IE Interfaces
Vol. 16, Special Edition, pp. 55-59, December 2003.

워크플로 시스템 간의 데이터 교환을 위한 이동 에이전시 설계

전수련 · 김선호†

명지대학교 산업시스템공학부

Design of Mobile Agency for Interoperability Among Workflow Systems

Soo-Ryun Chun · Sun-Ho Kim

Department of Industrial & Systems Engineering, Myungji University, Yongin, 449-728

For the interoperability between workflow systems, we have proposed data exchange scenarios based on mobile agents. The scenarios include sequence diagrams and methods defined by UML. The mobile agents are designed based on the mobile agent standard MASIF developed by OMG.

Keywords: workflow, mobile agent, MASIF, interoperability

1. 서 론

최근 PC와 인터넷의 보급으로 기업의 정보자원은 분산되고 이기종 하에서 운영되는 특징을 나타내고 있다. 기업의 분산은 워크플로 시스템 간의 상호운영을 더욱 더 요구하게 되며 서버에 즉시 응답을 요구하게 된다. 이때 상호응답 방식은 지금까지 이용되는 C/S(client/server) 방식과 이동 에이전트 방식을 고려할 수 있다. C/S 방식은 서버의 부하를 줄일 수 있는 장점이 있으나 분산환경에서 데이터 전송(interaction)시 네트워크의 부하가 과중하고 대기시간이 길어지는 단점들이 있다. 이동 에이전트 방식은 지능성, 이동성을 이용하여 수행할 실행 프로그램과 데이터를 한 번에 이동하여 로컬 시스템 서버 내에서 데이터를 주고 받은 후 네트워크를 통해 다시 돌아오게 된다. 이 경우 네트워크에서의 대기시간을 줄이고 네트워크 이용률을 높일 수 있다. 그러므로, 갈수록 복잡해지는 네트워크 환경에서 이기종 워크플로 시스템 간의 상호운영 능력을 높이기 위해서 이동 에이전트 기술과 워크플로를 결합하는 노력이 이루어지고 있다(Yang, 2001; 王新穎 2002). 본 논문에서는 이동 에이전트 기술의 장점을 이용하여 워크플로 시스템 간에 데이터를 교환하는 시스템을 설계하였다. 이 시스템은 OMG (Object Management Group)의 MASIF(Mobile Agent System Interoperability Facility) 표준에 의해 설계되었으며, WfMC의 Wf-XML을 교환할 수 있도록 이동 에이전트 관리 기능들을 포함하고 있다.

2. 워크플로 상호운영 관련 연구동향

워크플로 엔진이나 시스템 간의 상호운영을 지원하기 위한 많은 연구들이 진행되고 있다. 이런 연구 동향은 크게 세 가지로 분류할 수 있다(Ahang, 2002). 즉, 계약(contract)기반의 워크플로 시스템(Crossflow.org), XML 양식 기반의 워크플로 시스템(Muehlen 2000), 에이전트기술을 이용한 워크플로 시스템이다(Yan, 2001; Wooldridge, 1995). 여기서는 이동 에이전트와 관련된 기술을 소개한다.

분산되고 이질적인 환경에서 워크플로 관리 시스템을 적용하려면 그 환경 고유의 특성을 고려해야 한다. 그러한 특성들은 구체적으로 거래 파트너의 자율성, 사용자의 유동성, 불안

†연락저자 : 김선호, 449-728 경기도 용인시 남동 산38-2, 명지대학교 산업시스템공학부, Fax : 031-330-6451,
E-mail : shk@mju.ac.kr

한 네트워크, 잦은 프로세스의 변화, 사용자의 참여에 의한 간섭 등 여러 가지가 있을 수 있다. 이동 에이전트를 이용한 워크플로 관리 시스템은 특히 네트워크가 불안정하거나 부하가 많이 걸리는 환경에서 적합한 것이며, 다음과 같은 이점을 가질 수 있다. (1) 이동 에이전트는 작업자로 하여금 유동성을 갖게 한다. (2) 워크플로의 품질, 성취도, 경제성을 높인다. 작업을 수행하기 위해 네트워크를 투명하게 사용하면서, 동시에 지역적인 자원을 충분히 활용할 수 있다. (3) 이동 에이전트는 JAVA 기반으로 되어 있기 때문에 추상적인 수준에서 접근의 동질성을 제공할 수 있다.

