경영정보학연구
제10권 제4호
2000년 12월

전자상거래 개발을 위한 코디네이션 접근 방식

김 희 웅*, Thomas W. Malone**

Coordination Approach for Electronic Commerce Development

Kim, Hee-Woong, Thomas W. Malone

Successful Electronic Commerce (EC) requires not only a transformation of the supply chain but a redesign of organizational processes as well. Just as the organizations in a supply chain must cooperate with each other to achieve a collective goal, so too do the departments and groups that comprise an organization depend on each other to achieve the organization's goals. A lack of coordination at either of these levels can result in poor performance and high coordination costs. In addition, changes in inter-organizational dependencies can affect the dependencies within the affected organizations. The lack of coordination across two levels results in a transition gap in transforming the inter-organizational design to the intra-organizational design in EC deployments.

The purpose of this research is to develop a modeling method for coordination, managing the linkage between inter- and intra-organizations in EC deployments. The method was applied to the EC development project of a cable TV home-shopping company. The coordination approach enabled us to manage dependencies among the coordination elements and identify the effect of a change on other elements, which became the basis for effective EC deployments. We will discuss the method and compare it with other dependency management methods.

* LG-EDS Consulting 부문
** MIT, Sloan School of Management

Ⅰ. 서 론

조직이란 공동의 목표 달성을 위해 서로 상호 의존적인 요소(프로세스, 자원, 부서, 직원 등)들의 집합체라 할 수 있다[8, 13, 16]. 조직 내부 요소들간의 상호 의존관계의 예로는 프로세스간, 프로세스와 자원들간의 관계 및 프로세스와 부서들간의 관계 등이 있다. 조직 간에도 공동의 목표를 달성하기 위해 공급사슬(Supply chain) 같은 형태로 서로 상호 협력 및 의존 관계를 가진다[20]. 조직 내부의 그리고 조직간 의존관계를 적절히 관리하는 것은 조직 설계 뿐만 아니라 조직의 성과를 증진시키는데 아주 중요한 사항이다[3, 11, 12, 21]. 이러한 의존 관계들에 대한 부적절한 코디네이션은 비효율적 의사소통 채널과 비부가가치적 활동들을 유발할 수 있으며 성과 감소 및 코디네이션 비용 증가와 같은 결과를 가져올 수 있다. 코디네이션의 중요성은 공항 관제탑의 예를 들면 알 수 있다. 공항 활주로에는 착륙하려는 항공기들과 이륙하려는 항공기들이 활주로라는 공유 자원에 대해 서로 다른 프로세스를 수행하며 존재한다. 이 과정에 관제탑에 의한 코디네이션이 부적절한 경우는 이착륙에 따른 시간 지연과 항공기 연료 낭비 및 고객 불만, 최악의 경우에는 항공기 사고라는 위험에 까지 노출되게 된다.

Business Process Redesign(BPR)[18]은 조직의 성과 개선을 목적으로 기능적 프로세스들을 단순화하는데 초점을 맞추었다. 하지만 대부분의 BPR 프로젝트에서는 프로세스 측면의 변화가 조직의 다른 부분에 어떻게 영향을 미치는지에 대한 충분한 고려가 이루어지지 않았다. 이러한 코디네이션 이슈는 BPR 프로젝트의 주요 실패 원인 중에 하나로 파악되었다[23]. 조직 내부 요소들간의 수평적 의존관계와 더불어, 조직들간에도 의존관계가 존재한다. 예를 들어, 공급사슬과 같은 조직간의 프로세스에 변화가 발생하면 이는 조직 내부의 프로세스 및 관련 자원 및 기타 요소에도 영향을 미친다. 이처럼 조직 내부와 조직간 각각의 레벨에서 수평적 의존관계가 존재하며, 두 레벨간 수직적 의존관계가 존재한다. 코디네이션은 시스템 분석 설계 및 개발 과정에 있어서도 중요한 역할을 담당한다. Clemons et al.(1995)는 구현이 완료된 정보시스템이 고객의 요구사항을 충족시키지 못한다는 내용의 기능적 리스크를 BPR 프로젝트의 주요 실패 원인 중의 하나로 제시하였다. 이는 재설계된 프로세스를 정보시스템을 통하여 성공적으로 지원하기가 어렵다는 것을 설명한다. 또한 사용자의 요구사항이 시스템 설계 및 개발 과정에 충분히 반영되지 못하여, 요구사항과 개발된 시스템 간에 전환 갭(transition gap)이 존재하기도 하였다[25]. 마찬가지로 수직적 그리고 수평적 코디네이션이 부적절할 경우에는 전자상거래 분석 및 설계 단계와 개발된 시스템 간에 전환 갭이 발생할 수 있다.

본 연구에서는 전자상거래 개발을 위하여 조직 간의 의존관계와 조직 내부의 수평적 의존관계 그리고 조직간 관계와 조직 내부간의 수직적 의존관계를 분석 및 설계하고 이를 시스템 개발에 연계시키는 방안을 제시하고자 한다. 전자상거래 도입은 조직 형태에 있어 수직적 통제, 수평적 코디네이션, 조직의 규모 및 구성 요소들에 있어 변화를 가져왔다[9]. 전자상거래는 공급사슬 참여 조직들[10] 뿐만 아니라 조직 내부에[19] 있어서도 전자적 연계를 지원해주므로, 조직간 의존 관계에서부터 조직 내부의 의존관계까지 관리가 필요한 대표적 분야라고 할 수 있다. 본 연구에서는 이러한 연구 배경을 바탕으로 코디네이션을 기반으로 하는 전자상거래 개발 방법을 제안하고자 한다. 이를 위해 코디네이션을 위한 의존 관계의 요소들을 파악한다. 이러한 요소들은 전자상거래 분석 및 설계를 위한 모델링 구성 요소가 된다. 이와 더불어 연관된 요소들간의 의존 관계 관리를 위하여 의존관계 유형에 따른 코디네이션 메커니즘을 제시

한다. 마지막으로 제안하는 방법과 홈쇼핑 적용 사례를 통해 그 타당성을 토의해 보겠다.

2. 코디네이션 이론

조직 내 요소들의 상호 의존관계는 조직이 목표를 어떻게 달성할 지에 대해 제한을 가하게 되며, 이러한 것이 바로 코디네이션 문제로 정의된다. 의존관계를 구조화하는 것이 조직 성과를 개선하는데 있어 결정적이므로, 이러한 코디네이션 문제는 Organizational Science 분야에 있어 중요한 이슈가 되어 왔다. Thompson(1967)은 reciprocal, sequential, pooled라는 세가지 유형의 의존관계 패턴과 그에 따른 코디네이션 메커니즘을 제시했다. 그는 조직 구조들은 reciprocal한 상호 의존관계로 가장 밀접한 그룹들을 이루려 하는 경향이 있으며, 그리고 나서 sequential한 상호 의존관계로 그룹핑되며, 마지막으로 pooled 상호 의존관계로 그룹핑된다고 제시하였다. Galbraith(1977)는 코디네이션을 조직 설계의 핵심 요소로 파악하였다. 이는 조직이 달성하고자 하는 목표, 조직 내에서 일하는 직원, 노동 분배 패턴들, 그리고 조직 단위간 코디네이션 관계에 있어서 일관성을 찾아나가는 것으로 간주되었다. Van de Ven et al.(1976)은 inter-personal, personal, group이라는 세가지 유형의 코디네이션을 파악하였으며, 상황에 어떤 유형이 사용되는지를 결정하는 상황 요소들을 토의하였다. 의존 관계를 대상으로 조직 차원의 그 개념에 대한 대부분의 연구들은 그러한 의존관계가 직원들 또는 조직 부서들로부터 발생한다고 여겼으며 대부분 직원-직원 간의 의존관계 패턴을 기술하였다. 하지만 조직 자체는 프로세스들의 집합이며[13, 17], 공동의 목표를 달성하기 위해 그들 간에 코디네이션이 필요한 것으로 생각될 수 있다.

코디네이션이란 "활동들간의 의존관계를 관리"[21]하는 것이라 정의될 수 있다. 그렇다면 조직의 변화는 존재하는 조직 요소들을 우선 분석하고 그들 간의 의존관계를 변화시킬 수 있는 방법을 찾아내는 것에서부터 시작된다고 할 수 있다. 코디네이션 이론은 "다양한 조직 환경에 걸쳐 여러 가지 유형의 의존관계 및 그에 따른 코디네이션 메커니즘을 파악하고 연구"[15]하는 것을 제안한다. 여기서 의존 관계(Dependency)란 복수 활동간에 물리적 자원 (예, 상품) 또는 정보적 자원(예, 주문)의 흐름(Flow), 자원의 공유(Sharing), 또는 자원의 구성(Fit)으로 구분되며 이러한 것들의 조합도 가능하다. 코디네이션 프로세스란 그러한 의존 관계를 관리하는 활동들로서 공유 자원의 관리, Producer/Consumer 관계, 동시성(Simultaneity) 제약사항 관리, 그리고 활동/하위 활동 간의 관계 관리 등이 있다[Malone and Crowston 1994]. 예를 들어, 같은 업종의 기업들은 생산, 판매, 마케팅, 배달 등 대부분 유사한 기본 프로세스들을 수행한다. 이러한 기본 프로세스들은 동일한 반면, 비즈니스 프로세스들은 기업마다 큰 차이가 있다: 어떻게 비즈니스 프로세스를 프로세스로 세분화 되는지, 프로세스들이 어떻게 연결되는지, 누가 어떤 프로세스를 담당하는지, 어떻게 자원들이 프로세스에 얼마만큼 배정되는지 등. 이러한 이슈들은 인해 코디네이션 이론의 대상 문제에 해당하며, 결국 기업의 성과를 가늠하게 된다.

