경영정보학연구
제11권 제1호
2001년 3월

B2B 워크플로우의 메시징 시스템 설계

서 창 교*, 김 정 삼**, 이 형 석***

Design of a Messaging System for B2B Workflow

Suh, Chang-Kyo, Kim, Jeong-Sam, Lee, Hyung-Seok

B2B (business-to-business) commerce has become the prime driver of contemporary electronic commerce. Under B2B commerce, corporations often must operate across organizational boundaries to share their business processes. Workflow management was proposed by Aalst (4, 5) to support several business partners that are involved in shared workflow processes in B2B commerce.

We designed a messaging system for B2B workflow, where heterogeneous workflow management systems on each organization for trade were integrated. Based on Aalst's example in (4, 5), we analyzed B2B workflow by using class diagram, use case diagram, activity diagram, and statechart diagram of UML, and designed the messaging system. We also demonstrated a prototype system which was implemented by using Java API and XML. To compare with a holistic system such as EDI systems, the messaging system allows the business partners in B2B commerce to communicate with each other by dedicated messages and integrate each B2B interoperable workflow without any restrictions

* 경북대학교 경영학부 부교수
** 데이콤 시스템 테크놀러지
*** 경북대학교 대학원 경영학과 박사과정

Ⅰ. 서 론

Kalakota & Whinston[14]은 전자상거래, 가상 조직(virtual organizations), 확대된 기업(extended enterprises) 등의 부상으로 다수의 조직을 넘나드는 비즈니스 프로세스가 계속 증가할 것이라고 예측하였다. 오늘날 크게 성장하고 있는 Business-to-Business(B2B) 전자상거래는 다수 조직의 연관된 비즈니스 프로세스의 대표적인 형태이며 B2B 전자상거래를 지원해 줄 수 있는 전통적인 정보기술로는 EDI(Electronic Data Interchange)를 들 수 있다.

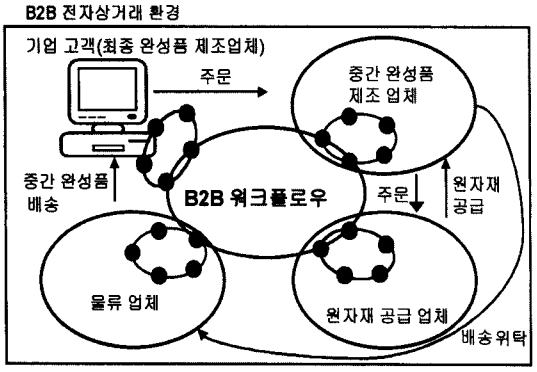
최근, 협업과 전자 문서 교환을 지원하는 워크플로우(workflow) 기술이 B2B 전자상거래에 적합한 새로운 정보기술로 인식되기 시작하였다[4-6, 17, 26, 27]. 조직 내의 자동화된 업무 프로세스 진행과 문서 교환의 목적으로 출발한 워크플로우 기술은 비즈니스 프로세스를 전사적으로 관리하여 이를 B2B 전자상거래에 적용하게 되면 조직 내부에서 외부로 이어지는 다수 조직의 비즈니스 프로세스 전체가 하나의 프로세스로 관리·통제·감시·최적화 될 수 있다. WfMC(Workflow Management Coalition)[25]에서는 워크플로우를 "일련의 절차에 따라 한 참여자에서 다른 참여자로의 문서와 정보 혹은 업무가 전달되는 비즈니스 프로세스의 전부 혹은 부분적인 자동화"로 정의하였다. 일반적으로, 워크플로우에 있어 자동화의 의미는 생산라인에서의 자동화와 같은 복잡하지만 정형화된 틀 속의 완벽한 자동화를 의미하지는 않으며, 워크플로우는 기계적이고 소프트웨어적으로 자동화된 프로세스뿐만 아니라 인간에 의한 수동적 업무와 외부 조직의 프로세스까지도 이어지는 전체 비즈니스 프로세스 관점에서의 비교적 비정형적인 자동화를 의미한다.

워크플로우는 여러 면에서 다른 정보기술과 공통점 혹은 차이점을 지닌다. 워크플로우 개념은 제조와 사무의 프로세스 개념에서 발전된 것으로[12], Hollingsworth[13]는 워크플로우 기술 도입의 주요 동기요인으로 최소의 리엔지니어링으로 비즈니스 프로세스에 유연성을 제공해 주는 것을 제안하였으며, Merz et al.[17]은 워크플로우 관리는 프로세스 위주의 작업들을 통합하는데 특별한 능력이 있는 기술로 평가하였다. 한편, Cichocki et al.[9]은 일부 워크플로우의 특성을 설명하면서 종업원에 의해 수행되는 거래처리 워크플로우가 CSCW(Computer Supported Cooperative Work)의 특질과 유사함을 지적하였으며, Bolcer & Taylor[7]는 Ad-hoc 워크플로우가 CSCW 영역에서 유래했다고 하였다. 워크플로우 관리(workflow management)는 비즈니스 프로세스의 자동화를 통해 정확하면서도 신속한 업무 처리를 지원하는 정보기술이며, 워크플로우 관리 시스템(workflow management systems)은 워크플로우를 분석, 모형화, 실행, 통제, 감시등을 수행하는 시스템을 의미한다. 현재 대략 200여 개의 상업적인 워크플로우 관련 제품이 존재하는데[5], 지금까지의 주된 응용 분야로는 정부나 교육기관과 같은 공공부문의 일반 관리와 서비스 업무, 금융·보험업과 정보통신 서비스업의 대 고객 업무, 제조업의 일반 관리 업무들이다[2]. 이러한 응용 영역 업무의 보편적인 특징은 대량의 복잡한 업무이면서, 비교적 정형화된 업무란 점이다.

하지만, 최근에 급격히 발전하고 있는 B2B 전자상거래는 워크플로우 기술의 응용 영역을 보다 확장하도록 요구하고 있다. 이는 기존의 워크플로우 개념이 주로 단일 조직 내부의 비즈니스 프로세스에만 초점을 맞추고 있는 반면, 새로운 워크플로우 개념은 B2B 전자상거래 환경과 같이 다수의 조직에 의해 공유 혹은 교환되는 비즈니스 프로세스에 초점을 맞출 필요가 있다는 것이다.

본 연구에서는 B2B 워크플로우를 "B2B 전자상거래 환경에서 상거래 수행과 협업을 목적으

로 다수의 조직이 협력하여 수행하는 워크플로우"로 정의하였다. B2B 워크플로우를 처음 시작되는 기업 고객의 입장에서 살펴보면, 최초 주문을 낸 후 최종적으로 자사로 배송되는 전 과정이 다수의 타 기업 워크플로우들과 함께 연결되어 형성된 일종의 가상적인 워크플로우를 자사의 비즈니스 프로세스인 것처럼 추적, 통제할 수 있어야 한다. 이러한 B2B 워크플로우의 특징은 독자적인 워크플로우 수행 능력을 가진 중간 완성품 제조 업체, 원자재 공급 업체, 물류업체 등과 같이 B2B 워크플로우 수행에 수동적으로 참여하고 있는 기업들에게도 마찬가지로 적용된다. 예를 들어 최종 완성품 제조업체인 기업 고객이 자사의 주문 워크플로우 수행의 일부로 협력관계에 있는 중간 완성품 제조업체에 주문을 하게 되면, 중간 완성품 제조 업체의 주문처리 워크플로우가 자동적으로 수행되고, 정해진 절차로 그 주문을 자동 처리해 나간다. 만일, 자사의 원자재 재고가 부족하면 마찬가지로 워크플로우 수행의 일부로 협력사인 원자재 공급 업체에 주문을 자동으로 하게 된다. 중간 완성품을 기업 고객에게 배송할 때는 자사의 주거래 물류업체에 배송을 위탁하게 되는데, 이러한 과정도 자사 워크플로우의 일부로 수행된다 (<그림 1> 참고).



<그림 1> B2B 워크플로우 예

다수의 조직이 참여하여 수행하는 워크플로우 연구 중에서 느슨히 연결된 구조(loosely coupled architecture)는 각 조직의 독립적인 워크플로우를 메시징 시스템을 통해 하나로 연결하는 방법으로 제시되었다[4, 5]. 이 방법의 장점은 각 조직이 운영하는 워크플로우 관리 시스템을 그대로 유지하면서, 추가적인 메시징 시스템의 개발 노력만으로, EDI 시스템과 같은 고비용의 시스템 개발 없이 B2B 전자상거래를 가능하게 하고, 거래 상대 기업과 절차가 바뀌더라고 유연하게 대처할 수 있게 한다는 점이다[8, 15]. 그러나, B2B 전자상거래 수행을 위한 느슨히 연결된 워크플로우 구조는 메시징 기술의 잠재력에도 불구하고, 현재까지 이의 구현과 관련된 연구와 응용은 미흡한 실정이다.

따라서, 본 연구에서는 워크플로우 및 메시징 기술과 관련한 연구에 대한 동시적인 고찰과 함께 실제 운용 가능한 메시징 시스템을 분석 및 설계하였으며, 프로토타입 시스템의 구현을 제안하였다. 이를 위해서 2장에서 이론적 배경으로 B2B 워크플로우의 개념과 메시징 시스템에 관한 연구들을 정리하였다. 3장 시스템 분석에서는 가상의 B2B 워크플로우를 설정하고, 이러한 B2B 워크플로우를 수행할 수 있도록 해줄 수 있는 메시징 시스템을 분석하였으며, 4장 시스템 설계에서는 실제 메시징 시스템이 가지는 구성요소 각각의 구조적 측면과 행동적 측면을 각각 설계하였다. 5장 프로토타입 시스템 구현에서는 설계된 시스템의 일부를 자바(Java) 언어로 실제 프로그래밍 하였고, 결론에서는 본 연구의 요약과 함께 한계점 및 향후 연구 방향을 정리하였다.

