이동 에이전트를 이용한 워크플로우 상호운영시스템 설계

Design of the Workflow Interoperability System Using Mobile Agent

김선호(Sun-Ho Kim)*, 전수련(Soo-Ryeon Jeon)*

초 록

기업의 구조가 복잡해지고 분산화 되면서 기업간 또는 부서간의 프로세스 자동화를 위해서 워크플로우 시스템간의 호환성이 중요한 이슈로 떠오르고 있다. 이러한 워크플로우 시스템간의 상호 운영성을 위하여 이 논문에서는 이동 에이전트 기반의 데이터 교환 시나리오를 제안한다. 이 시나리오는 UML을 이용하여 정의되는 순차 다이어그램과 방법을 포함하고 있다. 여기서 이동 에이전트는 OMG에서 제시하는 이동 에이전트 표준인 MASIF를 기반으로 설계되었다. 그리고, 이동 에이전트를 생성하고 이동시키는 프로토타입을 개발하여 이동 에이전트가 워크플로우간에 상호 호환이 가능함을 보여주었다.

ABSTRACT

As the structure of an enterprise becomes complicated and distributed, the interoperability between workflow systems for the process automation among partners or departments is rising as an important issue. For the interoperability between workflow systems, we propose a data exchange scenario based on mobile agents. The scenario includes sequence diagrams and methods defined by UML. The mobile agents are designed based on the mobile agent standard, MASIF, developed by OMG. In order to prove the applicability of the mobile agent technology to workflow interoperability, we have developed a prototype function which creates and transfers mobile agents.

키워드 : 워크플로우, 이동 에이전트, MASIF, 상호 운영성

Workflow, Mobile Agent, MASIF, Interoperability

본 연구는 명지대산업기술연구소의 2003년도 연구비를 지원받았음.

* 명지대학교 산업시스템공학부

1. 서 론

워크플로우 시스템은 업무조직이 복잡해지고 분산되여 감에 따라 업무의 흐름 및 정보의 교환에 많은 문제점이 생기게 되였다. 그리하여 워크플로우 시스템간의 상호 운영성(interoperability) 여부가 중요한 과제로 떠오르고 있다.

상호 운영을 위한 응답 방식은 지금까지 이용되는 C/S (client/server) 방식과 이동에이전트 방식을 고려할 수 있다. C/S 방식은 서버의 부하를 줄일 수 있는 장점이 있으나 분산 환경에서 데이터 전송(interaction)시 네트워크의 부하가 과중하고 대기시간이 길어지는 약점들이 있다. 이동 에이전트 방식은 지능성, 이동성을 이용하여 수행할 실행 프로그램과 데이터를 한번에 이동하여 로컬시스템 서버 내에서 데이터를 주고 받은 후 네트워크를 통해 다시 돌아 오게 된다. 이 경우 네트워크에서의 대기시간을 줄이고 네트워크 이용률을 높일 수 있다. 그러므로, 갈수록 복잡해지는 네트워크 환경에서 이기종 워크플로우 시스템간의 상호 운영 능력을 높이기 위해서 이동 에이전트 기술과 워크플로우를 결합하는 노력이 이루어지고 있다[1,14].

본 논문에서는 이동에이전트 기술의 장점을 이용하여 워크플로우 시스템간에 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 이동에이전트 관리 기능들을 포함하고 있다.

2. 워크플로우 상호 운영 연구동향

워크플로우 엔진이나 시스템 간의 상호 운영을 지원하기 위한 연구 동향은 크게 세가지로 분류할 수 있다. 즉, 계약(contract)기반의 워크플로우 시스템, XML양식 기반의 워크플로우 시스템, 에이전트기술을 이용한 워크플로우 시스템이다[3].

계약기반의 워크플로우 시스템은 가상기업들이 각각 동등한 위치에서 계약관계에 놓이며, 각 기업의 서비스가 캡슐화되고, 개별적으로 수행된다는 개념에 기초를 둔 것이다. 전형적인 계약기반의 워크플로우 시스템은 CrossFlow 이다[4]. 이 계약기반의 워크플로우 시스템은 tight-coupled 기업에적합하다. XML 기반의 워크플로우 시스템은 프로세스 정보를 XML 형식으로 교환하는 모델이다. WfMC에서 제안한 Wf-XML은 대표적인 사례이다[6].

