정보기술응용연구
제2권 제1·2호
2000년 6월

지능형 에이전트와 페트리네트 모형화 기술을 활용한 ERP 유지보수 방법론

권 오 병*, 이 재 준**

요 약

ERP가 기업의 정보를 통합하는데 유용한 시스템이라는 것에는 의심할 바가 없다. 그러나 많은 회사들은 ERP시스템을 도입하는 것을 주저하고 있다. 주요한 이유는 유지보수를 포함한 ERP 구축에 엄청난 총비용이 들기 때문이다. 특별히 프로세스를 리엔지니어링(reengineering) 하거나, 시스템이 각 기업에 경영지식을 담도록 하는 맞춤화 작업이 적절한 시간에 수정 보완되어지지 않는다면 ERP가 내놓는 성과의 최적화의 결과를 보장할 수 없다. 그러므로 본 논문은 ERP시스템의 사용자와 데이터베이스에 의해 발생되는 환경적 변화 그 자체를 수용하는 ERP시스템을 구축하는데 목표를 두고 있다. ERP시스템이 최적의 성과를 유지하도록 하기 위해 페트리넷 이론(petri-net theory)과 지능형 에이전트 기술(Intelligent agent technology) 을 사용하였다. 지능형 에이전트는 데이터 베이스를 통합하고 예외적인 변화 상황을 발견하며, 그 변화가 ERP시스템의 성과에 어떤 영향을 주게 되는지에 대해서 각 에이전트끼리 자동적으로 협동하게 된다. 해당 에이전트의 역동성은 페트리넷으로 표현되었다. 새롭게 제안된 ERP시스템은 BPR(business process reengineering)의 프로세스들을 최적 상태로 유지하도록 만들 것이다. 마지막으로 제안된 ERP 유지 보수 시스템의 타당성을 설명적인 물류부분 사례를 통해 제시하였다.

* 한동대학교 경영경제학부 교수
** 포항공대 산업공학과

1. 서 론

96년 이후 한국은 주요 대기업을 중심으로 경쟁력 강화, 업무 개선, 원가 절감, 물류의 최적화, 정보의 효용화 등을 기대하며 전사적 자원관리(ERP, Enterprise Resource Planning)기술을 도입하기 시작하였다. 전사적자원관리 기술의 타당성을 점검하던 한국 기업들은 전사적자원관리가 주는 효용가치에 의문을 던지기보다 자사에 효과적으로 적용하는 방법을 찾는데 주력하고 있다. 이와 더불어 전사적자원관리시스템은 보고형이 아닌 신속한 의사결정을 위한 정확한 관리형 데이터들의 흐름을 제공하려는 시도가 있다. 특별히 한국의 기업들은 국제 표준화를 따르는 것이 요구되고 있다. 전사적자원관리의 도입은 위의 문제를 가진 한국의 기업에게 표준화된 비즈니스 프로세스, 재무제표 상의 연결성과 투명성을 제공함으로 시장에서의 긍정적인 역할을 담당하게 될 것으로 기대하고 있다.

그런데 전사적자원관리가 기업에 도입되어 성공적으로 운영되고 있는지에 대한 고려는 새로운 문제로 부각된다. 특히 일반적인 소프트웨어에 비하여 전사적자원관리는 다음과 같은 특성때문에 전통적인 유지보수 방법론을 그대로 적용하기 어렵다. 첫째, 전사적자원관리에서 가정하고 있는 모범 사례가 어떤 상황에서나 최선이 아니며, 환경이 변하면서 수시로 변경될 수 있다. 둘째, 전사적자원관리는 단회적 패키지이며 환경변화에 민감하지 못하기 때문에 책임성이 결여될 수 있다. 이것은 전사적자원관리 시스템이 정교하기 때문이기도 하다. 세째, 일반적인 소프트웨어에서는 자료 구조와 프로세스에 대해서 관심을 가지지만 전사적자원관리는 업무 실적 평가에 의하여도 유지보수 시점을 결정해야 한다. 넷째, 기존 소프트웨어보다 컨설팅 비용이 많이 소요되기 때문에 유지 보수 비용이 더 많이 든다[4]. 전사적자원관리 구현에 드는 비용요소는 크게 패키지 비용, 하드웨어 비용, 컨설팅 비용 등으로 나눌 수 있다. 일반적으로 컨설팅 및 교육 비용이 전사적자원관리 도입 시 드는 막대한 패키지 비용보다 더 든다고 알려져 있다. 이러한 유지보수 비용은 속도와 유연성이 강조되는 업무일수록 더욱 중요한 문제로 부각된다 [21, 24]. 그러나 전사적자원관리 시스템의 특성에 맞는 유지보수 방법론에 대해서는 아직 연구된 바가 없다. 특히 보편적 업무가 아니거나 변화가 잦은 업무가 많은 기업의 전사적자원관리는 해당 기업의 특성을 반영하면서 동시에 전체적인 무결성 및 불일치성을 유지할 수 있는 새로운 형태가 되어야 한다. 이를 위한 요구 조건을 다음과 같이 열거한다. 첫째, 특성화 관련 규칙이나 지침 등이 지식의 형태로 표현되어야 한다. 둘째, 특성화 규칙, 지침의 변경이 명료하게 이루어져야 한다. 세째, 특성화 규칙, 지침의 변경 시점에 대한 시스템 자체적인 진단이 가능해야 한다. 넷째, 한 부분의 변경이 다른 부분에 미치는 영향

에 대해 평가되고 반영되어야 한다. 다섯째, 영역 관리자에 의한 변경과 시스템적으로 내장된 자료 및 지식의 변동으로 인한 변경이 모두 허용되어야 한다.

결국 위의 요구 조건은 지식의 체계적 관리, 자율적인 시스템 유지 보수 능력을 가진 전사적자원관리 시스템을 의미한다. 그리고 이러한 요구 조건을 만족시킬 수 있는 정보기술로는 지능형 에이전트 기술이 유력하다.

본 논문의 목적은 현재의 전사적자원관리의 효율적 유지보수를 위해 연속적 자기 점검을 통하여 지식 관리 기능결여 및 의사결정의 결여 등을 해결하는 새로운 전사적자원관리 구축방법론을 제시하는 데 목적이 있다. 이를 위해 전사적자원관리 데이타베이스와 프로세스를 감시하여 유지보수 시점을 인지하고 자율적으로 유지보수를 하거나 유지보수 필요성에 대해서 관리자에게 통보하는 지능형 에이전트 기술을 활용하고자 한다. 이를 위해 컨설팅 지능이 능동성과 자율성을 기반으로 한 에이전트들을 전사적자원관리시스템에 내재하도록 한다. 또한 예외 상황 발생시 에이전트의 능동적인 활동으로 진단하고 사용자에게 통보하여 대응 작업이 수행되도록 한다. 또한 에이전트의 동적인 작동성을 모형화하기 위해 페트리네트 이론을 활용하였다. 페트리네트로 모형화된 에이전트들에 대한 성능 분석은 페트리네트 자체의 분석 방법을 사용할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 전사적자원관리 시스템의 유지 보수 관련한 이슈에 대해서 보다 상세히 고찰해 보고, 제3장에서는 유지 보수 문제를 극복하기 위하여 채택한 지능형 에이전트 기술에 대해서 기술하되 자율성을 지닌 에이전트와 전사적자원관리의 대상 업무인 환경과의 상호 작용성에 집중하였다. 제4장에는 본 논문에서 제안하는 지능형 전사적자원관리의 틀을 제시하고 그의 구성 요소가 되는 대표적인 에이전트들을 상술하였다. 그리고 구현 방안과 추후 연구방향을 제5장과 제6장에서 기술하였다.