시스템 분석 및 설계 단계에서는 객체 지향적 분석·설계 방법을 채택하여, 예제 시나리오를 UML(the Unified Modeling Language)의 클래스도(class diagram), 쓰임새도(use case diagram), 활동도(activity diagram), 순차도(statechart diagram) 등을 기초로 B2B 워크플로우를 설정하고 분석한 후, 이를 지원하는 메시징 시

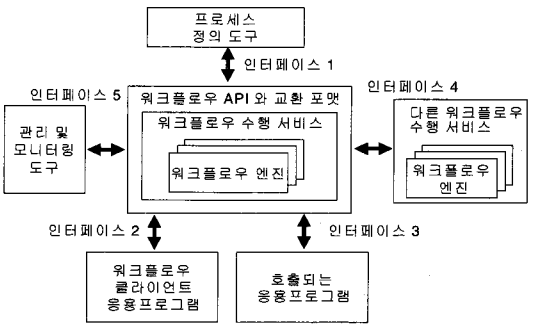
스템을 추가적으로 분석·설계하였다. 프로토타입 구현 프로그래밍 언어로는 객체지향 언어로 널리 쓰이고 있는 자바를 이용하였고, 메시지 포맷으로는 EDI를 비롯해 최근 많은 영역에서 메시지 포맷 언어로 널리 쓰이는 XML(the eXtensible Markup Language)을 이용하였다.

Ⅱ. 이론적 배경

2.1 워크플로우 관리시스템

워크플로우와 관련된 표준 제정은 WfMC에서 주로 맡고 있다. WfMC는 IBM사, MS사, 핸디소프트사 등 전세계 약 200여개의 워크플로우 개발업체들의 연합으로 운영되는 비영리단체로서 워크플로우 기술과 관련된 표준을 제정, 배포하고 있다. 워크플로우와 관련된 또 다른 표준은 CORBA(Common Object Request Broker Architecture) 표준을 제정한 OMG(Object Management Group)에 의해 제정되었다. 이 표준은 WfMC의 참조 모델을 기초로 워크플로우 어플리케이션이 가지게 될 객체들의 명세를 제시한 것이다[19].

WfMC에서 제시한 참조 모델로서 워크플로우 관리 시스템의 일반적 구성요소는 크게 여섯 개로 나뉘어 있다[21] (<그림 2> 참고). 가장 핵심적인 역할은 중앙의 워크플로우 수행 서비스로서 구성요소 전부를 하나의 시스템으로 묶어 주는 역할을 한다. 나머지 다섯 개의 구성요소들은 인터페이스를 통해 워크플로우 수행 서비스와 교류하게 되는데, WfMC에서는 각 인터페이스 명세를 표준 문서로 제공한다[21, 22, 23, 24, 25, 26, 27]. 이러한 인터페이스를 표준으로 정한 이유는 워크플로우 관리 시스템의 여섯 개 구성 요소가 상이한 업체에 의해 개발될 수 있기 때문이다. 따라서, 각 구성요소들이 해당 인터페이스를 정확히 구현하였다면, 상이한 업체가 개발한 구성요소들로도 전체 시스템을 구성할 수 있다. 워크플로우 수행 서비스 외 나머지 다섯 가지 구성요소 각각을 설명하면 다음과 같다.



<그림 2> WfMC 참조 모델[WfMC, 1995]

첫 번째, 프로세스 정의 도구들은 워크플로우의 원형인 프로세스 정의를 인터페이스 1을 통해 BNF(Backus-Naur Form)와 유사한 형식 언어인 WPDL(Workflow Process Definition Language)로 작성하게 하는 도구이다. 대개 사용자 측면을 고려하여 프로세스 정의를 GUI 소프트웨어로 작성할 수 있게 개발되는데, 내부 형식 언어가 반드시 WPDL일 필요는 없으나, 대개 WPDL 형식으로 변환하는 기능을 가진다.

두 번째, 워크플로우 클라이언트 응용프로그램은 워크플로우 참여자 혹은 수행자가 현재 수행 중인 워크플로우에서 자신들이 처리해야 할 워크플로우의 작업 목록을 워크플로우 수행 서비스에 요청하는 기능을 인터페이스 2를 통해 제공한다.

세 번째, 호출되는 응용프로그램들은 워크플로우에 참여할 수 있는 조직 내 외부의 ERP, EDI, MIS 등으로서, 해당 시스템을 워크플로우에 참여시키기 위해서는 인터페이스 3의 명세를 구현 할 필요가 있다.

네 번째, 다른 워크플로우 수행 서비스는 대개 해당 부서 혹은 사업부의 워크플로우 엔진과 연동되는 조직 내부 혹은 외부의 워크플로

우 엔진들이다. 인터페이스 4를 구현하는 내용과 정도에 따라 워크플로우 엔진들 간에 다양한 수준의 상호운용성이 가능하다. 이러한 상호운용성은 대개 조직 내부에서의 작업 부하 분산 혹은 부서간 업무 통합 등을 목적으로 이루어진다.

마지막으로, 관리 및 모니터링 도구들은 프로세스 정의 혹은 워크플로우 원형으로 인터페이스 5를 통해 프로세스 인스턴스 혹은 워크플로우 사례를 생성하고, 이를 워크플로우 전체 수준에서 추적, 감시, 제어한다.

2.2 B2B 워크플로우

Alonso et al.[6]은 WISE(Workflow based Internet SErvices) 프로젝트 사례를 통해 워크플로우 관리 시스템을 B2B 전자상거래를 수행할 수 있는 주된 정보기술로 제안하고, 다수의 조직이 연관되는 비즈니스 프로세스를 설계하기 위해 고려해야할 점들을 제시하였다. WISE 프로젝트는 스위스 국립 과학 재단 지원으로 1997년 12월부터 29개월 동안 3개 학술 기관과 2개 기업의 참여로 수행되었으며, 이 프로젝트는 가상 기업 비즈니스 프로세스를 정의하고, 수행하며, 감시하는 기능을 가진, 시스템 형태의 B2B 전자상거래를 위한 기초 기반 구조를 설계, 구축, 검사하는 것을 목표로 하였다. WISE 프로젝트는 가상 비즈니스 프로세스(virtual business process), 가상기업(virtual enterprise) 그리고 교역 공동체(trading community)를 전자상거래의 주요 요소로 평가하였다. 이들은 독립적인 다수의 워크플로우 관리 시스템이 아닌, 하나의 워크플로우 관리 시스템을 통해 다수의 조직에 걸친 비즈니스 프로세스가 수행됨을 전제로 하고 있다.

그러나, B2B 전자상거래 환경에서는 둘 혹은 그 이상의 워크플로우 엔진이 작업을 통합하기 위해 교류하고 협동하는 능력을 의미하는 워크플로우의 상호운용성이 매우 중요하다. 이러한, 상호운용성 개념은 동일한 조직의 동일한 워크플로우 엔진 간 상호 운용뿐 만 아니라, 상이한 조직의 상이한 워크플로우 엔진 간 상호 운용 모두를 포괄한다. WfMC에서는 상호운용성과 관련해 두 가지의 표준을 제시하고 있는데, 하나는 Internet e-mail MIME Binding[26]이며, 또 다른 하나는 Wf-XML Binding[27]이 그것이다. e-mail을 사용하는 경우와 XML을 사용하는 경우를 비교하면, e-mail을 사용할 경우 반드시 담당자의 개입이 있어야 한다. 그러나, XML을 이용할 경우, 워크플로우간에 전달되는 메시지를 담당자의 개입이 없이도 자동으로 처리할 수 있으므로 담당자가 휴가를 갔거나, 자리에 없을 경우에도 자동적으로 트랜잭션이 처리될 수 있는 장점이 있다. 이러한 WfMC의 상호운용성의 특징은 상호 운용되기 위한 각 워크플로우가 상호 운영 가능할 만큼 충분히 WfMC 표준에 따라 설계된 시스템이어야 하므로, 결국, 상호 운영을 위해 많은 개발 노력을 필요로 한다. 또한, WfMC의 상호 운용성은 기능상 유사한 워크플로우 간의 완벽한 연결에 초점을 맞추었기 때문에, 상이한 조직의 이질적인 워크플로우 간 유연한 연결에는 적합치 않다고 할 수 있다.

따라서, Aalst[5]는 WfMC에서 워크플로우 상호운용성이 가능하도록 표준화를 추진하고 있으나, 그 표준은 조정 구조(coordination structure)의 내용물이 아닌 기술적 이슈만을 다루고 있음을 지적하고, 주로 구문론적인(syntactical) 기술적 문제 이전에 여러 조직이 연루된 워크플로우를 모형화 하기 위해 구성개념들의 의미론(semantics)이 필요하다고 제안하였다. 또한, Aalst[4]는 현재까지의 전자상거래 응용프로그램들이 주로 비즈니스 문서의 전면(front side)에만 초점을 맞추고 있으며, EDI 기반의 시스템들은 거래 처리 자료의 표현에만 초점을 두고 있음을 지적하였다. 그의 연구는 비즈니스 문서의

후면(back end)이 개별적인 거래 처리 혹은 거래 처리 자료보다는 비즈니스 프로세스를 반영하는 점에 착안하여, 문서의 후면에 연구의 초점을 두어, 워크플로우 상호 운용성의 형태를 다음과 같이 여섯 가지로 분류하고, B2B 전자상거래 수행에 있어 각각의 유용성을 설명하였다.

첫 번째는 처리력 공유(capacity sharing)인데, 이것은 단일의 워크플로우가 분산된 비즈니스 파트너의 작업들을 중앙 집권적으로 통제하는 형태이다.

두 번째는 연결된 실행(chained execution)으로, 이 형태는 하나의 비즈니스 프로세스가 다수의 하부 프로세스로 나뉘어 지고, 이것들이 다수의 비즈니스 파트너로 나뉘어 연속적으로 수행되는 형태이다. 이때, 하부 프로세스들은 비즈니스 파트너 각자가 통제한다.

세 번째는 하청 계약(subcontracting)이다. 이것은 한 비즈니스 파트너가 하부 프로세스를 다른 비즈니스 파트너에게 하청을 주는 형태이다.

네 번째는 사례 전달(case transfer)이다. 이것은 비즈니스 파트너 각자가 워크플로우 프로세스 정의(process definition)의 사본, 즉 사례를 가지는 형태이다. 하지만, 그 사례는 특정시점에 하나의 비즈니스 파트너에만 존재한다.

다섯 번째는 확장된 사례 전달(extended case transfer)이다. 이것은 사례 전달 형태에, 각 비즈니스 파트너에 머무는 동안의 사례 변이를 허용한 형태이다.