에이전트기술은 워크플로우 시스템에서 세가지 형태로 응용될 수 있다, 즉, 에이전트 접목형 (agent-enhanced), 에이전트 기반 (agent-based), 이동 에이전트 기반 (mobile agent-based) 이다[9]. 에이전트 접목형은 에이전트 기술이 워크플로우 관리시스템에 응용된 형태이다. 많은 기존의 WFMS는 이 기술을 응용하였다 [13]. 에이전트 기반형은 프로세스 로직이 워크플로우 엔진 속에 있는 것이 아니라 에이전트 속에 포함되어 에이전트로 하여금 독자적으로 프로세스 분석, 자동화, 결합, 감시 등을 할 수 있게 한다[13]. 이동 에이전트는 특히 네트워크가 불안정하거나 부하가

많이 걸리는 환경에서 적합하다 [1].

이동 에이전트 관한 표준은 MASIF와 FIPA(foundation for intelligent physical agents)가 있다. FIPA[5]는 원격 통신 서비스를 기반으로 지능 에이전트간의 통신을 목적으로 개발되었다. 이러한 이유로 인하여 이동성에 대해서는 많이 고려하지 않고 있다[8]. FIPA는 에이전트 통신 파라다임를 채택하였으며 인공지능 기술과 연계시킬 수 있는 장점이 있다.

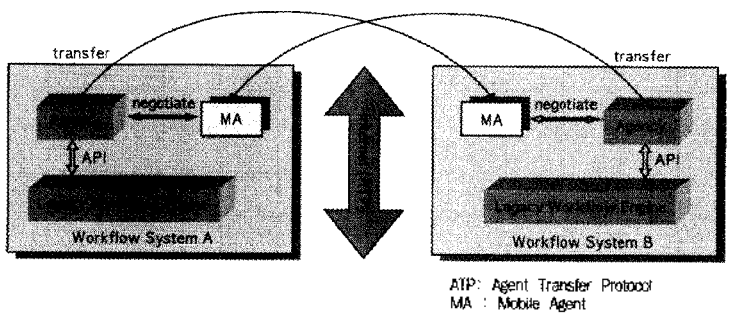
OMG는 에이전트 플랫폼 간의 상호 운영성을 촉진하기 위해 1998년 MASIF를 표준으로 채택하였다[7]. 이것은 이동 에이전트 파라다임을 채택하여 응용 컴포넌트의 동적(dynamic) 및 자율(autonomous)적인 교환, 대체, 수정, 갱신을 필요로 하는 환경에 적합하다. 본 논문에서는 이동 에이전트의 이동성에 중심을 두고 있으므로 MASIF 표준을 채택하여 시스템을 설계하였다.

3. 워크플로우 시스템간의 상호운영을 위한 에이전시 설계

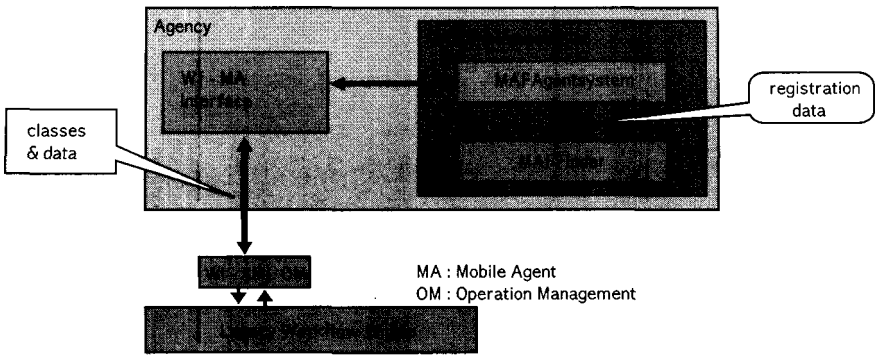
3.1 시스템 모델

설계된 시스템모델은 <그림 1>과 같다. 일반적인 워크플로우 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, 인터넷상에서 ATP (Agent Transfer Protocol)[10]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 전해 주고 소멸된다.