2. 전사적자원관리 시스템의 유지 보수 이슈

전사적자원관리의 세부기능은 비즈니스 모델링 기능과 정보 하부구조 기능, 그리고 전사조직 기능으로 분류되며, 비즈니스 모델링 기능은 다시 선진 프로세스 제공, 각 기업에 적용하도록 하는 커스터마이징, 모의실험 기능으로 세분화되고, 정보 하부구조 기능은 통합화, 표준화, 외부조달, RAD 기능으로, 전사조직 능력 기능은 최적화 및 유연성 기능을 세분화된다. 특히 전사적자원관리의 향후 발전 전망으로 최적화 의사결정을 지원하는 전사적 컴퓨터 모델과 프로세스의 성능의 분석 및 평가를 위한 시뮬레이션 기능 등을 제시하고 있다 [5]. 또한 기존의 기업 정보시스템과 전사적자원관리와의 통합을 위한 시도들도 계속되고있다 [20,

8]. 특히 전사적자원관리모형과 기존 실시간 생산시스템의 통합을 위해 ISA S88 batch control standard 등이 제안되고 있다 [23]. 이러한 시도들 때문에 제조업체의 전사적자원관리는 CIM보다 더 포괄적인 전사적 통합시스템으로 인식되고 있다 [16] 또한 전사적자원관리와 문제해결 알고리듬과의 결합을 시도하는 노력들도 시작되었다 [9].

각 부서의 입장에서 볼 때의 최상의 소프트웨어를 구매하지만 IT부서에서는 통합 문제에 어려움을 가진다. 자료들의 불일치성, 각 부서의 통합에 대한 저항 등은 유지보수 비용의 증가를 유발하였다. 따라서 전사적자원관리와 같이 초기에 통합된 소프트웨어를 구입하는 것이 전통적인 시스템 통합 접근법보다 유지보수 비용이 적게 들 것을 예상된다. 또한 실제로 제조실행관리 (MES, Manufacturing Execution Systems), 공급사슬관리(SCM, Supply Chain Management), 제품정보관리(PDM, Product Data Management), 지리정보시스템(GIS, Geographic Information System) 등 주변 정보시스템과의 통합 연구 및 상용화가 활발히 진행되고 있다 [4]. 그러나 전사적자원관리와 같은 통합 소프트웨어에는 모든 기업에 공통적으로 적용될 기능 외에 특정 기업에 특성화된 기능들이 추가되어야 하는 부담을 가진다. 예를 들어 특정 사용자 편의의 인터페이스 구축, 기업마다 차별적인 정책 및 모형 관리는 범용화 된 전사적자원관리에서 지원될 수 없다.

따라서 전사적자원관리의 효과적 유지보수를 위해서는 다음과 같은 고려가 추가적으로 필요된다. 첫째로 전사적자원관리는 모범 사례(Best practice)에 근거하여 작성된다는 것이다. 이는 적용한 모범 사례가 더 이상 최적이 아닐 때에는 유지보수를 해야 함을 의미한다. 따라서 채택된 모범 사례의 최적성에 대한 사후분석을 지속적으로 수행하는 기능이 필요하다. 두번째로, 유지보수의 자율성에 대한 필요성이다. 전사적자원관리 자체적으로 유지보수 계획을 수립하고 유지보수의 범위 및 파급효과에 대해 분석하고 스스로 유지보수가 가능한지에 대한 평가를 한 후에 가능한 경우 자체적 유지보수 능력을 가지도록 하는 것이다. 특히 잘못된 전사적자원관리의 도입은 유지보수비용을 초래하여 결국 더 많은 소유비용을 유발하는 것으로 알려져 있다 [13]. 이를 어느 정도 해결하기 위해 최근 전사적자원관리의 컴포넌트화 작업이 추진되어 위험을 감소시키는 방법 등이 제안되고 있다. 예를 들어 SAP같은 경우는 인터넷, 데이터 웨어하우징, 인적 자원 관리, 계획, 판매 자동화 컴포넌트를 98년 중반에 개발한 바 있으며, 타 전사적자원관리에서도 비슷한 컴포넌트를 개발하여 제공하고 있다 [14]. 그러나 컴포넌트 위주의 전사적자원관리도 특정 범주 내에서의 일반화이기 때문에 기업 특성화를 완전히 반영하지는 못하고, 이에 따라 의도하지 않은 예외 상황이 발생에 대응하기가 어렵다 [1].

또한 기업 특성화를 충실히 반영하는 전사적자원관리를 개발하는 것은 전사적

자원관리 자체가 전사적 혹은 반전사적이기 때문에 한 부분의 반영이 다른 부분에 어떠한 영향을 주는지에 대해서 관리할 수 없다면 전체 시스템의 무결성 혹은 불일치성을 보장하기 어렵고, 이에 따라 유지보수 비용이 증가할 것으로 예상하고 있다 [25].

3. 지능형 에이전트

에이전트는 본질적으로 에이전트 기반 프로그래밍(Agent-oriented programming)과 객체 지향성에서 취할 수 있는 강점을 가지고 있다. 우선 에이전트 기반 프로그래밍의 강점은 첫째, 상충된 목적을 가지는 에이전트 사이의 자율적인 조정이 가능하다는 것이다. 둘째, 인터넷 상의 의사결정을 위해 다음의 기술이 적합하다. 이기종 간의 운동성을 가진 에이전트 기반 프로그래밍은 인터넷의 자료를 의사결정자에게 필요한 자료를 적시에 전달한다. 셋째, 에이전트 사이의 협동 개념이 자율적 조정의 활동에 알맞다.

다음은 객체 지향성이 주는 강점으로 그 중 캡슐화 기능과 객체 버저닝의 특징이 있다. 먼저, 캡슐화(Encapsulation)는 데이터와 행동을 하나의 패키지에 결합하여, 객체의 사용자로부터 데이터의 구현을 감추는 방식이다. 캡슐화의 특징은 기업 내부에서는 상대방 부서 내용에 대한 완전한 사전지식 없이 협동이 가능토록 한다. 각 부서를 대표하는 에이전트 자체 내에 모형을 보유하여 외부에서는 이 모형에 대해서 자세히 알 필요가 없기 때문이다. 둘째, 객체 버저닝(Object Versioning)이다. 객체는 현재 자기가 어떻게 보이는지에 대한 정보를 저장한다. 이것을 객체의 상태라고 한다. 또한 객체는 객체끼리 구별되는 식별의 요소를 갖는다. 각각의 객체는 항상 식별이 다르다. 이것이 각각에게 영향을 끼치게 되는 것이다. 이로 인해 형상관리(configuration management)가 가능하며 이는 시스템 유지 보수에 중요한 조건이다.