서로 다른 유형의 의존관계를 분석하기 위해서 목표, 활동(Activity), 수행자(Actor), 그리고 자원과 같은 코디네이션 요소들이 제시되었다. Crowston(1994, 1998)은 코디네이션 요소들을 직무(Task)와 자원(Resource)과 같은 두가지 범주로 구분하였다. 이와 더불어 직무-자원간의 코디네이션 메커니즘을 제안하였다. 직무란 목표를 달성하는 것과 활동을 수행하는 것을 포함한다. 자원이란 그러한 활동에 의해 사용되거나 영향을 받는 개체들을 포함한다. 직무와 자원간의 의존관계에 따라 다양한 코디네이션 프로세스들이 제시되었다. 예로서 공유 자원의 관리, 활

동들에 있어서 생산자(producer)/소비자(consumer) 관계 관리, 자원 활용에 있어 동시성(simultaneity) 제약사항 관리, 직무/세부 직무 간의 관계 관리 등이 있다.

의존 영역에 따라 네가지 유형의 -- 조직간(inter-organizational), 조직 부서간(inter-departmental), 조직 내 직원간(inter-personal), 개인적(personal) -- 코디네이션이 존재할 수 있다. 조직간 코디네이션 모드는 조직간의 커뮤니케이션에서 유발한다. 이 유형의 초점은 채널 설계이다. 조직간 의존관계에서는 상품과 정보와 같이 교환되는 항목(item), 장소와 조직과 같은 공간적 요소(spatial factors), 프로세스 플로우와 같은 시간적 요소(temporal factors)와 같이 최소 세가지 요소가 관련된다[20]. 여기서 공간적 요소는 프로세스가 어디에서 그리고 함축적으로 누구에 의해 수행되는지를 나타내기 위한 것이다. 전자상거래 환경에서는 각 참여 기업의 물리적 위치는 기존의 공급사슬에서의 기업의 지역적 위치에 비해 상대적으로 의미가 적다. 따라서 새로운 전자상거래 환경에서는 이러한 공간적 요소는 누구에 의해 프로세스가 수행되며 함축적으로 어디에서 수행되는지를 나타내는데 활용할 수 있다

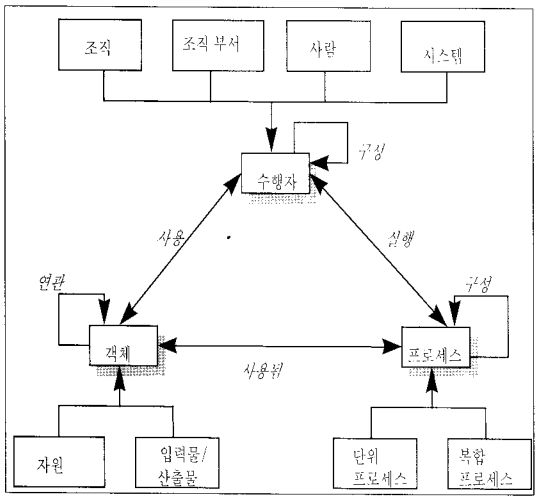
조직의 목표를 하위 목표로 세분하는 것과 더불어 이의 달성을 위해 업무들을 조직 부서에 배정하는 것은 조직 부서간 그리고 업무들간의 의존관계 관리를 필요로 하는 것을 의미한다. 이처럼 코디네이션을 위해서는 복수 개의 조직 구성 요소들간의 상호 의존관계를 관리해야 한다. 이는 한 조직 구성요소의 변화는 다른 의존관계의 요소에 영향을 미치기 때문이다. 조직 부서 내에서는 프로세스를 담당하는 수행자에게는 역할이 주어진다. 세분화되어서 조직 부서에 할당되어지는 목표를 달성하기 위해서는 조직 내 직원간 모드에서 서로 다른 역할의 수행자들 간에 코디네이션이 필요하다. 마지막으로 다양한 직무와 역할들이 한 수행자에 배정될 수 있다. 개인 모드에서는 그 수행자 또한 그러한 임무와 역할들에 대한 코디네이션이 필요하다.

3. 모델링을 통한 코디네이션 방법

코디네이션, 즉 요소들간의 상호 의존관계 관리를 위한 모델링 접근 방식을 개발하였다. 이를 위해 우선 코디네이션 요소들을 파악하고, 코디네이션을 위한 모델을 제안한다. 코디네이션을 위한 모델은 코디네이션 요소들간의 상호운용 가능성(Interoperability) 테스트와 더불어 여러가지 의존관계 관리 및 전자상거래 설계에도 사용될 수 있다.

3.1 코디네이션 프레임워크

코디네이션 프레임워크를 개발하기 위해 조직 모델[8]로부터 조직 구성 요소들을 파악하였다. Levitt(1965)이 제안한 조직 모델은 커뮤니케이션 및 권한(authority)과 관련하여 *구조(structure)*, 상품과 서비스를 생산하기 위한 *직무(task)*, 직무를 수행하는 *수행자(actor)*, 그리고 직무 수행에 사용되는 자원으로서의 *기술(technology)*로 구성된다. 이 네가지 요소들은 서로 상호 의존적 관계를 가지며, 한 요소에의 변화는 다른 요소들에게 상호 연관된 변화를 유발시킨다. 요소들간의 수직적 구조 관계를 고려하여, 제안된 요소들을 *수행자(Actor)*, *프로세스(Process)*, *객체(Object)*로 재구성하였으며, 그들 간의 상호 의존관계를 <그림 1>과 같이 정리할 수 있다. 첫번째, 조직 부서 단위 간에 수직 구조 관계가 있으며 이는 조직 모델의 구조와 유사하다. 각 조직 부서 단위는 직원과 시스템 서버들로 구성된다. 이러한 이유로 인해, 구조 관계를 포함하는 조직 부서 단위와 *수행자*들을 통합하여 광의의 *수행자(Actor)*를 제시하였다. 두번째, 복합 프로세스는 단위 활동들로 구성되므로, 비즈니스 프로세스



<그림 1> 코디네이션 요소들간의 상호 연관관계

와 활동들을 통합하여 *프로세스*를 제시하였다. 이는 Levitt의 조직 모델에서 *직무*와 유사하다. 세번째, *객체*는 프로세스의 입력물 또는 산출물과 같은 자원으로 사용되며 다른 객체와 관계를 가진다. 이 객체는 Levitt 모델의 *기술*을 포함하는 자원과 유사하다. 복합 *프로세스*가 조직 또는 조직 부서 단위와 같은 상위 수준의 *수행자*에 의해 수행되는 반면, 단위 *프로세스*는 직원 또는 시스템 서버 등과 같은 개별 *수행자*에 의해 수행된다.

코디네이션 요소들간의 상호 의존관계는 코디네이션 모드에 따라 바뀌게 된다. 조직간 모드에서는 조직들 간의 커뮤니케이션 채널 관리에 초점을 맞춘다. 이 모드에서의 코디네이션을 위해 조직을 *수행자*로, 커뮤니케이션 항목을 *객체*로, 그리고 조직간 상호작용 활동들을 *프로세스*로 하는 모델링 방식이 필요하다. 조직 부서간 모드는 비즈니스 프로세스를 통하여 부서간의 의존관계에 초점을 맞춘다. 이 모드에서의 코디네이션을 위해서는 비즈니스 프로세스를 *프로세스*로, 조직 부서들을 *수행자*로, 그리고 프로세스들 간의 입력물과 산출물 및 자원들을 *객체*로 하는 모델링 방식이 필요하다. 조직내 직원간 코디네이션 모드는 활동들을 통하여 직원간의 의존관계에 초점을 맞춘다. 이 모드에서는 활동들을 *프로세스*로, 각 직원 또는 활동 참여자들을 *수행자*로, 그리고 활동과 관련된 입력물/산출물 및 관련 자원들을 *객체*로 하는 모델링 방식이 필요하다. 개인 모드는 한 수행자에 의해 수행되는 다양한 활동들과 관련 자원들간의 의존관계에 초점을 맞춘다. 이 모드에서의 코디네이션을 위해서는 단위 활동들을 *프로세스*로, 활동과 관련된 입력물/산출물 및 자원들을 *객체*로 하는 모델링 방식이 필요하다.

3.2 코디네이션 메커니즘

제시한 세가지 코디네이션 요소들과 그들간의 의존 관계를 바탕으로, 조직 내의 의존 관계를 파악하고 코디네이션 메커니즘을 제시할 수 있다. 코디네이션 요소인 프로세스, 수행자, 객체간에 여러 종류의 의존관계가 (프로세스-프로세스, 프로세스-객체, 프로세스-수행자, 수행자-객체, 수행자-수행자, 객체-객체) 존재한다. 이러한 의존 관계 중, 수행자와 객체간의 관계는 프로세스와 수행자 간의 관계와 프로세스와 객체간의 관계를 통해 파악될 수 있다. 각각의 의존 관계와 연관된 코디네이션 메커니즘이 <표 1>에 정리되어 있다.