여섯 번째는 느슨한 결합(loosely coupled)이다. 이것은 프로세스를 여러 조각으로 나누고, 이것들을 각 비즈니스 파트너가 병렬로 수행한다. 각 하부 프로세스 정의는 지역적이며, 단지 비즈니스 파트너 간 통신에 사용되는 프로토콜만이 공개된다.

워크플로우의 여섯 가지 상호 운용 형태에서, 처리력 공유 형태는 통제가 분산되지 않았고, 연결된 실행은 복잡한 상거래에 적용할 수 없는 너무 단순한 연결형태라는 이유로 다수의 기업이 참여하는 B2B 전자상거래에 적합하지 않으며, 하청 계약 형태는 EDI 시스템과 같이 주도하는 기업이 있는 경우에만 적합한 형태이다. Aalst[4]는 사례 전달, 확장된 사례 전달, 느슨한 결합 등 나머지 세 가지 형태의 상호운용성을 B2B 전자상거래에 적합한 워크플로우 구조로 평가하고, 가상적 시나리오를 이용해 이들 세 가지 구조의 조직간 운용 가능한 워크플로우를 각각 모델링 하였다.

Aalst[4, 5]가 B2B 전자상거래에 가장 적합한, 기업간 운용되는 워크플로우 구조로 제시한 느슨히 연결된 구조에서는 참여하는 조직의 워크플로우들이 독립적으로 수행되고, 서로 간에 메시지를 비동기적으로 교환하며, 하나의 B2B 워크플로우를 수행한다. 이러한 구조는 각 조직에 존재하는 워크플로우의 자율성을 유지하면서, 메시지 처리 기능의 추가만으로 B2B 워크플로우를 수행할 수 있다. 느슨히 연결된 구조에서 워크플로우간 연결을 가능하게 하는 것은 각 조직에 존재하는 메시징 시스템이다. 이러한 메시징 시스템의 특징은 단순히 메시지를 교환하기만 하는 것이 아니라, 워크플로우의 요청에 의해 전송 가능한 메시지를 작성하고, 전달받은 메시지의 내용 일부를 해석하며, 해석된 내용을 토대로 워크플로우와 상호 작용해야 한다.

2.3 메시징 시스템

컴퓨터 소프트웨어 혹은 응용 프로그램간의 자동화된 메시지 교환 문제는 소프트웨어 개발에 있어 항상 고려되어야 할 부분이다. 좁게는 동일한 시스템의 동일 패키지 내의 다수 모듈간의 통신에서, 넓게는 상이한 조직의 상이한 시스템 내 상이한 응용프로그램 간 통신까지 모두를 고려할 필요가 있다. 특히, 기업 내에 다수의 이질적인 시스템과 응용프로그램들이 속속 도입되면서, 기존 시스템과 신규 도입 시

스템간의 연결 문제는 항상 존재하게 되는데, 메시징 시스템은 CORBA, DCOM(Distributed Component Object Model), RPC(Remote Procedure Call) 등과 함께 이러한 문제의 해결책으로 등장하였다.

메시징 시스템이 CORBA, DCOM, RPC 등과 비교되는 점은, 응용 프로그램간 연결을 위해 메시징 시스템이라는 중간 매개체가 존재하고, 그 처리과정이 느리다는 단점이 있지만, 동기적인 연결뿐만 아니라 비동기적인 연결로 응용프로그램간 병렬 수행이 가능하고, 무엇보다 이질적인 응용 프로그램들의 연결과 통합을 가장 쉽게 할 수 있다는 장점이 있다. 결국, 메시징 시스템은 이질적인 시스템과 응용프로그램을 보유한 기업 간 자동화된 상거래 수행을 필요로 하는 오늘날의 B2B 전자상거래 환경에서 그 중요성이 더해지고 있다.

전자상거래를 위한 자동화된 메시지 처리를 위한 접근법에는 자연어 처리, EDI, 태그(tag)를 사용한 메시지 시스템(tagged-message systems), FLBC(Formal Language for Business Communication) 등 네 가지 일반적 접근법이 가능한데, 각 접근법을 비교 설명하면 다음과 같다[15, 18].

첫 번째, 자연어 처리 접근법에서 메시지의 내용은 우리가 일상적으로 사용하는 자연어로 이루어져 있다.

두 번째, EDI 프로토콜은 주문서, 선하증권 등과 같은 종이 기반 문서의 기업간 흐름을 컴퓨터간의 정보 교환으로 대체하기 위해 개발된 것이다. EDI 시스템의 메시지는 태그를 사용한 메시지 시스템 접근법의 메시지 바디(body) 부분이 메시지 헤더(header)에 통합된 형태이다.

세 번째, 태그를 사용한 메시지 시스템 접근법에서 메시지는 태그와 내용(contents)으로 분리되는데, 태그는 의미적 접근이 가능하지만, 내용은 의미적 접근이 불가능하다. 또한 메시지는 메시지 헤더와 메시지 바디로 구성되는데, 메시지 헤더는 메시지의 타입, 메시지의 고유한 식별자, 관련된 키워드 등으로 구성되고, 메시지 바디는 전달되어 실제 보여질 전자 문서로 구성된다.

마지막, FLBC 접근법은 앞선 세 가지 접근법에 비해 훨씬 더 많은 표현력을 가진다. FLBC 메시지는 메시지를 의미, 내용, 배경 등 세 개의 주요한 부분으로 나뉘어 지고, FLBC 시스템(혹은 MMS(Message Management Systems))에서 메시지는 해석 프로시저의 입력이 되는 일련의 단언(assertions) 혹은 선언(declarations)으로 구성된다.

FLBC는 응용 프로그램 혹은 소프트웨어 에이전트(intelligent agents) 간 통신을 위한 언어로 널리 알려진 Finin at al.[11]의 KQML (Knowledge Query and Manipulation Language)에 비교되는 것으로서, 상거래 목적의 응용 프로그램 간 통신에 초점을 맞추어 연구된 것이며 MMS는 응용프로그램 간 FLBC 메시지 교환을 담당하는 시스템이다. 이상의 전자상거래를 위한 네 가지 메시지 교환 접근법 중에서 가장 유연하고 적절한 표현력을 지닌 FLBC 접근법이 가장 우수하다.

3. 시스템 분석

3.1. 객체지향 개발 방법론

객체지향 개발 방법론은 해당 시스템을 구성하는 객체들과 그 객체들이 어떻게 행동하고, 교류하는가에 초점을 맞춘다. 이 개발 방법론의 첫 단계는 시스템을 구성하는 비즈니스 객체들을 인지하는 것이며, 그 다음 단계는 객체의 속성과 메서드(method), 그리고 다른 객체와의 관계 등을 이용해 각 클래스를 기술함으로써, 객체 모델을 생성하는 것이다[10].

본 연구는 객체지향 시스템 분석 및 설계 방

법을 채택하였다. 우선 메시징 시스템의 분석 및 설계에 앞서, 보다 구체적인 목표 시스템을 위해 예제 시나리오를 설정하였고, 이 예제 시나리오를 기초로 분석을 수행하였다. 메시징 시스템은 독자적으로 수행되는 것이 아닌 지역의 워크플로우와 긴밀하게 연결되어 실행되는 것이므로, 관련된 워크플로우를 우선적으로 분석하고, 그 다음으로 메시징 시스템을 분석하였다. 분석 단계에서 인지된 다수의 객체들을 세 가지로 분류하였다. 또한, 분석 단계에서 인지된 객체들의 기본적인 속성과 메서드들의 자세한 내용은 설계 단계에 나타내었다. 설계 단계에서는 인지된 객체들간의 구조적 관계와 행동적 교류에 대해 상세하게 서술하였다. 또한 설계 단계에서는 분석 단계에서 인지하지 못했던 객체들과 속성 및 메서드를 추가하였다.

본 연구의 분석과 설계에서 사용한 UML은 1997년 OMG가 UML을 표준 언어로 지정한 이후 소프트웨어 산업의 지배적인 모델링 언어로 등장하였다[16]. UML은 소프트웨어 청사진을 작성하는 표준언어로서 소프트웨어 중심 시스템의 산출물을 가시화, 명세화, 문서화하는데 사용될 수 있다. 하지만, UML은 단지 언어이므로 개발방법론의 한 부분이며, 개발 과정과는 독립적이다. UML은 여러 형태의 도해들을 이용해 시스템의 다양한 측면을 각각 모델링할 수 있게 하며, 각 도해들은 별개의 것이 아니라 서로 연결되어 있다는 것이 특징이다. UML 1.0의 경우 표준적으로 9개의 도해를 제시하고 있는데, 이 모두가 반드시 쓰여져야 하는 것은 아니며, 필요에 따라 선택하여 사용할 수 있다[3].

본 연구에서는 분석과 설계 단계에서 UML의 쓰임새도, 활동도, 클래스도, 순차도 등을 이용하였다. 분석된 객체는 실제 객체 클래스와 객체 인스턴스로 구분된다. 객체 클래스는 어떤 사물의 원형 혹은 부류를 의미하며, 객체 인스턴스는 원형으로부터 나온 하나의 사례 혹은 어떤 부류에 속하는 일례를 의미한다. 분석 단계에서 작성한 쓰임새도에서 쓰임새를 실현하는 것을 협력이라 한다. 이때 협력은 UML 요소들간의 논리적인 구성으로 나타낼 수 있는데, 대개 그 요소들 간의 구조적 모습을 보여주기 위해서는 클래스도를, 행동적 모습을 보여주기 위해서는 교류도를 각각 이용한다[3]. 교류도에는 협력도와 순차도가 있는데, 동일한 대상을 모델링 하면서 구조적 관계에 초점을 둔다면 협력도가, 시간적 순서에 초점을 둔다면 순차도가 각각 유용하다.

클래스도를 작성할 때 다음과 같은 규칙을 본 연구에서는 공통적으로 적용하였다.

첫 번째, 속성의 형(type) 명칭이 int, double과 같이 소문자로 시작하는 경우는 기본형을 나타내고, String, Message와 같이 대문자로 시작하는 경우는 클래스형을 나타낸다.