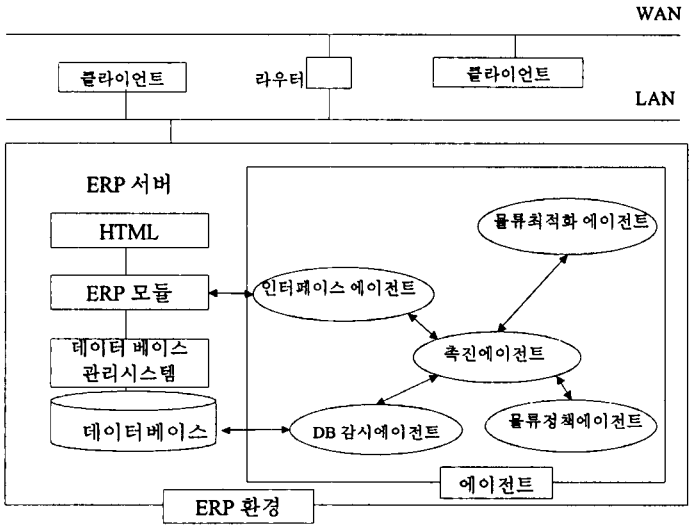
지능형 에이전트는 응용 영역에 따라 메타 지식 혹은 임무 수행을 위임받은 컴퓨터 프로그램 등으로 다양하게 정의되어 왔다 [2, 12]. 또한 기존의 지능형 에이전트에 대한 연구는 현재 네트워크 분야 등 공학 전반 분야에 응용되고 있다 [11, 15, 22].

본 논문에서는 지능형 에이전트를 동적으로 변화하는 전사적자원관리 환경에서 유지보수를 자율적으로 수행하기 위한 목적으로 제작된 독립적인 프로그램으로 정의하고자 한다. 또한 의사소통과 업무 수행에 대한 동적 지식을 표현하고 관리하기 위해 페트리 네트를 활용하고자 한다.

4. 지능형 전사적자원관리의 개념적 틀

4.1 개념적 틀

에이전트 기술을 도입하여 기존의 전사적자원관리에 자체 유지보수 기능을 추가한 새로운 개념적 틀은 다음 [그림-1]과 같다. 단, 전사적자원관리시스템 자체가 방대하여 우선 물류관리 모듈만 고려했다.



[그림-1] 물류시스템을 예제로 에이전트를 부가한 전사적자원관리의 개념 구조도

위의 [그림-1]에서 나타난 에이전트 기반의 전사적자원관리시스템은 에이전트-환경 동적 시스템(Dynamic systems on agent environment) 체제를 따르고 있다 [3, 6, 19]. 전사적자원관리와 다른 응용시스템과 통합하는 방법은 크게 API등과 같은 인터페이스를 통하여 파일 등을 공유하는 방식과 한 쪽에서 RPC(Remote Procedure Call)등을 활용하여 다른 시스템의 데이터베이스에 접근하여 자료를 참조하는 방식 등이 소개되고 있다 [10]. 본 논문에서는 양자를 모두 활용하여 DB감시 에이전트의 경우에는 필요한 경우 대상 전사적자원관리시스템의 데이터베이스에 접근하도록 하고 인터페이스 에이전트의 경우에는 각 전사적자원관리 모듈과 대화할 수 있게 하려고 한다. 전사적자원관리서버는 에이전트의 입장에서는 환경의 영역에 속한다.

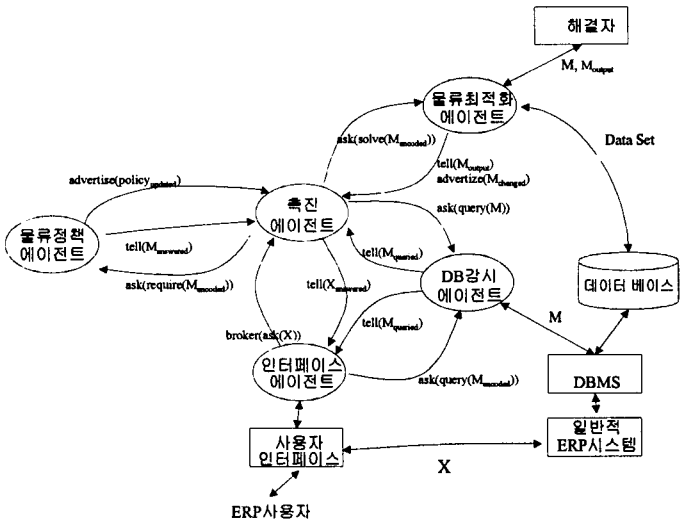
한편 에이전트는 데이터 뿐만 아니라 관련 모형과 방법론을 보유하고 있다. 이

때 유지보수의 대상이 되는 전사적자원관리 모듈과 데이터베이스는 각각 인터페이스 에이전트와 DB감시에이전트에 의하여 이상 정보에 대해 습득한다. 인터페이스 에이전트는 사용자로부터 문제를 입력 받아 문제의 형태가 업무 처리에 있는 경우 직업 전사적자원관리시스템과 상호작용을 하여 정보를 제공하기도 하고, 정보 요구 결과를 클라이언트에 제공하기도 한다. 이때 정보의 이상성 여부는 각 에이전트 내에 보유되어 있는 이상성 평가 지식에 의하여 결정된다.

본 전사적자원관리 구조도에 속하는 에이전트들에 대한 설명은 다음과 같다. DB감시에이전트는 데이터베이스에 저장된 데이터의 예외 상황과 오류들을 감시하는 역할을 하고 감시 결과를 촉진 에이전트에게 전달하는 역할을 한다. 촉진 에이전트는 인터페이스 에이전트로부터 요청된 문제를 접수하여 적절한 해답을 찾아 반응하기까지의 전과정을 스케줄링하며 다른 에이전트로부터 자신들이 해결할 수 있는 일에 대한 정보를 받아 반복적으로 작업을 요청하는 에이전트이다. 따라서 촉진 에이전트는 이상성 여부와 이상성이 감지된 부분에 대한 정보를 접수받아 이를 해결하기 위한 문제해결 에이전트들에게 전달하는 임무를 맡는다. 본 예제에서는 물류최적화 에이전트와 물류정책 에이전트가 문제해결을 위한 에이전트에 해당된다. 물류최적화 에이전트는 물류최적화 모형을 보유하고 있으며, 이와 함께 모형에 대해 사후 분석할 수 있는 지식을 가지고 있다. 촉진에이전트로부터 온 정보가 자신이 보유한 모형의 변화를 초래할 것인지를 분석해 보고 만약에 수정의 필요성이 있을 경우에는 스스로 자신의 모형을 변경한 다음에 최적화알고리즘 에이전트에 모형을 넘겨주어 최적해를 회신받는다. 이 같은 에이전트는 전사적자원관리의 모듈별로 즉 생산, 재무, 회계, 인사 등의 종류별로 존재하고 있으며, 따라서 각 에이전트는 자신이 어떤 형태의 모형을 인식할 수 있는지에 대한 정보를 촉진 에이전트에게 알린다. 촉진에이전트로부터 요청을 받으면 수송최적화 에이전트는 능동적인 활동을 통해 출력물을 획득하게 된다. 이때 계산 프로시저를 보유하고 있는 최적화 알고리즘 에이전트와의 상호작용을 통해 최적의 값을 도출하게 된다. 이렇게 도출된 최적의 값은 촉진에이전트로 회신하게 된다. 최적화알고리즘 에이전트는 모형을 접수받아 그 모형의 종류와 가정을 분석하고 이에 적합한 알고리즘을 선택하는 지식을 가지고 있으며, 필요한 경우에는 모형을 LINDO와 같은 해결자(Solver)와의 인터페이스가 가능한 형태 (예: MPS format)로 변환한다. 물류정책 에이전트는 물류관리 관련 통합 규칙들을 보유하고 있다. 정책 수정 요청이나 새로운 정책 입력을 가능하게 하며, 또한 최적화 에이전트로부터 유도된 최적해가 기존 정책에 위배되는지에 대한 점검을 하여, 그 결과를 촉진 에이전트로 이동시키는 역할을 한다.