여기서는 이동 에이전트 시스템의 핵심인 에이전시를 설계하였다. 에이전시와 워크플로우 엔진의 구조는 <그림 2>와 같다. 여기서 워크플로우 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



<그림 1> 이동에이전트를 이용한 워크플로우 시스템간의 상호운영 [14]



〈그림 2〉 MASIF 기반의 에이전시 구조

Implementation 모듈은 이동에이전트를 관리하는 부분으로 MAFAgent System 과 MAFFinder 인터페이스로 구성되어 있다. MAFAgentSystem는 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-XML 과 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 이것으로부터 클래스와 관련 데이터를 받아 Wf-XML OM에게 넘겨주는 역할도 한다.

3.2 MASIF기반의 에이전시 설계

여기서는 임의의 두 워크플로우 시스템간에 이동 에이전트를 주고 받는 프로세스와 관련 메쏘드(method)를 설계하였다. 프로세스는 UML의 순차 다이어그램으로 설계되었다.

3.2.1 이동 에이전트 생성 및 등록

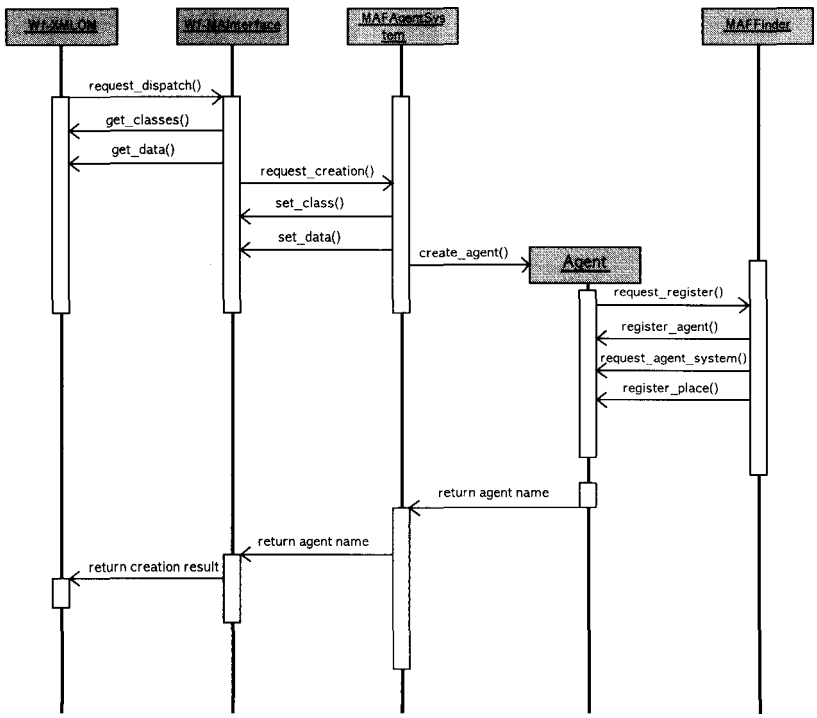
Wf-XML OM과 에이전시간의 데이터 교환은 Wf-MA Interface를 통해 이루어진다. 워크플로우 엔진에서 외부 서브 프로세스로 연결하는 요청을 할 때 Wf-MA Interface는 Wf-XML OM에서 클래스와 관련 데이터를 받은 후 이동 에이전트의 구조에 맞게 데이터를 변환하여 MAFAgent System에 넘겨 주면서 이동 에이전트를 생성할 것을 요청한다.

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 메쏘드는 〈그림 3〉과 〈표 1〉에 요약되어 있다. 〈표 1〉에

있는 메쏘드들 중 비고란에 "MASIF"로 표기된 것은 MASIF 표준에 정의된 메쏘드들이며 "추가"로 표기된 것들은 이 논문에서 추가로 제시된 메쏘드들이다.

