다중변수 기반 에이전트 중재 전자상거래 협상모델 및 프레임워크 설계

(Design of Multi-Attribute Agent-Mediated Electronic Commerce Negotiation Model and its Framework)

정 목 동 †

(Mokdong Chung)

요 약 오늘날의 1세대 구매 에이전트는 상품의 전체 특징 변수(attribute)에 대해서보다는 주로 판매자들이 제시한 가격만 비교해서 구매행위를 대행해 주고 있으며, 간혹 가격 이외의 변수에 대한 비교를 해주는 에이전트의 경우에도 협상(negotiation) 과정에서 전체 변수를 적절하게 고려해주는 협상 모델은 찾아보기 힘들다. 따라서 전자 상거래의 협상 모델(negotiation model)을 가격 변수뿐만 아니라 상품의 전체 변수로 확장 시켜 주는 것이 절실히 요구되고 있다. 본 논문에서는 유틸리티(utility) 이론과 간결한 휴리스틱스(simple heuristics)에 바탕을 두어서 가격, 상품의 특성, 보장 기간, 서비스 정책 등에 대해서 협상을 벌이는 에이전트 중재에 의한 전자 상거래 협상 모델을 제시하고, 이를 구현하는 프레임워크 Pmart의 원시 시스템에 관해서 논한다. Pmart는 객체 지향 기술에 바탕은 둔 소프트웨어 재사용과 확장성을 제공하고 있으며, 네트워크 투명성과 플랫폼 독립성을 위하여 Java와 CORBA를 이용하여 구현하고 있다.

Abstract Today's first generation shopping agent is limited to comparing merchant offerings usually on price instead of their full range of attributes. Even in the full range comparison, there is not a good model which considers the overall features in the negotiation process. Therefore, the negotiation model needs to be extended to include negotiations over the more attributes. In this paper, we propose a negotiation model in the agent-mediated electronic commerce to negotiate over prices, product features, warranties, and service policies based on utility theory and simple heuristics. We will describe a prototype agent-mediated electronic commerce framework called Pmart. This framework provides the software reuse and the extensibility based on the object-oriented technology. It is implemented on Windows-based platforms using Java and CORBA for the network transparency and platform independence.

1. 서 론

Forrester Research 에서 평가하기로는 인터넷을 통한 상품 판매는 1996년에 약 6억불, 1997년에 20억불을 넘어섰고, 2001년에는 170억불에 달할 것으로 전망하고 있다[1]. 이처럼 전자 상거래는 전통적인 경제 행위의 지축을 흔들 정도로 우리의 일상 생활에 깊숙이 파고들고 있는 실정이다.

그렇지만 오늘날의 1세대 전자 상거래에 해당하는 OnSale[2], UCE[3], Amazon[4] 등 많은 웹사이트에서 상품을 사고 팔 수는 있지만 이들 중 어떤 곳에서도 자율적인 에이전트가 사용자를 대신해서 최적의 협상을 해주는 곳은 드물다. 더구나 오늘날의 1세대 구매 에이전트는 상품의 전체특징 변수(attribute)에 대해서보다는 주로 판매자들이 제시한 가격만 비교해서 구매행위를 대행해 주고 있으며, 간혹 가격 이외의 변수에 대한 비교를 해주는 에이전트의 경우에도 협상(negotiation) 과정에서 전체 변수를 적절하게 고려해주는 협상 모델은 찾아보기 힘들다. 따라서 전자 상거래의 협상 모델(negotiation model)을 가격 변수뿐만 아니라 상품의 전체 변수로 확장 시켜 주는 것이 절실히 요구되고 있다. 즉, 차세대 전

· 이 논문은 1999년도 부경대학교 연구년교수 지원에 의하여 연구되었음.
† 정 회 원 : 부경대학교 컴퓨터공학과 교수
mdchung@pknu.ac.kr
논문접수 : 2000년 12월 7일
심사완료 : 2001년 8월 20일

자 상거래 협상 모델은 지능형 소프트웨어 에이전트에 바탕을 두어야하며, 상품의 비교도 가격 변수뿐만 아니라 상품의 전체 변수로 확장되어야 한다. 이를 위해서는 소프트웨어 에이전트 기술이 해결해야할 과제가 많이 있는데 예를 들면, 불분명한 상품 내용, 개인적인 선호도 파악, 복잡한 목표 설정, 거래 환경의 동적인 변경 등의 문제이다. 차세대 전자 상거래 시스템에서 XML 같은 표준 문서 등이 채택되면 상품과 서비스, 구매자와 판매자의 프로파일, 부가 서비스, 보안 지불, 그리고 인터넷 EDI 등의 표준화가 이루어 질 수 있을 것이다. 여기서 에이전트는 소비자들의 만족도를 증가시키고 유연한 B2B(Business-to-Business) 거래를 보장하며 결과적으로 전 단계에서 거래 비용을 줄여주는 역할을 수행하게 된다. 이와 같이 전자상거래와 지능형 에이전트의 결합은 상당한 경제적 이익을 안겨줄 것이고, 특히 에이전트 중재에 의한 전자 상거래(Agent-mediated Electronic Commerce)는 구매자와 판매자 사이의 협상을 보다 용이하게 해줄 수 있다.

따라서 본 논문에서는 지능형 멀티 에이전트에 바탕을 두고 가격, 상품의 특성, 보장 기간, 서비스 정책 등에 대해서 협상을 벌이는 에이전트 중재에 의한 전자 상거래 프레임워크 *Pmart*(***Pukyong-mart***)를 제시한다. 이를 위한 협상 모델은 영역 지식과 일반 지식을 동시에 사용할 수 있는데, 영역 지식은 MAUT(Multi-Attribute Utility Theory)[5]에 바탕을 두고 있고 일반 지식은 기존의 구매기록(purchase history)과 간결한 휴리스틱스(simple heuristics)[6]에 바탕을 두고 있다. 제시하는 프레임워크는 객체 지향 기법과 소프웨어 컴퍼넌트(software component)에 바탕을 둔 소프트웨어 재사용(software reuse) 기능을 제공하고 있다.

논문의 구성은 1장 서론에 이어서 2장 관련연구, 3장 Pmart의 협상 모델, 4장 Pmart의 설계 및 구현, 그리고 5장에서 결론과 향후 연구에 대해서 논한다.

2. 관련 연구

MIT Media Lab은 Kasbah[1,7]라고 불리는 에이전트 기반 마켓(market)을 개발했는데, 여기서 사용자들은 상품의 거래 행위를 본인의 취향에 따라 협상하고 거래하는 에이전트에게 위임할 수 있게 되었다. 구매 에이전트와 판매 에이전트가 일치가 된 다음, 구매 에이전트가 취하는 행위는 판매자에게 입찰(bid)을 보내는 것이다. 그러면 판매 에이전트는 yes나 no로 답을 하게된다. Kasbah는 구매자에게 3가지 협상 방법을 제공해 주는데, anxious, cool-headed, frugal 등이 있다. 이 협상은 멀티-에이전트, 상호 거래 등의 형태를 띠고 있지만 단순한 종간 함수를 사용하고 있으며, 또한 범용 마켓 인프라 구조를 목표로 하지 않았고, 분산 환경과 대화 모델을 제공하지 못한다.

MIT의 Tete-a-Tete[8,9]는 소비자가 복잡한 물건을 구매하는 경우에 가격 이외의 다른 조건을 고려 할 수 있도록 해주는 에이전트 중재에 의한 구매 시스템이다. 대부분의 온라인 협상 시스템이 가격에 대해서만 협상하는 것과는 달리 Tete-a-Tete 에이전트들은 거래의 여러 항목에 대해서 협상을 벌인다. 이 시스템은 의사 결정 보조 모듈과 함께 통합적인 협상 모델을 사용하여 판매자와 구매자 모두에게 개선된 온라인 상거래 환경을 제공하고 있다. 의사 결정 보조 모듈은 MAUT와 분산 제약 만족(distributed constraint satisfaction)에 바탕을 둔 제품 최적화(product customization)를 사용하고 있다. 그러나 이 시스템은 불확실한 정보를 가지고 의사 결정을 할 때에는 문제점을 안고 있다.

AuctionBot[10]는 University of Michigan에서 개발한 범용 인터넷 경매 서버이다. 사용자들은 경매 형태와 매개 변수를 선택함으로써 상품을 거래할 새로운 경매를 구성할 수 있다. 구매자와 판매자는 새로 구성된 경매에서 다각적인 분산 프로토콜(multilateral distributive negotiation protocols)에 따라서 입찰을 할 수 있다. 그러나 협상에서 전체 변수에 대해 협상해주지 못한다.