4.2 에이전트 구성

전사적자원관리 유지보수를 위한 에이전트 시스템은 다음 [그림-2]와 같이 구성된다.



[그림-2] 에이전트 구성도

위의 구성도에 소개된 에이전트를 요약 설명하면 다음 [표-1]과 같다.

[표-1] 에이전트의 기능 설명

에이전트명	기능설명
촉진에이전트	각 에이전트들 간의 의사소통을 지원하는 에이전트
물류최적화 에이전트	물류 최적화 모델을 활용하여 원하는 결과를 획득하는 에이전트
물류정책 에이전트	모형 규칙을 형성하는 에이전트
DB감시 에이전트	DB의 오류 및 경영지원에 필요한 자료의 예외상황를 감시, 경고하는 에이전트
사용자 인터페이스 에이전트	모형 혹은 모형 수행 결과를 사용자가 볼 수 있게, 하이퍼텍스트 형태로 생성하고 결과물을 서버로 운송하는 사용자 인터페이스 지원을 위한 에이전트

물류최적화 에이전트

물류최적화 에이전트에서 모형 변화관리에 대한 지식은 페트리네트(Petri net)의 형태로 표현될 수 있다. 에이전트의 행태를 페트리네트로 표현하는 것은 다음과 같은 몇가지 이유에서다 [17].

첫째, 페트리네트는 시스템의 동적 특징을 표현할 수 있다. 따라서 에이전트가 다양한 이벤트의 발생에 따라서 자율적으로 반응하고 협동하는 성질을 자연스럽게 표현할 수 있다. 특히 전이 행렬(Transition matrix)을 통한 동적 행태 표현은 각 이벤트와 활동 내용의 연관성을 표현해 준다. 둘째, 페트리네트는 그래프 이론으로 증명된 유용한 분석 알고리즘을 가지고 있다. 예를 들어 안정성 분석(safety analysis)은 에이전트 시스템이 도중에 중지하는 현상을 사전에 확인할 수 있다. 또한 도달 가능성 분석(reachability analysis)은 에이전트의 기능 중 작동하지 않는 부분이 있는지에 대해서 사전에 분석할 수 있게 해준다. 세째, 페트리네트는 예측, 통제, 최적화 등을 가능하게 하여 전사적자원관리 시스템 관리에 유용한 정보를 제공해준다.

페트리네트의 기본형은 다음과 같이 정의된다 [18].

G = (P, T, I, O, Mo)

단, P 및 T는 P T , P T = 인 플레이스(place)와 트랜지션(transition)의 유한 집합

I는 임의의 트랜지션의 입력 플레이스를 지정하는 함수, I: P x T {0, 1}

O는 임의의 트랜지션의 출력 플레이스를 지정하는 함수, O: P x T {0, 1}

Mo는 Mo: P N을 의미하는 최초 마킹(initial marking)이며 이때 N은 비음수의 집합

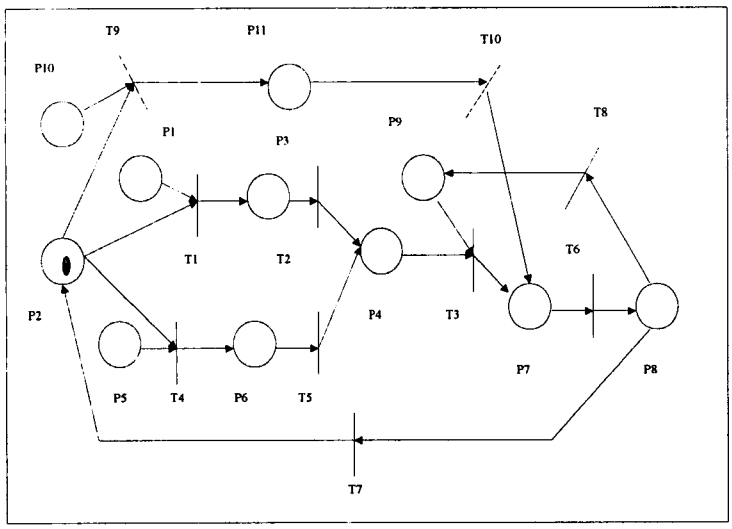
위의 정의에 따라 모든 에이전트들은 자율적인 조정을 위하여 다음 (1)과 같이 페트리네트의 형태로 동적 통제 지식을 가진다.

```
CLASS Agent
{
    IS-A: (none)
    INSTANCE-OF: (none)
    ATTRIBUTES
        Transition matrix:
                array [1, 2, 3,  , m] [1, 2, 3,  , n] of integer;
        Initial marking: array [1, 2, 3,  , m] of integer;
        Current marking: array [1, 2, 3,  , m] of integer;
} ---------------------- (1)
```

위의 정의에 따라 물류최적화 모형의 행동에 대하여 페트리네트로 모형화하면 다음 [그림-3]과 같이 표현된다. 또한 [그림-3]에서 소개되는 P와 T에 대한 설명은 다음 [표-2]와 같다.