프로세스와 프로세스 간의 수평적 의존 관계는 자원의 사용 유형에 따라 분류된다. 첫번째, 선행 프로세스의 산출물이 후속 프로세스의 입력물로 활용되는 경우(Flow)가 있다. 두번째, 자원이 두개의 프로세스에 의해 공유되는 경우(Share)가 있다. 세번째, 서로 다른 자원이 하나의 프로세스에 같이 필요한 경우(Fit)가 있다. 예를 들어, 프로세스 간의 선행(Flow) 관계에 대한 관계 관리를 위해서는 프로세스 간의 순서 정의를 필요로 하며, 선행 프로세스의 산출물이 후속 프로세스에서 원하는 사양과 동일한지 확인이 필요하고, 선행 프로세스에서 후속 프로세스로의 자원 전달 관리가 필요하다. 이러

<표 1> 의존 관계 유형 및 관련 코디네이션 메커니즘[1)]

의 존 관 계		코디네이션 메커니즘
프로세스 - 프로세스	한 프로세스의 Output이 타 프로세스의 Input(Flow) 1. 같은 특성의 경우 2. 충돌할 경우	1.1 업무의 순서를 정함 1.2 전달 객체의 사용 가능성확인 1.3 전달 또는 객체/수행자 관리 2.1 충돌(conflict)을 회피하기 위해 프로세스들의 순서를 재정리 2.2 수행할 한 업무 선택
	프로세스들이 공통 Input을 공유(Share) 1.공유 가능한 입력물 2.재사용 가능한 입력물 3.재사용이 안되는 입력물	1.1 충돌이 없음 2.1 충돌을 인지 2.2 이 객체 Input의 사용 계획을 정함 3.1 수행할 한 업무 선택
	프로세스들이 공통 Output을 공유 (Fit) 1. 같은 특성의 경우 2. 중복(overlapping) 3. 충돌하는 경우	1.1 중복 프로세스 파악 1.2 프로세스들을 통합 또는 수행할 한 업무 선택 2.1 상호 동의하는 결과를 협상 3.1 수행할 한 업무를 선택
프로세스 - 객체	프로세스에 의해 요청되는 객체	1.1 필요한 객체들을 파악 1.2 가용한 객체들을 파악 1.3 특정 객체 집단을 선택 또는 설계 1.4 객체들을 프로세스에 배정 1.5 객체들간의 의존 관계를 관리
프로세스 - 수행자	프로세스를 수행하는 수행자	1.1 필요한 수행자를 파악 1.2 가용한 수행자들을 파악 1.3 특정 수행자 그룹을 선택 1.4 수행자들을 프로세스에 배정 1.5 수행자들간의 의존 관계 관리
수행자 - 객체	프로세스 실행에 관여하는 수행자와 객체	1.1 프로세스 수행자와 프로세스 관련 객체 파악 1.2 프로세스-수행자 코디네이션 메커니즘 적용 1.3 프로세스-객체 코디네이션 메커니즘 적용 1.4 프로세스를 중심으로 객체와 수행자간의 관계 설정

한 프로세스들은 아주 다양한 방법으로 수행될 수 있다.

프로세스와 객체간의 의존관계 관리를 위한 코디네이션 메커니즘으로서, 프로세스 수행에 가용한 객체들을 우선 파악하고 그 프로세스에 배정한다. 그리고 추가된 객체와 기존 객체들간의 관계도 수정한다. 유사하게 프로세스와 수행자간의 의존관계 관리를 위한 코디네이션 메커니즘도 가용한 수행자들을 프로세스에 배정하고, 기존 수행자와 추가된 수행자들간의 관계를 갱신하는데 초점을 둔다. 수행자와 객체간의 의존관계 관리를 위한 코디네이션 메커니즘은 동일한 프로세스를 중심으로 서로 연결된 수행자와 객체들에 대해 프로세스-수행자, 프로세스-객체 관계에서의 코디네이션 메커니즘을 각각 적용한 후, 프로세스를 중심으로 객체와 사용자간의 관계를 재정의하는데 초점을 둔다.

3.3 코디네이션 기반 모델링

의존관계 관리를 위해, 도식적 코디네이션 기반 모델링 방법과 개별 코디네이션 요소에 대한 명세(specification) 방법을 제안한다. 기존의 프로세스 맵과 유사한 도식적 모델은 프로세스 플로우를 따라 세개의 코디네이션 요소간의 상호 의존관계를 표현하는데 사용된다. 이 도식적 모델링 방법에는 세개의 주요 모델링 요소와 (ACTOR, PROCESS, OBJECT) 한 개의 추가적 요소가 (BRANCHING) 있다. ACTOR는 도식 모델 위에 그 이름만으로 표현된다. PROCESS는 사각형으로 표현된다. OBJECT는 프로세스들간의 플로우를 따라 모델링된다. BRANCHING은 프로세스 플로우의 조건 분기를 나타내기 위해 원형으로 표현된다. 이러한 모델 구성 요소들은 시간 및 프로세스 흐름에 따른 수평축과 수행자 측면의 수직 축 기준으로 작성된다.

도식모델 상의 코디네이션 요소들은 BNF를 이용하여 명기할 수 있다. PROCESS는 9개의 요소들로 – *process name, input, output, precondition, postcondition, procedure, actor, resource, constraint* - 명세화 된다. Input은 프로세스를 수행하기 위해 필요한 객체로서, 대개 선행 프로세스에서 전달된다. Output은 프로세스를 수행하고 나온 산출물을 의미하며, 이는 뒤에 연결되는 프로세스에 Input으로 사용된다. Input과 Output은 OBJECT들로 정의된다. 이러한 OBJECT들은 프로세스들간의 의존관계 유형에 따라 'AND' 또

1) *Crowston*(1998)이 제시한 내용을 개선

는 'OR' 조건으로 정의된다. Precondition 은 프로세스를 수행하기 위해 필요한 상태를 의미한다. Precondition이 만족되고 필요한 Input이 입력되어야만 프로세스는 활성화될 수 있다. Postcondition은 프로세스를 수행하고 난 후의 결과 상태에 해당한다. 이는 문자열과 선택적으로 관련 OBJECT로 명기된다. Subprocesses는 복수의 세부 활동들간의 절차로 명기된다. 프로세스 플로우를 위해서 3가지 기호를 -- THEN 의미로서 ';', AND 의미로서 '&', 그리고 OR 의미로서 '||' -- 사용한다. Procedure 명세 데이터는 워크플로우 스케쥴링을 위한 기반으로 활용될 수 있다[1, 4]. 프로세스와 수행자간의 의존관계는 performed_by 요소를 이용하여 나타낼 수 있다. 프로세스를 수행하기 위해서는 자원들이 필요하게 된다. 자원은 시스템, 기계, 자료, 또는 도구 등 여러가지 예가 있을 수 있다. 프로세스 명세의 input/output 그리고resource 요소는 프로세스와 객체간의 의존관계를 나타낸다. Constraint요소는 관련된 프로세싱 규칙 또는 코디네이션 메커니즘을 표현하는데 사용된다. 수행자와 객체간의 의존관계는 프로세스를 통해 관리할 수 있다.

```
<PROCESS>::=  <process_name>  <input>  <output>
              <precondition> <postcondition>
              [<procedure>] <performed_by>
              <resource> <constraint>
    <process_name>::= String
    <input>::= [ ( ] <OBJECT> [ ) ] [AND|OR <input>]
    <output>::= [ ( ] <OBJECT> [ ) ] [AND|OR
              <output>]
    <precondition>::= [ ( ] String | <OBJECT> String
              [ ) ] [AND|OR <precondition>]
    <postcondition>::= [ ( ] String | <OBJECT>
              String [ ) ]
              [AND|OR <postcondition>]
    <procedure>::= <PROCESS> ( | ; | & | || | )
              <procedure>
    <performed_by>::= <ACTOR>
    <resource>::= <OBJECT>
    <constraint>::=<processing_rule> | <coordination_
              mechanism>
    <processing_rule>::= String
    <coordinaiton_mechanism>::= String
```

OBJECT는 세개의 요소들로 -- object name, attributes, relationship -- 명세화 된다. 객체의 속성(attributes)은 object_spec 요소로 정의된다. 복수 개의 속성들은 '{'와 '}' 기호를 이용하여 나타내며, 그것들은 '+' 기호를 이용하여 통합 표시된다. 하나의 OBJECT는 다른 OBJECT와 관계를 가질 수 있으며, 이는 relationship 요소를 이용하여 표시된다. 이러한object_spec과 relationship 요소는 데이터베이스 설계에 사용될 수 있다.

```
<OBJECT>::= <object_name> <object_spec> <related_with>
    <object_name>::= String
    <object_spec>::= [{] <object_attribute> [}] [+
              <object_spec>]
    <object_attribute>::= String
    <related_with> ::= {<OBJECT>}
```

ACTOR는 세개의 요소로 -- actor name, role, 선택적으로 hierarchy ? 명세화 된다. 수행자는 서로 다른 프로세스를 수행함에 있어 고객 또는 서비스 제공자와 같이 서로 다른 역할을 가질 수 있으며, 이는 role_name으로 정의된다. 수행자 간에는 수직적 구조, hierarchy가 존재할 수 있으며, 이는 consist_of 요소로 정의된다.

```
<ACTOR>::= <actor_name> <role> [<consist_of>]
    <actor_name>::= String
    <role>::= {<role_name>}
    <role_name>::= String
    <consist_of>::= {<ACTOR>}
```

이러한 명세(Specification) 자료는 다음의 두가지 목적으로 사용될 수 있다. 첫번째, 코디네이션

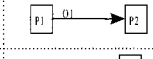
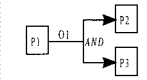
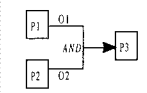
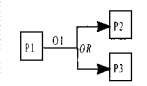
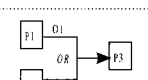
요소들간의 여러 가지 의존관계를 -- PROCESS-OBJECT, PROCESS-ACTOR, OBJECT-ACTOR, PROCESS-PROCESS, OBJECT-OBJECT, ACTOR-ACTOR -- 형식화 측면에서 명세화하고 관리할 수 있다. 도식 모델만을 이용해서는 요소들간의 의존관계를 충분히 파악하고 관리하는 데 한계가 있다. 도식 모델은 프로세스 플로우 또는 데이터 플로우에 초점을 맞춘다. 이로 인해 개념적 설계 모델을 시스템으로 구현하는 데 있어 쉽게 갭을 유발시킬 수 있다[5, 6]. 개념적 설계로부터 유도된 명세 데이터는 객체지향 시스템 개발 특히, 워크플로우 시스템 개발을 위한 기반으로 활용할 수 있다[24]. 두번째, 제시한 세 개의 코디네이션 요소들이 서로 밀접하게 의존관계를 가지므로, 조직 변화 또는 공급사슬 변환으로부터 발생하는 한 요소에의 변화는 다른 연관 요소에 영향을 끼친다. 명세 데이터를 이용하면 이러한 변화의 영향을 받는 연관된 요소들을 파악하는데 활용할 수 있다.

