전자상거래 시스템 모듈의 페트리넷 모형

임재걸
동국대학교 컴퓨터학과
e-mail: yim@wonhyo. dongguk.ac.kr

Petri Net Models of the modules of An E-Commerce System

Jaegeol Yim
*Dept of Computer Science, Dongguk University

요약

본 논문은 mobile agent 시스템 설계의 예를 보인다. 본 시스템 설계에서 구축될 시스템의 사양은 자연어로 묘사되어 있다고 가정한다. 본 시스템 설계 과정은 다음과 같다. 첫째, 시스템 사양으로부터 필요한 사이트들을 분별하고, 둘째, 사이트간에 송수신 되는 정보들을 분별해, 이들 정보 개체들을 각각 한 개의 agent로 구현한다. 셋째, 각 Agent가 수행할 과업을 Petri net으로 정확히 표현한다. Agent가 정의되는 위치에 따라 필요한 통신량이 좌우됨으로 통신량을 최소화시키는 관점에서 Agent 정의의 위치를 결정한다.

1. 서론

인터넷 사용이 보편화됨에 따라 최근 인터넷 기반의 전자상거래가 급속히 신장하고 있다. 전자상거래 시스템은 사용자가 원하는 물품을 당연히 신속하게 찾아 주어야 한다. 또한 같은 물건이라면 가장 저렴한 가격으로 물품을 제공하는 사이트를 찾아 주어야 한다. 이러한 과업을 수행하는 도구로 소프트웨어 에이전트를 들 수 있다[1].

[2]에 의하면 “Agent란 센서를 통하여 환경을 지각하고 작동자를 이용하여 환경에 동작하는 모든 것”이라고 정의되어 있다. 이러한 agent의 종류에는 “전자비서 에이전트”, “상업 에이전트”, ..., “이동 에이전트” 등 여러 가지가 있다[3].

이러한 에이전트 중 전자상거래 시스템에서 가장 저렴한 가격의 물품을 찾아 주는 에이전트는 이동에이전트라야 한다. 이동 에이전트를 개발하기 위한 도구로는 IBM의 Aglets[4], General Magic의 Telescript[5, 6], ObjectSpace의 Voyager[7,8] 등이 있다. 본 논문에서는 Aglets를 염두에 두고 시스템을 설계한다.

2. 설계 방법

본 전자상거래 시스템 설계의 기준에 대하여 클래스 설계 기준과 클래스 배당시 고려할 사항에 대하여 나누어 고찰한다.

클래스 설계의 목표는 정확성과 확장성이다. 정확성이란 시스템이 의도한 대로 정확하게 작동하는 것을 의미하며 이를 위하여 구현된 시스템의 검증이 용이해야 한다. 시스템 검증이 용이하려면 모듈들이 서로 가능한 독립적이어야 한다. 따라서, 클래스들은 자신에 속한 데이터와 관련된 정보를 자신의 내부에 숨기고 작동에 필요한 최소한의 정보만 외부에 노출시켜야 한다.

본 mobile agent 시스템에서 정보를 어떤 호스트로부터 다른 원거리 호스트로 전송하는 방법에는 agent 자체에 데이터 멤버로 첨부시켜 agent 이동과 함께 옮아가게 하는 방법과 mobile agent 사이에 메시지 전송으로 정보를 전송하는 방법의 두 가지 방법이 있다. 정보 숨김 지침에 따르면 전자의 방법을 최대로 사용해야 한다. 즉, 호스트간에 정보를

교환해야 한다는 시스템 요구 사항이 있으면 이 정보와 관련된 mobile agent를 설계하고 구현해야 하며, 실제 정보의 교환을 가능한 mobile agent에 의해서만 수행될 수 있어야 한다.

Mobile agent 시스템 설계에서 클래스 설계 지침에 이어 두 번째로 중요한 것은 mobile agent를 구성하는 클래스들을 어디에 위치시키는 가 하는 것이다. Mobile agent 시스템은 다수의 agent로 구성되며 이들은 여러 호스트를 옮아 다니면서 수행되고, 수행 중에 agent들간에 원거리 통신을 통하여 상호작용 함으로 클래스를 어느 호스트에 위치시키는 가에 따라 시스템 성능이 좌우될 뿐 아니라 정확성 검증과 유지 보수의 용이성에도 영향을 미친다.