[표-2] 물류최적화 에이전트의 플레이스와 트랜지션

P1	i의 변화	T1	행의 증감
P2	에이전트의 초기화 상태	T2	새 Si(공급지)값 검색
P3	행이 증감된 클래스	T3	모형 인스턴스 작성
P4	검색완료	T4	열의 증감
P5	j의 변화	T5	새 Dj(수요지)값 검색
P6	열이 증가된 모델 클래스	T6	알고리듬 통한 해 도출
P7	완성된 모형	T7	실행가능해 결정
P8	도출된 해	T8	모형 인스턴스 결정
P9	모형 재수정 요청	T9	DB값 검색 요청
P10	최적 값 요청	T10	모형 인스턴스 작성(정상)
P11	검색된 자료 집합		



[그림-3] 물류 최적화 에이전트의 페트리네트 표현

[그림-3]의 모형을 전이 행렬(Transition matrix)로 표현하면 다음 [그림-4]와 같다.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	-1	0	0	0	0	0	0	0	0	0
P2	-1	0	0	-1	0	0	1	0	-1	0
P3	1	-1	0	0	0	0	0	0	0	0
P4	0	1	-1	0	1	0	0	0	0	0
P5	0	0	0	-1	0	0	0	0	0	0
P6	0	0	1	1	-1	0	0	0	0	0
P7	0	0	0	0	0	-1	0	0	0	1
P8	0	0	0	0	0	1	-1	-1	0	0
P9	0	0	-1	0	0	0	0	1	0	0
P10	0	0	0	0	0	0	0	0	-1	0
P11	0	0	0	0	0	0	0	0	1	-1

[그림-4] 물류최적화 에이전트의 전이행렬

한편 물류최적화 에이전트를 객체로 표현하면 다음 (2)와 같다.

```
CLASS 물류최적화_Agent
{
    IS-A: (Agent)
    INSTANCE-OF: (none)
    ATTRIBUTES
        Logistics_model {
                goal: minimize;
                objective_function: Minimize total cost;
                decision_variable {
                        name: string;
                        indices: string;
                        non-negativity: [yes | no];
                        domain: [Real | Integer | Binary];
                        };
                CONSTRAINT[1]: Supply limit;
                CONSTRAINT[2]: Demand requirement;
        }
    OPERATIONS
        Interpret(input: M_encoded; output: ModelTemplate);
        Instantiate_Model(input: ModelTemplate, DataSet;
```

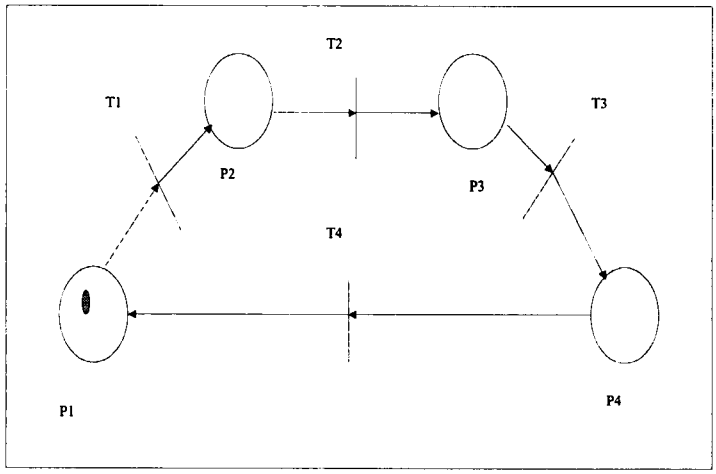
output: SourceModel);
MPS_Formatter(input: SourceModel; output: MPS_format_file);
Update_Model(input: SourceModel, UpdateInformaion;
output: UpdatedModel);
Ask(solve($M_{MPS\_format\_file}$));
Tell(M_{output});
Advertise($M_{changed}$);
} ----------(2)

물류최적화 에이전트는 전이행렬에 따라 각 행과 열이 증가하게 됨에 따라 작업이 진행되게 된다. ATTRIBUTE 내의 Logistics_model은 본 에이전트가 내재하고 있는 물류 비용 최적화를 위한 수리적 모형이다. 먼저 촉진 에이전트로부터 문제 M이 최적화 모형에 맞게 변형된 정보인 $M_{encoded}$ 를 받으면 이를 해결하고 또한 본 에이전트가 이해할 수 있는 형태의 모형 틀(Model Template)을 Interpret이라는 오퍼레이션을 통해 작성한다. 그런데 모형 틀에는 모형 구조 및 요소들에 대한 정보만 들어있으며 실제 모형 구축은 Instantiate_Model 오퍼레이션에 의해 수행된다. Instantiate_Model은 데이타베이스로부터 해당 모형에 들어갈 데이타들을 입력받아 모형 틀을 모형 인스턴스로 제작하는 역할을 한다. 또한 LINDO, LINGO, MODLER 등과 같은 최적화 알고리듬을 소유하고 있는 특정 문제풀이자(solver)의 입력 형태를 생성하기 위해 MPS_Formatter가 사용된다. 이렇게 모형관리 및 실행을 하는 이유는 에이전트가 사용되는 환경에는 다중 문제풀이자가 있으며 따라서 모형과 문제풀이자를 독립시키기 위함이다. 생성된 MPS형태의 모형 화일은 ask(solve($M_{MPS\_format\_file}$))를 통해 문제풀이자에게 보내어지고 문제풀이자는 이를 실행하여 최적해 결과값을 다시 리턴해 준다. 입수된 최적해 출력화일은 다시 촉진에이전트에게 tell(M_{output}) 오퍼레이션을 활용하여 보내져서 촉진에이전트의 유지보수를 위한 의사결정에 참고되도록 한다.

뿐만 아니라 물류최적화 에이전트는 자신이 보유하고 있는 모형이 과연 유의한지를 점검하는 자율적 기능을 가진다. 여기서 주요하게 이루어지는 일은 새로운 공급지와 수요지를 검색한다. Update_model 오퍼레이션은 데이터베이스의 내용에 의해 모형의 구조에 변화가 생기는 여부를 파악하고 필요한 경우 모델을 재작성 하고, 그 사실을 Advertise($M_{changed}$)을 통하여 촉진에이전트에게 전하게 된다. 이것은 모델의 변화 내용을 각 에이전트가 알 수 있도록 하는 것이다.

촉진 에이전트

촉진에이전트의 행태를 페트리네트로 표시하면 다음 [그림-5]와 같다.



[그림-5] 촉진에이전트의 페트리네트 표현

또한 관련된 P와 T에 대한 설명은 다음 [표-3]과 같다.

[표-3] 촉진에이전트의 플레이스와 트랜지션

P1	초기화 상태	T1	변화 검색
P2	변화 요청, 완료 내용	T2	모형이름, 모형사용위치 기록
P3	목적에이전트를 향한 메시지	T3	목적에이전트에 맞게 형식 재구성
P4	목적파일에 맞게 재구성된 메시지	T4	메시지 전달

촉진 에이전트(facilitator)를 객체로 표현하면 다음 (2)과 같다.