Magnet[11]은 University of Minnesota에 의해서 개발되었는데, 지향하는 목표는 멀티에이전트 마켓 영역에서 계획, 계약, 스케쥴링, 수행 등을 통합하는 시멘틱 모델을 개발하는 것이다. 역시 전체 변수에 대해 협상해주지 못한다.

SICS MarketSpace[12]는 Swedish Institute of Computer Science 에서 Java를 이용하여 개발되었다. 이 시스템은 사용자가 자신의 관심사를 기술하는 것을 도와주는 개인적인 조수로서 구성되어 있다. 두 에이전트가 웹 시장을 형성하고 있고, 사용자와 서비스 에이전트가 관심사를 등록하고, 일치되는 에이전트를 찾을 수 있도록 해주는 디렉토리 서비스를 제공하고 있다. 역시 전체 변수에 대해 협상해주지 못한다.

스페인의 CSIC에서 개발한 Fish Market[13]는 downward bidding protocol로 구현한 생선 판매 경매 시스템이다. 역시 전체 변수에 대해 협상해주지 못한다.

한편, 웹나라[14], 샵바인더[15], daum 경매[16] 등 국내의 전자상거래 환경의 협상에서도 아직은 가격 위주의 비교만 하는 1세대 전자상거래를 하고 있는 실정이다.

3. Pmart 협상 모델

3.1 Pmart 협상 모델

상품의 전체 변수에 대해서 협상하는 방법으로 세 가지를 생각 할 수 있는데, 첫 번째 접근 방법은 단지 MAUT 에만 의존하는 방법이다. 두 번째 방법은 MAUT를 먼저 사용하고 몇 번 실패 후에는, 가령 *numS*번, 기존의 구매기록과 간결한 휴리스틱스에 의존하는 방법이다. 구매기록은 한 상품에 대한 지금까지의 많은 사람들의 성공적인 거래 기록을 담고 있는 데이타베이스이다. 따라서 이것은 한 상품의 구매에 관한 일반적인 지식으로 볼 수 있다. 마지막 방법은 MAUT를 사용하지 않고 바로 구매 기록과 휴리스틱스로 구성된 일반 지식만 사용하는 방법이다.

본 협상 모델에서는 세 가지 방법을 모두 쓸 수 있지만 이 중 두 번째 방법에 초점을 맞추는 이유는 다음과 같다. 첫째, 특수 지식에서 일반 지식으로 추론해 나가는 것은 일반적인 추론 기법에서 보았을 때도 타당성이 있다. 둘째, MAUT 기반 협상을 몇 번 사용 한 다음에도 계속 실패하면 기존의 구매기록과 휴리스틱스에 바탕을 두고 있는 일반 지식 기반 협상에 의존하는 것이 보다 우수해 보인다.

그러므로 본 연구에서는 2 단계 의사결정 과정을 제시하고 있다. 즉, 첫째 단계에서는 MAUT에 바탕을 둔 영역 지식을 사용하고, 두 번째 의사 결정 과정에서는 구매기록과 휴리스틱스에 바탕을 둔 일반 지식을 사용하고 있다. 이 모델에서는 과거의 유사 상황을 관찰하기 위해서 일종의 사례 기반 추론 기법을 사용하고 있다. 여기서 MAUT와 간결한 휴리스틱스에 대해서 살펴보자.

3.1.1 유틸리티 함수 선정 과정

MAUT(Multi-Attribute Utility Theory)는 다중 변수에 대한 의사 결정 문제(decision problem)에서 유틸리티(utility)를 통한 정량적인 의사 결정 방법이다. 유틸리티 분석(utility analysis)은 의사결정자(decision maker)가 원하는 제비뽑기(lottery)의 결과(consequence)를 분석해주는 분야로써 의사 결정자는 이들 결과에 대한 개인의 선호도(preference)를 유틸리티 수(utility number)로 표현하고 있다. 결국 유틸리티는 0과 1사이의 상대적인 값으로서 $u(x^o)$와 $u(x^*)$를 각각 가장 선호하지 않는 결과 유틸리티와 가장 선호하는 결과 유틸리티라고 두면 $u(x^o)=0$, $u(x^*)=1$로 나타낼 수 있다. 결과에 대한 유틸리티 수의 대입은 기대 유틸리티 (expected utility)를 최대화 시켜주는 쪽으로 이루어져야한다. 즉, 기대 유틸리티의 최대화는 의사 결정자의 최적 행동의 기준이 된다. 유틸리티 함수를 선정하는 데 일률적인 방법이 있는 것은 아니지만 공통적으로 사용될 수 있는 과정은 대체로 다음과 같다[5].

선정 준비: 결과 Q를 실수 x에 사상시키는 평가 함수를 $\boldsymbol{X}$라고 하면, $x=\boldsymbol{X}(Q)$이다. x 값의 크기와 바람직한 정도와의 관계를 정할 수 있다. 즉, 의사 결정자가 결과 x_1과 결과 x_2 중에서 어떤 것을 선호하는지 확인해 볼 수 있다. 그림1은 두 개의 특정 변수(two-attribute) $\boldsymbol{Y}$와 $\boldsymbol{Z}$에 대한 결과 공간을 보여주고 있는데, 결과 $\boldsymbol{T}$와 결과 $\boldsymbol{S}$ 중에서 의사결정자가 선호하는 것을 질의를 통해서 확인해볼 수 있다. 결과 $\boldsymbol{Q}$는 $y=y_1$ 이고 $z=z_1$인 결과이고, 결과 $\boldsymbol{R}$은 $y=y_2$ 이고 $z=z_2$인 결과이다.

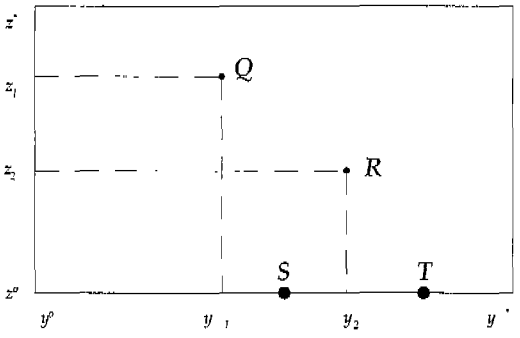

그림 1 두 특정 변수 결과 공간

독립성 확인: $\boldsymbol{Y}$와 $\boldsymbol{Z}$가 덧셈 독립(additive independence), 유틸리티 독립(utility independence)인지 확인해본다. $\boldsymbol{Y}$와 $\boldsymbol{Z}$가 덧셈 독립이라는 의미는 임의의 제비뽑기를 두 번 했을 때 상호 선호도 비교(paired preference comparison)는 이들의 한계 확률 분포(marginal probability distributions)에만 의존하고, 결합 확률 분포(joint probability distributions)와는 무관하다는 것을 의미한다. 즉 $\boldsymbol{Y}$와 $\boldsymbol{Z}$가 상호 독립이라는 것이다. $\boldsymbol{Y}$와 $\boldsymbol{Z}$가 유틸리티 독립이라는 의미는 z가 주어졌을 때 $\boldsymbol{Y}$에 대한 제비뽑기의 조건 선호(conditional preferences)가 z 값의 변화와는 무관하다는 것이다.

정성적인 성질 확인: 유틸리티 함수가 단조(monotonic) 함수인지 확인한다. x_k가 x_l 보다 크면 x_k는 항상 x_j 보다 좋은 것인지 확인하고, 다음으로 유틸리티 함수 u가 모험 회피(risk averse), 모험 중립(risk neutral), 모험 노출(risk prone) 중에서 어떤 것인지 결정한다. 우선 임의의 x와 h에 대해서 의사 결정자의 선호도를 확인하는데, 제비뽑기 $<x+h,\ x-h>$와 기대결과(expected consequence)

x 중에서 어떤 것을 좋아하는지 확인한다. 제비뽑기 $<x+h, x-h>$는 같은 확률로서 $x+h$와 $x-h$를 선택할 수 있다는 것을 의미한다. 이런 시험을 여러 다른 x와 h 값에 대해서 반복한 결과, 만약 의사 결정자가 주로 제비뽑기를 선호하면 모험 노출이라고 추정할 수 있고, 기대 결과 x를 선호하면 모험 회피라고 간주 할 수 있다. 제비뽑기와 기대 결과 사이에 특별한 선호가 없으면 모험 중립이라고 본다.

유틸리티 함수 결정: 간단한 예로 의사 결정자의 유틸리티 함수가 x에서 단조 증가이고, 감소적인 모험 회피라고 가정하면, 이러한 특징을 만족시키는 유틸리티 함수는 다음과 같다.