이동 에이전트에 관한 표준은 MASIF와 FIPA(Foundation for Intelligent Physical Agents) 두 가지가 있다. FIPA(FIPA 2002)는 원격 통신 서비스를 기반으로 지능 에이전트 간의 통신을 목적으로 개발되었다. 이러한 이유로 인하여 이동성에 대해서는 많이 고려하지 않은 단점이 있다(Standard). 1995년 OMG는 에이전트 플랫폼 간의 상호운영성을 촉진하기 위해 1998년 MASIF를 OMG의 표준으로 채택하였다. FIPA는 에이전트 통신 패러다임를 채택하였으며 인공지능 기술과 연계시킬 수 있는 장점이 있다. MASIF는 이동 에이전트 패러다임을 채택하여 응용 컴포넌트의 동적(dynamic) 및 자율(autonomous)적인 교환, 대체, 수정, 갱신을 필요로 하는 환경에서 더 적합하다. 그래서 이 논문에서는 MASIF를 채택하여 사용하였다(OMG 2000).

3. 워크플로 시스템 간의 상호운영을 위한 에이전시 설계

3.1 시스템 모델

설계된 시스템 모델은 <그림 1>과 같다. 일반적인 워크플로 엔진에서 생성된 프로세스 정보는 Wf-XML 데이터 형태로 생성된다. 이것은 API(Application Program Interface)를 이용하여 MASIF 기반 하에 설계된 에이전시 (agency)로 전달된다. 에이전시는 Wf-XML로 정의된 프로세스 데이터를 포함하는 이동 에이전트를 생성하며, 인터넷 상에서 ATP(Agent Transfer Protocol)을 통해 이동 목적지로 이동(transfer)시킨다. 이동된 이동 에이전트는 에이전시와 Wf-XML 데이터와 관련 정보를 주고 받으며 트랜잭션을 수행(negotiate)한다. 트랜잭션을 완료한 이동 에이전트는 원래 보냈던 서버로 다시 돌아와 수행 결과를 에이전시에 전해주고 소멸된다. 여기서는 이동 에이전트 시스템의 핵심인 에이전시를 설계하였다. 에이전시와 워크플로 엔진의 구조는 <그림 2>와 같다. 여기서 워크플로 시스템은 엔진 부분과 Wf-XML을 생성 및 관리하는 Wf-XML OM(Operation Management) 모듈로 구성되어 있다. MASIF 표준의 의해 설계된 에이전시는 MAF Implementation 모듈과 Wf-MA Interface 모듈로 구성되어 있다. 여기서 MAF Implementation 모듈은 이동 에이전트를 관리하는 부분으로 MAFAgentSystem과 MAFFinder 인터페이스로 구성되어 있다. MAFAgentSystem은 클래스와 관련 데이터를 운반하기 위한 이동 에이전트를 생성, 중지, 종료, 재생, 이동, 수신하는 기능들을 포함하고 있다. MAFFinder는 생성된 이동 에이전트, 이동 에이전트의 위치, 이동 에이전트 시스템 정보를 등록시키는 기능을 가지고 있다.

Wf-MA Interface는 Wf-XML OM에서 Wf-XML과 관련 데이터를 주고받게 된다. MASIF 표준에서는 이러한 Wf-XML 데이터를 클래스 (class)로 정의하고 있으므로 앞으로는 클래스라고 부르기로 한다. Wf-MA Interface는 클래스와 관련 데이터를 MAFAgentSystem에게 넘겨주며 또한, MAFAgentSystem으로부터 클래스와 관련 데이터를 받아 Wf-XML OM에게 넘겨주는 역할도 한다.

3.2 MASIF 기반의 에이전시 설계

여기서는 임의의 두 워크플로 시스템 간에 이동 에이전트를 주고 받는 프로세스와 관련 메소드(method)를 설계하였다. 프로세스는 UML의 시퀀스 다이어그램(sequence diagram)으로 설계되었다.