Wf-XML OM은 request_dispatch() 을 통해 Wf-MA Interface에게 프로세스데이터를 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 데이터를 가져온다. 그리고 MAFAgentSystem에게 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_names 과 code_base 에 맞도록 매핑시킨다. class_names은 이동 에이전트를 인스턴스화 시키는 데 필요한 클래스 이름의 리스트이다. code_base는 스트링 값으로 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

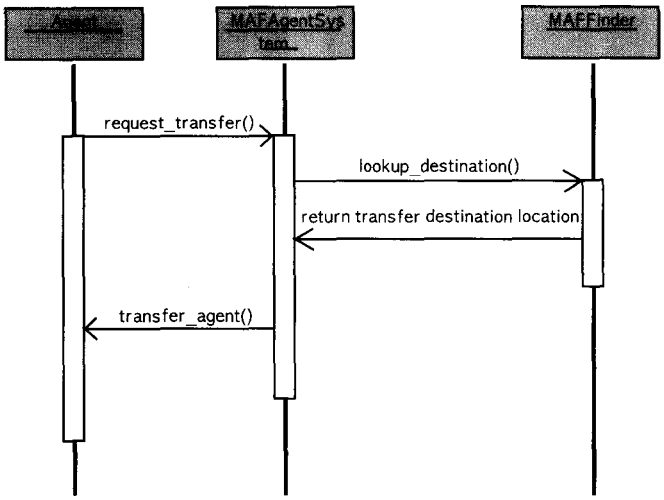
MAFAgentSystem은 create_agent()를 이용하여 이동 에이전트를 생성한다. 생성된 이동 에이전트는 register_agent()를 이용하여 MAFFinder에 등록요청을 하게된다. 그러면 MAFFinder는 register_agent()와register_agent_system()을 이용하여 이동 에이전트에 대한 정보와 에이전트 시스템에 대한 정보를 등록하게 된다. 그리고, register_place()를 이용하여 이동 에이전트의 place 정보를 등록하게 된다.

3.2.2 에이전트 이동

에이전트를 이동시키기 위해 우선 이동목적지 에이전시의 위치 정보를 얻어야 한다. 이 정보는 앞에서 이미 MAFFinder에 등록되었으므로 에이전트 이름을 통해 목적지 에이전시의 주소를 찾을 수 있다. 에이전트를 이동시키는 프로세스와 메쏘드는 〈그림 6〉에 나타나 있다. 이동 에이전트가MAFAgentSystem에게 request_transfer()를 통해 자신의 이동을 요청한다. 이 요청은 받은 MAFAgentSystem

〈표 1〉 이동 에이전트를 생성 및 등록을 위한 메쏘드와 파라미터

메쏘드	메쏘드 정의	파라미터	비고
request_dispatch()	프로세스 데이터전송	workflow_engine_id, process_id, activity_id	추가
get_classes()	관련 클래스들 가져오기	class list; class location	추가
get_data()	관련 데이터 가져오기	target_workflow_engine_id, target_process_definition_name, context_data	추가
request_creation()	이동 에이전트 생성 요청	class_name, class_location, data_location	추가
set_classes()	클래스를 파라미터로 변환	class_names, code_base	추가
set_data()	데이터를 파라미터 agent에 맞게 변환	agent, class_provider	추가
create_agent()	이동 에이전트 생성	agent_name, agent_profile, agent, place_name, arguments, class_ names, code_base, class_provider	MASIF
request_register()	등록 요청	agent_name	추가
register_agent()	이동 에이전트 등록하기	agent_name, agent_location, agent_profile	MASIF
register_agent_system()	시스템 정보 등록하기	agent_system_name, agnet_ system_location, agent_system_info	MASIF
register_place()	place정보 등록하기	place_name, place_location	MASIF



〈그림 4〉 에이전트를 이동 프로세스

〈표 2〉 에이전트 이동 위한 메쏘드와 파라미터

메쏘드	메쏘드 정의	파라미터	비고
request_transfer()	이동 에이전트의 이동 요청	agent_name	추가
lookup _destination()	이동목적지 찾기	agent_name	추가
transfer_agent()	이동 에이전트 보내기	agent_name	추가