3.4 의존관계 유형과 일관성 테스트

공급사슬 변환과 새로운 조직 변화는 코디네이션 요소들 뿐만 아니라 그들 간의 의존관계에도 변화를 가져온다. 비록 대부분의 조직 변화 방법들은 변화의 원칙과 지침들을 제시했지만, 변화 대상 요소들이 다른 관련 요소에 어떤 영향을 미치는지에 대한 고려를 충분히 하지 않았다. 또한 변화 대안에서 새로운 의존 관계에 놓이게 되는 코디네이션 요소들간에 상호 운용가능성이 있는지 확인하기가 쉽지않다. 이러한 이유로 인해, 코디네이션 요소들간에 일관성 테스트가 필요하게 된다.

일관성 테스트를 위해 <표 2>와 같이 프로세스들간의 5가지 의존관계와 그에 따른 일관성 조건들을 제시하였다. 앞장에서 제시한 Flow, Share, Fit이외에 Flow와 Share의 변형인 Branching과 Flow와 Fit의 변형인 Altering을 추가하였다.

<표 2> 프로세스들간의 의존관계 유형 및 일관성 조건

의존관계 유형		일관성 조건
Flowing (P1;P2)		{Output(P1) -> Input(P2)} ^ {Postcondition(P1) -> Precondition(P2)}
Sharing (P1;(P2&P3))		{Output(P1) -> [Input(P2) ^ Input(P3)]} ^ {Postcondition(P1) -> [Precondition(P2) ^ Precondition(P3)]}
Fitting ((P1&P2);P3)		{[Output(P1) ^ Output(P2)] -> Input(P3)} ^ {[Postcondition(P1) ^ Postcondition(P2)] -> Precondition(P3)}
Branching P1;(P2|P3)		{Output(P1) -> [Input(P2) v Input(P3)]} ^ {Postcondition(P1) -> [Precondition(P2) v Precondition(P3)]}
Altering (P1|P2);P3		{[Output(P1) v Output(P2)] -> Input(P3)} ^ {[Postcondition(P1) v Postcondition(P2)] -> Precondition(P3)}

Flowing은 프로세스들의 순차적 실행으로서, 프로세스 P1을 수행하고 뒤따르는 프로세스 P2를 수행하는 것을 의미한다. 이러한 순차적 관계에서 프로세스들간에 일관성이 존재하기 위해서는, P1의 output은 P2의 input에 포함되어야 하며, P1의 postcondition은 P2의 precondition을 만족시켜야 한다.

*Sharing*은 선행하는 프로세스의 *output*이 뒤따르는 복수개의 프로세스들에 의해 *input*으로 공유됨을 의미한다. 이때 프로세스들간에 일관성을 유지하기 위해서는 <표 1>에서 보듯이, 선행 프로세스 P1의 *output*은 뒤따르는 프로세스 P2와 P3의 *input*으로 활용되어야 한다. 그리고 P1의 *postcondition*은 P2와 P3의 *precondition*들을 만족시켜야 한다.

*Fitting*은 선행하는 두개의 프로세스 P1과 P2의 output들이 뒤따르는 프로세스 P3의 input으로 사용되는 경우이다. 일관성 유지를 위해서는 P1의 *output*과 P2의 *output*은 P3의 *input*에 포함되어야 하며, P1의 *postcondition*과 P2의 *postcondition*은 P3의 *precondition*을 만족시켜야 한다.

*Branching*은 분기 조건에 따라 선행하는 프로세스의 *output*이 뒤따르는 프로세스들 중, 하나

의 프로세스 *input*에 포함되어야 하는 경우이다. 분기가 이루어지는 후속 프로세스는 선행 프로세스의 *postcondition*에 따라 결정된다. Sharing 유형에서는 선행 프로세스의 *output*, O1이 후속 프로세스 P2와 P3에 사용되지만, Branching 유형에서는 O1이 P2나 P3 중의 한 프로세스에 사용된다. 이와 더불어, P1의 *postcondition*은 플로우에 따라 P2 또는 P3의 *precondition*을 만족시켜야 한다.

*Altering*은 선행 프로세스들 중, 하나의 프로세스 *output*이 후속 프로세스의 *input*으로 사용되는 경우이다. 선행하는 프로세스들 중의 한 프로세스가 후속 프로세스의 수행을 시작하게 한다. Fitting 유형에서는 P1과 P2의 output들이 공동으로 P3의 input으로 사용되지만, Altering 유형에서는 P1의 output 또는 P2의 output이 P3의 input으로 사용된다. 이와 더불어, P1의 postcondition 또는 P2의 postcondition 중의 하나가 P3의precondition을 만족시킨다.

5가지 기본 의존관계 유형을 바탕으로 프로세스들간의 다양한 의존관계 유형들을 프로세스 절차(procedure)로 모델링 할 수 있다. 하지만 대부분의 경우, 의존 관계들은 표의 기본 유형들과 같이 단순하지 않다. 일관성 테스트를 위해서는 복잡한 프로세스 절차를 기본 의존 관계 유형으로 분리시켜야 한다. 분리시키는 것은 ':' 단위로 이루어 질 수 있다. ':'을 기준으로 앞 부분과 뒷 부분은 하나의 분리(parsing) 그룹을 형성한다. 예를 들어, 하나의 절차가 (P1;(P2&P3);P4;P5)으로 정의된다며, 이는 세개의 분리 그룹으로 구분될 수 있다: Sharing (P1;(P2&P3)), Fitting ((P2&P3);P4) 그리고 Flowing (P4;P5). 이 경우, 일관성 테스트는 각 분리 그룹별로 수행된다.

4. 적용 사례

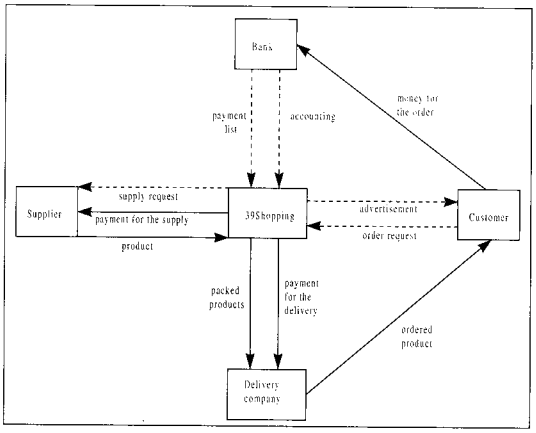
제안된 방법의 검증을 위해, 케이블 홈쇼핑 업체인 S 홈쇼핑의 전자상거래 설계에 코디네이션 기반 모델링 방법을 적용하였다. 케이블 홈쇼핑 업체의 기본 업무는 케이블 네트워크를 통해 광고를 방송하고 고객으로부터의 전화를 통해 주문을 받고 처리하는 것이다. 홈쇼핑 업무 지식과 절차들을 바탕으로 이 회사는 전자상거래를 위한 시스템을 개발하기로 결정하였다.

4.1 현 상황 분석

공급사슬은 <그림 2>처럼 공급사슬 참가자들 사이의 상호작용(interaction) 측면에서 분석 및 모델링된다. 상호작용 분석을 이용하여 객체를 통한 수행자들 즉, 업체들간의 의존 관계를 파악할 수 있다. 현 상황에는 다섯 수행자가 참여한다: S 홈쇼핑, 고객, 공급자, 배달회사, 그리고 은행. 상호 거래에 관여하는 객체는 두가지 유형으로 표시되는데, 정보(information) 유형은 점선으로 물질적(material) 유형은 실선으로 표시된다.

홈쇼핑 거래를 위해, S 홈쇼핑은 상품 광고 방송을 하기 전에 공급자들로 부터 상품을 공급받는다. 케이블 TV 네트워크를 이용하여 방송을 한 후, 고객들은 전화를 통해 S 홈쇼핑으로 주문을 한다. 고객이 주문에 대해 현금으로 지불을 한다면, 고객들은 지정된 기간 내에 은행을 직접 방문하여 대금 납부를 하게 된다. S 홈쇼핑은 은행으로부터 지급 목록을 받고 주문에 대한 납부를 확인한 후, 주문 상품을 포장하고 배달회사 또는 우체국을 통해 배달 업무를 수행한다. 만약 신용카드로 지급을 신청한다면, 카드 조회를 실행하고 배달 업무를 수행한다. 이러한 주문 프로세스 수행과 더불어, S 홈쇼핑은 상품 대금을 공급업체에게 그리고 배달비용을 배달업체에게 지급한다. 본 논문에서는 내용의 간결성을 위해 전자상거래 이후의 사후 서비스 프로세스는 생략하였다.