$u(x)=h+k(-e^{-ax}-be^{-cx})$, 여기서 a, b, c, k는 양의 상수이다.

일반적으로 유틸리티 함수 $u(x_1, x_2,..., x_n)$가 덧셈 독립, 유틸리티 독립이면 u는 다음과 같다.

$u(x_1, x_2,..., x_n)=k_1u_1(x_1)+k_2u_2(x_2)+...+k_nu_n(x_n)$, 여기서 모든 i 에 대해서 $u_i(x^o_i)=0$, $u_i(x^*_i)=1$. $u_i(x^o_i)$와 $u_i(x^*_i)$는 각각 가장 선호하지 않는 결과 유틸리티와 가장 선호하는 결과 유틸리티이다.

3.1.2 간결한 휴리스틱스의 이용

간결한 휴리스틱스(simple heuristics)는 1995년 독일, 미국, 영국에서 심리학, 수학, 컴퓨터과학, 경제학, 생물학 등 학제간의 연구를 위해서 설립된 ABC(Adaptive Behavior and Cognition) 연구 그룹의 주된 이론이다 [6]. 이 연구소에서는 제한적 추리(bounded rationality)와 불확실성 하에서 좋은 의사 결정에 관한 연구를 해오고 있다. 간결한 휴리스틱스(simple heuristics)를 이용하려는 이유는 상품의 특징 변수와 관련된 사용자들의 선호도를 개량적으로 정확히 예측한다는 것이 쉬운 일이 아니기 때문이다. 추리(rationality)에는 두 종류가 있는데 무제한적 추리(unbounded rationality)와 제한적 추리이다. 무제한적 추리에 바탕을 두고 있는 의사 결정 모델에서는 실제로 사람이 겪고있는 시간, 지식, 계산 능력 등의 제한이 거의 없다. 이 모델은 전통적으로 확률 이론에 바탕을 두었고, 기대 유틸리티의 최대화와 Bayesian 모델이 대표적인 것이다.

제한적 추리에 바탕을 두고 있는 의사 결정 모델에서는 만족화(satisficing)와 fast and frugal 휴리스틱스가 있다. 만족화는 가능한 경우의 선택을 찾아가고, fast and frugal 휴리스틱스는 여러 종류의 결정을 내리기 위해서 정보나 계산을 거의 이용하지 않는다. 그리고 이 휴리스틱스는 적응가능한 선택을 위해서 최소한의 시간, 지식, 계산만을 필요로 한다. 의사 결정을 위해서 적은 시간과 지식을 필요로 한다는 것은 한가지 이유에 의한 결정(one reason decision making)의 형태를 띠고 있다. 이 결정에서는 결정을 위해서 여러 가지 이유의 결합이 아니라 단지 하나의 정보만 필요하다는 것을 의미한다. 왜 한가지 이유에 의한 결정이 타당성이 있느냐 하면, 첫째, 서로 다른 이유의 결합은 공통의 단위로 변경을 해야 하는데 이는 상당히 비용이 많이 드는 작업이다. 둘째, 모든 믿음, 소망 등에 대해서 개량적인 확률이나 유틸리티가 가능하다고 가정해도, 이것은 수학적으로 편리한 가정일 뿐이지 실 세계는 그렇지 못한 경우가 많이 있다.

서로 다른 환경은 각기 특수한 fast and frugal 휴리스틱스를 필요로 한다. 그렇지만 약간의 차이점 때문에 모두 다른 휴리스틱스를 필요로 한다면 처리할 수 없을 정도의 많은 휴리스틱스를 만들어내어야 할 것이다. fast and frugal 휴리스틱스는 간결성 때문에 이런 어려움을 극복해주고 있고, 새로운 환경에서도 쉽게 일반화할 수 있도록 해주고 있다. 간결한 휴리스틱스 중에서 대표적인 것은 Take The Best가 있는데, 이는 특징변수(cue)를 차례로 조사하여 두 개체를 차등화 시켜 줄 수 있는 변수를 발견하면 이 변수가 추론의 근거가 되고 나머지 특징변수는 모두 무시한다. Take The Best는 특히 훈련 집합(training set)의 규모가 적을 때 다중 회귀(multiple regression)를 능가한다 [6].

3.2 알고리즘

이 절에서는 구매 에이전트와 판매 에이전트 사이의 협상 알고리즘 NEGOTIATE를 제시한다. NEGOTIATE에서 영역 지식만 쓰기를 원하면 $numS=n$ 으로 하고, 영역 지식과 일반 지식을 함께 사용하려고 하면 $0 < numS < n$으로 두며, 일반 지식만 사용하길 원하면 $numS=0$으로 둔다. NEGOTIATE 알고리즘은 정책을 쉽게 바꿀 수 있다는 면에서 융통성이 있다. 다음은 협상 알고리즘 NEGOTIATE를 나타낸다.

3.3 협상 시나리오

상품 P에 관한 데이타베이스 $GoodDB$와 환산된 x_i 값을 표1로 가정하자. 상품 특징 변수(attribute 혹은 cue) 로서 $\mathbf{X_1}$: 가격, $\mathbf{X_2}$: 배달 시간, $\mathbf{X_3}$: 보증 기간을 가정한다. 문제의 간결성을 위해서 3개의 특징 변수만 가정한다. 표1에서 각 특징 변수의 최선 값(x^*_i)과 최악 값(x^o_i)을 선택하여 이를 토대로 0과 1사이의 값으로 환산된 x_i 값을 구한다. 예를 들면 x^*_1은 $35 이고 x^o_1은 $60이므로 환산된 x_1 값을 구하는 공식은 $(60 - x_1) / 25$ 이다. 3.1.1절의 덧셈 독립, 유틸리티 독립을 가정하면 유틸리티 함수는 $u(x_1, x_2, x_3)=k_1u_1(x_1)+k_2u_2(x_2)+$

```
function NEGOTIATE (problem) returns a solution or failure
  inputs: problem        : 협상 문제
  static:
      GoodDB             : 상품과 사용자의 선호도를 담고 있는 데이타베이스
        G_i              : i 번째 상품, 1 ≤ i ≤ n
        T_ijk            : step 1에서 상품 G_i 선택, step 2에서 G_j 선택, 협상 후에 구매사가 G_k를
                           구입하는 경우의 거래
        n(T_ijk)         : 지금까지 성공적으로 거래된 T_ijk 회수
        CaseBase         : n(T_ijk)를 담고있는 기록 데이타베이스, n(T_ijk) 값에 대해 내림차순으로 정렬
        numS             : 영역 지식을 사용한 회수
  local variables:
      V(G)={G_1, G_2, ... , G_n}     : 전체 상품 집합
      V(S): MAUT에 의해 선택되는 V(G) 부분집합, 유틸리티 값으로 내림차순 정렬
      V(R): step1 에서 선택되지 못한 나머지 상품 집합, 즉 V(G) - V(S)
  step 1 // MAUT 기반 영역 지식 이용
    PREFERENCE() // MAUT에 따라서 사용자와의 대화에 의해서 유틸리틱 함수를 결정
    for each element in V(S) do
      negotiate the element
      if succeeded then SUCCESS(); end
    if n=numS then FAIL() // MAUT만 사용할 경우
// 협상이 실패하면 영역 지식 기반 협상에서 일반 지식 기반 협상 정책 변경
  step 2 // 구매기록 데이타베이스 CaseBase와 간결한 휴리스틱스 기반 일반 지식 이용
    for each element G_k in V(R) where n(T_ijk) > 0 in CaseBase do
                    // CaseBase는 정렬 되어있으므로 첫 항목은 항상 n(T_ijk) 최대값
      negotiate G_k and remove it from V(R)
      if succeeded then SUCCESS(); end
    // 더이상 그런 G_k 가 없으면
    CUES_ORDER() // 특징변수의 우선 순위는 사용자와의 대화에 의해서 결정된다.
    for each element G_k in V(R) // 협상하지 않았던 나머지 모든 상품에 대해서
      G_k ← TAKE_THE_BEST(GoodDB) // TakeTheBest 휴리스틱스
      negotiate G_k
      if succeeded then SUCCESS(); end
    FAIL() // 모든 협상에서 실패
```

```
function PREFERENCE() returns a utility function
// MAUT에 따라서 사용자와의 대화에 의해서 유틸리티 함수를 결정
  static u(x_1,x_2,..,x_n)     : 유틸리티 함수
  u(x_1,x_2,..,x_n) ← k_1u_1(x_1)+k_2u_2(x_2) +...+ k_nu_n(x_n) //모든 i에 대해 u_i(x°_i)=0, u_i(x*_i)=1, k_i는 상수
          // u_i(x_i)는 사용자와의 대화에 의해서 다음과 같이 결정된다.
          if risk prone then b(2^cx -1)
          else if risk neutral then bx
          else if risk averse then b log_2(x+1)  // b, c > 0 인 상수
  return u(x_1,x_2,..,x_n)
```

```
function TAKE_THE_BEST(GoodDB) returns G_k // 최선은 취하고, 나머지는 무시
  inputs GoodDB        : 상품과 사용자의 선호도를 담고 있는 데이타베이스
    if the most important preference is ω, then G_k is selected in GoodDB whose value of ω is
      the most desirable
    return G_k
```

```
function SUCCESS() returns a solution
  add n(T_ijk) in CaseBase by 1, and terminate the negotiation with success
  return a solution //exit(0)
```

```
function FAIL() returns failure
  Terminate the negotiation in fail.
  return failure // exit(1)
```

$k_3u_3(x_3)$와 같이 정의된다. 여기서 모든 i에 대해서 $u_i(x^o_i)=0$, $u_i(x^*_i)=1$이다.