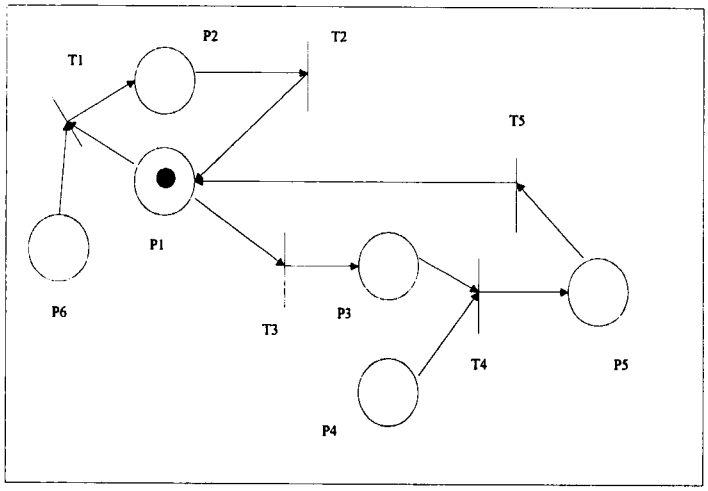
```
CLASS Facilitator
{
    IS-A: (Agent)
    INSTANCE-OF: (none)
```

```
    ATTRIBUTES
        name: string
        location: string
        description: string
    OPERATIONS
        search_linked_agents (output: linked_agents);
        modify_linked_agents ( input: linked_agents;
                                    output: modified_agents);
        reformat(input: M,  mode; output: M_encoded);
        interpret(X, M);
        tell(X_answered);
        ask(query(M_encoded));
        ask(require(M_encoded));
        ask(solve(M_encoded);
} ------------------(2)
```

촉진 에이전트는 각 에이전트의 변화 내용이나 문제 풀이 내용을 다른 에이전트에게 알려주는 작업을 한다. 각 에이전트의 변화에 따른 타 에이전트의 변경 필요성은 Search_linked_agents에 의해 이루어지며 먼저 각 에이전트의 변화 내용을 받아온다. 또는 전사적자원관리사용자로부터 사용자 인터페이스를 통해 요청이 들어올 수 있는데, 이때 문제의 입수는 인터페이스 에이전트의 broker(ask(X))에 의해 이루어진다. 이때 촉진 에이전트는 Interpret 오퍼레이션을 구동하여 문제 형태로 변환하게 된다. Reformat은 인지된 문제 M을 해결하기 위해 M을 해당 에이전트가 인식할 수 있는 모드로 변환하는 일을 하는데, 본 시스템에서는 SQL형태, 최적화 모형 형태, 그리고 추론 형태의 세 가지의 모드가 있다. 변환된 문제 ($M_{encoded}$)는 ask 오퍼레이션에 의해 각 해당 에이전트에 보내어진다. 결국 해결된 문제 혹은 기타 사용자에게 알려야 하는 정보는 tell($X_{answered}$) 에 의해 인터페이스 에이전트에게 알려지게 된다.

인터페이스 에이전트

인터페이스 에이전트의 동적 모형은 다음 [그림-6]과 같다.



[그림-6] 인터페이스 에이전트의 페트리네트 표현

또한 관련된 P와 T에 대한 설명은 다음 [표-4]와 같다.

[표-4] 인터페이스 에이전트의 플레이스와 트랜지션

P1	초기화 상태	T1	사용자정보 변화 검색
P2	새 사용자 정보	T2	사용자변화 정보 DB검색에이전트로 전달
P3	사용자 프로화일	T3	사용자 정보 생성
P4	사용자 필요 발생	T4	사용자 필요 검색
P5	변화된 사용자 필요 메시지	T5	사용자 필요 메시지를 촉진에이전트로 전달
P6	사용자 입력		

인터페이스 에이전트에 대한 표현은 다음 (3) 과 같다.

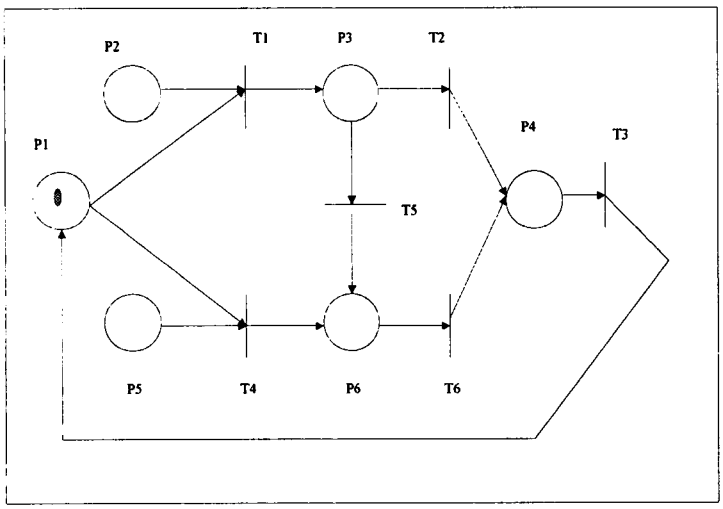
```
CLASS Interface_Agent
{
    IS-A: (Agent)
    INSTANCE-OF: (none)
    ATTRIBUTES
        name: string;
        location: string;
```

```
        description: string;
    OPERATIONS
        broker(ask(X));
        ask(query(M));
        generate_interface_from_facilitator(input: X_answered, mode;
                                    output: generated_format);
        generate_interface_from_db(input: M_queried, mode;
                                    output: generated_format);
} --------------------(3)
```

인터페이스 에이전트는 ask(query(M)) 오퍼레이션을 통해 DB감시 에이전트에게 사용자 정보를 요구하고 생성하도록 하기도 하며, 또한 사용자의 정보 필요를 broker(ask(X)) 오퍼레이션을 통해 촉진에이전트로 하여금 각 에이전트에게 알리도록 하는 활동을 하기도 한다. 또한 촉진에이전트 혹은 DB감시에이전트로부터 회신되는 결과 내용은 다시 사용자가 사용하는 사용자 인터페이스에 맞게 재구성되도록 각각 generate_interface_from_facilitator 혹은 generate_interface_from_db 오퍼레이션을 구동한다.

물류정책 에이전트

물류정책 에이전트는 물류 정책을 검색하고, 유효결정에 따라 변경하거나 수정하는 일을 한다. 다음 [그림-7]은 물류 정책 에이전트의 페트리네트 표현이다.



[그림-7] 물류정책 에이전트의 페트리네트 표현

관련된 P와 T에 대한 설명은 다음 [표-5]와 같다.

[표-5] 물류정책 에이전트의 플레이스와 트랜지션

P1	에이전트 초기화 상태	T1	정책 검색
P2	기존의 정책 변화	T2	정책 유효결정
P3	검색완료	T3	정책 변경 및 수정
P4	정책 수정 요청	T4	정책 증가
P5	새 정책 입력	T5	정책 무효결정
P6	새 정책 작성 요청	T6	새 정책 검색

물류정책 에이전트를 객체로 표현하면 (4)와 같다.

```
CLASS Outbound_policy_agent
{
    IS-A: (Agent)
    INSTANCE-OF: (none)
    ATTRIBUTES
        name: string;
        location: string;
        description: string;
        outbound_policy_rules: *file
    OPERATIONS
        add_policy(added_policy);
        validate_policy(input: validated_policy; output: results);
        query_policy(input: policy; output: results);
        update_policy(input: policy; output: updated_policy);
        get(M_encoded);
        tell(M_answered);
        advertise(updated_policy);
        reasoning(input: facts, reasoning_mode; output: results);
} ---------------------------------- (4)
```