<그림 2>의 상호작용 분석을 통해, 대부분의 상호작용들이 S 홈쇼핑을 중심으로 밀집되어

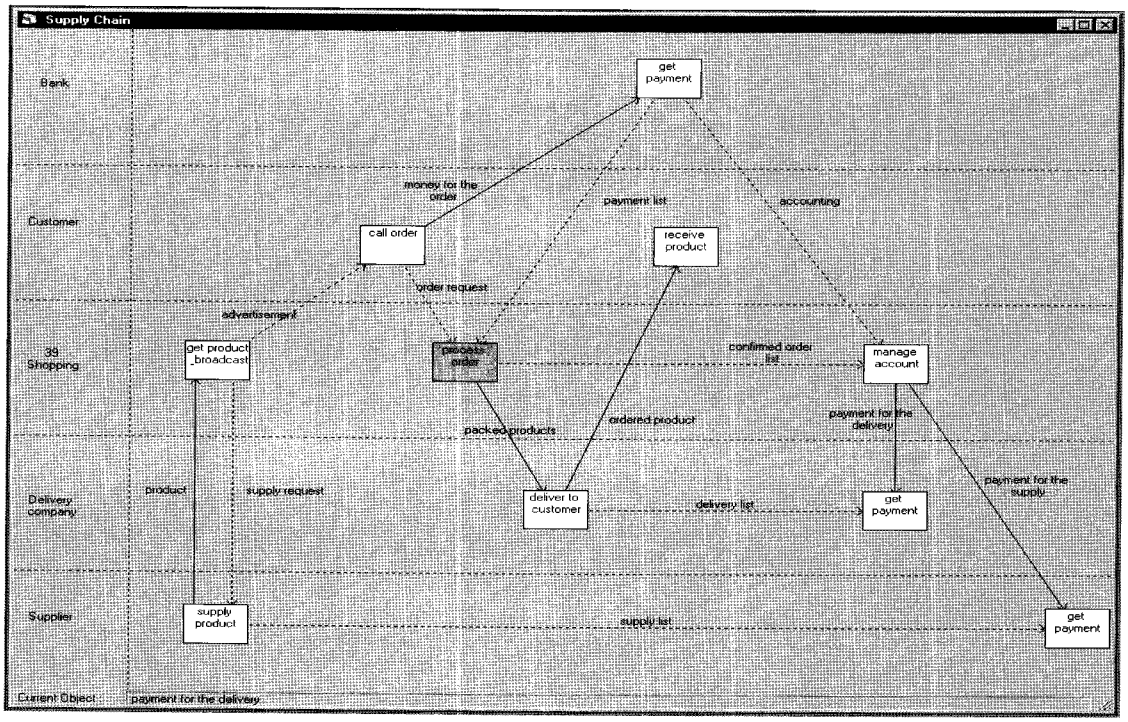


<그림 2> 현재의 공급사슬에 대한 상호 작용 분석

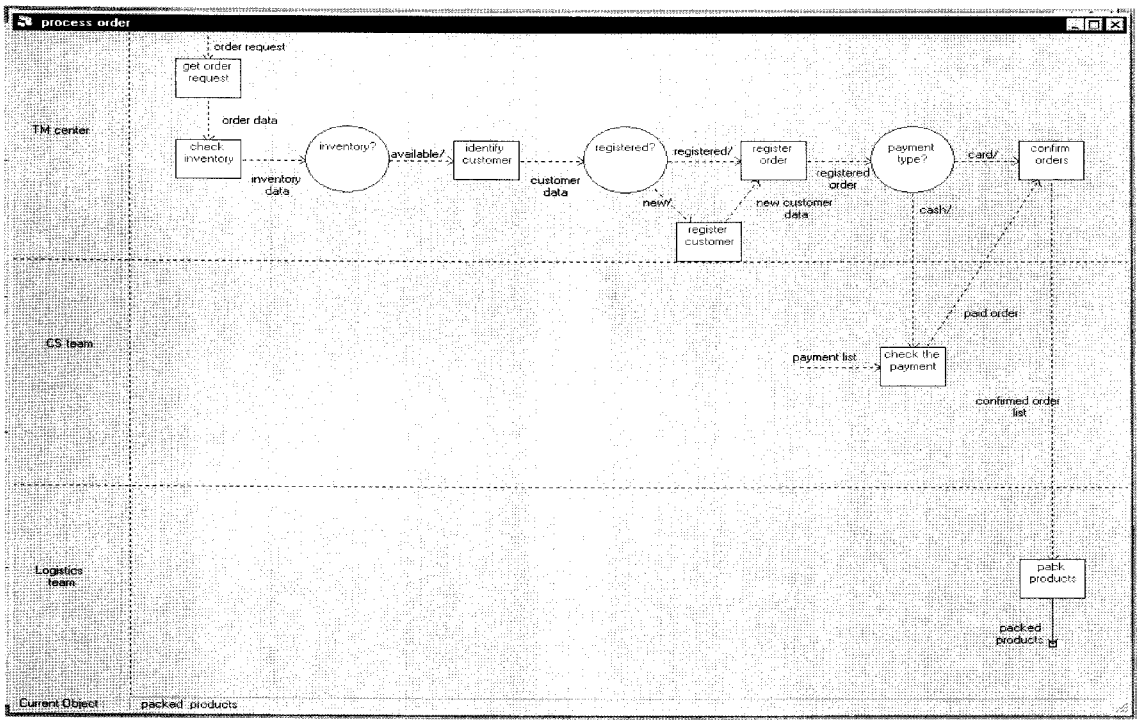
있음을 알 수 있다. 이는 다른 공급사슬 참여 업체에 비해, S 홈쇼핑이 더 많은 코디네이션 노력이 필요하다는 것을 의미한다. 코디네이션 노력이 더 필요할수록, 더 많은 행정 및 관리 활동들이 필요하며 이는 결국 더 많은 코디네이션 비용을 유발시킨다. 또한, 정보 객체 또는 물질 객체들이 부가가치 증진에 별로 기여하지 않는 프로세스 또는 장소(조직)를 거쳐야 함으로 인해 더 많은 시간을 필요로 하게 된다.

코디네이션을 위한 모델링을 위해 비쥬얼베이직을 이용한 모델링 도구를 개발하였다. 공급사슬 참가 업체들간의 조직간 프로세스 중심의 의존관계는 개발 도구를 이용하여 <그림 3>과 같이 분석할 수 있다. 상호작용 분석은 그림에서 보듯이 객체를 통해 수행자들 간의 상호 의존관계를 나타내는 반면, <그림 3>의 도식적 코디네이션 모델은 프로세스들, 프로세스와 객체들, 프로세스와 수행자들, 수행자와 객체들간의 상호 의존관계를 나타낸다.

프로세스 명세(specification)와 더불어, <그림 3>의 "process order" 프로세스의 내부 의존관계는 모델링 도구의 explosion 기능을 이용하여 <그림 4>와 같이 분석되고 모델링될 수 있다. 또한 "process order"는 다음과 같이 명세화 된다.



<그림 3> 조직간 의존관계: 현 상황



<그림 4> 조직내 의존관계: "Process order"

Process::= <process_name: process order> <input: order request AND payment list> <output: confirmed order list AND packed products> <precondition: (order request) occurred> <postcondition: (confirmed order list) prepared AND (packed products) prepared> <performed_by: 39 Shopping> <procedure: get order request; check inventory; (Stop || identify customer); (||register customer); register order; (||check the payment);confirm orders > <resource: order management system, customer database, product database, order database> <performed_by: 39 Shopping> <constraint: get order request only through telephone call>

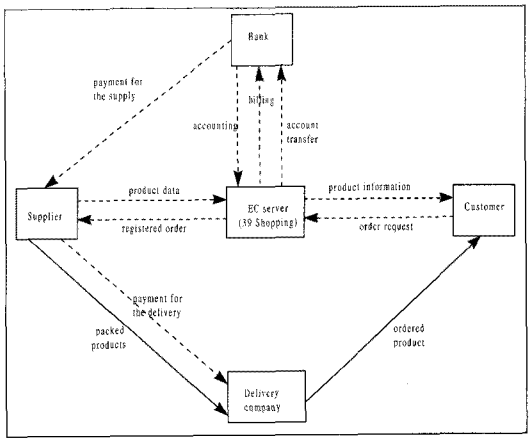
이 프로세스는 S 홈쇼핑의 세개 부서에 걸쳐 수행된다: Tele-Marketing (TM) center, Customer Service (CS) team, 그리고 Logistics team. 고객들로부터 주문을 받은 후, TM center의 텔레마케터들은 재고를 확인하고 고객 정보를 파악한다. 재고가 있으면, 고객이 신규 고객일 경우 고객 데이터를 등록하거나 기존 고객인 경우는 정보를 확인한 후, 텔레마케터들은 주문 내역을 등록한다. 주문에 대한 대금 납부가 현금으로 이루어 지는 경우, 주문을 처리하기 이전에 은행으로부터 팩스로 전달 받은 지급 리스트를 바탕으로 CS팀이 납부 여부를 확인한다. CS팀은 이외에도 고객 불만 관리와 같이 다른 고객 서비스 업무를 수행한다. 본 적용 사례에서는 간결성을 위해 이러한 고객 서비스 활동들을 포함시키지는 않았다. 마지막으로, 확정된 주문 리스트는 상품을 포장하고 배달하기 위한 입력 자료로 사용된다.

<그림 4>의 코디네이션 기반 모델은 조직 내 부서간 모드를 나타내지만, 직원 간의 코디네이션 모드에서도 프로세스 플로우에서 같은 의존관계를 나타낼 수 있다. 단 하나의 차이점은 조직 내 부서간 모드에서 수행자는 부서들인 반면, 직원간 모드에서의 수행자는 직원이라는 점

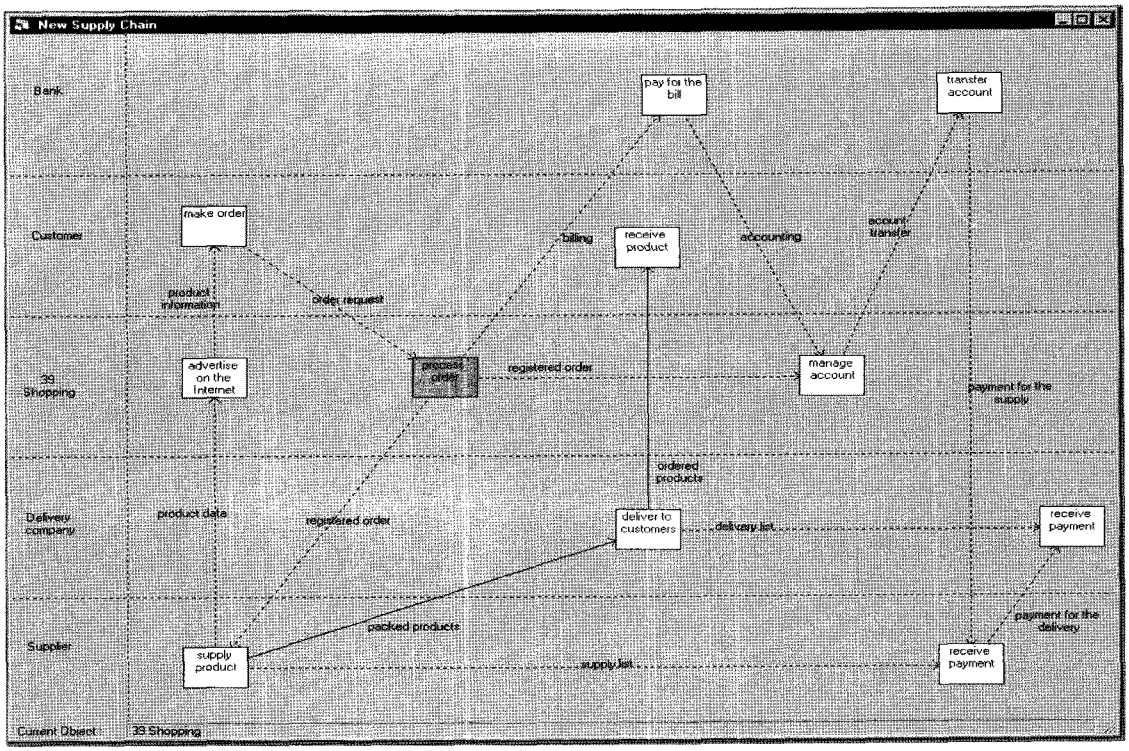
이다. 이와 더불어 모델링 도구를 이용하면 각 수행자에 대한 개인적 모드에서 의존관계를 분석하고 모델링 할 수 있다.