표 1 상품P에 대한 GoodDB와 환산된 xi 값

	G_1	G_2	G_3	G_4	G_5	G_6	G_7
x_1 (price)	$45	$60	$50	$35	$40	$55	$45
환산된 x_1	.6	0	.4	1	.8	.2	.6
x_2 (delivery)	6	4	1	14	3	7	7
환산된 x_2	.61	.77	1	0	.85	.54	.54
x_3 (warranty)	2	6	5	1	2	3	2
환산된 x_3	.2	1	.8	0	.2	.4	.2

이제 k_i 값을 결정하기 위해서 몇몇 의미 있는 정성적 질문을 한다. 가령 $\boldsymbol{X_2}$ 와 $\boldsymbol{X_3}$ 가 최선 값 x'_2 와 x'_3 이 되는 것보다 $\boldsymbol{X_1}$이 최선 값 x^*_1 이 되는 것을 의사 결정자가 선호하면 $k_1 > k_2+k_3$ 이다. 즉 $k_1 > .5$ 이다. 다시 말하면 이 의사결정자는 배달 시간(x_2)이나 보증 기간(x_3)이 좋은 것 보다 가격 싼 것(x_1)을 선호한다는 의미이다. 또한 $\boldsymbol{X_3}$ 가 x^o_3 에서 x^*_3 로 가는 것 보다 $\boldsymbol{X_2}$ 가 x^o_2 에서 x^*_2 로 가는 것을 더 선호한다면 $k_2 > k_3$ 이다. 즉 보증 기간 보다 배달 시간이 빠른 것을 선호한다는 의미이다. 일반적으로 특정 변수의 수가 많을수록 확률에 의존하지 않고 k_i 의 특성을 이끌어 낼 수 있다 [5]. 만약 k_i=.6으로 평가되었다면 의사결정자는 (x^*_1, x^o_2, x^o_3) 과 제비뽑기 <(x^*_1, x^*_2, x^*_3), .6, (x^o_1, x^o_2, x^o_3)> 사이의 어떤 결정도 무관하다는 것을 의미한다. 제비뽑기 <(x^*_1, x^*_2, x^*_3), .6, (x^o_1, x^o_2, x^o_3)>은 (x^*_1, x^*_2, x^*_3)일 확률 .6과 (x^o_1, x^o_2, x^o_3)일 확률 .4를 의미한다. k_1 이 .6 이므로 k_2+k_3=.4이다. 이제 의사결정자가 (x^o_1, x^*_2, x^o_3)와 <(x^o_1, x^*_2, x^*_3), p, (x^o_1, x^o_2, x^o_3)>의 선택을 무관하게 할 수 있는 확률 p에 대한 질의를 한다. 만약 의사 결정자의 답이 .7이라면 공식 k_2=p(k_2+k_3) [5]에 의해서 k_2=.28이 된다. 결과적으로 유틸리티 함수 $u(x_1, x_2, x_3)$=.6$u_1(x_1)$+.28$u_2(x_2)$+.12$u_3(x_3)$ 이 된다. 이제 $u_i(x_i)$를 결정하기 위해서 제비뽑기 <$x+h$, $x-h$>와 기대결과 x의 선호도를 질의한다. 만약 x를 선호하는 것으로 확인되면 의사결정자는 모험 회피이며, 각 $u_i(x_i)$는 $b\boldsymbol{log}_2(x_i+1)$의 형태 (concave)가 된다. 즉 $u(x_1,x_2,x_3)$= $0.6\boldsymbol{log}_2(x_1+1)+0.28\boldsymbol{log}_2(x_2+1)+0.12\boldsymbol{log}_2(x_3+1)$이 된다. 여기서 x_i 값은 표1의 환산된 값이다. NEGOTIATE 알고리즘의 상세한 적용 예는 아래와 같다.

function NEGOTIATE(*problem*)

step 1:

PREFERENCE() // 협상 시나리오에 따라 유틸리티 함수 결정 :

$u(x_1,x_2,x_3)=0.6\boldsymbol{log}_2(x_1+1)+0.28\boldsymbol{log}_2(x_2+1)+0.12\boldsymbol{log}_2(x_3+1)$

$numS$=3 가정.

유틸리티 함수의 값에 따라 G_5, G_3, G_4의 순으로 선정. 따라서 $V(S)$={G_5, G_3, G_4}.

구매/판매 에이전트가 $V(S)$에 있는 원소의 순으로 협상. 모든 협상 실패 가정.

// *numS번 협상후에도 실패하였으므로 협상정책을 영역지식 기반에서 일반지식 기반으로 변경*

step 2

*CaseBase*의 초기 값으로 표 2 가정.

step 1에서 선택되지 못한 상품 집합 $V(R)$={G_1, G_2, G_6, G_7}이다. $n(T_{ijk})$ 값이 0 보다 큰 것은 G_2, G_3, G_4 이지만 G_3, G_4는 step1에서 이미 실패하였으므로 $V(R)$의 원소 중에서 협상 대상인 상품은 G_2 뿐이다. 에이전트는 G_2와 협상. 협상 실패 가정.

CUES_ORDER() // 변수의 우선 순위가 의사결정자에 의하여 (x_1, x_2, x_3)으로 선택되었다고 가정

TAKE_THE_BEST() // *TakeTheBest 휴리스틱스 호출*

우선 순위에 따라 G_1, G_7, G_6 의 순으로 선택됨. G_7과 G_1의 x_1 값이 같지만 두 번째 우선 순위의 특징변수 x_2 에서 G_1이 G_7 보다 우수하므로 G_1이 선택된다.

G_1과의 협상을 실패로, G_7과의 협상을 성공으로 가정하면, *CaseBase*는 표 3과 같이 된다. 표 3은 표 2에서 T_{337}을 추가한 결과인데, T_{337}의 의미는 step1에서 G_5가 최초로 선택되고, step 2에서 G_7가, 최종적으로 구매자가 C_1를 구매한 구매 기록을 나타낸다. 이때 $n(T_{ijk})$ 값이 큰 G_k가 우선적으로 추천된다. 왜냐하면 $n(T_{ijk})$ 값이 큰 G_k가 구매자들에게 보다 인기가 있다고 가정할 수 있기 때문이다. 다만 각 step에서 복수개의 협상이 이루어져도 처음으로 선택된 상품만 저장하는데 이는 처음 선택된 상품의 유틸리티 값이나 $n(T_{ijk})$의 값이 가장 크기 때문이다.

표 2 G_7 구매 전의 *CaseBase*

T_{ijk}	$n(T_{ijk})$
T_{122}	3
T_{123}	2
T_{554}	2

표 3 G_7 구매 후의 *CaseBase*

T_{ijk}	$n(T_{ijk})$
T_{122}	3
T_{123}	2
T_{534}	2
T_{527}	1

4. Pmart의 설계 및 구현

4.1 Pmart 계층구조

Pmart는 CORBA를 미들웨어로 하여 웹 환경에서 전자상거래 서비스를 제공하는 응용 프레임워크이다. 그림 2는 분산 환경에서의 프레임워크 Pmart의 계층 구조를 나타낸 것이다. Pmart는 응용 계층(Application layer), 프레임워크 계층(Framework layer), 미들웨어 계층(Middleware layer) 등으로 구성된다. 응용 계층은 Pmart External, 응용 지식(Domain knowledge) 등 가변 부분으로 구성되고, 프레임워크 계층은 Client, Server, DB Server 등 불변 부분으로 구성된다. 따라서 응용 영역이 변하면 응용 계층은 변화 시켜 주어야 하지만, 프레임워크 계층은 재 사용할 수 있어서 소프트웨어 재사용(software reuse)이 가능하다.