은 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 MAFAgent System에게 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 이동에이전트는 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 도착일시만 다시 등록한다. 등록 후, 이동 에이전트는 request_do()를 이용하여 원하는 클래스의 실행 을 MAFAgentSystem에게 요청한다. 요청을 받은 MAFAgentSystem은 Wf-MA Interface에게 prepare_do()를 이용하여 실행 준비를 요청한다. 이 요청을 받은 Wf-MA Interface는 receiv_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 워크플로우 엔진과 연계하여 실행한다. 실행 결과들은 Wf-MA Interface에 넘겨준다. 이 결과들은 성공적인 실행 결과 또는 실패한 결과와 실패 원인 또는 예외상황 코드 등을 포함할 수 있다. Wf-MA Interface는 받은 결과를 에이전트 구조에 맞게 파라미터 agent 로 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. 이 프로세스와 메쏘드는 〈그림 5〉와 〈표 3〉에 나타나 있다.

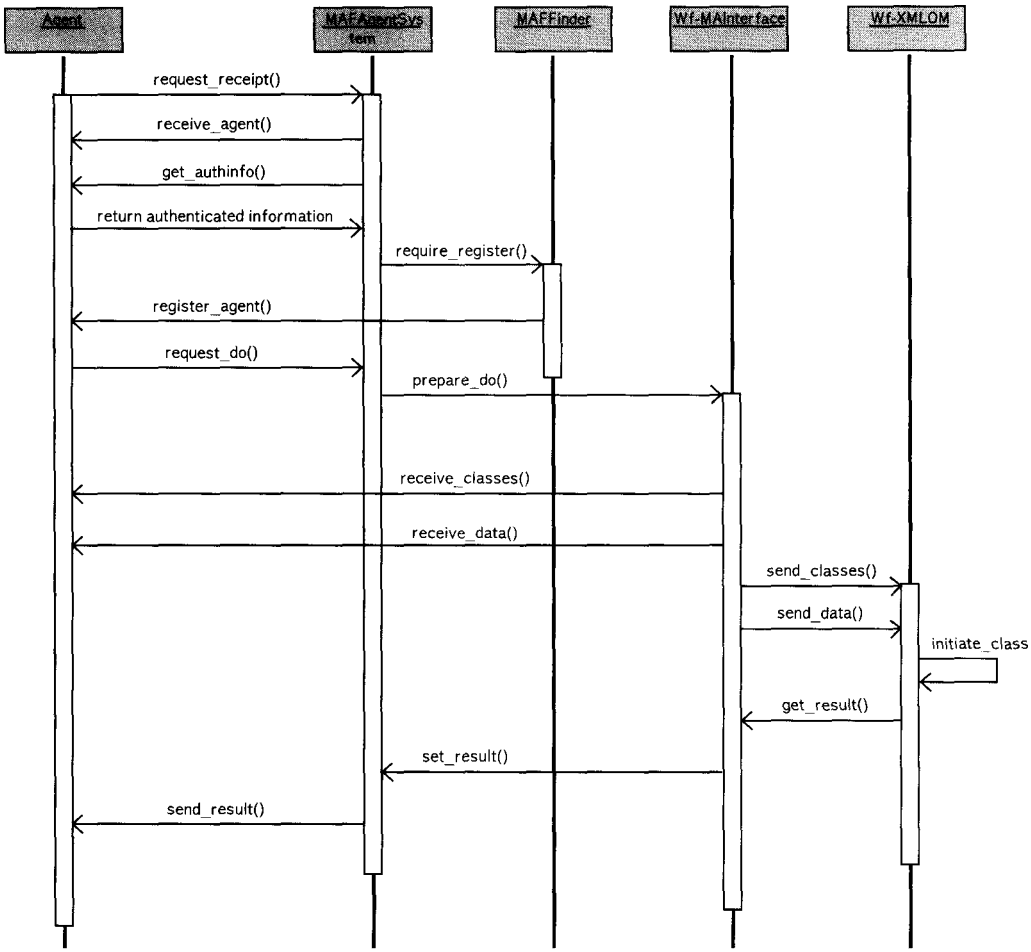
이동 에이전트가 실행 중에 클라이언트는 상황에 따라 이동 에이전트의 실행을 suspend_agent()를 이용하여 중지시킬 수 있다. 이동 에이전트의 수행이 완료되어 돌아오면 MAFAgentSystem은 terminate_agent()를 이용하여 에이전트를 소멸시킨다.