두 번째, 속성이 클래스형인 경우, 그 속성을 하나의 UML 클래스로 표현하고, 이를 UML 연관을 이용해 해당 속성을 가진 클래스와의 관계를 구체적으로 표시한다. 단, 그 속성이 표준 클래스 라이브러리에 포함된 것인 경우 불필요한 복잡성을 피하기 위해, 기본형과 마찬가지로 UML 클래스로 표현하지 않는다.

세 번째, 모든 클래스는 자신이 보유한 속성 값을 읽을 수 있는 getter 메서드와 속성 값을 설정할 수 있는 setter 메서드를 가지는데, 이것들은 속성과 함께 항상 존재하는 메서드로 간주하고, 이 또한 불필요한 복잡함을 피하기 위해 클래스도에 표시하지 않았다.

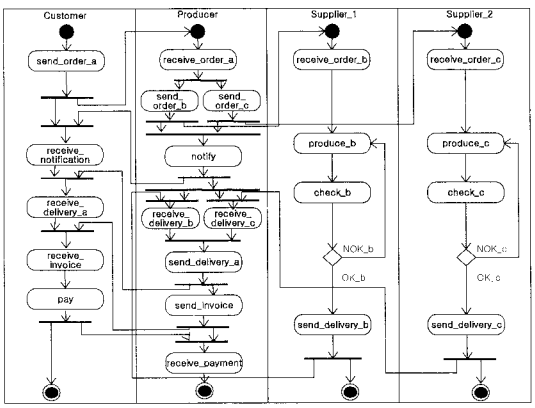
3.2. B2B 워크플로우 분석

본 연구에서는 B2B 워크플로우를 예시적으로 보여주기 위해 Aalst[4, 5]의 가상 시나리오를 이용하였는데, 이 시나리오는 EDI를 위한 표준적 시나리오를 제공하는 Open-EDI 표준[8]에 기초하고 있다. 시나리오를 다시 정리하면 다음과

같다.

> *Customer, Producer, Supplier_1, Supplier_2 등 네 개의 조직이 B2B 전자상거래에 참여한다. Customer가 제품 a를 주문하기 위해 주문서를 Producer에 보낸다. Producer는 제품 a 생산에 필요한 b와 c를 Supplier_1과 Supplier_2에 각각 주문하고, Customer에게 주문한 제품 a 의 공급이 가능하다는 통지서를 보낸다. Supplier_1은 제품 b를, Supplier_2는 제품 c를 각각 생산한다. 생산된 b, c 두 제품이 Producer로 배송 되면, Producer는 이것들을 조립하여 a 제품을 완성하고 이를 물품 내역서와 함께 Customer에게 배송 한다. 다음으로 대금청구서가 Customer에 전달되면, Customer는 물품 대금을 Producer에 지급한다[5, p.69].*

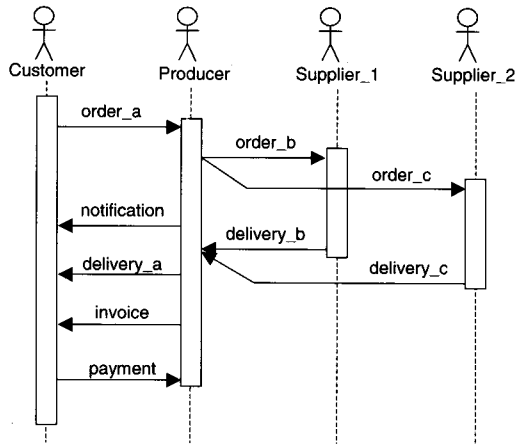
<그림 3>은 이상의 B2B 워크플로우 시나리오를 UML 활동도로 모델링 한 것이다. 활동도는 구획이 네 개로 나뉘어져 있는데, 각 구획은 B2B 워크플로우에 참여하는 Customer, Producer, Supplier_1, Supplier_2의 워크플로우 각각을 의미한다. B2B 워크플로우의 시작은 맨 좌측의 Customer에서 시작된다. 활동도에서 검은 원은 워크플로우의 시작을, 모서리가 둥근 사각형은 활동을, 화살표는 활동의 전이를 각각 나타낸다. 활동간의 전이는 선행하는 활동의 수행이 완료되면 자동으로 발생한다. 전이가 발생하면, 다음 활동으로 넘어 가게 되는데, 이때, 동시분할이 발생할 수 있다. 동시분할은 동시합류와 함께 검은 막대로 표시된다. 동시분할은 하나의 활동에서 다수의 활동으로 동시적인 전이가 이루어짐을 의미하고, 전이된 후의 각 활동은 병렬로 수행된다. <그림 3>에서 send_order_a라는 활동이 완료되면, 자동적으로 전이가 발생하고, 다음 활동으로 넘어 가기 전에 검은 막대를 만나게 되는데, 이때, 두 개의 동시적인 전이가 발생한다. 여기서의 동시분할은 Producer 측의 워크플로우를 시작하게 만든다. 동시분할 이후 Customer의 워크플로우와 Producer의 워크플로우는 병렬로 수행되며, 후에 Producer의 notify 활동이 수행된 후 전이를 통해 Customer 측 워크플로우에서 동시합류가 발생한다. Supplier_1과 Supplier_2의 워크플로우는 각각 Producer의 send_order_b와 send_order_c에 의해 시작되는데, 이 또한 병렬로 수행된다. Supplier_1과 Supplier_2의 워크플로우 각각의 내부에서는 조건 분기가 발행하는 것을 보여주는데, check_b와 check_c 활동 수행 후 그 결과에 따라 아래쪽으로 계속 진행 될 수도 있고, 다시 위쪽의 produce_b 혹은 produce_c로 되돌아갈 수도 있다. Supplier_1과 Supplier_2는 각각 send_delivery_b와 send_delivery_c 활동이, Customer는 pay 활동이, Producer는 receive_payment 활동이 완료되면, 각각의 워크플로우가 종료되면서 전체 B2B 워크플로우도 종료된다.



<그림 3> B2B 워크플로우 활동도

B2B 워크플로우 활동도가 B2B 워크플로우 전체를 구체적으로 보여 주고 있는 반면, 워크플로우 간 메시징을 보여주기에는 불필요하게 복잡한 면이 있다. <그림 4>는 UML 순차도를 이용해 메시징 부분만을 따로 모델링 한 것이다. 순차도의 최상단은 Customer, Producer, Supplier_1, Supplier_2 등 각 참여 조직을 나타낸 것이며, 각 참여 조직에서 뻗어 내린 점선은 시간 순서를 나타내고, 그 위를 덮은 사각형 막대는 해당 워크플로우가 수행 중인 상태임을 나

타낸다. 각 워크플로우간 주고 받는 메시지는 화살표로 나타내었다.



<그림 4> 메시징 순차도

3.3 메시징 시스템 분석

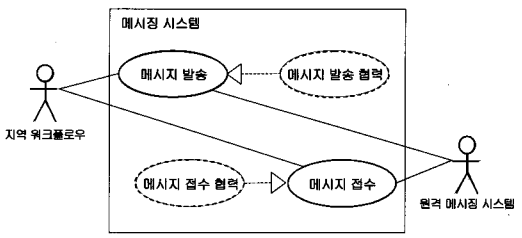
3.3.1 메시징 시스템 개요

메시징 시스템은 항상 클라이언트의 요청을 기다리며 대기해 있는 인터넷 서버 시스템과 유사하다. 지역의 워크플로우 수행 서비스가 활동 수행 요청을 하거나, 원격의 메시징 시스템이 메시지를 보내 올 때까지는 항상 대기 상태에 놓여 있고, 원격의 상대 메시징 시스템이 메시지를 전달해 오거나, 로컬의 워크플로우 수행 서비스가 활동 수행 요청과 같은 호출이 오면 그때 본격적인 기능을 수행한다. 이때 다수의 동시적인 요청을 수행하기 위해서는 매번 요청시 새로운 작업 흐름을 생성하고, 그 다음의 새로운 요청에 대비해 다시 대기 상태에 놓여 있을 필요가 있다. 이는 메시징 시스템의 양측에 있는 워크플로우들이 병렬로 수행되며, 쌍방에게 비동기적인 메시지를 전송하는 경우 필수적인 기능이다. 메시징 시스템은 크게 두 부분으로 나눌 수 있는데, 하나는 메시지 발송과 관련된 부분이며, 다른 하나는 메시지 접수와 관련된 부분이다.

느슨히 연결된 구조에서의 메시징 시스템은 인터넷과 같은 공공의 네트워크를 통한 메시지 교환을 수행할 수 있어야 하고, 전송 가능한 포맷으로 메시지를 작성 할 수 있어야 하며, 접수된 메시지의 내용을 일부 혹은 전부를 해석할 수 있어야 한다. 메시징 시스템이 메시지를 해석한 후 지역 워크플로우의 새로운 프로세스 인스턴스가 수행되거나, 수행되고 있는 프로세스의 특정 활동이 수행된다. 메시지 내용 중에서 주문서와 같은 부분은 메시징 시스템이 해석하지 않고 워크플로우에 전달한다. 이때 워크플로우 관리 시스템은 주문서와 관련된 활동을 수행해야 한다. 한편, 워크플로우 측면에서 메시징 시스템은 일종의 행위자(actor) 혹은 자원(resource)으로 볼 수 있는데, 이는 메시징 시스템이 조직 외부에 있는 특정 B2B 워크플로우 활동을 담당하는 것으로 볼 수 있기 때문이다.

