을 반영하는 것임으로, 모델에 대한 이러한 해석은 오해를 가져올 수 있다. 만일 릴레이션 혹은 연결관계가 이러한 방식으로 해석되면, 엔티티들은 개체에 관하여 우리가 가지고자하는 정보가 아니라 개체 그 자체를 표현하는 것으로 볼 수 밖에 없는 것이다.
이 모델을 해석하는 올바른 방법은, 이러한 연결관계가 현실세계에서 유지되는 관계를 관리하기 위한 정보를 표현하고 있다는 것에 우리의 인식을 집중할 필요가 있다. 따라서 이러한 연결관계는 " 모델에서 '한 부서는 하나 혹은 그 이상의 사원을 고용한다.' 혹은 그 반대 '각 사원은 정확히 한 부서를 위해서 일한다' 라는 연관관계를 관리하기위하여 이를 지원하는 정보를 가지고 있다" 라고 읽는 것이 더 정확할 것이다.
연결관계를 읽는 좀 더 직관적인 또 다른 방법은 " '한 부서는 하나 혹은 그 이상의 사원을 고용한다' 혹은 그 반대로 '각 사원은 정확히 한 부서를 위해 일한다' 라는 업무 룰은 정보시스템에 의해 유지되거나 강제된다." 라고 읽는 것이다.
오직 정보 시스템에 의해 유지되는 이들의 관계와 업무 룰에 관해서만 모델화하는 원칙을 실행하는 것은 두 가지 중요한 면에서 도움이 된다. 첫째는, 모델 작업자가, 표현될 수 있는 현실적 개체 사이에 존재할 수 있는, 말 그대로 무한한 수를 모델화하려는 경향을 피하

게 하며, 모델 작업자의 관심을 정보 시스템에 의해 지원될 수 있는 관계에만 집중하게 만든다. 둘째, 이러한 원칙은 의사 결정자와 모델 연구자가 모델의 올바름을 판단하고, 정보관리 정책이 업무 룰을 지원하는 정도를 판별할 수 있게 한다. IDEF1 모델 작업자는 추가적으로 이와 함께 원시데이터 항목 리스트와 흡사한 각각의 분리된 비즈니스 룰 리스트를 구성함으로써 모델 검토자를 도울 수 있는데 이는 IDEF1 AS-IS 모델 혹은 TO-BE 모델 결과와 비교될 수 있으며 이 경우 정보관리 정책이 바람직한 작업 방식을 얼마나 훌륭하게 지원하는지를 모델을 통해 쉽게 평가할 수 있다.

IDEF1 의 룰과 절차는, 새로운 시스의 구현을 목표로하는 조직에 의해 현재 관리되는 정확한 정보모델의 구축을 지원한다. 동시에 또한 TO-BE 시스템에서 요구되는 정보의 정의에 대한 메커니즘으로 기능하기도 한다. 그러나 IDEF1 은 정보기술과는 관계 없이 디자인 된 방법이다. 따라서 새로운 정보시스템 구현을 위한 디자인이 시작되고 관계형 데이터베이스의 구축이 결정된 경우에는 IDEF1X, 객체 지향적 방법의 시스템 구현에 대해서는 IDEF4 방법이 사용된다. IDEF14 에 관하여는 다음 절에서 설명한다.

5.6.2 IDEF1X 모델링 지침

IDEF1X 방법과 연관된 기본적 개념은, 현실세계의 사물. 사람. 장소. 사건 등에 관한 자연언어적 사실에 관한 모델화를 논리적 데이터 구조의 모델화로 연결하고자 하는 것이다. 이것은 오직 현실세계의 사물에 대한 정보 이미지에만 초점을 맞추는 IDEF1 의 목표(사물 자체에 대한 것도 아니며 또한 사물에 관해 유지되는 정보를 표현하고자 하는 데이터의 구조에 관해서도 아닌)와는 전혀 다른 것이다. IDEF1X 의 활용성은 종종 이들 현실세계 데이터 항목들의 집합과 연관된 데이터 특성을 설계하려는 시도에 있어서 확대되는데 이는 릴레이션과 애트리뷰트라는 항목에 의해 달성된다.