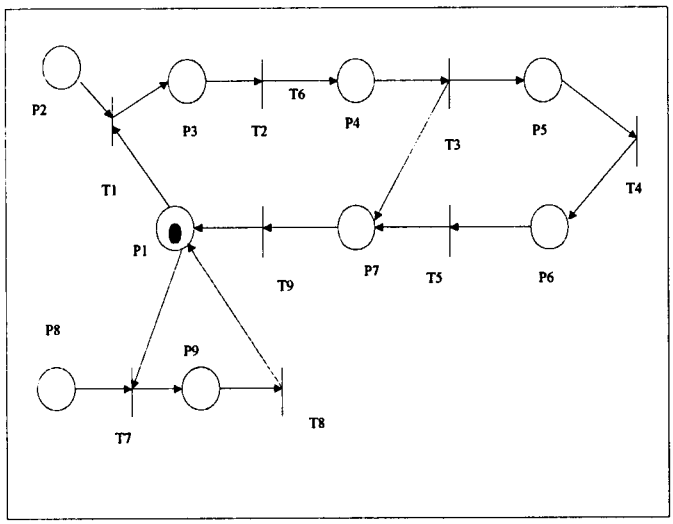
촉진에이전트로부터 받은 $M_{encoded}$ 는 물류 정책에 대한 규칙에 대한 요구사항으로서 다른 에이전트로부터 획득한 사실(facts)의 집합체이다.

outbound_policy_rules이라고 하는 자체 규칙을 내포하고 있는 물류정책 에이전트는 reasoning오퍼레이션을 통해 추론 결과를 확보해 나간다. 이때 reasoning_mode는 전방위 추론법 혹은 후방위 추론법 중 어느 하나를 지정하도록 한다. $M_{encoded}$ 는 또한 메타 지식, 즉 물류정책 규칙의 관리에 대한 요구 사항일 수도 있다. 이의 종류로는 정책의 추가, 유효성 분석, 검색 및 수정 등이 있다. 물류정책에이전트는 이러한 촉진 에이전트로부터 오는 요구 사항에 따라 각각 add_policy, validate_policy, query_policy, update_policy 오퍼레이션을 구동시킨다.

이러한 오퍼레이션은 또한 물류정책 에이전트 자체의 판단에 따라 자율적으로 구동될 수도 있다. 이를 위해서 우선 정기적으로 query_policy 를 통해 기존의 정책을 검색한다. 기존의 정책은 validate_policy를 거쳐 유효성을 판단된 후 더 이상 유효하지 않다고 판단되면 update_policy 에 수정 및 변경 요청을 한다. update_policy에서는 요청된 정책을 수정 변경하여 초기값을 돌린다. 수정 내용은 advertise 오퍼레이션에 의해 촉진 에이전트를 거쳐 필요한 에이전트로 전달되게 된다.

DB감시 에이전트

DB감시 에이전트를 기술하는 페트리네트 표현은 아래 [그림-8]과 같다.



[그림-8] DB 감시 에이전트의 페트리네트 표현

또한 관련된 P와 T에 대한 설명은 다음 [표-6]과 같다.

[표-6] DB감시 에이전트의 플레이스와 트랜지션

P1	초기값	T1	정보 검색
P2	정보 검색 요구	T2	SQL문 작성
P3	정보 검색 완료	T3	예외상황검색
P4	검색된 테이블	T4	예외상황 분석 및 보고
P5	발견된 예외상황	T5	예외상황 긴급 수정
P6	분석, 보고된 예외상황	T6	정상 메세지 전달(예외상황 없음)
P7	정상	T7	정보 입력 및 수정
P8	사용자 정보 입력 및 변화 요구	T8	정보 저장 메시지 전달
P9	새 정보 입력 테이블		

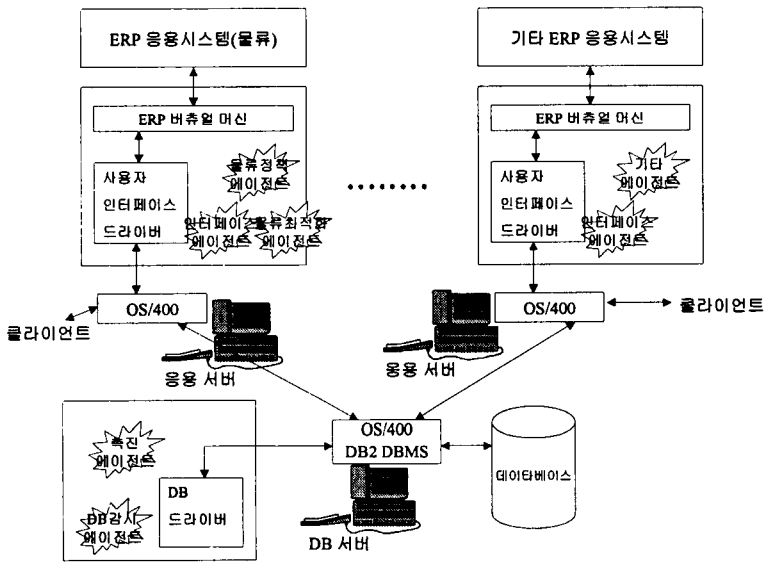
DB감시 에이전트는 요구 내용을 토대로 generate_SQL에서 SQL문을 작성한다. 또한 이 에이전트는 DBMS에 접근하여 analyze_exceptions 이 DB의 예외상황을 분석하도록 한다. DB감시 에이전트에 대한 표현은 (5)에 나타나 있다.

```
CLASS DB_Montoring_Agent
{
    IS_A: (Agent)
    INSTANCE_OF: (none)
    ATTRIBUTES
        name: string;
        description: string;
        model_file_structure: data;
    CONSTRAINTS
        tell(M_queried);
        generate_SQL(M);
        analyze_exceptions(M);
} ---------------------------------- (5)
```

5. 구현 방안

현재 본 논문의 에이전트 아키텍쳐를 반영한 프로토타입 구현이 진행 중이며 AS/400 상에서 구동되는 전사적자원관리 시스템을 채택하였다. 전사적자원관리 플랫폼으로는 AS/400e 170시리즈를 채택하였으며 데이타베이스는 DB2를 사용하고 에이전트 프로그램은 자바 스크립트를 활용할 예정이다. 또한 인터페이스 설계 및 자바 프로그래밍 지원을 위해 VisualAge for Jave 2.0과 DB2 관리를 위하여 Net.Data를 사용할 것이다. 구현 결과는 다음 [그림-9]와 같이 될 예정이다.

한 조직 내에 복수개의 응용 서버가 존재할 수 있으며 데이타베이스 서버를 보유하도록 한다. 에이전트들은 응용서버와 데이타베이스 서버에 분산된다. 우선 특정 응용 영역에 관련된 에이전트 (예: 물류최적화 에이전트, 물류정책 에이전트, 인터페이스 에이전트)는 응용 서버에, 전체적인 관리와 데이타베이스에 관련된 에이전트 (예: 촉진 에이전트, 데이타 감시 에이전트)들은 데이타베이스 서버에 연결된다. 상세한 프로토타입 구현 방법은 본 논문의 범위를 벗어나므로 생략하였다.