3.3.2 메시징 시스템의 문맥 모델링

UML 쓰임새도는 사용자와 시스템간의 교류를 보여주는 시스템 문맥 모델링에 유용하다. <그림 5>에서 사각형은 메시징 시스템 전체를 나타내며, 사각형 외부에는 지역 워크플로우와 원격 메시징 시스템이 존재한다. 사각형 내부에서 실선 타원으로 나타낸 메시지 발송과 메시지 접수는 UML 쓰임새이며, 이는 시스템 분석 단계에서 발견되는 시나리오를 의미한다. 점선 타원으로 나타낸 메시지 발송 협력과 메시지 접수 협력은 UML 협력으로서 메시지 발송 쓰임새와 메시지 접수 쓰임새의 구체화를 의미한

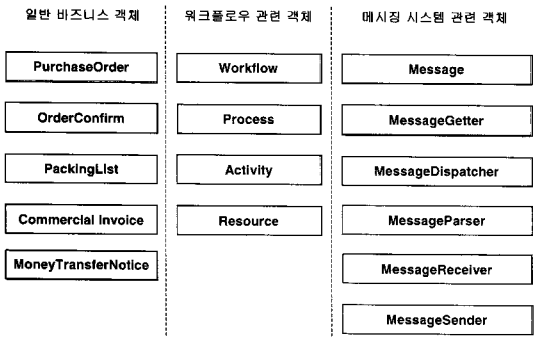


<그림 5> 메시징 시스템 쓰임새도

다. 설계 단계에서 협력은 클래스도와 순차도 등으로 나타낼 수 있는데, 이들 각각은 협력의 구조적 측면과 행동적 측면을 보여준다. 메시징 시스템에는 메시지 발송 협력과 메시지 접수 협력 두 가지가 있다.

3.3.3 메시징 시스템 객체

객체지향 개발 방법론에서 분석 단계의 주요 산출물은 인지된 객체들이다. 앞서 B2B 워크플로우, 메시징 시스템 및 메시지 등의 분석을 통해 정의된 객체는 <그림 6>과 같다. 인지된 객체들은 일반 비즈니스 객체, 워크플로우 관련 객체, 메시징 시스템 관련 객체 등 세 종류로 분류된다. 첫 번째 일반 비즈니스 객체는 종이 기반(paper-based)의 실무나 정보 시스템에서 사용되던 객체들이다. 두 번째, 워크플로우 관련 객체는 워크플로우와 관련된 것으로서 보다 상세한 목록은 OMG[19]를 통해 획득할 수 있다. 마지막으로, 메시징 시스템 관련 객체가 있는데, 이들 중 MessageDispatcher, MessageSender, MessageReceiver, Message는 메시지 발송 협력과 관련된 것이고(<그림 11> 참고), MessageGetter와 MessageParser는 메시지 접수 협력과 관련된 것들이다 (<그림 13> 참고).



<그림 6> 메시징 시스템 객체

해당기업 간에 어떠한 메시지가 교환되는지는 메시징 순차도를 통해 분석할 수 있다. 예를 들어 Customer와 Producer 간에는 다섯 개의 메시지가 서로 교환되게 되는데, 이 다섯 개의 메시지로 하나의 거래가 이루어진다. 예를 들어, t1이라는 거래를 수행하기 위해 m1(order_a), m2(notification), m3(delivery_a), m4(invoice), m5(payment) 등의 메시지들을 순서대로 주고 받을 필요가 있다면, 이 메시지들은 t1 거래로 묶이게 되고, t2 혹은 t3와 같은 다른 거래의 메시지와는 구분되어야 한다. 따라서, 교환되는 메시지가 어떤 거래에 속하는지 명시해야할 필요가 있는데, 이를 위해 메시지는 "transaction_id"와 같은 식별자를 포함해야 한다. <그림 7>은 메시지로 구성된 거래를 형식 언어의 일종인 EBNF(Extended Backus-Naur Form)로 구문 정의한 것이다.

```
<transaction>          ::=   TRANSACTION
                                 <transaction_id>
                                 {<message>}
                             END_TRANSACTION
<transaction_id>       ::=   <identifier>
<identifier>           ::=   TEXT_PLAIN
<message>              ::=   MESSAGE
                                 <header>
                                 <body>
                             END_MESSAGE
<header>               ::=   HEADER
                                 <name>
                                 <sender>
                                 <receiver>
                             END_HEADER
<name>                 ::=   <keyword>
<sender>               ::=   TEXT_PLAIN
<receiver>             ::=   TEXT_PLAIN
<keyword>              ::=   TEXT_PLAIN
<body>                 ::=   BODY
                                 <purchase_order> |
                                 <order_confirm> |
                                 <packing_list> |
                                 <commercial_invoice> |
                                 <money_transfer_notice>
                             END_BODY
<purchase_order>       ::=   TEXT_PLAIN | TEXT_XML | <uri>
<order_confirm>        ::=   TEXT_PLAIN | TEXT_XML | <uri>
<packing_list>         ::=   TEXT_PLAIN | TEXT_XML | <uri>
<commercial_invoice>   ::=   TEXT_PLAIN | TEXT_XML | <uri>
<money_transfer_notice> ::=  TEXT_PLAIN | TEXT_XML | <uri>
<uri>                  ::=   TEXT_PLAIN
```

<그림 7> 메시지 구조(EBNF)

어떤 거래에 속하는 지를 명시하기 위한 식별자 외에 메시지는 전체적으로 메시지 헤더와 메시지 바디로 나뉘어 진다. 메시지 헤더 부분은 키워드 역할을 하는 명칭(name), 수신자(receiver) 및 발신자(sender) 등이 포함되고, 메시지 바디 부분은 해당 메시지와 관련된 문서가 포함된다. 바디 부분에는 해당 문서가 평이한 텍스트 혹은 XML 형태로 포함될 수도 있고, 하이퍼링크를 의미하는 URI(Uniform Resource Identifier) 형태로 포함될 수도 있다. 메시징 순

<표 1> 예제 메시지

	헤 더 부 분			바 디 부 분
거래 식별자	메시지명(name)	발신자(sender)	수신자(receiver)	첨부 문서(contents)
t1	order_a	Customer	Producer	주문서(purchase_order)
t2	order_b	Producer	Supplier_1	〃
t3	order_c	Producer	Supplier_2	〃
t1	notification	Producer	Customer	주문확인서(order_confirm)
t1	delivery_a	Producer	Customer	물품내역서(packing_list)
t2	delivery_b	Supplier_1	Producer	〃
t3	delivery_c	Supplier_2	Producer	〃
t1	invoice	Producer	Customer	상업송장(commercial_invoice)
t1	payment	Customer	Producer	입금통지서(money_transfer_notice)

차도에 표시된 예시적 시나리오의 아홉 개 메시지는 <표 1>에 정리하였다.

Ⅳ. 시스템 설계

분석 단계에서 발견된 메시지 발송 협력과 메시지 접수 협력을 설계 단계에서는 구조적 모습을 보여 주기 위해 클래스도를 이용하였고, 행동적 모습을 보여주기 위해 교류도의 일종인 순차도를 이용하였다. 두 개의 협력 외에 설계 시 구체화해야 할 것은 메시징 시스템에서 교환되는 메시지 그 자체인데, 메시지의 설계는 XML DTD(Document Type Definition)를 이용한 메시지 구조 설계와 UML 클래스도를 이용한 객체 설계로 구분하였다.

4.1 메시지 설계

4.1.1 메시지 구조

XML 문서는 그 자체에 문서의 구조를 정의하여 문서 검증을 가능하게 한다. 이때, 문서의 구조 정의는 XML DTD를 통해서 이루어지는데, XML DTD의 작성이 그 문서 구조의 설계가 된다. 본 연구에서의 메시지는 XML 포맷으로 작성되며, 그 구조의 설계를 위해 <그림 8>과 같이 DTD로 작성하였다. 앞서, 분석 단계에서 메시지의 개략적인 구조를 EBNF를 통해 나타내었는데, 기본적인 내용이 DTD에 모두 포함되었으며, 그 외에 메시지 바디에 포함될 내용이 무엇인지, 그리고, 내용의 포맷 및 언어, 확장을 고려한 전달 인자 등을 메시지 헤더의 구성요소에 포함시켰다. 특히, 거래를 식별할 수 있는 거래 식별자(TransactionID)도 메시지 헤더에 포함 시켰다.

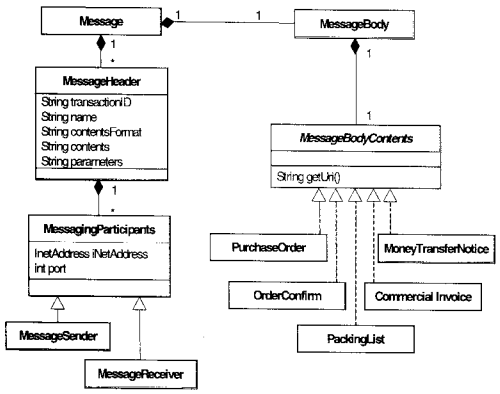
```
<!ATTLIST Message ID CDATA #REQUIRED>

<!ELEMENT MessageHeader
    (TransactionID, MessageName, MessageSender,
    MessageReceiver, ContentsFormat, Contents, Language, Parameters?)
    >
    <!ELEMENT TransactionID (#PCDATA)>
    <!ELEMENT MessageName (#PCDATA)>
    <!ELEMENT MessageSender (#PCDATA)>
    <!ELEMENT MessageReceiver (#PCDATA)>
    <!ELEMENT ContentsFormat (#PCDATA)>
    <!ELEMENT Contents (#PCDATA)>
    <!ELEMENT Language (#PCDATA)>
    <!ELEMENT Parameters (#PCDATA)>

<!ELEMENT MessageBody
    (PurchaseOrder|OrderConfirm|PackingList|CommercialInvoice|
    MoneyTransferNotice)
    >
    <!ELEMENT PurchaseOrder  ANY>
    <!ELEMENT OrderConfirm  ANY>
    <!ELEMENT PackingList  ANY>
    <!ELEMENT CommercialInvoice  ANY>
    <!ELEMENT MoneyTransferNotice  ANY>
```

<그림 8> 메시지의 구조(XML DTD)

4.1.2 메시지 객체

XML DTD가 전달시 파일 형태로 외부로 드

러나는 실제 메시지의 구조를 보여 주고 있는 반면, 메시징 시스템 내부에서의 메시지 구조(즉 모습)를 보여 주지는 못한다. <그림 9>는 메시지가 시스템 내부에서 어떠한 구조로 이루어져 있는지를 보여 주는 UML 클래스도이다. 그림의 주된 구성은 사각형 모양의 UML 클래스로 이루어지며, 그 내부는 다시 속성과 메서드로 구성된다. 단 사각형 모양은 동일하나 클래스 명이 이탤릭체로 된 것은 인터페이스를 나타낸다.