4.2 새로운 코디네이션을 바탕으로 한 전자상거래 개발

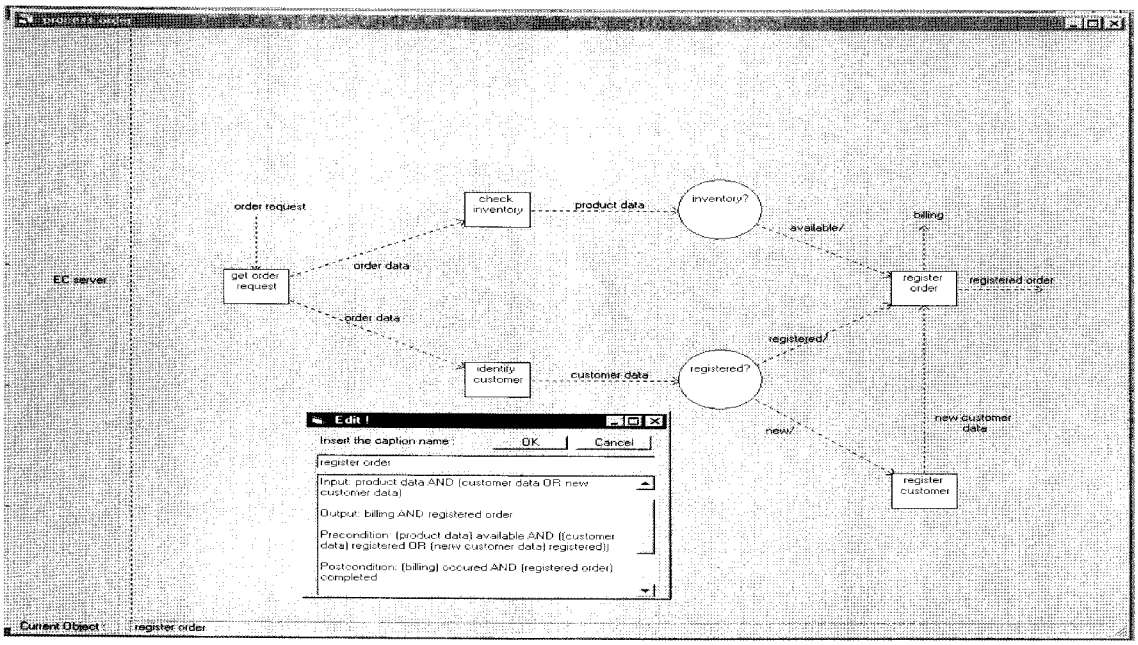
전자상거래 설계를 위해서는 우선 공급사슬 참가 기업들 간의 조직간 의존관계를 효율적으로 재설계하고 이와 연계하여 조직 내부의 의존관계를 재설계할 필요가 있다. 현재의 공급사슬은 대부분의 상호작용 활동들이 S 홈쇼핑에 집중되어 있음을 보여주었다. 따라서 일부 상호작용 활동들을 공급사슬의 타 참여 업체들로 이전시키는 것이 필요하다. 이를 위해 S 홈쇼핑은 고객의 주문리스트를 공급회사에 직접 전달하는 것을 고려했다. 공급회사는 이 자료에 따라 주문 상품들을 포장하고, 직접 배달회사와 협력하여 고객에게 배달하는 것을 고려했다. 이와 더불어 S 홈쇼핑은 고객에게 신용카드로 대금 납부하는 것을 요청한다. 공급업체에 대한 대금 지불은 은행을 통해 S 홈쇼핑의 계좌 이체 방



<그림 5> 변환된 공급사슬에서의 상호작용 분석



<그림 6> 조직간 의존관계: 변경된 사례



<그림 7> EC에 의한 의존관계: "Process Order"

식을 통해 이루어지도록 했다.

변환된 공급사슬에 대한 상호작용 모델은<그림 5>에 나와 있는 바와 같다. 이전의 상황과 비교해 볼 때, 변환된 공급사슬은 S 홈쇼핑 이외의 다른 관련 업체들간의 상호작용이 증가 했음을 보여준다. 이는 S 홈쇼핑에게 행정 및 관리 활동이 줄어들고, 코디네이션 노력이 감소하게 되는 것을 의미한다. 이와 더불어 이전 공급사슬 환경에서 부가가치 추가 효과가 적었던 물질적 객체의 흐름이 6개에서 2개로 단축되었다.

변환된 공급사슬은 <그림 6>과 같이 조직간 의존관계에 변화를 유발시켰다. 인터넷을 통해 고객으로부터 주문을 받은 후, S 홈쇼핑의 "process order" 프로세스는 상품 공급업체에 공급 주문을 하게 된다. 공급 주문에 따라 공급업체는 배달업체를 통해 상품을 고객에게 발송한다. S 홈쇼핑에 대한 공급업체의 지불 요청은 은행에 발송된다. S 홈쇼핑은 등록된 고객주문 자료와 은행으로부터의 회계자료를 바탕으로 공급업체에게 주문처리에 대한 비용을 지불한다. S 홈쇼핑으로부터 상품 대금을 받은 후, 공급업체는 배달업체에 배달에 대한 비용을 지불해야 한다.

전자상거래 적용을 위하여 재설계된 S 홈쇼핑의 내부 프로세스 "process order"의 내부 의존관계는 <그림 7>과 같이 모델링 될 수 있다. 그리고 그 명세 내용은 다음과 같다.

Process::= <process_name: process order> <input: order request> <output: registered order AND billing> <precondition: (order request) occurred > <postcondition: (billing) occurred AND (registered order) completed> <resource: EC system, product database, customer database, order database> <performed_by: EC Server> <procedure: get order request; ((check inventory; (Stop ||)) & (identify customer; (|| register customer))) ; register order > <constraint: get order request only through Internet>

순차적으로 수행되던 활동들이 -- "get order request", "check inventory", "identify customer"

-- 우선 병렬적으로 수행되게 된다. 그리고 이전의 3개 활동이 -- "check payment", "confirm orders", "pack products" -- 프로세스 흐름상에서 없어지거나 관련 업체로 이전되었다. 이와 더불어, 프로세스 수행에 있어 수행 규칙, 조건, 객체 명세, 그리고 수행자 등에 여러 가지 변화가 발생하였다. 이러한 변화 내용은 모델링 도구를 이용하여 <그림 7>의 "register order"와 같이 그 명세 데이터로 정의할 수 있다. "register order" 프로세스의 두개 산출물 즉, "registered order" 산출물은 공급자에게 "billing"은 은행에 전달된다. 또한 "register order" 산출물은 S 홈쇼핑 내의 "manage account" 프로세스를 수행하기 위해 회계부서(accounting department)로 전달된다. 하위 프로세스 및 관련 요소들에 대한 명세 내용은 Appendix 1을 참조하기 바란다.

코디네이션 모델을 이용하여 새로운 프로세스 흐름 및 의존 관계를 설계한 후, 프로세스 흐름상에서 코디네이션 요소들간의 일관성을 확인해야 한다. 일관성 테스트를 위해, 새로운 "process order" 프로세스의 procedure를 -- get order request; ((check inventory; (Stop ||)) & (identify customer ; (|| register customer))) ; register order -- 분리(parse)해야 한다. 그 결과 다음과 같이 4개의 부분으로 분리되며, 각 부분별로 일관성 테스트가 이루어질 수 있다:

1) Sharing: get order request; ((check inventory; (Stop ||)) & (identify customer; (|| register customer)))
2) Branching: check inventory; (Stop ||)
3) Branching: identify customer; (|| register customer)
4) Fitting: ((check inventory; (Stop ||) & (identify customer; (|| register customer))) ; register order

일관성 테스트 (Appendix 2) 결과, 모든 코디네이션 요소가 서로 상호 작용이 가능함을 알 수 있다. Procedure 명세 내용의 parsing 그룹과 일관성 조건은 언제 그리고 어떻게 프로세스가 활성화되는지를 나타낸다. 즉, 분리된 procedure에서 세미콜론(;)의 오른쪽 활동은 왼쪽 활동이 완료되고 전달되는 객체 및 조건을 바탕으로 하는 일관성 조건이 만족해야만 활성화될 수 있다. 워크플로우 개발과 더불어, 데이터베이스에 대한 정규화(normalization)가 필요하다. OBJECT 명세 데이터를 바탕으로 데이터베이스를 정규화 및 재설계를 수행하였으며, 다음은 그와 같이 재설계된 데이터베이스 모델 중에 간단한 예를 보여 주고 있다. :

Customer(customer-no, customer-name, telephone, customer-address)
Customer_Card(card-no, card-type, valid-through, customer-no)
Payment(blling-no, billing-amount, paid-date)
Product(product-no, product-name, product-spec, supplier)
Order(order-no, customer-no, billing-no)
Order_Product(order-no, product-no, order-amount)

4.3 적용 결과

공급사슬 참가 업체들간의 전자적 연계(Electronic chaining) 측면에서, 은행은 인터넷을 통해 S 홈쇼핑과 공급자들과 통합되게 된다. 공급 업체들도 S 홈쇼핑과 은행 및 배달업체들과 인터넷을 통해 연계되게 된다. 이전 상황과 비교해 볼 때, 더 많은 작업량이 공급업체에 부과된다. 이와 더불어, 공급업체들은 배달업체와 직접적 협력이 필요하게 된다. 하지만 공급사슬 참여 업체들간의 성공적인 전자적 연계를 위해서는 강력한 파트너쉽이 요구된다. 모든 주문 및 상품 관련 데이터들이 인터넷을 통해 다루어지므로, 참가 업체들간의 코디네이션이 더욱 중요하게 된다.