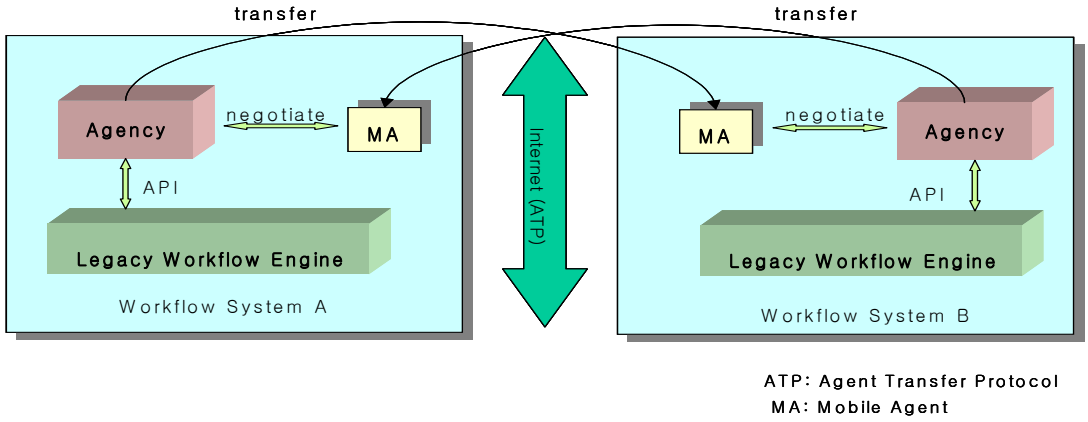

그림 1. 이동 에이전트를 이용한 워크플로 시스템 간의 상호운영(Yang 2001).

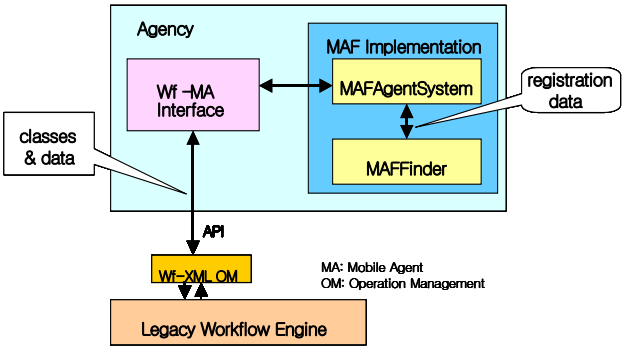

그림 2. MASIF 기반의 에이전시 구조.

3.2.1 이동 에이전트 생성 및 등록

Wf-XML OM과 에이전시 간의 데이터 교환은 Wf-MA Interface를 통해 이루어진다. 워크플로 엔진에서 외부 서브 프로세스로 연결하는 요청을 할 때 Wf-MA Interface는 Wf-XML OM에서 클래스와 관련 데이터를 받은 후 이동 에이전트의 구조에 맞게 데이터를 변환하여 MAFAgentSystem에 넘겨 주면서 이동 에이전트를 생성할 것을 요청한다.

Wf-MA Interface로부터 이동 에이전트 생성 요청이 오면 MAFAgentSystem은 이동 에이전트 생성 작업을 수행한다. 이동 에이전트가 생성되면 자신의 위치(location)를 MAFFinder에 등록해야 한다. 이 정보는 다른 곳으로 이동한 에이전트를 찾아 통신할 때 사용된다. 또한 이동 에이전트를 소멸시킬 때는 먼저 MAFFinder에 등록 정보를 삭제해야 한다.

이것을 위한 프로세스 및 관련 메소드는 <그림 3>에 요약되어 있다. Wf-XML OM은 request_dispatch() 을 통해 Wf-MA Interface에게 프로세스 데이터를 전송 요청한다. 이때에 workflow_engine_id, process_id, activity_id 등의 파라미터가 전해진다. Wf-MA Interface는 get_class( )와 get_data( )를 이용하여 보내기 위한 클래스와 관련 데이터 (workflow_ engine_id, process_id, activity_id, target_ workflow_engine_id, target_process_defini- tion_ name, class_list, class_location, context_ data 등)들을 가져온다. 그리고 MAFAgent System에게 이동 에이전트를 생성하는 것을 request_creation( )을 이용하여 요청한다. 이때, set_classes( )와 set_data( )를 이용하여, 신규 생성할 이동 에이전트의 구조에 맞게 클래스들과 데이터를 변환하여 전달한다. 여기서 클래스들은 이동 에이전트의 파라미터인 class_names과 code_base에 맞도록 매핑시킨다. class_names은 이동 에이전트를 인스턴스화시키는 데 필요한 클래스 이름의 리스트이다. code_base는 스트링값으로 주어지며 이 값은 에이전트 클래스가 실제 어떤 디렉토리에 있는지를 알려주어 에이전트를 생성하거나 이동할 때 쓰인다. 그리고, 관련 데이터는 이동 에이전트의 파라미터인 agent와 class_provider에 맞도록 매핑한다. 여기서 agent는 다른 요소에는 들어 있지 않은 데이터를 보내주기 위한 파라미터이며, class_provider는 클래스를 생성하는 제공자를 나타낸다.