<그림 9> 메시지 클래스도

Message 클래스는 메시지 객체들의 클래스를 의미하며, MessageHeader 클래스와 MessageBody 클래스로 구성된다. MessageBody 클래스는 MessageBodyContents 인터페이스를 구현하는 클래스로 구성된다. MessageBodyContents 인터페이스를 구현하는 클래스는 분석 단계에서 Customer와 Producer 간에 주고 받는 다섯 가지의 문서를 각각 의미한다. 또한 MessageBodyContents 인터페이스에 getUri() 메서드를 선언해 두었는데, 이는 PurchaseOrder와 같은 객체가 인터넷 상의 특정 위치에 존재하는 파일을 원천으로 해서 생성되도록 하기 위함이다. MessageParticipants는 메시지 교환의 쌍방 모두를 지칭하는 UML 클래스이며, 자식 클래스로 MessageSender 클래스와 MessageReceiver 클래스를 두고 있다. 이러한 상속 구조는 메시지 교환 쌍방에 공통적인 속성과 메서드를 부모 클래스에 두고, 발송자 혹은 수신자에 특수한 속성과 메서드는 자식 클래스에 둠으로써 구현시 코드의 작성을 최소화하고, 설계의 변경이나 추가를 용이하게 할 수 있다.

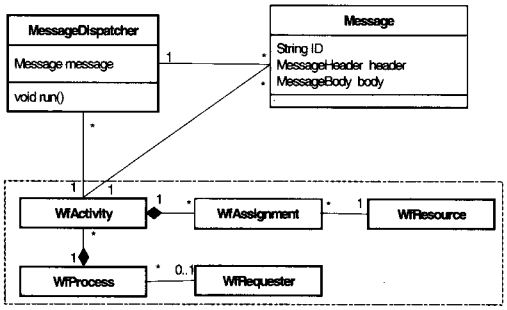
UML의 연관 관계는 객체 클래스가 객체 인스턴스가 되었을 때에 UML 연결이 됨을 의미하고, 실선의 양끝에는 역할, 다중성, 집합연관 여부, 복합연관 여부 등을 나타낼 수 있다. Message 클래스와 MessageParticipants 클래스 간의 연관관계에서 보여주는 일대다 관계는 하나의 Message 객체가 다수의 MessageParicipants 객체와 관계됨을 나타낸다. MessageHeader 클래스와 Message 클래스간의 연관관계와 MessageBody 클래스와 Message 클래스간의 연관관계를 나타내는 실선에서 Message 쪽에 검은색 마름모형이 더해 진 것은 이 둘 간에 강한 연관이 있음을 의미한다. 이러한 강한 연관을 복합연관이라 하며, MessageHeader와 MessageBody 객체의 생명주기가 Message 객체의 생명주기에 종속됨을 의미한다. 클래스를 나타내는 사각형이 가는 실선이 아닌 굵은 실선인 경우 그 클래스가 활성클래스임을 나타내며, 활성클래스는 독자적인 프로세스 혹은 쓰레드 객체를 가지는 클래스이다.

4.2 메시지 발송 협력 설계

4.2.1 구조적 측면

메시지의 발송은 워크플로우 수행 중에 필요에 의해 발생되는데, 워크플로우 관련 객체 중 하나가 메시지 발송 객체를 호출함으로써 시작된다. <그림 10>은 메시지 발송 협력을 클래스도로 나타낸 것이다. 그림의 점선 사각형 내부의 클래스들은 워크플로우와 직접 관련된 클래스들간의 구조적 관계를 나타내고 있다. 본 연구에서는 WfActivity 클래스의 객체 인스턴스가 MessageDispatcher의 run() 메서드를 호출함으

로써 메시지 발송을 수행시키는 것으로 설계하였다.

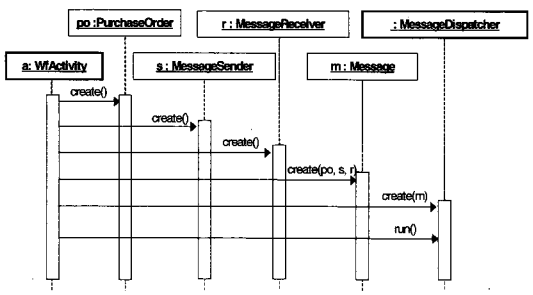


<그림 10> 메시지 발송 협력 클래스도

4.2.2 행동적 측면

메시지 발송 협력을 나타내는 클래스도가 클래스들 간의 구조적 관계를 보여주는 반면, 순차도는 교류도의 일종으로 객체간에 전달되는 메시지의 시간적 순서를 나타낸다.

<그림 11>은 메시지 발송 협력을 나타내는 순차도인데, 메시지 발송 협력의 행동적 측면을 시간 순서로 설명하면 다음과 같다. 활성객체인 a가 create() 메서드 호출로 po, s, r 등을 각각 생성하고, 이것들을 매개 변수로 m 객체를 생성한 후, 다시 이 m 객체를 매개변수로 익명의 MessageDispatcher 객체를 생성하며, 마지막으로 이 익명의 객체가 가진 run() 메서드를 호출한다. 익명의 객체는 독자적인 프로세스 혹은 쓰레드로 전달받은 메시지를 메시지 수신자에게 네트워을 통해 발송한다.

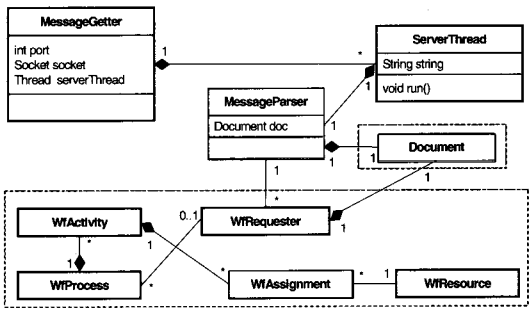


<그림 11> 메시지 발송 협력 순차도

4.3 메시지 접수 협력 설계

4.3.1 구조적 측면

메시지 접수 협력에는 MessageGetter, ServerThread, MessageParser 등의 메시징 시스템과 직접 관련된 클래스와 외부 시스템의 클래스 혹은 표준 클래스 등으로 구성된다 (<그림 12> 참고).



<그림 12> 메시지 접수 협력 클래스도

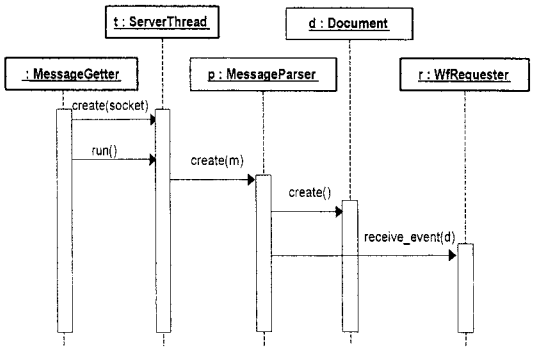
MessageGetter 클래스는 메인 메서드를 가진 활성클래스로 운영체제로부터 독자적인 프로세스를 할당받아 수행된다. 하나의 MessageGetter 객체에 대해 다수의 ServerThread가 연관된다. ServerThread 클래스는 일종의 활성클래스이며, MessageGetter 객체로부터 전해 받은 메시지 제어권을 이용해 해당 메시지를 전송 받고, 이 메시지를 전달인자로 MessageParser 객체를 생성한다. MessageParser 클래스는 메시지를 해석하고, 이를 Document 객체로 생성한다. 생성된 Document 객체는 객체 상태로 워크플로우에 직접 전달 될 수 있으며, 파일이나 데이터베이스 테이블로 저장될 수 있다. MessageGetter와 ServerThread, ServerThread와 MessageParser, MessageParser와 Document 객체간에는 복합연관 관계가 존재하는데, 이는 한 객체의 생명주기가 다른 객체의 생명주기에 종속되어 있음을 의미한다. 특이할 점은 네트워으로부터 전해 받은 메시지는 워크플로우 수행과 관련된 키워드

가 포함되며, 메시지 해석 후에는 필연적으로 워크플로우와 관련되게 된다. 본 설계에서는 워크플로우 객체 중의 하나인 WfRequester[19]의 특정 메서드를 호출하는 것으로 하였다.

4.3.2 행동적 측면

메시지 접수 협력은 전형적인 클라이언트-서버 환경에서의 서버 시스템의 수행과 매우 유사하다. 이러한 프로그램의 특징은 네트웍으로부터 계속적으로 전송되어 오는 메시지를 신속히 접수하고, 안전하게 사후 처리를 할 수 있어야 한다는 점이다. 본 설계에서는 신속하고, 안전한 메시지 접수를 위해 쓰레드(thread)를 이용하였다.

<그림 13>은 메시지 접수 협력을 순차도로 나타낸 것이며, 시간적 순서로 이를 설명하면 다음과 같다. 익명의 MessageGetter 객체가 지역의 워크플로우 수행과 함께 하나의 쓰레드로 수행된다. 이 객체는 네트웍 상으로부터의 메시지를 계속 기다린다. 네트웍 상에서 메시지 전송이 감지되면, 해당 메시지에 대한 제어권을 전달 인자로 ServerThread 객체를 즉시 하나 생성하고, run() 메서드로 이를 실행시킨 후, 다시 네트웍 상으로부터의 메시지를 계속 기다린다. 쓰레드의 일종인 ServerThread 객체는 수행 시작 후 넘겨받은 메시지 제어권으로 네트웍으로부터 전송되어 오는 메시지를 모두 읽는다. 그런 후 접수받은 문자 스트링 형식의 메시지를 전달인자로 MessageParser 객체를 생성한다. MessageParser 객체는 넘겨받은 메시지의 헤더 부분을 해석하고, 바디 부분의 내용을 Document 객체로 생성한다. 그리고, MessageParser 객체는 해석된 메시지 내용을 토대로 지역의 워크플로우를 실행하여야 하는데, 이를 위해, WfRequester 객체의 receive_event() 메서드를 호출하며, 이때의 전달인자는 Document 객체인 d가 된다.