명세 데이터를 포함한 도식적 코디네이션 모델은 S 홈쇼핑의 전자상거래 개발에 있어 워크

플로우 및 데이터베이스 설계를 위한 기반이 되었다. 전자상거래 도입에 따라 공급사슬 참가 업체들간에 그리고 참가 업체 내부에 새로운 변화들이 발생하였다. 첫번째, S 홈쇼핑 측면에서는 집중되었던 상호작용(interaction) 업무가 10개에서 7개로 감소하였다. 결과적으로, S 홈쇼핑의 코디네이션 업무 부담은 다른 공급사슬 참가 업체들로 전가된다. 공급업체는 배달업체와 더불어 S 홈쇼핑의 재고 및 물류 담당 부서 역할을 담당한다. 두번째, S 홈쇼핑은 텔레마케터(tele-marketer)들이 없이 고객 주문을 접수하고 처리할 수 있다. 케이블-TV 홈쇼핑회사에서는 시간대에 따라 텔레마케터 인원을 적절히 배정하는 것이 중요한 의사결정 사항이었다. 이는 시간대에 따라 고객 주문량이 다르고, 주문처리 수용량은 텔레마케터 인원 수에 의존하기 때문이다. 이런 이유로 인해, S 홈쇼핑은 비록 텔레마케터들의 활용도가 평균 60퍼센트 미만이지만 그 인원을 감축하지 못했다[17]. 세번째, S 홈쇼핑 내 조직 요소들간의 상호 의존관계는 워크플로우 시스템 개발과 더불어 재구성되었다.

5. 토 의

조직간 의존관계는 제시된 세개의 코디네이션 요소들을 이용하여 성공적으로 모델링하고 관리할 수 있었다: 객체, 수행자, 그리고 프로세스. 이러한 코디네이션 요소들은 Spek and Sol(1997)이 조직간 의존관계에서 중요하다고 제시한 세 가지 요소 즉, 교환되는 항목(item), 참가 업체 및 지리적 위치와 관련한 공간적 요소, 그리고 프로세스 상에서 조직들간의 순차적 관계와 일치한다. 조직 내 의존 관계는 -- 조직 간, 조직 내 부서간, 조직 내 개인간, 개인적 -- 제안된 코디네이션 요소들을 이용하여 프로세스 흐름을 따라 관리될 수 있다. 공급사슬의 변경처럼 조직간 의존관계에서의 변화들은 도식적 모델과 명세 데이터를 이용하여 관리될 수 있다. 변화로 인해 다른 요소에 끼치는 효과는 명세 데이터를 이용하여 파악할 수 있으며, 이를 이용하여 비즈니스 프로세스 변화 또는 워크플로우 변화 대안에 대한 가능성(feasibility) 체크할 수 있다.

조직들간 그리고 조직 내의 코디네이션과 더불어, 조직간 그리고 조직 내의 프로세스들간의 연계 또한 제안된 코디네이션 기반 모델링 방법을 이용하여 가능하다. 새롭게 코디네이트 된 공급사슬은 조직간의 경계에서 상호 작용 활동들에 영향을 미친다. 새롭게 설계된 조직간 상호작용 활동들은 조직 내부 특히, 부서간 레벨에서 관련 요소들에 대한 코디네이션을 요구하게 된다. 새로운 조직 설계 안은 개인간 그리고/또는 개인적 코디네이션 모델과 더불어 시스템 개발에 연계될 수 있다. 마지막으로, 코디네이션 모델을 바탕으로 워크플로우와 데이터베이스 설계가 가능하고 이와 더불어 전자상거래를 설계할 수 있었다.

조직간 그리고 조직 내부의 부서간 업무간 의존 관계가 적절치 못하면 성과에 부정적 영향을 미칠 뿐만 아니라 상대적으로 높은 코디네이션 비용을 유발시킨다. 이처럼 업무들 간의 코디네이션 뿐만 아니라 조직 내부의 부서간, 부서와 업무간, 업무와 관련 객체간의 코디네이션은 조직 설계의 주요 요인으로 제시되고 있다[22]. 물질적 흐름과 정보적 흐름의 분류에 기반한 코디네이션 방법은 의존관계를 재설계하고 코디네이션 작업량을 추정하는데 유용하다. 본 연구에서는 전자상거래 설계 과정에 물질적 객체의 이동 거리 및 이동 횟수를 가능한 줄이려고 노력하였으며, 이와 관련하여 물질적 객체를 정보적 객체로 대체시키려 하였다.

제안된 코디네이션 접근 방식은 다른 코디네이션 모델과 비교될 수 있다. 첫번째, Process Handbook 프로젝트[22]의 Dependency Diagram과 비교될 수 있다. Dependency Diagram은 활동(activity)들과 자원(resource)들과의 의존관계

를 모델링하는 것이 주 목적인 반면, 수행자와 관련하여 연관된 의존관계를 관리할 수 없다. 두번째, organizational coordination model[7]과 비교할 수 있다. 본 연구에서는 조직의 프로세스들과 그들의 코디네이션에 대한 다각적 분석을 위해 상호작용(interaction) 모델, 프로세스 모델, 그리고 수행자 모델 제시하였다. 상호 작용 모델은 수행자들 그리고/또는 데이터베이스와 같은 저장소(repository) 간의 항목 교환을 기술한다. 프로세스 모델은 프로세스 흐름과 수행자와 저장소들의 활동 간의 상호 의존관계를 서술한다. 수행자 모델은 개별 수행자들의 활동들 간에 상호 의존관계를 표현한다. 이러한 세 가지 모델들은 비즈니스 상황의 다양한 측면을 모델링 하는데 사용될 수 있지만, 이 접근법은 개인간 코디네이션 모드에 초점을 둔다. 이와 더불어, 조직 변화를 시스템 설계로 변형하는 것에 대한 고려가 없다. 세번째, 제안된 방법은 다른 워크플로우 모델들[1, 4]과 비교될 수 있다. 활동들과 다른 코디네이션 요소들간의 상호 의존관계는 워크플로우 모델을 이용하여 관리될 수 있다. 워크플로우 명세 데이터는 시스템 구현을 위한 기반이 된다. 하지만, 대부분의 워크플로우 모델들의 사용은 개인간 또는 개인 모드에서의 관리에 한정된다. 마지막으로, 본 연구에서의 접근 방식은 IS 아키텍쳐[2]와 비교될 수 있다. IS 아키텍쳐 개발의 주요 초점은 정보시스템의 청사진 설계에 있다. 이 접근 방식은 비록 기능(function), 데이타, 조직, 그리고 통제 아키텍쳐 간의 코디네이션을 지원했지만, 조직간 그리고 조직 내부 부서간 그리고 시스템 개발까지의 연계에 대해 충분한 고려를 하지 않는다.

6. 결 론

전자상거래 설계 및 개발에 따라 기업들은 다른 공급사슬 참가 업체들과 새로운 형태의 상호작용 및 다른 구성 단위들과의 새로운 조직적 형태를 경험해 왔다. 전자상거래의 적용은 공급사슬의 변형 뿐만 아니라 조직 프로세스의 변화까지 유발하므로, 의존 관계에 있는 요소들 간의 관계를 관리하는 것은 성공적 전자상거래의 대표적 특성인 높은 성과와 낮은 코디네이션 비용을 위해 필수가 된다. 제안된 코디네이션 기반 모델링 방법은 공급사슬 참가 업체들 간의 수평적 의존관계와 조직 내부 요소들간의 수평적 의존관계를 관리할 수 있다. 또한 조직간 설계와 조직 내부의 설계 또한 전자상거래 시스템 개발까지의 수직적 연계도 관리할 수 있다. 제안된 코디네이션 접근 방식은 일관성 테스트와 명세 데이터를 이용하여 조직간 또는 조직 내에서의 한 변화가 다른 의존 관계의 요소에 어떤 영향을 미치는지 파악할 수 있게 해준다. 이는 효과적 전자상거래 설계를 위한 기반이 된다. 본 논문에서는 내용의 간결성을 위해 주문 처리 프로세스에 초점을 맞추고, 고객 서비스 관련 프로세스는 생략하였다. 고객 서비스 및 불만 처리를 위해서는 여러 부서 간의 협력 즉, 코디네이션이 필요하다. 이러한 내용이 전자상거래 설계에 반영되어 고객 요구 사항에 대한 처리와 더불어 진행 상황에 대한 관리 및 통제도 체계적으로 수행할 수 있다.

본 연구의 의의로는 이론과 실무 측면에서 다음의 몇 가지를 들 수 있다. 이론적 측면에서, 조직간 그리고 조직 내부간의 수평적 그리고 수직적 의존관계를 관리하기 위해 코디네이션 이론을 적용하였다. 제안한 코디네이션 접근방식에서, 조직 모델을 바탕으로 코디네이션 요소 및 그들 간의 상호 의존관계를 파악하였다. 다음으로 코디네이션 기반 모델링 방법을 제시하였다. 이 방법은 조직간, 조직 내 부서간, 조직 내 개인간, 그리고 개인적 코디네이션을 위해 활용된다. 실무적 측면에서의 시사점은 전자상거래의 개념과 시스템 개발 간의 변환 갭을 감소시키는데 있다. 개념 레벨에서 지적된 변화와

코디네이션 요소들간의 상호 의존관계는 쉽게 파악되고 시스템 설계에 반영이 용이하다. 명세 데이터와 더불어 제시한 코디네이션 모델은 목표로 하는 전자상거래의 워크플로우 시스템과 데이터베이스의 기반이 된다.