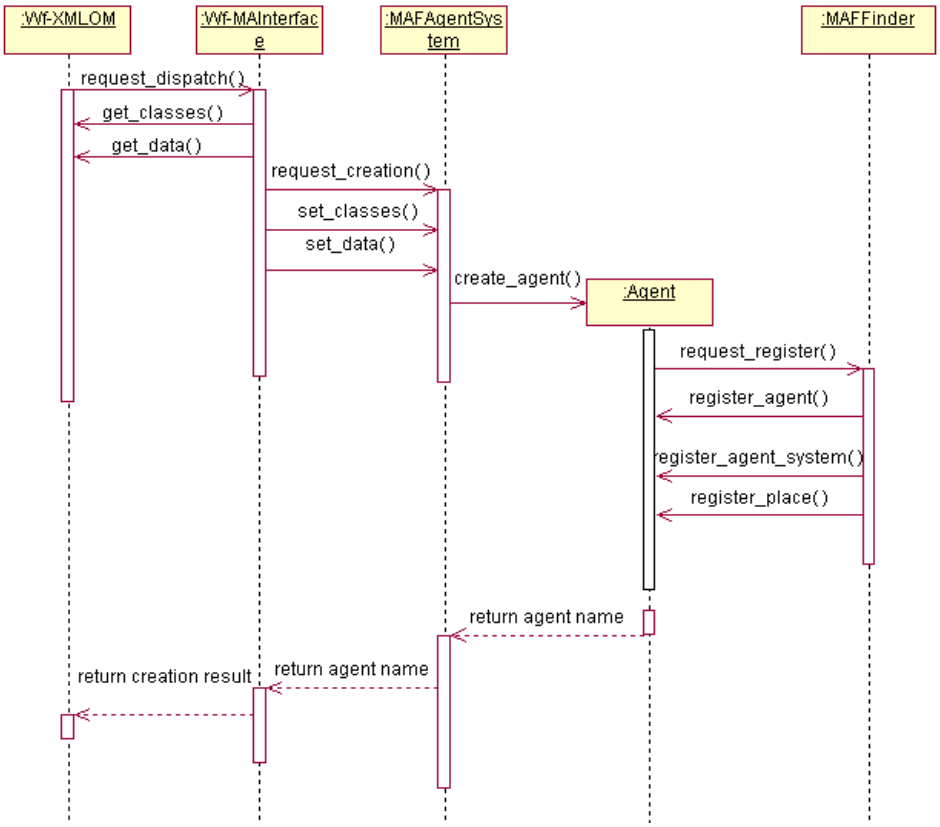

그림 3. 이동 에이전트를 생성 및 등록 프로세스.

MAFAgentSystem은 create_agent( )를 이용하여 이동 에이전트를 생성한다. 이때, 생성된 이동 에이전트는 agent_name, agent_profile, agent, place_name, arguments, class_ names, code_base, class_provider 파라미터 정보를 포함하게 된다. 생성된 이동 에이전트는 request_register( )를 이용하여 MAFFinder에게 등록 요청을 하게 된다. 그러면 MAFFinder는 register_agent( )와 register_agent_system( )을 이용하여 이동 에이전트에 대한 정보와 에이전트 시스템에 대한 정보를 등록하게 된다. 그리고, register_place( )를 이용하여 이동 에이전트의 place 정보를 등록하게 된다.

3.2.2 에이전트 이동

에이전트를 이동시키기 위해 우선 이동 목적지 에이전시의 위치 정보를 얻어야 한다. 이 정보들은 앞에서 이미 MAFFinder에 등록되었으므로 에이전트 이름을 통해 목적지 에이전시의 주소를 찾을 수 있다. 에이전트를 이동시키는 프로세스와 메소드는 <그림 4>에 나타나 있다. 이동 에이전트가 MAFAgentSystem에게 request_transfer( )를 통해 자신의 이동을 요청한다. 이 요청을 받은 MAFAgentSystem은 lookup_destination ( )을 이용해서 MAFFinder에게 목적지 정보를 요청한다. 목적지 정보를 받으면 MAFAgentSystem은 transfer_agent( )를 이용하여 이동 에이전트를 이동시킨다.