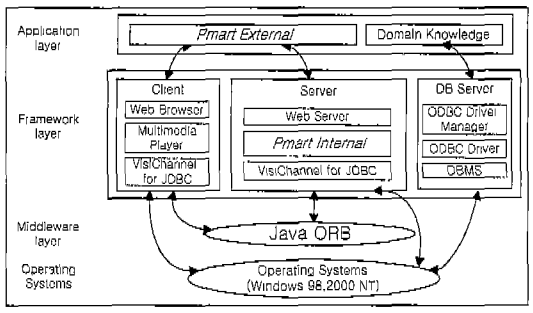

그림 2 분산환경에서의 *Pmart* 계층 구조

4.1.1 Pmart External과 응용 지식

Pmart External은 응용 영역의 변화에 민감한 에이전트들로 구성되고, 응용 지식(Domain Knowledge)은 특정 영역에 관한 지식이다. Pmart External 부분과 응용 지식을 변화 시킴으로써 다른 형태의 전자 상거래 시스템을 구성할 수 있다.

4.1.2 Client

Client는 Web Browser, Multimedia Player, VisiChannel for JDBC Client 등으로 구성된다. Java 가능한 웹 브라우저를 통해 사용자 인터페이스인 애플릿을 실행할 수가 있다. 데이타베이스 접근을 위해서는 VisiBroker for Java의 API를 통해 애플릿의 클라이언트 객체를 생성하고 클라이언트 ORB를 통해 서버에 요청하게 된다 [17,18]. Java가 WWW을 객체 지원이 가능한 대화형 시스템으로 변형하였지만, 클라이언트/서버 환경에서 객체의 위치에 무관하게 작동되는 편리한 기법을 제공하는 데에는 어려움이 있다. 그래서 범용 네트워크 컴퓨팅 플랫폼(platform for universal network computing)을 형성하기 위해서 Java는 프로그래밍 언어와 이동 코드 시스템(mobile code system)으로 사용하고, CORBA는 통합 기술로 사용 가능하다 [19]. CORBA의 stub 클래스의 역할은 클라이언트들이 호출할 수 있는 프록시 객체(proxy objects)를 제공해 주고 있다. 프록시 객체는 요구하는 객체 호출을 marshaled form으로 전송하게 된다. marshaled form은 ORB의 인프라 구조(ORB's infrastructure)를 통해서 구현 객체로 전송된다.

4.1.3 Server

서버의 구성은 웹 서버, 영역 지식에 독립인 Pmart Internal, 서버 ORB, VisiChannel for JDBC Server로 구성된다. VisiChannel for JDBC는 클라이언트의 JDBC 프로그램이 서버의 ODBC data source에 접근할 수 있도록 해주는 다-계층(multi-tiered) 구조이다 [17]. CORBA의 skeleton 코드는 구현 객체, CORBA 서버, ORB 등을 결합해주는 역할을 한다. skeleton 클래스는 서버로 넘어온 객체 호출을 unmarshall 해서 해당 객체를 찾아주는 역할을 한다. 만약 데이타베이스 접근 요청일 경우 VisiChannel for JDBC를 통해 데이타베이스에 접근하게 된다.

4.1.4 DB Server

데이타베이스 서버는 ODBC Driver Manager, ODBC Driver, DBMS 등으로 구성된다. ODBC Driver Manager는 ODBC API를 구현하고 있으면서, ODBC 드라이버를 필요할 때 동적으로 적재한다. VisiChannel for JDBC Client 는 JDBC 응용 프로그램과 애플릿이 서버에 설치된 ODBC 드라이버를 사용할 수 있도록 해준다. VisiChannel for JDBC를 통한 접근은 ODBC Driver Manager와 ODBC Driver를 통하여 수행된다. 서버 측의 ODBC 드라이버는 DBMS와 직접 교류하게 해준다.

4.1.5 ORB와 운영 체제

CORBA는 HTTP/CGI가 제공해주고 있는 현재의 웹-기반 시스템의 클라이언트/서버 기능을 변경하고 있다

[17,18]. Java와 CORBA의 결합 즉, Java ORB는 Java 컴퍼넌트가 클라이언트 컴퍼넌트와 서버 컴퍼넌트로 분산 될 수 있도록 해주고, 다운로드 시간을 줄임으로써 웹 클라이언트/서버 모델이 더욱 각광을 받도록 해주고 있다. 운영체제는 Windows기반의 Windows98, 2000, NT를 이용할 수 있다.

4.2 Pmart Internal과 External

제안하는 프레임워크 Pmart의 전체 구조는 그림 3과 같다. 전자 상거래 프레임워크 Pmart는 특정 영역에 독립인 Pmart Internal과 영역에 의존하는 Pmart External로 구성되어있다.

4.2.1 Pmart Internal

Pmart Internal은 하나의 조정 에이전트(Facilitator)와 3개의 내부 에이전트인 구매 에이전트(Buyer Agent), 판매 에이전트(Seller Agent), 정보 에이전트(Information Agent)로 구성된다. Pmart 내의 조정 에이전트는 Agent Name Server의 기능과 메타지식을 이용하여 시스템 내에 존재하는 에이전트들간의 통신을 돕는 기능을 한다. 각 에이전트는 생성될 때 조정 에이전트에 에이전트 이름 및 그 에이전트가 할 수 있는 기능과 주소를 등록한다. 조정 에이전트는 어떤 에이전트로부터 문제 풀이를 요청 받았을 때, 메타 지식을 이용해서 그 에이전트에게 해결책을 알려준다. 따라서 각 에이전트는 조정 에이전트의 도움으로 원하는 에이전트를 실제 위치와 무관하게 찾을 수 있다. 제안하는 에이전트의 구조는 멀티에이전트 시스템 구조로서 하나의 에이전트로 해결하지 못하는 복잡한 문제의 해결을 위해서 여러 응용 에이전트간의 협동을 통하여 문제를 해결하고 있다.

작업 에이전트(Task Agents)는 작업 영역에 관한 지식을 가지며, 문제 풀이 계획을 세움으로써 의사 결정을 돕고, 다른 소프트웨어 에이전트와 질의/정보 교환을 하며, 대부분의 자율적인 문제 풀이 과정을 수행한다. 그림 3의 구매 에이전트, 판매 에이전트 등이 여기에 속한다. 주요 기능은 우선 인터페이스 에이전트로부터 사용자가 위임한 작업을 접수하고, 사양을 해석하여 문제풀이 목표를 생성하며, 목표를 만족시키는 부 목표(subgoal) 계획으로 분할하고, 마지막으로 계획 수행을 위하여 작업 에이전트 혹은 정보 에이전트와 조정 작업을 한다.

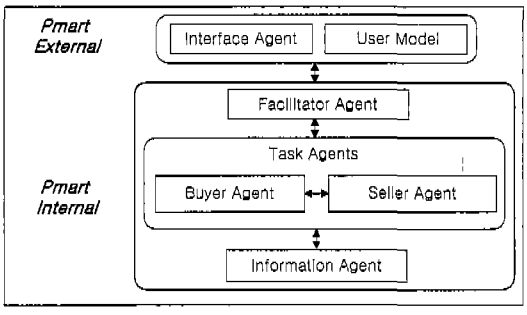

그림 3 Pmart의 구조

판매 에이전트(Seller Agent)는 판매 세션이 시작될 때 다운로드되는 애플릿이며, 전자 상거래 서비스를 전체적으로 관리한다. 이 에이전트는 판매와 협상에 관한 지식이 있으며 추론 엔진과 판매자의 행동을 분석하는 기능을 가진다.

구매 에이전트(Buyer Agent)도 구매 세션이 시작되었을 때 다운로드되는 애플릿이며, 구매와 협상에 관한 지식, 추론 엔진, 구매자의 행위를 분석할 수 있는 기능을 가진다. 협상이 끝나면 구매/판매 에이전트는 조정 에이전트와 통신하여 사용자 모델을 갱신한다.

정보 에이전트는 이형질의 정보 집합에 접근을 하며, 연관된 정보에 대한 모델을 가지고, 소스 선택, 정보 접근, 충돌 해결, 정보 반환 등의 일을 한다.

구매, 판매, 조정 에이전트 사이의 통신은 KQML 메시지 패킷에 바탕을 두고 있으며, Marketplace [20]의 에이전트 간 통신 개념을 수정하여 사용한다.