IDEF1X 의 의도는, '하나의 사원이 정확히 한 부서를 위하여 일한다'와 같은 업무 룰들이 기업의 데이터베이스에 표현된 것과 같이 업무 를을 어떻게 모델화 할 것인가 하는 것이다. 그러나 이러한 초점이, 현실세계의 개체 그 자체를 모델화 하는 것과 혼동되어서는 안된다. IDEF1X 로 현실세계를 모델화 하는 것은 의심스러운 결과를 가져올 수 있으며 또한 데이터베이스 디자이너에게 거의 혹은 전혀 쓸모 없는 결과가 될 것이다.

예를 들어 종종 IDEF1X 모델 검토자는, 특정 엔티티들에 들어있는 애트리뷰트들이 현실세계에서 사용되는 특정 양식에 들어 있는 각각의 항목과 정확히 일치하는지를 확인하고자 한다. 이는 매우 잘못된 일반적인 IDEF1X 의 적용 사례로서 현재의 '정보집합(즉 양식화 된 장표)'을 그대로 아무런 재 구성이나 생략 없이 IDEF1X 엔티티로 변환하여 모델화 하는 것 매우 잘못된 방식이다.

혹자는 그러한 모델링 방식이 유용하다고 주장할 지도 모른다. 그렇게 구축된 IDEF1X 모

델은 사실상 기존의 양식에 표시된 모든 항목을 정확하게 모델에 반영할 수 있었다. 따라서 충분히 정당화 될 수 있는 것 같이 보이기도 한다. 그러나 “그러한 방식으로 양식을 모델화 함으로써 어떤 디자인 활동이 촉진 되었는가?” 하는 것이다. 달리 말하자면 “일련의 정보요구로 부터 실행 가능한 관계형 테이블의 논리적 설계를 결정하는데 있어서 그런 종류의 모델 작업이 도움되는가?” 하는 것이다.

기존의 양식에 나타나는 형식과 일치하게 테이블, 혹은 릴레이션을 관계형 데이터베이스로서 작성하는 것이 합당한가? 이러한 접근방식은 제한된 범위의 요소들만이 모델화 된 아주 작은 규모의 데이터베이스 구현에 있어서는 거의 영향을 주지 않을 것이다. 그러나 대규모 데이터베이스 구현의 경우에는 너무도 많은 다양한 양식들은 이를 검토하는 것 만으로도 수 많은 모델작업자가 필요할 것이며 또한 이들 모든 양식들과 각각의 양식을 구성하는 항목들을 관리하는데 더 많은 노력을 필요로 할 것이다. 또한 그러한 모델이 설사 설계 사양으로 채택되어 데이터베이스로 구현된다 할지라도 그 결과는 현재의 종이문서로 이루어진 시스템을 단순히 컴퓨터화한 것이 될 것이다. 이러한 종이문서 시스템의 단순한 자동화는 CIM 실패의 역사중에서도 가장 잘못된 사례로 널리 알려져 있다.

이러한 예에 기초해 볼 때, 디자인 방법을 잘못 적용하는 경우 혼란스럽고 빈약한 설계결과가 될 수 밖에 없다는 것은 명백하다. 한가지 가능한 해결책은, 엔티티가 개체 그 자체가 아니라 현실세계의 개체에 관한 정보를 표시해야 한다는 규약을 수립하고 이를 지키는 것이다.

방법은 세 가지 다른 측면에서 필요하다. 우선, 사람. 장소. 사건 등의 사이에 존재하는 관계 및 현실세계에 관하여 알고 있는 것을 효과적으로 파악하기 위한 방법이 필요하다. 둘째, 기존의 그리고 예상되는 정보관리 요구를 파악하기 위한 방법이 필요하다. 세 째, 정보기술을 적용하기위하여 소프트웨어 시스템 디자인을 지원하기 위한 훌륭한 방법이 필요하다.

다음절에서 논의하는 ‘IDEF3 프로세스 모델링 방법’이나 ‘IDEF5 존재론적 설명 방법’은 첫 번 째 필요에 맞추어 특별히 디자인 된 것이다. ‘IDEF0 기능모델’ 그리고 ‘IDEF1 정보모델’은 특별히 두 번째 필요에 상응하는 목적을 가지고 있다. ‘IDEF1X 데이터 모델’ 그리고 ‘IDEF4 객체 지향적 디자인’은 세 번째 필요성에 맞추어 개발되었다. 불행히도 이들 세 가지 영역 사이의 뚜렷한 구분을 유지하지 못한 상태로 많은 모델화 언어들이 사용되고 있다.