3.2.3 이동 에이전트 처리

이동 에이전트가 이동 목적지에 도착하면 이동 에이전트는 request_receipt( )를 이용하여 목적지에 있는 MAFAgentSystem에게 도착하는 것을 알려 준다. MAFAgentSystem은 receive_agent( )를 이용하여 이동 에이전트를 받는다. 받은 후에 get_authinfo( )를 이용하여 인증 정보를 점검한다. 인증을 통과하지 못한 이동 에이전트를 다시 클라이언트에게 되돌려 보낸다. 반대로 인증을 통과하면 이동 에이전트는 request_register( )를 이용하여 MAFFinder에게 이동 에이전트 등록을 요청한다. 그러면 MAFFinder는 register_agent()를 이용하여 이동 에이전트 정보를 등록한다. 등록하는 이동 에이전트가 다른 곳에서 작업을 마치고 돌아온 이동 에이전트이면 같은 agent_name으로 도착 일시만 다시 등록한다.

등록 후, 이동 에이전트는 request_do( )를 이용하여 원하는 클래스의 실행을 MAFAgentSystem에게 요청한다. 요청을 받은 MAFAgentSystem은 Wf-MA Interface에게 prepare_do( )를 이용하여 실행 준비를 요청한다. 이 요청을 받은 Wf-MA Interface는 receive_ classes( )와 receive_data( )를 이용하여 프로세스와 관련된 클래스 및 관련 데이터를 받는다. 받은 클래스들과 관련 데이터를 Wf-XML OM이 사용할 수 있는 구조로 변환하여 send_classes( )와 send_data( )를 이용하여 Wf-XML OM에게 넘겨준다. Wf-XML OM은 initiate_classes( )를 이용하여 받은 클래스를 워

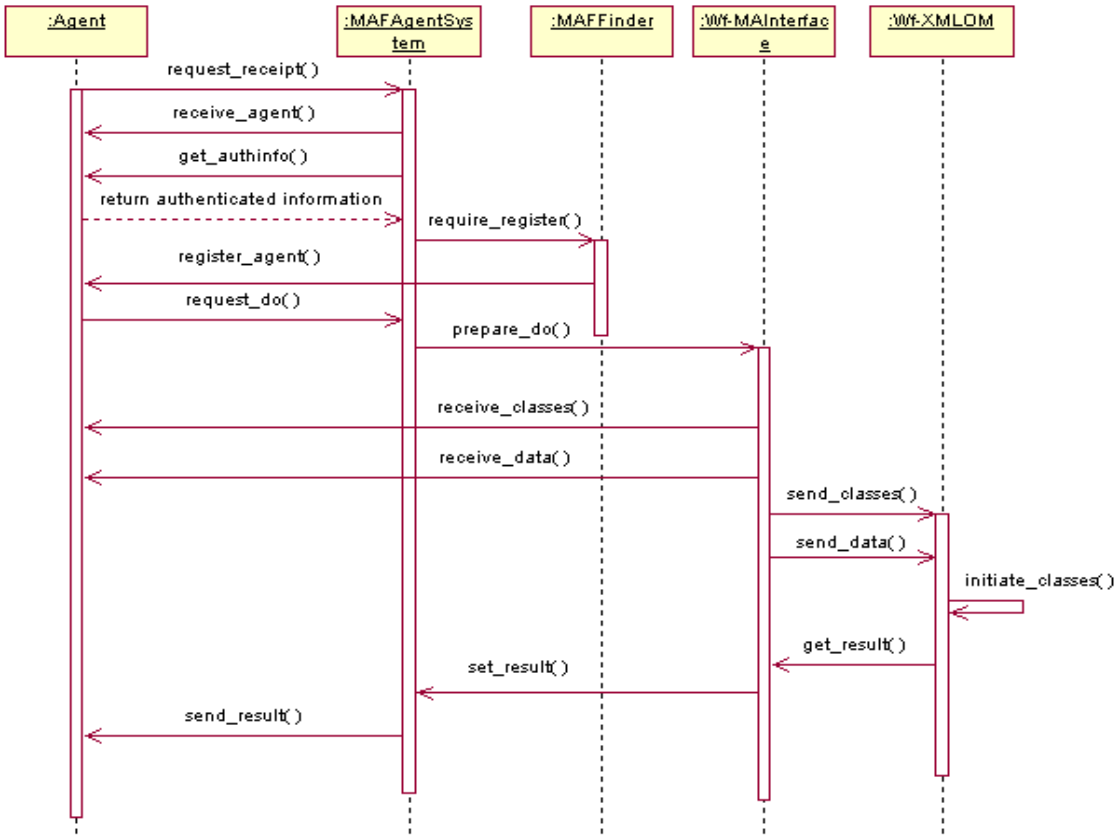

그림 4. 이동 에이전트 처리를 위한 프로세스.