4.2.2 Pmart External

Pmart External은 인터페이스 에이전트(Interface Agent)와 사용자 모델(User Model)로 구성되며 영역 변경 시 수정해야 하는 부분이다. 인터페이스 에이전트는 사용자의 요구를 받고 결과를 되돌려 주는 방식으로 사용자와 상호 작용을 하며, 사용자의 작업을 조정하기 위해서 사용자의 기호를 알아서 모델링하여 사용한다. 주요 기능은 우선 해당 정보를 사용자로부터 획득하고, 사용자에게 결과와 설명을 포함한 관련 정보를 제공하며, 필요한 경우 상거래 도중에 추가 정보를 요구하거나 사용자의 승인을 요구한다. 이 에이전트는 세션 중에서 가장 먼저 클라이언트로 다운로드 되는 자바 애플릿으로서 사용자 모델의 내용을 가져오거나 저장하기 위해 정보에이전트와 통신한다.

사용자 모델은 구매자와 판매자에 관한 지식과 협상에 관한 지식인 구매 기록 데이타베이스 *CaseBase*를 가지고 있다. 통신상의 효율을 위해서 서버 측과 클라이언트 측에 동시에 위치시킨다. Pmart에서 가장 빈번한 통신이 일어나는 곳이 인터페이스 에이전트, 구매 에이전트, 판매 에이전트, 사용자 모델 부분이므로 거래 시

이 부분을 전부 클라이언트에 위치시키고 거래가 끝날 경우 클라이언트 측의 사용자 모델 내용이 일관성을 위해서 서버의 사용자 모델에 복사된다.

4.3 Pmart의 구현

구현 환경은 다음과 같다.

서버 : Windows NT 4.0 Server
클라이언트 : Windows 98/2000
웹 서버 : Internet Information Server3.0
DB 서버 : NT Oracle 7
웹 Browser : Netscape Communicator 4.7./Internet Expleror 5.0
개발 소프트웨어 : JDK1.2.2, Symantec Visual Cafe, VisiBroker for Java, VisiChannel for JDBC

제안하는 전자상거래 Pmart는 *n*개의 웹 클라이언트들이 Java ORB인 *VisiBroker for Java 3.2*를 통해서 복수의 웹 서버와 DB 서버의 서비스를 받는다. *CORBA(Common Object Request Broker Architecture)*는 *OMG(Object Management Group)*에서 정의한 분산 객체간에 서비스를 제공하거나 획득할 수 있도록 해주는 통신 환경의 표준사양으로서, 가장 큰 장점은 분산환경에서 클라이언트/서버간의 인터페이스만 정의되면 서비스 요구나 결과 값의 전달이 하부 통신 메카니즘에서 투명하게 이루어진다는 것이다. 이와 같이 CORBA는 네트워크 투명성을 제공하고 있고, Java는 구현 투명성을 제공하는데 현재 이들 두 모델을 접목한 Java ORB가 유용하게 사용되고 있다.

원격 데이타베이스를 연결하기 위해서 *Inprise*사의 VisiChannel for JDBC 1.0을 사용한다. VisiChannel for JDBC는 클라이언트의 JDBC 프로그램이 ODBC data source의 데이타를 접근할 수 있도록 해주는 다-계층 아키텍처이다. JDBC를 이용하여 데이타베이스 연결 자바 프로그램을 작성할 수 있고 이들을 인터넷과 인트라넷에 분산시킬 수 있다. 기존의 ODBC 드라이버를 이용함으로써 JDBC 응용 프로그램이나 애플릿이 컴퓨터 네트워크 상에 산재해있는 ODBC data source의 데이타를 접근할 수 있게 된다.

JDBC(Java Database Connectivity)는 데이타베이스 및 플랫폼 독립 표준 API를 정의하여 자바 프로그램에서 인터넷/인트라넷상의 데이타베이스를 접근 할 수 있도록 하고 있다. JDBC 모델에서 자바 프로그램은 JDBC 클래스를 사용하여 데이타베이스와 교신하는 JDBC 드라이버를 적재한다. JDBC는 어떤 SQL 스트링이라도 DBMS에 전달되어 실행되도록 한다. ODBC(Open Database Connectivity)는 데이타베이스에 저장된 데이타를 접근하는 데이타베이스 독립 표준 API이다. VisiChannel for JDBC Client는 JDBC 응용 프로그램과 애플릿이 서버에 구축되어 있는 ODBC 드라이버의 특성을 사용할 수 있게 해준다. VisiChannel for JDBC Client는 Java 클래스들로 구성되어 있는데 이 클래스들은 서버로부터 필요시 자동으로 다운로드 되든지 혹은 클라이언트에 따로 구축될 수 있다. VisiChannel for JDBC Server는 ODBC 드라이버 매니저를 통하여 data source에 접속한다. 서버 측의 ODBC 드라이버는 DBMS에 직접 연결된다. ODBC Driver Manager는 ODBC API를 구현하고 있고 필요한 정보를 제공하고 있으며, 필요시 ODBC 드라이버를 동적으로 적재한다.

Pmart 원시시스템을 구현하는데 있어서 본 실험실에서 개발한 분산 환경을 바탕으로 하였으며 [21], 표준 KQML의 에이전트간 통신 기능을 Java method로 구현한 Marketplace [20]를 수정하여 사용하였다. 본 연구실에서 구축한 분산환경은 *VisiBroker for Java 3.2*를 바탕으로 하고 있지만, 현재는 *VisiBroker for Java 4.5*를 가지고 작업을 하고 있다.

그림 4는 Pmart 에이전트들 사이의 전형적인 동작 모델을 나타내고 있다. 구매 에이전트와 판매 에이전트가 전자상거래 시스템의 다른 에이전트들과 통신하기 위해서 사전에 조정 에이전트에 등록되어야 한다.

사용자가 자신의 고유 에이전트를 만들어서 상거래를 시작시켜 주면, 하나의 조정에이전트(Facilitator Agent)가 생성되며, 판매 에이전트는 조정 에이전트에게 판매자로부터 의뢰 받은 상품을 광고한다(그림4 Seller Agent의 2). 구매 에이전트는 구매자로부터 요청 받은

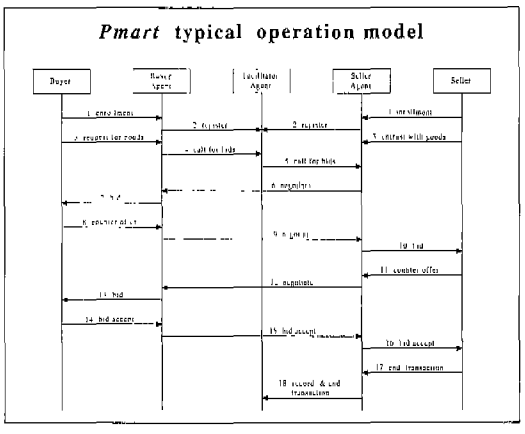

그림 4 *Pmart*의 전형적인 동작 모델

상품의 구매 목록을 만든다(그림 4 Buyer Agent의 2). 하나의 구매 에이전트는 원하는 상품을 구매하기 위해서 조정 에이전트의 도움으로 입찰(call for bids)을 보낸다(그림 4의 4,5). 즉, 조정 에이전트에게 구매하고 싶은 물건을 팔려고 하는 판매 에이전트를 추천해 줄 것을 요구한다. 만약 해당 판매 에이전트가 복수이면, 조정 에이전트는 기록되어 있는 전체 정보를 구매 에이전트에게 넘긴다. 그러면 구매 에이전트는 본 논문에서 제안한 협상 알고리즘 NEGOTIATE에 의해서 판매 에이전트와 협상을 시작한다. 현재의 협상에 만족하지 못하면 서로 역제의(counter offer) 할 수 있다(그림4의 8, 11). 만약 구매 에이전트가 판매 에이전트의 수정된 입찰을 받아들이는 경우에는 입찰 수락(bid accept)을 판매 에이전트에게 보내서 거래를 마무리하고(그림4의 15), 판매 에이전트는 거래 결과를 구매 기록 데이타베이스 *CaseBase*에 보관하고 거래를 종료한다(그림4의 18).

그림 5는 구현된 Pmart의 초기 화면이다. 보기 편리한 GUI 구현은 현재 진행중이며 현재는 엔진 부분만 구현되어 있다. 각 에이전트는 생성될 때 이름, 기능, 주소 등을 조정 에이전트에게 등록한다. 그러므로 조정 에이전트에서는 원하는 거래의 초기 등록 정보를 확인 할 수 있다. Pmart 윈도우는 판매 에이전트와 구매 에이전트 사이의 협상 내용을 보여 주고 있다.