<그림 13> 메시지 접수 협력 순차도

V. 프로토타입 시스템 구현

5.1 구현 언어와 시스템 환경

5.1.1 구현 언어

본 연구의 프로토타입 시스템의 프로그램들은 자바 언어로 구현되었다. 자바는 대표적인 객체 지향적 프로그래밍 언어이기 때문에 본 연구에서 적용한 객체 지향적 분석·설계를 실제 코드화하기에 적합하였다. 또한, 자바 소스 코드 및 컴파일된 코드는 운영체제와는 독립적인 것이기 때문에, 다양한 운영체제 환경에서 실행시키는 것이 가능하다. 자바 언어로 시스템을 구현하기 위해서는 표준 라이브러리와 컴파일 및 실행 도구가 필요하다. 본 연구에서는 대표적 자바 표준 라이브러리인 Java API 1.2[20]와 컴파일 및 실행 도구가 포함된 썬 마이크로시스템즈사의 Java 2 SDK을 이용하였다.

본 연구의 메시지 구현 포맷은 XML로 하였는데, XML은 일반적인 문자 포맷과 호환될 수 있다. XML은 현재 가장 많이 이용되는 웹 문서 표준 포맷인 HTML을 대체하기 위한 표준 마크업 언어이며, SGML의 복잡성을 피하면서, 동시에 HTML의 제약을 극복할 수 있다는 장점을 가지고 있다[1]. XML의 응용 분야는 매우 다양하나 최근 XML의 주목할 만한 활용분야는 B2B 전자상거래 분야인데, 특히 자동화된 문서

교환 혹은 메시징에서 많이 활용되고 있다. XML 형태의 문서를 작성하고, 작성된 XML 문서를 해석하기 위한 라이브러리는 썬 마이크로시스템즈사의 Project X Core API를 이용하였다.

5.1.2 구현 시스템 환경

자바의 특성상 특정 구현 환경에 구애받지 않으나, 본 연구의 프로토타입 시스템은 고유한 IP(Internet Protocol) 주소를 가지는 인터넷 상의 윈도우즈 98 컴퓨터와 레드햇 리눅스 6.0 서버에서 수행되었다. 프로토타입 시스템 모두는 소스의 변경과 재 컴파일 없이 두 대의 시스템에 동일하게 설치되었고, 실행시 명령행 인자만을 상이하게 주었다.

5.2 프로토타입 시스템 소프트웨어

예제 워크플로우 엔진은 전체 프로토타입 시스템의 일부로 구성되어 있는데, 메시징 시스템의 시험적 작동을 위한 기능만을 소프트웨어로 구현한 것이다. 분석 단계의 예제 시나리오를 다시 정리하면 다음과 같다.

> *Customer(Company_A)가 Producer(Company_B)에 order_a 메시지를 보내고, 자신은 Producer로부터 notification 메시지를 기다린다. order_a 메시지는 주문서(purchase order)를 포함하고 있으며, notification은 주문확인서(order confirm)를 포함하고 있다. order_a 메시지를 받은 Producer는 자신의 워크플로우를 시작하고, 몇 가지 활동을 수행한 후, notify 활동을 수행하게 되는데, 이 활동을 통해 notification 메시지가 Customer에 전달된다. notification 메시지를 전달 한 후 Producer의 워크플로우는 계속 자신의 활동을 수행하고, Customer의 워크플로우도 Producer로부터 notification 메시지를 전달받은 후 자신의 워크플로우를 계속 수행한다[5, p69].*

B2B 워크플로우 시나리오에 나타나는 Customer와 Producer 간에 교환되는 메시지는 order_a, notification, delivery_a, invoice, payment 등이 있는데, 본 연구에서는 프로토타입 메시징 시스템의 작동을 시험하기 위해 이 모두를 XML 문서로 작성하였다. <그림 14>는 그 중에서 order_a 메시지를 보여 주고 있으며, 이는 설계 단계에서 XML DTD로 작성한 메시지의 구조를 그대로 따르고 있다. order_a를 비롯한 다섯 개의 메시지를 나타내는 XML 파일과 XML DTD 파일의 메시지 바디 부분에 포함된 주문서와 같은 내용은 실무에서 많이 사용되는 형식을 참고하였다.

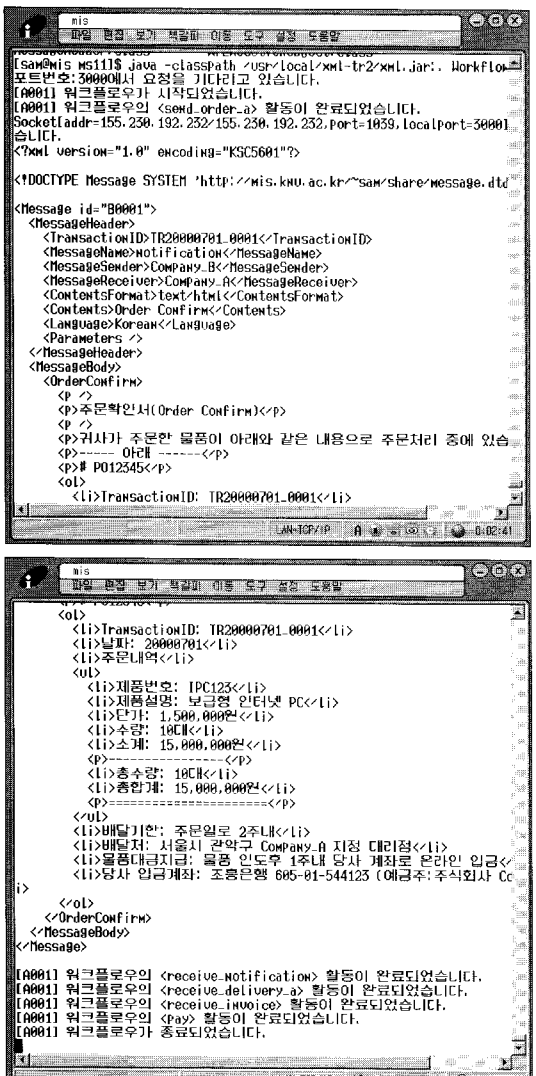
```
<?xml version='1.0' encoding="KSC5601"?>

<!DOCTYPE Message SYSTEM "http://mis.knu.ac.kr/~sam/share/message.dtd">

<Message id='A0001'>
    <MessageHeader>
        <TransactionID>TR20000701_0001</TransactionID>
        <MessageName>order_a</MessageName>
        <MessageSender>Company_A</MessageSender>
        <MessageReceiver>Company_B</MessageReceiver>
        <ContentsFormat>text/html</ContentsFormat>
        <Contents>Purchase Order</Contents>
        <Language>Korean</Language>
        <Parameters>
            <PriceUnit>Korean Won</PriceUnit>
        </Parameters>
    </MessageHeader>
    <MessageBody>
        <PurchaseOrder>
            <Date>20000701</Date>
            <OrderContact>박만수</OrderContact>
            <Company>주식회사 Company_B</Company>
            <Address>대구시 남구</Address>
            <Items>
                <ProductNo>IPC123</ProductNo>
                <Description>보급형 인터넷 PC</Description>
                <UnitPrice>1500000</UnitPrice>
                <Qty>10</Qty>
                <Total>15000000</Total>
            </Items>
            <Delivery>
                <Address>서울시 관악구 Company_A 지정 대리점</Address>
            </Delivery>
        </PurchaseOrder>
    </MessageBody>
</Message>
```

<그림 14> 예제 메시지(주문서)

5.3 실행 화면

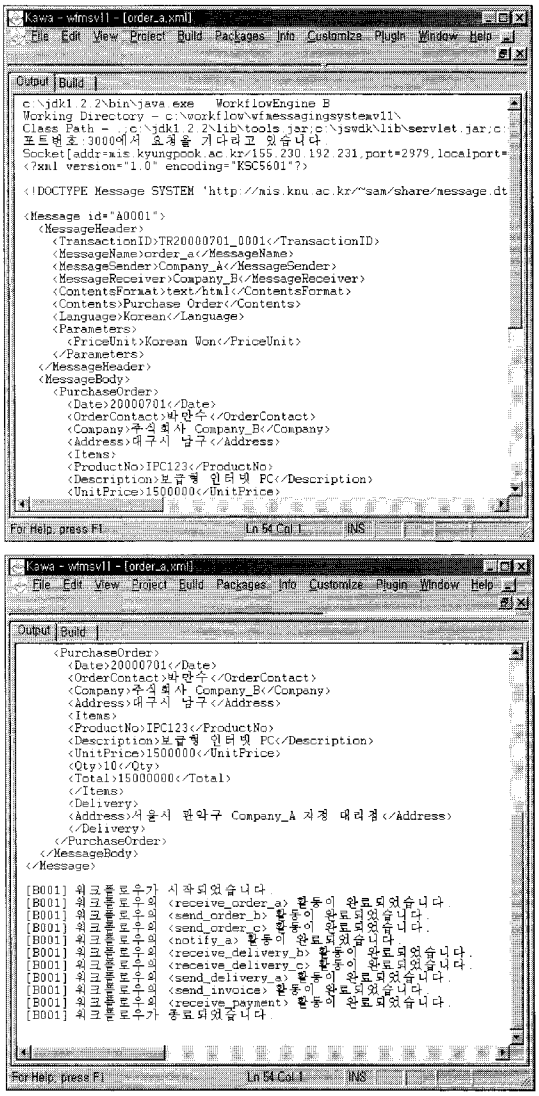
프로토타입 시스템은 하나의 실행 파일에 의해 수행되는데, 실행시 명령행 전달 인자를 A로 주면, Company_A사 워크플로우와 메시징 시스템이 수행되고, 전달 인자를 B로 주면, Company_B사의 워크플로우와 메시징 시스템이 실행되도록 하였다. Company A사의 워크플로우는 명령문 실행 후 즉시 수행되나, Company_B사의 워크플로우는 Company_A 사로부터 order_a 메시지를 전달받아야 실제 실행된다.

프로토타입 시스템은 인터넷으로 연결된 두 대의 컴퓨터에 설치하였고, 실행시 명령행 전달 인자를 A와 B로 각각 다르게 주었다. <그림 15>는 Company_A사의 워크플로우와 메시징 시스템이 실행되는 모습을 보여 준다.