〈참 고 문 헌〉

[1] A. Cichocki, A. Helal, Rusinkiewicz, and D. Woelk, Workflow and process automation: Concepts and Technology, Kluwer Academic Publishers, 1998.

[2] A. Sheer, Architecture Of Integrated Information Systems, Springer-Verlag, 1992.

[3] A.H. Van de Ven, A.L. Delbecq, and R. Koenig, Determinants of coordination modes within organizations, American sociological review, 41, (April 1976) 322-338.

[4] D. Georgakpoulos, M. Hornick, and A. Shethe, An overview of workflow management: From process modeling to workflow automation infrastructure, Distributed and Parallel Databases, 3, (1995) 119-153.

[5] D.A. Garvin, The processes of organization and management, Sloan Management Review, (Summer 1998) 33-50.

[6] E.K. Clemons, M.E. Thatcher, and M.C. Row, Identifying sources of reengineering failures: A study of the behavioral factors contributing to reengineering risks, Journal of Management Information Systems, 12, 2, (1995) 9-36.

[7] G. Vreede and D. Eijck, Modeling organizational coordination, Simulation & Gaming, 29, 1, (1998) 60-87.

[8] H.J. Levitt, Applied organizational change in industry: Structural, Technological, and Humanistic Approach, Handbook of Organizations, James G. March (Eds.), Rand McNally: Chicago, 1965, pp. 1144-1170.

[9] J. Fulk and DeSanctis, Electronic communication and changing organizational forms, Organization Science, 6, 4, (July-August 1995) 337-349.

[10] J. Sivori, Evaluated receipts and settlement at Bell Atlantic, Communications of the ACM, 39, 6, (June 1996) 24-28.

[11] J.D. Thomspon, Organizations in Action: Social science bases of administrative theory, New York: McGrow-Hill, 1967.

[12] J.R. Galbraith, Organization design, Addison Wesley, Reading, MA, 1977.

[13] J.R. Galbraith, Competing with Flexible Lateral Organizations, Addison-Wesley Publishing Co., 2nd(Eds.), 1993.

[14] K. Crowston, A taxonomy of organizational dependencies and coordination mechanism, Working paper, Center for Coordination Science, MIT, 1994.

[15] K. Crowston, A coordination theory approach to organizational process design, Organization Science, 8, 2, (Mar-April 1998) 157-175.

[16] K.D. Mackenzie, The Process Approach to Organizational Design, Human Systems Management, 8, (1989) 31-43.

[17] KAIST, IS Architecture Development for the Integrated Information Systems at 39 Shopping, Project Report, Korea Advanced Institute of Science and Technology, 1997.

[18] M. Hammer, Reengineering Work: Don't Automate, Obliterate, Harvard Business

Review, (July-August 1990) 104-112.

[19] R. Kalakota and Whinston, Electronic Commerce: A manager's guide, Addison-Wesley, 1997.

[20] R. Spek and H. Sol, Multimodeling, a group approach to modeling interaction within interorganizational systems, Proceedings of the Thirtieth Annual Hawaii International Conference on System Science, IV, (1997) 179-188.

[21] T.W. Malone and K. Crowston, The interdisciplinary study of coordination, ACM Computing Surveys, 26, 1, (March 1994) 87-119.

[22] T.W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. Osborne, and A. Bernstein, Tools for Inventing Organizations: Toward a handbook of organizational processes, Management Science, 43, 3, (1999) 425-443.

[23] V. Grover, S. Jeong, W. Kettinger, and J. Teng, The implementation of business process reengineering, Journal of Management Information Systems, 12, 1, (1995) 109-144.

[24] Jintae Lee, et al., "The PIF Process Interchange Format and Framework," Working paper # 194, Center for Coordination Science, Sloan School of Management, MIT, 1997.

[25] Nilsson, A.G., "Business modeling as a base for information systems development", Proceedings of the third international conference on information systems developers workbench ? Methodologies, Techniques, Tools, and Procedures, Gdansk, Poland, September 1992.

Appendix 1: "Process Order"의 세부 명세 내용

Process::= <process_name: get order request> <input: order request> <output: order data> <precondition: (order request) given> <postcondition: (order data) prepared> <resource: EC system> <performed_by: EC Server> <constraint: get order request through Internet>

Process::= <process_name: check inventory> <input: order data> <output: product data> <precondition: (order data) prepared> <postcondition: (product data) available OR (product data) not available> <resource: product database> <performed_by: EC Server>

Process::= <process_name: identify customer> <input: order data> <output: customer data> <precondition: (order data) prepared> <postcondition: (customer data) exist OR (customer data) new> <resource: customer database> <performed_by: EC Server>

Process::= <process_name: register customer> <input: customer data> <output: new customer data> <precondition: (customer data) new> <postcondition: (new customer data) registered> <resource: customer database> <performed_by: EC Server>

Process::= <process_name: register order> <input: product data AND (customer data OR new customer data)> <output: registered order> <precondition: ((customer data) exist OR (new customer data) registered) AND (product data) available> <postcondition: (registered order) completed> <resource: order database> <performed_by: EC Server>

Object::= <object_name: order request> <object_spec: {product_no + amount} + customer_no|(customer_name+telephone_no)> <related_with: customer data, product data>

Object::= <object_name: order data> <object_spec: customer_no + customer_name + telephone_no +{product_id+ product_name+ order_amount} + delivery_address> <related_with: customer_data, product data>

Object::= <object_name: customer data> <object_spec: customer_no + customer_name + telephone + {card_type + card_no + valid_through}> <related_with: order data>

Object::= <object_name: new customer data> <object_spec: customer_no + customer_name + telephone + {card_type + card_no + valid_through}> <related_with: order data>

Object::= <object_name: registered order> <object_spec: order_no + customer_no + {product_no + product_name+ order_amount} + delivery_address + order_date + billing_no + billing_amount> <related_with: customer data, product data>

Actor::= <actor_name: EC Server> <role: order processing>

Appendix 2: 일관성 테스트

1) Sharing: get order request; ((check inventory; (Stop||)) & (identify customer; (|| register customer)))

• Output(get order request) -> (Input(check inventory; (Stop||)) ^ Input(identify customer; (|| register customer))) : 승인

① Output(get order request) = {order data}

② Input(check inventory; (Stop||) = {order data}

③ Input(identify customer; (|| register customer)) = {order data}

• Postcondition(get order request) -> (Precondition(check inventory; (Stop||) ^ Precondition(identify customer; (|| register customer))) : 승인

① Postcondition(get order request) = {(order data) prepared}

② Precondition(check inventory; (Stop||) = {(order data) prepared}

③ Precondition(identify customer; (|| register customer)) = {(order data) prepared}

2) Branching: check inventory; (Stop||)

• Output(check inventory) -> Input(Stop||): 승인

① Output(check inventory) = product data

② Input(Stop||) = product data

• Postcondition(check inventory) -> Precondition (Stop||): 승인

① Postcondition(check inventory) = (product data) available OR (product data) not available

② Precondition(Stop||) = (product data) not available

3) Branching: identify customer; (|| register customer)

• Output(identify customer) -> Input(|| register customer) : 승인

① Output(identify customer) = {customer data}

② Input(|| register customer) = {customer data}

• Postcondition(identify customer) -> Precondition (|| register customer) : 승인

① Postcondition(identify customer) = (customer

data) exist OR (customer data) new

② Precondition(|| register customer) = (customer data) new

4) Fitting: ((check inventory; (Stop ||) & (identify customer; (|| register customer))) ; register order

• Output((check inventory; (Stop ||) & (identify customer; (|| register customer))) -> Input (register order): 승인

① Output((check inventory; (Stop ||) & (identify customer; (|| register customer))) = {product data AND (customer data OR new customer data)}

② Input(register order) = {product data AND (customer data OR new customer data)}

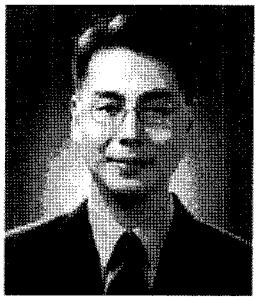
• Postcondition((check inventory; (Stop ||) & (identify customer; (|| register customer))) - > Precondition (register order): 승인

① Postcondition((check inventory; (Stop ||) & (identify customer; (|| register customer))) = ((product data) available AND (customer data) exist) OR (new customer data) registered

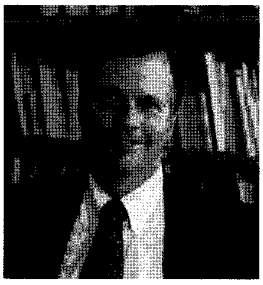
② Precondition (register order) = ((customer data) exist OR (new customer data) registered) AND (product data) available

◈ 이 논문은 2000년 4월 9일 접수하여 1차 수정을 거쳐 2000년 12월 11일 게재확정 되었습니다.

◆ 저자소개 ◆



김희웅 (Kim, Hee-Woong)
한국과학기술원에서 경영공학 박사 학위를 취득하고, MIT의 Sloan School of Management에서 (Post-Doc) 연구원으로 근무하였으며. 현재 LG-EDS 컨설팅 부문에서 근무하고 있다. 연구 분야는 System Dynamics를 이용한 지식 경영 및 경영 전략 수립, e-Business 설계, 정보시스템 아키텍쳐 개발 등이다. 그 동안 Information & Management, Decision Support Systems, Journal of Information Technology Management, International Journal of Flexible Manufacturing Systems 등에 논문들이 게재되었으며 ICIS, HICSS, DSI, European Simulation Symposium 등의 학회에서 논문을 발표하였다.



Thomas W. Malone 박사
Stanford 대학에서 공학박사 학위를 취득하고 현재 MIT, Sloan School of Management에서 교수로 재직 중에 있다. 그리고 Sloan 소속의 Center for Coordination Science의 책임자이다. 그의 연구 분야는 정보시스템을 활용한 새로운 형태의 조직 설계에 초점을 맞추고 있다. 그는 50여 편의 논문을 저널에 게재했으며, Management Science, Journal of Organizational Computing 등 여러 저널의 편집위원을 담당하고 있다.