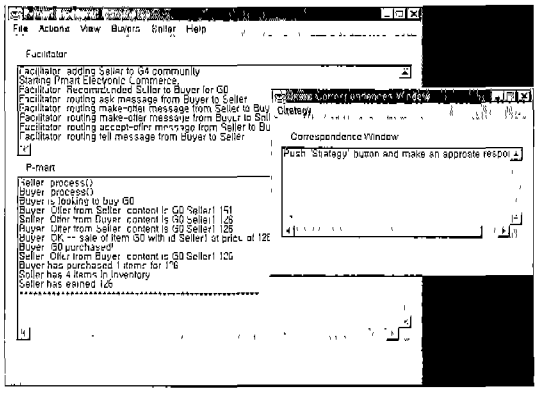

그림 5 Pmart의 초기화면

그림 6은 협상이 진행되면서 영역 지식과 일반 지식 중에서 영역 지식을, 유틸리티 함수 중에서 모험 회피(risk averse)를 단계적으로 선택했을 때 나타나는 화면이다. 의사 결정자는 단조 감소 유틸리티 함수가 오목 다각형(concave)일 때 또 그때 만 위험 회피이다[5]. 이런 종류의 특성을 만족시키는 유틸리티 함수는 $u(x) = h + k(-e^{-ax} - be^{-cx})$이다. 여기서 a, b, c, k는 양의 상수이다. 다섯 개의 미지수가 있으므로 다섯 개의 식이 필요한데, 이는 사용자와의 대화에 의해서 본 논문의 3절과 같이 해결한다.

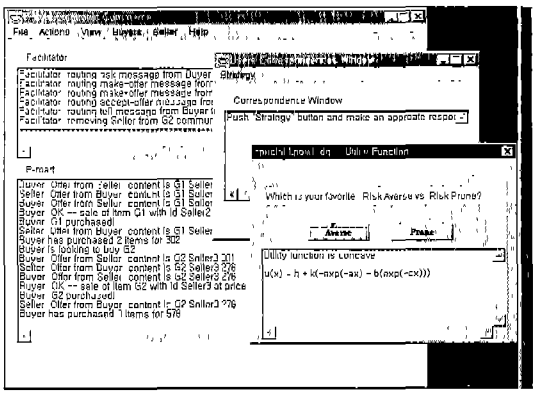

그림 6 특수지식/모험회피를 선택한 경우

4.4 Pmart와 타 시스템의 비교

Pmart와 비교할 수 있는 시스템은 MIT의 Kasbah와 Tete-a-Tete이다. 나머지 시스템들은 협상과정이 단순하거나 전체변수에 대한 협상과정이 없는 실정이다. 따라서 Kasbah와 Tete-a-Tete 시스템의 협상과정을 Pmart의 협상과정과 비교해서 설명한다. 또한 전자 상거래 시스템은 아니지만 노사(勞使) 문제의 협상과정을 모의 프로그램하고 있는 PERSUADER[22] 시스템과도 비교한다.

4.4.1 Kasbah 시스템의 협상 과정

Kasbah에서 사용자들은 상거래를 본인의 취향에 따라 협상하고 거래하는 에이전트에게 위임한다. 구매 에이전트와 판매 에이전트가 일치된 다음, 구매 에이전트가 취하는 행위는 판매자에게 입찰(bid)을 보내는 것이다. 그러면 판매 에이전트는 yes나 no로 답을 하게된다. Kasbah는 구매자에게 3가지 협상 방법을 제공해 주는데, anxious, cool-headed, frugal 등이 있다. 이들은 시간이 지남에 따라 판매자 입장에서 값을 깎아 주거나 구매자 입장에서 값을 더 올려주는 함수인데 각각 선형, 2차원, 지수 함수에 해당한다. 이 협상은 멀티-에이전트, 상호 거래 등의 형태를 띠고 있지만 단순한 증감 함수를 사용하고 있어서 사용자들의 취향을 제대로 반영할 수 없다는 단점이 있다. 또한 협상 대상이 가격에만 제한되어 있어서 가격이외의 협상이 필요한 경우에는 사

용할 수 없다. 실험 환경은 MIT 대학 내에서 벼룩 시장과 같이 책과 음악에 관한 것을 사고 팔 수 있는 C2C 형태의 모습을 띠고 있다.

4.4.2 Tete-a-Tete 시스템의 협상 과정

대부분의 온라인 협상 시스템이 가격에 대해서만 협상하는 것과는 달리 Tete-a-Tete 에이전트들은 거래의 여러 변수에 대해서 협상을 벌인다. 이 시스템은 의사 결정 보조 모듈과 함께 통합적인 협상 모델을 사용한다. 의사 결정 보조 모듈은 MAUT와 분산 제약 만족(distributed constraint satisfaction)에 바탕을 둔 제품 최적화(product customization)를 사용한다. 제품 최적화를 통해서 같은 제품이라 할 지라도 여러 다른 사양에 따라서 발생하는 수많은 경우의 수를 취급 가능한 수로 조절할 수 있다는 장점이 있다. 그러나 이 시스템은 불확실한 정보를 가지고 의사 결정을 할 때에는 협상에서 좋은 결과를 내기 어렵다. 왜냐하면 협상의 기본이 MAUT에만 바탕을 두고 있고 사용자의 불확실한 행위에 대해서는 효율적인 대처 방안이 없기 때문이다.

4.4.3 PERSUADER 시스템의 협상 과정

PERSUADER는 노사문제에서 중재자의 행위를 모의 프로그래밍한 시스템이다. 이 시스템은 세 개의 에이전트 즉 회사, 노조, 중재자 에이전트로 구성되어 있다. 중재자 에이전트는 현재의 협상과 유사한 과거의 사례에서 추론하는 사례 기반 추론(Case Based Reasoning)과 MAUT에 바탕을 둔 협상을 진행한다. 사례 기반 추론을 사용하는 목적은 기존 인공지능의 계획(planning)과 전문가 시스템의 단점을 극복하기 위한 것이었다. 즉 문제 풀이를 할 때마다 유사한 경우에도 처음부터 모든 단계를 다시 시작해야 하는 규칙 기반 전문가 시스템의 한계를 극복하고 PERSUADER는 새로운 경험이 들어올 때마다 CASE 메모리를 갱신한다. 메모리 갱신은 자동적으로 지식 획득과 학습을 가능하게 하여서 협상의 효율성과 질을 높일 수 있다. 또한 MAUT와 함께 사용함으로써 사례 기반 협상을 정량적으로 평가할 수 있는 프레임워크를 제공하고 있다.

4.4.4 Pmart 시스템의 협상 과정

Kasbah는 사용자의 선호도를 확인하는 과정에서 3종류의 단순한 함수 즉, 선형, 2차, 지수 함수만으로 표현함으로써 사용자의 선호도를 제대로 표현할 수 없고 협상도 단지 가격에 대해서만 이루어진다는 단점이 있다. 이에 비해서 Pmart는 MAUT, CaseBase, 간결한 휴리스틱스 등을 사용함으로써 사용자의 선호도를 보다 구체적으로 확인 할 수 있고 가격 이외의 변수에 대해서도 협상 해준다.

Kasbah 보다 개선된 협상 알고리즘을 제시하는 Tete-a-Tete는 MAUT와 제품 최적화를 사용하는데, 한 상품의 수많은 경우의 수를 제품 최적화를 통해서 취급 가능한 수로 줄였다는 점에서는 타 시스템보다 뛰어나지만, 협상 과정이 대개 불확실성 하에서 이루어지고 있는 만큼, 과거 거래 행위에 대한 사례 연구나 휴리스틱스 등의 사용을 못했기 때문에 불확실성 하에서 의사 결정을 하는데는 문제점이 있다. 반면에 Pmart 에서는 협상과정에서 2단계 의사 결정 과정을 사용하는데, 첫 번째 단계에서는 영역 지식을 이용하기 위하여 MAUT를 사용하지만, 몇 번 실패하면 둘째 단계에서 지금까지의 성공적인 구매 기록 CaseBase와 간결한 휴리스틱스를 사용할 수 있다. 이 모델에서는 과거의 유사한 성공적인 상황을 고찰하기 위해서 사례 기반 추론을 제안하고 있다. CaseBase는 상품 G_k의 지금까지 성공한 거래회수 $n(T_{ijk})$를 담고 있는 구매 기록 데이타베이스로서 앞으로 유사 상황의 거래에 의미 있는 자료가 될 수 있다. 그러나 두 번째 단계인 CaseBase를 이용해서도 협상이 실패하면 마지막으로 간결한 휴리스틱스에 기반한 협상을 벌인다. 간결한 휴리스틱스(simple heuristics)는 1995년 독일, 미국, 영국에서 심리학, 수학, 컴퓨터과학, 경제학, 생물학 등 학제간의 연구를 위해서 설립된 ABC(Adaptive Behavior and Cognition) 연구 그룹의 주된 이론이다. 이 연구 그룹에서는 제한적 추리(bounded rationality)와 불확실성 하에서 좋은 의사 결정에 관한 연구를 해오고 있다. 간결한 휴리스틱스 중에서 대표적인 것으로 Take The Best가 있는데, 이는 특히 훈련 집합의 규모가 적을 때 다중 회귀를 능가한다 [6].