<그림 15> Company_A사의 메시징 시스템 실행 화면

Company_B사의 예제 워크플로우와 메시징 시스템은 인터넷 상의 다른 컴퓨터에서 실행되었는데, <그림 16>은 Company_B사의 예제 워크플로우와 메시징 시스템이 실행되는 것을 보여준다. 그림에 나타난 XML 형태의 메시지는 메시지를 보내거나 받을 때, 그 내용을 화면 출력한 것이다. 예를 들어, <그림 15>에는 Company_A가 Company_B에 order_a 메시지를 보내기 전에 화면 출력한 내용이 나타나며, <그림 16>에는 Company_B가 Company_A로부터 받은 order_a 메시지의 내용을 다시 화면 출력한 것이 나타난다.



<그림 16> Company_B사의 메시징 시스템 실행 화면

Ⅵ. 결 론

본 연구에서는 워크플로우 상호운용성, 기업 간 운용되는 워크플로우, B2B 전자상거래 환경의 워크플로우에 관한 선행 연구를 통해 워크플로우가 B2B 전자상거래를 지원하는 정보기술로서의 워크플로우의 잠재력을 밝히고, B2B 전자상거래 환경에서의 가상의 B2B 워크플로우를 설정하고, 이를 수행하기 위한 메시징 시스템을 분석, 설계, 프로토타입을 구현하였다.

B2B 워크플로우는 "B2B 전자상거래 환경에서 상거래 수행과 협업을 목적으로 다수의 조직이 협력하여 수행하는 워크플로우"를 의미한다. 메시징 시스템을 통해 다수 워크플로우를 연결할 때의 장점은 각 조직에서 독립적으로 운영되는 워크플로우 관리 시스템을 그대로 이용하여 B2B 전자상거래를 수행할 수 있고, EDI 시스템에 비해 거래 대상 기업이나 거래 내용의 변경을 용이하게 할 수 있다는 점이다.

시스템의 분석 및 설계 단계에서는 객체지향적 방법을 채택하였고, 표준 모델링 도구인 UML을 주로 이용해 메시징 시스템을 모델링하고 상세화 하였다. 분석 및 설계 단계의 주된 결과물은 UML 활동도, 쓰임새도, 클래스도, 순차도, XML DTD 등이며, 각 도해는 시스템의 다양한 측면을 투영하였다. 마지막, 프로토타입 시스템 구현 단계에서는 분석 및 설계된 시스템의 일부를 자바 언어로 프로그래밍 하였다.

B2B 워크플로우 분석, 메시징 시스템의 분석, 설계 및 프로토타입 구현 단계에서 채택한 객체지향 개발 방법론, UML, 자바, XML 등은 널리 사용되는 표준화된 방법론과 도구들로써 본 연구의 관련 연구 및 응용을 위해 쉽게 공유될 수 있다.

그러나, 본 연구에서 설계한 메시징 시스템은 Aalst[4]의 시나리오를 바탕으로 시스템의 구조적·행동적 측면을 중심으로 분석되었으나, 실제 응용을 위해서는 보다 상세한 설계를(예를 들면, 가격설정 또는 재고확인 등이 필요한 경우의 사용자의 개입을 지원하기 위해서, 메시지를 접수하여 parsing을 거친 후 적절한 사용자의 개입을 요청하도록 하는 제약조건의 추가 등) 필요로 한다.

본 연구를 보다 발전시키기 위한 향후 연구 방향을 요약하면 다음과 같다.

첫 번째, 보다 복잡하고 다양한 B2B 워크플로우 모델을 (예를 들면, 다수의 구매자와 판매자가 참여하는 e-Marketplace 등) 분석한다.

두 번째, 실제 운영되는 워크플로우 관리 시스템을 대상으로 워크플로우 및 메시징 시스템을 모델링한다.

마지막으로, 보다 정교한 분석과 설계를 시도한다. 이 경우 서로가 연결된 더 많은 UML 도해들을 작성하고, 그 내용 또한 보다 정교하게 표현할 수 있어야 하는데, 현재 개발되어 있는 상용 툴이 그 역할을 할 수 있다.

〈참 고 문 헌〉

[1] 김용권 역, Harold, E., *XML Bible*, 정보문화사, 1999.

[2] 안승해, 백창현, *워크플로우*, 시사컴퓨터, 2000.

[3] 임춘봉, 신인철, 심재철 공역, Booch, G., Rumbaugh, J., and Jacobson, I., *UML 사용자 지침서*, 인터비전, 1999.

[4] Aalst, W., "Process-oriented architectures for electronic commerce and interorganizational workflow," *Information Systems*,

Vol. 24, No.8, 1999, pp. 639-671.

[5] Aalst, W. "Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries," *Information & Management*, Vol. 37, No. 2, 2000, pp. 67-75.

[6] Alonso, G., Fiedler, U., Hagen, C., Lazcano, A., Schuldt, H., and Weiler, N., "WISE: business to business E-Commerce," *9th International Workshop on Research Issues on Data Engineering*, Sydney, Australia, March, 1999, pp. 23-24.

[7] Bolcer, G.A. and Taylor R.N., "Advanced workflow management technologies," http://www.ics.uci.edu/pub/endeavors/docs/AdvancedWorkflow.pdf, 1998.

[8] Bons, R.W.H., Lee, R.M., Wagenaar, R.W. and Wrigley, C.D., "Modelling inter- organizational trade procedures using documentary petri nets," in *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences, Vol. III*, IEEE Computer Society Press, 1995, pp. 189-198.

[9] Cichocki, A., Helal A., and Woelk D., *Workflow and Process Automation Concepts and Technology*, Kluwer Academic Publishers, 1998.

[10] Dewitz, S. D., *Systems Analysis and Design and the Transition to Objects*, McGraw-Hill, 1996.

[11] Finin, T., Fritzson, R., McKay, D., and McEntire, R., "KQML as an agent communication language," *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg, MD, USA, November, 1994, pp. 456-463.

[12] Georgakopoulos D. and Hornick M., "An overview of workflow management: from process modeling to workflow automation infrastructure," *International Journal of Distributed and Parallel Databases*, Vol. 3, No. 2, 1995, pp.119-153.

[13] Hollingsworth, D., "Workflow - A model for integration", *ICL Systems Journal*, Vol. 12, No. 2, November, 1997, pp. 213-232.

[14] Kalakota, R. and Whinston, A. B., *Frontiers of Electronic Commerce*, Addison-Wesley, Reading, MA, USA, 1996.

[15] Kimbrough, S.O. and Moore, S.A., "On automated message processing in electronic commerce and work support systems: speech act theory and expressive felicity," *ACM Transactions on Information Systems*, Vol. 15, No. 4, October, 1997, pp. 321-367.

[16] Kobryn, C., "UML 2001: a standardization odyssey," *Communications of the ACM*, Vol. 42, No. 10, October, 1999, pp. 29-37.

[17] Merz, M., Liberman, B., and Lamersdorf, W., "Using mobile agents to support interorganizational workflow management," *International Journal on Applied Artificial Intelligence*, Vol. 11, No. 6, September, 1997, pp. 551-572.

[18] Moore, S.A. "Categorizing automated messages," *Decision Support Systems*, Vol. 22, No. 3, March, 1998, pp. 213-241.

[19] Object Management Group, *Workflow Management Facility*, Workflow Revision Task Force V 1.3 Report, 2000. (dtc/2000-02-05)

[20] Sun Microsystems, "Java Message Service API," (http://www.javasoft.com/products/jms/index.html), June, 2000.

[21] Workflow Management Coalition, *The Workflow Reference Model V 1.1*, WfMC Specification, 1995. (WfMC-TC-1003)

[22] Workflow Management Coalition, *Interface 4 - Interoperability Abstract Specification V 1.0*, WfMC Specification, 1996. (WfMC-TC-1012)

[23] Workflow Management Coalition, *Interface 2 - Workflow Client Application Programming Interface (Interface 2 & 3) Specification V 2.0*, WfMC Specification, 1998a. (WfMC-TC- 1009).

[24] Workflow Management Coalition, *Interface 5 - Audit Data Specificatiosn, V 1.1*, WfMC Specification. 1998b. (WfMC-TC-1015)

[25] Workflow Management Coalition, *Interface 1 - Process Definition Interchange V 1.1*, WfMC Specification, 1999. (WfMC-TC-1016-P)

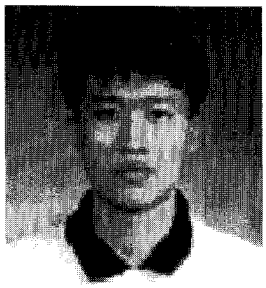
[26] Workflow Management Coalition, *Interface 4 - Interoperability Internet e-mail MIME Binding V 1.2*, WfMC Specification, 2000a. (WfMC-TC-1018)

[27] Workflow Management Coalition, *Interface 4 - Interoperability Internet XML Binding, 1.0 Beta*, WfMC Specification. 2000b. (WfMC-TC-1023)

◈ 이 논문은 2000년 9월 25일 접수하여 1차 수정을 거쳐 2001년 2월 6일 게재확정되었습니다.

◆ 저자소개 ◆

서창교 (Suh, Chang-Kyo)
공동저자 서창교는 포항공과대학교 산업공학과에서 경영정보시스템 전공으로 석사 및 박사학위를 취득하였다. 한국과학기술원 시스템공학연구소의 연구원으로 근무하였으며, 텍사스 주립대(UTHSCSA) 조교수, 계명대학교 조교수를 거쳐, 경북대학교 경영학부에 부교수로 재직중이다. 주요관심분야는 의사결정지원시스템, 전자상거래, 데이터웨어하우징 등이다.



김정삼 (Kim, Jeong-Sam)
공동저자 김정삼은 경북대학교 대학원 경영학과에서 경영정보시스템 전공으로 석사 학위를 취득하였다. 현재, 데이콤 시스템 테크놀러지에서 데이타웨어하우징 기반의 데이콤 다차원 고객 수익성 분석 시스템을 개발 중이며, 주요 관심분야는 데이터웨어하우징, 데이터마이닝, B2B 전자상거래, 객체지향 시스템 개발 등이다.

이형석 (Lee, Hyung-Seok)
공동저자 이형석은 경북대학교 대학원 경영학과에서 경영정보시스템 전공으로 석사 학위를 취득하고 현재 박사과정에 재학 중이다. 주요 관심분야는 전자상거래, 데이터베이스, 전사적자원관리 등이다.