3.2.4 이동 에이전트의 상태보기

이동 에이전트의 상태를 보려 할 때 우선 클라이언트의 MAFAgent System은 MAFFinder로부터 이동 목적지 (서버)를 알아낸다. 그리고 서버의MAF AgentSystem로부터 통해 이동 에이전트의 상태 정보를 얻게 된다.

4. 이동 에이전트 프로토타입

본 논문에서는 이동 에이전트의 활용 가능성을 파악하기 위하여 MASIF 표준 기반으로 설계된 기능의 일부인 이동 에이전트 생성과 이동 기능을 구현하였다. 여기서는 IBM의 Aglet 툴 [11]을 사용하여 프로토타입을 개발하였다. 〈그림 6〉은 개발된 프로토타입을 보여주고 있다. 여기서 이동 에이전트인 WfAglet을 생성(create)하며 http://202.30.101.53으로 이것을 전송(dispatch)한다. 이 프로토타입을 통하여 이동 에이전트를 이용하여 워크플로우 정보를 교환하는 가능성을 검증하였다.



〈그림 5〉 이동 에이전트 처리를 위한 프로세스

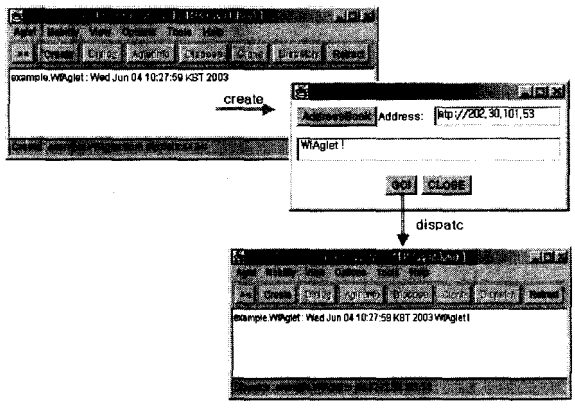
5. 결 론

본 연구에서는 분산 환경에서 인터넷을 통해 워크플로우 시스템간에 데이터를 교환할 수 있는 이동 에이전트 시스템의 프로세스와 메쏘드를 MASIF 기반 하에 설계하였다. 그리고 일부 기능을 프로토타입으로 구현해 가능성을 보여 주었다. 여기서 프로세스와 메쏘드를 UML의 시퀀스 다이어그램으로 표현한 것은 이동 에이전트의 실행 시나리오를 보여주기 위한 것이다. 이것을 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, 메쏘드, 프로세스 등을 도출하는 것이 바람직하다.

이동 에이전트 방식은 처리할 데이터가 많고 상호 교환이 자주 일어나며 대역폭

〈표 3〉 이동 에이전트 처리를 위한 메쏘드와 파라미터

메쏘드	메쏘드 정의	파라미터	비고
request_receipt()	도착하는 것을 알려주기	agent_name	추가
receive_agent()	도착한 이동 에이전트를 받기	agent_name; agent_profile; agent;place_name;class_names;code_base;agent_sender	MASIF
get_author()	이동 에이전트인증정보를 얻기	agent_name	MASIF
request_register()	받은 이동 에이전트 등록 요청	agent_name	추가
register_agent()	받은 이동 에이전트를 등록하기	agent_name	MASIF
request_do()	실행클래스 시작을 요청	agent_name; class_name	추가
prepare_do()	필요한 클래스와 데이터 요청	agent_name; class_name	추가
receive_classes()	수행에 필요한 클래스 받기	agent_name	추가
receive_data()	관련데이터 받기	agent_name	추가
send_classes()	클래스를 Wf-XMLOM에게 넘기기	class name; class location	추가
send_data()	관련데이터를 Wf-XMLOM에게 넘기기	workflow id; process defination name; ContextData	추가
initiate_classes()	시작클래스를 수행하기	class name; class location	추가
get_result()	실행결과정보를 보내기	process instance id, ResultData, Status, Exception	추가
set_result()	결과정보를 인수 agent에 매핑	agent_name; agent	추가
send_result()	인수 agent변화 내용 저장하기	agent_name; agent	추가