따라서 제안한 협상 알고리즘 NEGOTIATE는 Kasbah와 Tete-a-Tete의 단점을 보완하고 PERSUADER 시스템의 협상 과정에서 효율적으로 사용되었던 사례 기반 추론을 도입함으로써 보다 성공적인 협상 결과를 기대할 수 있다. 다만 MIT의 Tete-a-Tete와 마찬가지로 Pmart도 실험실 환경에서 구현되고 있기 때문에 최종 구현 후에 많은 사용자들의 반응과 협상 성공률 등을 비교 검토해 보아야 정량적인 비교가 가능할 것이다. 현재로서는 타 시스템과의 정량적인 비교는 어려운 실정이다. 표 4는 국내외 주요 전자상거래 시스템의 특징과 협상 알고리즘을 비교하고 있다.

5. 결 론

오늘날의 1세대 구매 에이전트는 상품의 전체 특징 변수보다는 주로 판매자들이 제시한 가격만 비교해서

표 4 국내외 주요 전자상거래 시스템의 특징 및 협상알고리즘 비교

시스템 / 항목	Tete-a-Tete	Kasbah	AuctionBot	FishMarket	Webnara Mart	Shopbinder	Pmart
개발처	MIT	MIT	U. of Michigan	Spain CSIC	한국 webnara.com	한국 shopbinder.com	본 연구실
특징	의사 결정은 MAUT와 제품 최적화	멀티에이전트 상호 거래 C2C 시스템으로 MIT 내에서 실험	범용 인터넷 경매 서버 경매 전략을 위한API 제공	경매 전략을 Java로 프로그램 할 수 있는 구조	고객의 특성에 따라 적절한 상품 추천	경매, 역경매, 공동구매 등 서비스를 순차적으로 적용	2 단계 의사 결정 과정 (1:MAUT 2:구매 기록 & 간결한 휴리스틱스)
협상 알고리즘	가격과 다른 변수 동시 비교	가격 변수만 비교	가격 변수만 비교	가격 변수만 비교	가격 변수만 비교	가격 변수만 비교	가격과 다른 변수 동시 비교
기타	불확실성 하에서 의사결정 문제점	문산 환경과 대화 모델을 제공못함	경매 형태 선택 => 새로운 경매 구성	Downward bidding protocol 사용	다양한상품 (표준,패키지, 맞춤) 유형	Meta-mall 구조	과거 사례와 휴리스틱스 이용

구매행위를 대행해 주고 있으며, 간혹 가격 이외의 변수에 대한 비교를 해주는 에이전트의 경우에도 협상 과정에서 전체 변수를 적절하게 고려해주는 협상 모델은 찾아보기 힘들다. 따라서 전자 상거래의 협상 모델을 가격 변수뿐만 아니라 상품의 전체 변수로 확장 시켜 주는 것이 절실히 요구되고 있다.

본 논문에서는 지능형 멀티 에이전트에 바탕을 두고 가격, 상품의 특성, 보장 기간, 서비스 정책 등에 대해서 협상을 벌이는 에이전트 중재에 의한 전자 상거래 프레임워크 *Pmart*를 제시하였다. 이를 위한 협상 모델은 영역 지식과 일반 지식을 동시에 사용할 수 있는데, 영역 지식은 MAUT에 바탕을 두고 있고, 상거래의 일반 지식은 지금까지의 성공적인 구매 기록의 Casebase와 간결한 휴리스틱스로 구성되어 있다. 그러므로 본 논문에서 제안하는 협상 알고리즘 NEGOTIATE는 영역 지식인 MAUT와 일반 지식인 CaseBase 그리고 간결한 휴리스틱스인 Take The Best에 바탕을 둔 2단계 의사결정 과정을 통해서 보다 성공적인 결과를 기대할 수 있다.

제시하는 프레임워크는 객체 지향 기법과 소프트웨어 컴퍼넌트(software component)에 바탕을 둔 소프트웨어 재사용(software reuse) 기능을 제공하고 있다. 따라서 Pmart의 가변 부분인 Pmart External의 내용을 변경함으로써 다른 종류의 전자 상거래 시스템을 구축할 수 있게 되었다.

향후 연구 방향은 현재의 협상 알고리즘을 지원할 수 있는 일반적인 경매 프레임워크를 구현하는 것과 B2C, B2B 등에 공통적으로 사용할 수 있는 화상회의 시스템도 추가할 계획이다.

참 고 문 헌

[1] R.H. Guttman and P. Maes, "Agent-Mediated Integrative Negotiation for Retail Electronic Commerce," *Lecture Note in Artificial Intelligence 1571, Agent Mediated Electronic Commerce,* 1998, pp. 70-90.

[2] OnSale:Live Online Auction House. http://www.onsale.com/

[3] United Computer Exchange. http://www.use. com/

[4] Amazon.com.http://www.amazon.com/exec/obidos/subst/home/home.html/

[5] R.L.Keeney and H.Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons, New York, NY, 1976.

[6] G.Gigerenzer et al., *Simple Hueristics That Make Us Smart,* Oxford University Press, New York, 1999.

[7] A.Chavez and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," *Proceedings of the first International conference on the Practical Application of Intelligent Agents and Multi-agent Technology (PAAM96),* London, UK, 1996, pp. 75-90.

[8] R.H.Guttman, *Merchant Differentation through Integrative Negotiation in Agent-mediated Electronic Commerce*, MS thesis, Media Art and Sciences, MIT, 1998.

[9] R.H.Guttman et al., "Agent-mediated electronic commerce: a survey," *Knowledge Engineering Review*, Vol. 13, No. 2, 1998, pp. 147-159.

[10] P.R.Wurman et al., "The Michigan AuctionBot: A Configurable Auctioon Server for Human and Software Agents," In *Proceedings of the Second International Conference on Autonomous Agents*

(Agents98), Minneapolis, MN, 1998, pp.301-308.
[11] E. Steinmetz et al., "Bid evaluation and Selection in the MAGNET Automated Contracting System," *Lecture Note in Artificial Intelligence 1571. Agent Mediated Electronic Commerce*, 1998, pp. 105-125.
[12] J. Eriksson et al., "SICS MarketSpace-An Agent-Based Market Infrastructure," *Lecture Note in Artificial Intelligence 1571. Agent Mediated Electronic Commerce*, 1998, pp. 41-53.
[13] J.A.Rodriguez-Aguilar et al., "Competitive Scenarios for Heterogeneous Trading Agents," In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, Minneapolis, MN, 1998, pp.293-300.
[14] http://www.webnara.com
[15] http://www.shopbinder.com
[16] http://www.daum.net
[17] VisiBroker for Java 3.2 Programmer's Guide, Visigenic Co.,ftp://ftp.visigenic.com/*private /vbj/ vbj32/vbj32.html*
[18] A. Vogel and K. Dubby, *JAVA Programming with CORBA, 2nd Edition*, Wiley Computer Publishing Co., NY, 1998.
[19] S.C.Lewandowski, "Frameworks for Component-Based Client/Server Computing," *ACM Computing Survey*, Vol. 30, No. 1, 1998, pp.3-27.
[20] J.P.Bigus and J.Bigus, *Constructing intelligent Agents with Java*, John Wiley & Sons, New York, NY, 1998.
[21] M. Chung et al., "Multiagent-based Distance Learning Framework using CORBA," In *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA99), Nassau, Bahamas, 1999, pp. 224-228.*
[22] K.P.Sycara, "Negotiation planning: An AI approach," *European Journal of Operation Research*, 46, 1990, pp.216-234.

정 목 동

1981년 경북대학교 컴퓨터공학과 학사. 1983년 서울대학교 컴퓨터공학과 석사. 1990년 서울대학교 컴퓨터공학과 박사. 1984년 ~ 1985년 금성반도체(주) 연구소 연구원. 1985년 ~ 1996년 부산외국어대학교 컴퓨터공학과, 부교수. 1999년 ~ 2000년 미국 Iowa State University, Visiting Professor. 1996년 ~ 현재 부경대학교 컴퓨터공학과, 부교수. 관심분야는 Intelligent Agent, Electronic Commerce, Distance Learning