크플로 엔진과 연계하여 실행한다. 실행 결과들은 Wf-MA Interface에 넘겨준다. 이 결과들은 성공적인 실행 결과 또는 실패한 결과와 실패 원인 또는 예외 상황 코드 등을 포함할 수 있다. Wf-MA Interface는 받은 결과를 에이전트 구조에 맞게 파라미터 agent로 변환시킨다. 변환된 결과는 set_result( )를 이용하여 MAFAgentSystem으로 보낸다. 이 결과는 다시 send_result( )를 이용하여 이동 에이전트에 전달된다. 이동 에이전트의 실행이 끝나면 이동 에이전트는 MAFAgentSystem에게 원래 보냈던 곳으로 이동시켜 줄 것을 요청하며 MAFAgentSystem은 이동 에이전트를 이동시킨다.

이동 에이전트가 실행중에 클라이언트는 상황에 따라 이동 에이전트의 실행을 suspend_agent( )를 이용하여 중지시킬 수 있다. 이동 에이전트의 수행이 완료되어 돌아오면 MAFAgentSystem은 terminate_agent( )를 이용하여 에이전트를 소멸시킨다.

3.2.4 이동 에이전트의 상태 보기

클라이언트 쪽의 에이전시는 이동 에이전트의 상태를 감시할 수 있다. 이동 에이전트의 상태를 보려 할 때 우선 클라이언트의 MAFAgentSystem은 lookup_agent( )를 이용하여 MAFFinder로부터 이동 목적지(서버)를 알아낸다. 그리고 서버의 MAFAgentSystem으로부터 get_agent_ status( )를 통해 이동 에이전트의 상태 정보를 얻게 된다.

4. 결론 및 추후 연구 과제

본 연구에서는 분산 환경에서 인터넷을 통해 워크플로 시스템 간에 데이터를 교환할 수 있는 이동 에이전트 시스템의 프로세스와 메소드를 MASIF 기반하에 설계하였다. 여기서 프로세스를 UML의 시퀀스 다이어그램으로 표현한 것은 이동 에이전트의 실행 시나리오를 보여주기 위한 것이다. 이것이 좀 더 상세히 설계하기 위해서는 이 시나리오를 이용하여 UP (unified process)와 같은 설계 프로세스를 따라 클래스 다이어그램, 시퀀스 다이어그램 등을 상세 설계하여 클래스, 속성, 메소드, 프로세스 등을 도출하는 것이 바람직하다.

참고문헌

Ahang, Y., Shi, M.,(2001), "Workflow Interoperability - Enabling e-Business," *Proceedings of the Sixth International Conference on Computer Supported Cooperative Work in Design*.

Crossflow.org, CrossFlow Project Overview, http://www.crossflow.org/flyer.html

Standard for Agents: MASIF and FIPA Specifications, http://www.icsuci.edu/~mingl/ agent.html

FIPA, (2002), FIPA Agent Management Specification, http://www.fipa.org/specs/fipa00023/SC00023J.pdf.

Muehlen, M.Z.,(2000), A Framework for XML-based Workflow Interoperability - The AFRICA Project, *Report of University of Muenster Department of Information Systems*.

OMG, (2000), Mobile Agent Facilities Specification v1.0, http://www.omg.org/docs/formal /00-01-02.pdf

Wooldridge and Jennings (1995), Intelligent Agents, Lecture Notes in Artificial Intelligence #890, Springer-Verlag,.

Yang, W., Li, S., Guo, M. (2001), "Mobile Agent: Enhancing Workflow Interoperability," *Proceedings of the 2001 International Conferences on Info-tech and Info-net*, V5, Beijing.

Yan, Y., Maamar, Z., Shen, W.,(2001) "Integration of Workflow and Agent Technology for Business Process Management," *Proceedings of the Sixth International Conferences on Computer supported Cooperative Work in Design.*

王新穎 (2002), 移動Agent實現技術, http://www.china-pub.com/computers/eMook/1391/info.htm.