〈그림 6〉 이동 에이전트 생성 및 이동 프로토타입

(bandwidth)이 부족한 경우에 적합하다. 그리고 네트워크상의 트랜잭션이 적어서 워크플로우 시스템간 원격 서버에 즉시 응답을 해줄 수 있는 장점이 있다. 반면에 이동 에이전트가 갖는 가장 큰 단점은 취약한 보안성이다. 자신의 컴퓨터에서 수행되는 다른 사람의 이동 에이전트는 바이러스처럼 나쁜 영향을 끼칠 수 있다. 또 다른 문제점으로는 수많은 서버 중에서 자신이 원하는 서비스를 제공하는 서버를 어떻게 효율적으로 찾고 돌아오는가 이다. 이동 에이전트 기술이 성공적으로 e-비즈니스 등 실무 영역에 응용되려면 우선 이러한 문제들이 해결되어야 할 것이다.

참 고 문 헌

[1] 김용훈, 최인준, 이창우, "가상기업을 위한 모빌 에이전트 바탕의 워크플로우 관리 시스템," 대한산업공학회/한국경영과학회 98춘계공동학술대회 논문집, Apr. 24-25, 1998.

[2] 王新穎, "移動Agent實現技術," "http://www.china-pub.com/computers/eMook/1391/info.htm", Aug. 7, 2002.

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

[4] CrossFlow Project Overview, "http://www.crossflow.org/flyer.html"

[5] FIPA, FIPA Agent Management Specification, http://www.fipa.org/specs/fipa00023/SC00023J.pdf, Dec. 3, 2002.

[6] Muehlen, M.Z., A Framework for XML-based Workflow Interoperability - The AFRICA Project, University of Muenster Department of Information Systems, 2000.

[7] OMG, Mobile Agent Facilities Specification v1.0, http://www.omg.org/docs/formal/00-01-02.pdf, Jan. 2, 2000.

[8] Standard for Agents: MASIF and FIPA Specifications, http://www.ics.uci.edu/~mingl/agent.html"

[9] Wooldridge, Jennings, "Intelligent Agents," Lecture Notes in Artificial Intelligence #890, Springer-Verlag, 1995.

[10] Lange, D.B., Aridor, Y, Agent Transfer Protocol--ATP/0.1, IBM Tokyo Research Laboratory, March 19, 1997.

[11] Oshima, M., Karjoth, G., One, G., Agelts Specification 1.1 Draft, IBM Tokyo Research Laboratory, Sep. 8, 1998.

[12] Work Group 1, Workflow Management Coalition - Interface 4: Interoperability Wf-XML Binding, WfMC TC-1023 Issue May 1, 2000.

[13] Yan, Y., Maamar, Z., Shen, W., "Integration of Workflow and Agent Technology for Business Process

Management," Proceedings of the Sixth International Conferences on Computer supported Cooperative Work in Design, 2001.

[14] Yang, W., Li, S., Guo, M., "Mobile Agent: Enhancing Workflow Interoperability," Proceedings of the 2001 International Conferences on Info-tech and Info-net, Volume:5, Beijing, 2001.

저 자 소 개

김선호 (E-mail : shk@mju.ac.kr)
1979. 서울대 산업공학과(학사)
1989. Pennsylvania State University 산업공학과(석·박사)
국방과학연구소, 한국기계연구원 근무
현재 명지대학교 산업시스템공학부 교수,
전자상거래 표준화 통합포럼, 전자카탈로그 기술 위원회 부위원장
관심 분야 워크플로우, BPEL, 전자카탈로그 표준화

전수련 (E-mail : vdsl8835@kornet.net)
2001. 중국 대련민족대학 컴퓨터학과(학사)
2003. 명지대학교 산업공학과(석사)
현재 성일엔지컴 근무
관심 분야 워크플로우, 모바일 에이전트