[그림-9] 시스템 구현 결과

6. 결 론

본 논문은 에이전트 기술을 사용하여 전사적자원관리시스템에게 자가진단 기능을 부여함으로 지능형 전사적자원관리시스템 구축이 가능함을 보였다. 각 에이전트는 전사적자원관리시스템과 상호작용을 하여 비용증가, 이익감소, 반응시간의 연장 등의 경영환경에 미치는 부정적 요인을 검색 및 진단한다. 이것은 전사적자원관리시스템과 동시에 고려될 유지 및 보수 비용을 크게 감소할 수 있을 것이다. 특히 에이전트의 진단 기능은 비용 뿐 만 아니라 컨설팅 기술력이나 인력의 수급 부족이 주는 문제점도 감소시킬 수 있다. 또한 에이전트의 동적 작용성을 모형화하기 위해 페트리네트를 사용하였다. 이는 설계된 에이전트 아키텍쳐가 어떤 특성을 가질지를 모의 실험하고 통제하는 것을 가능하게 한다.

전사적자원관리시스템 자체적으로 외부환경의 변화에 적응하고 컨설팅 지능을 가짐으로 능동적인 전사적자원관리를 구축하는 것은 현 전사적자원관리의 과제이다. 앞으로 프로세스 내에 예외 상황이나 환경적 변수가 다양해지기 때문이다. 이형질성의 시스템에도 접근이 가능한 에이전트의 이동성은 다양한 환경변수와 환경의 변화에도 대처 가능하다. 또한 전사적자원관리의 유지보수와 관련하여 시스템을 자체 점검하는 연구도 이루어져야 한다. 이것은 촉진에이전트와 같은 중앙통제 및 감시역할을 하는 에이전트의 활동이 시스템 내에서 자유로이 의사소통 하는 수단으로 사용되어질 방법에 관한 연구이다.

참고문헌

[1] Appleton, L., How to Survive ERP, *Datamation*, Vol. 43, No.3, 1997, pp. 50.

[2] Bates, J., The role of emotion in believable agents, *Communications of the ACM*, Vol. 37, No. 1, 1994, pp. 122-125.

[3] Brian, Y. and B. Randall, Spatial Learning for Navigation in Dynamic Environments, Working Paper, *Case Western Reserve University*, 1995.

[4] Celarier, M. and Harris, R., Plucking More Profit From Production, *CFO*, Vol.15, No.1, 1999, pp. 85.

[4] Dilger, K. A., Execution and the Enterprise, *Manufacturing Systems*, 17(5), 1999, p. 34-38.

[5] Evans, M., Bragg, S. and T. Klevers, *Ovum Evaluates ERP for Manufacturers*, Ovum Ltd., 1997.

[6] Hyacinth S. N. and T.N. Divine, Agents of change in future communication systems, *British Telecommunications LAB*, 1998.

[8] Kappelhoff, R., Integration of ERP to the final control elements, *ISA Transactions*, 36(4), 1997, pp. 229-38.

[9] Kazanskii, D.L.,The ERP approach for enterprise simulation, *Seti i Sistemy Svyazi*, 1998, pp. 52-59.

[10] Lawrence, G. S., Integrating APS and ERP is Getting Easier, *Automotive Manufacturing & Production*, 111(5), 1999, p. 50-52.

[11] Levy, A.Y., Sagiv, Y., and D. Srivastava, Towards efficient information gathering agents, In O. Etzioni, editor, *Software Agents-Papers from the 1994 Symposium* (Technical Report SS-94-03), 1994, pp. 64-70, AAAI Press.

[12] Maes, P. Agents That Reduce Work and Information Overload, *Communications of ACM*, Vol. 37, No. 7, 1994, pp. 31-40.

[13] Marion. L., Best of breed reborn, *Datamation*, Vol. 44, No. 4, 1998, http://www.datamation.com/PlugIn/issues/1998/april/04erp.html.

[14] Marlene, P., How Midsize Companies Are Buying ERP, *Journal of accountancy*, Vol. 188, No. 3, 1999, pp. 41.

[15] McGregor, S.L. Prescient Agents, In D. Coleman, editor, *Proceedings*

of Groupware-92, 1992, pp. 228-230.

[16] Morris, J.S. and Morris, L.J., Problems in CIM implementation: a case study of nine CIM firms, *Computers & Industrial Engineering*, 27, 1994, pp. 147-50.

[17] Mu, D.J. A Petri Net Synthesis Theory for Modeling Flexible Manufacturing Systems, *IEEE Proceedings*, 1997, pp. 169-183.

[18] Peterson, J.L., *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Inc., N.J., 1981.

[19] Randall D. B., A Dynamical Systems Perspective on Agent-environment Interaction, *Artificial Intelligence* , Vol. 72, No. 2, 1995, pp 173-215.

[20] Seki, Y., *Information unification between enterprise resource planning system and production control system,* Yokogawa Technical Report, no. 25, 1998.

[21] Sidney, H. Jr, How fast is your business? *Manufacturing Systems*, 17(8), 1999, pp. 1-9.

[22] Steeb, R., S., Cammarata, F.A., Hayes-Roth, P., Thorndyke, W., and R.B. Wesson, Distributed Intelligence For Air Fleet Control, In Bond, A.H. and Gasser, L., editors, *Readings in Distributed Artificial Intelligence*, 1988, pp. 90-101.

[23] Strothman, B.L., What's next after ERP? Prepare for NBO!.(Networked Business Objects), *InTech*, 42(1), 1995, pp. 19-20.

[24] Tim, S., Consulting's New Era, *Industry Week*, 248(15), 1999, p. 24-31.

[25] Zigmont, J., Configuring Success, *Datamation*, Vol. 44, No. 12, 1998, http://www.datamation.com/PlugIn/issues/1998/october/10confi.html.

Applying Intelligent Agent and Petri Net Modeling Technology to ERP Maintenance

O Byung Kwon , Jae Jun Lee

Even though there is no doubt that ERP(Enterprise Resource Planning) system is a prevailing solution for integrating corporate information, many companies still hesitate adopting ERP system because of a great deal of cost including maintenance cost. In special, unless consulting knowledge that is infused into process reengineering phase or adequately embedded in customized ERP system is upgraded on time, then we cannot guarantee the optimality of system performance. Hence, this paper aims to construct an ERP system that adapts itself to environmental changes that are issued by database and users. To do so, we adopt intelligent agent technology and Petri net theory. The agents autonomously cooperate each other to investigate databases and to find any exceptional changes and analyze how the changes will affect ERP performance. The dynamics of the agents are represented as Petri nets. The newly proposed ERP system is to make corresponding BPR processes maintain optimality. To show the feasibility of the proposed ERP maintenance system, logistics component is described as an illustrative example.

◆ 저자소개 ◆

권오병 (O Byung Kwon)

현재 한동대학교 경영경제학부 조교수로 재직 중이다. 서울대 경영대학(1988)을 졸업하고, 한국과학기술원 경영과학과 공학석사(1990), 공학박사(1995)를 취득하였다.
1995 ~ 1996년 중국 연변과학기술대학 경영정보과 교수를 역임하였다. 현재 경실련 경제정의 연구소 연구위원이기도 하다. 주요 관심분야로는 의사결정지원시스템, 전자상거래, 에이전트 기술 등이 있다.

이재준 (Jae Jun Lee)

현재 포항공과대학교 정보통신대학원 석사과정에 재학중이다. 한동대학교 경영경제학부 2000년 졸업하였고, 관심분야로는 전자상거래, 지능형 에이전트, ERP 등이 있다.