클래스의 위치는 "생성지에 위치"하기, "수행지에 위치"하기, "서버에 위치"하기, 등 세 가지 유형이 존재한다. 생성지란 agent가 create된 호스트를 말하며, 수행지란 agent가 dispatch 되는 호스트를 말한다. Mobile agent 시스템을 설계할 때에는 이들 세 가지 유형을 잘 조화시켜서 가장 효율적인 시스템이 되도록 한다. 각 시스템의 장단점은 다음과 같다.

"생성지에 위치"시키는 유형에서는 mobile agent를 create 할 때 그 호스트에 있는 클래스들만을 사용한다. 따라서, 이 agent가 dispatch되면 이들 클래스들도 수행지로 옮아 가게된다. 이러한 유형은 여러 개의 수행지에서 똑 같은 작업이 수행 될 때 유용하다. 이렇게 되려면 수행되는 작업이 수행지의 환경에 의존적이면 안 된다. 즉, mobile agent가 수행 host의 file을 access 해야 한 다면 "생성지 위치"를 채택하면 안 된다.

"수행지에 위치"시키는 유형에서는 agent를 create하기 위하여 수행지로부터 클래스를 갖고 와야 한다. 따라서, agent를 dispatch할 때 클래스들도 옮아가야 할 필요는 없다. 이러한 유형은 수행 작업은 수행지에 의존적이지만 생성지와 수행지간의 인터페이스는 동일 할 때 유용하다. 예를 들어, 생성지에서 정보 수집을 위하여 다수의 agent를 서로 다른 host로 보낸다면 각 host에서 사용하는 데이터베이스 관리 시스템을 비롯하여 환경이 서로 다르므로 "수행지 위치" 유형이 적당하다. 각 호스트마다 환경이 다름으로 구현된 클래스는 서로 다르더라도 사용 방법인 인터페이스는 서로 같게 할 수 있다. 따라서, "수행지 위치" 유형이 유용하다. 반대로, 똑 같은 agent가 여러 host로 파견되어야 한다면 이 유형은 사용할 수 없다.

끝으로 "서버에 위치" 유형에서는 생성지나 수행지에서 사용할 클래스들을 서버에 위치시킨다. 따라서, 생성지에서 agent를 create 할 때에도 서버로부터 클래스를 load 해야 하고, 수행지에 agent가 파견될 때에도 수행지로 클래스가 load 되어야 한다. 이러한 유형은 다수의 생성지와 수행지에서 사용될 수 있는 library 클래스나 다수의 생성지에 의하여 공용될 고객 mobile agent 클래스에 유용하다. 또한 생성지와 수행지가 지정되어 있지 않은 경우에도 이러한 유형이 유용하다. 다른 유형에 대한 단점은 통신량이 더 많아진다는 것이다.

이상을 고려한 본 system 설계 방법은 다음과 같다. 첫째, 시스템에서 교환되는 정보의 유형을 파악하여 이들 각각을 agent로 설계한다. 둘째, 클래스가 수행할 작업이 수행지의 환경에 독립적이면 "생성지에 위치" 유형을 채택한다. 셋째, 클래스가 수행할 작업이 수행지의 환경에 의존적이면 "수행지에 위치" 유형을 채택한다. 넷째, 클래스 라이브러리나 고객 mobile agent 클래스는 "서버에 위치" 유형을 채택한다.

3. 전자상거래 모델

전자상거래에 관련된 개체에는 그림1에 보이는 바와 같이 고객, 중개인 그리고 상인이 있다. 상인은 물품을 진열하고, 재고와 고객을 관리하며, 고객과 상거래 하는 기능을 구비한다. 따라서, 상인은 자신의 데이터베이스를 운용해야한다.

고객은 상인으로부터 직접 물품을 구입할 수도 있지만 보통 중개인을 통하여 최저가의 물품을 구입하고자 희망한다. 중개인은 몇 명의 고정된 단골 상인으로부터 물품을 조달 받을 뿐만 아니라 다른 중개인에게도 물품 조달을 요청한다. 고객은 어느 중개인과도 거래가 가능하다.

고객이 물품을 구입하는 시나리오는 다음과 같다. 그림1의 "1 menu"와 같이 고객이 임의의 중개인과 접속하면 중개인은 취급 품목의 메뉴를 고객에게 보여준다. 이 중개인을 다른 중개인과 구분하기 위하여 중개인1이라 하자. 메뉴는 고객의 편의를 돕기 위한 수단일 뿐, 반드시 메뉴에 등재된 물품만이 구입 가능하다는 의미는 아니다. 메뉴에 등재된 물품이든 아니든 고객은 메뉴를 참조하여 특정 물품에 대한 견적을 중개인에게 요청한다 (2. 견적요청).

중개인은 자신의 단골 상인에게 견적을 요청하

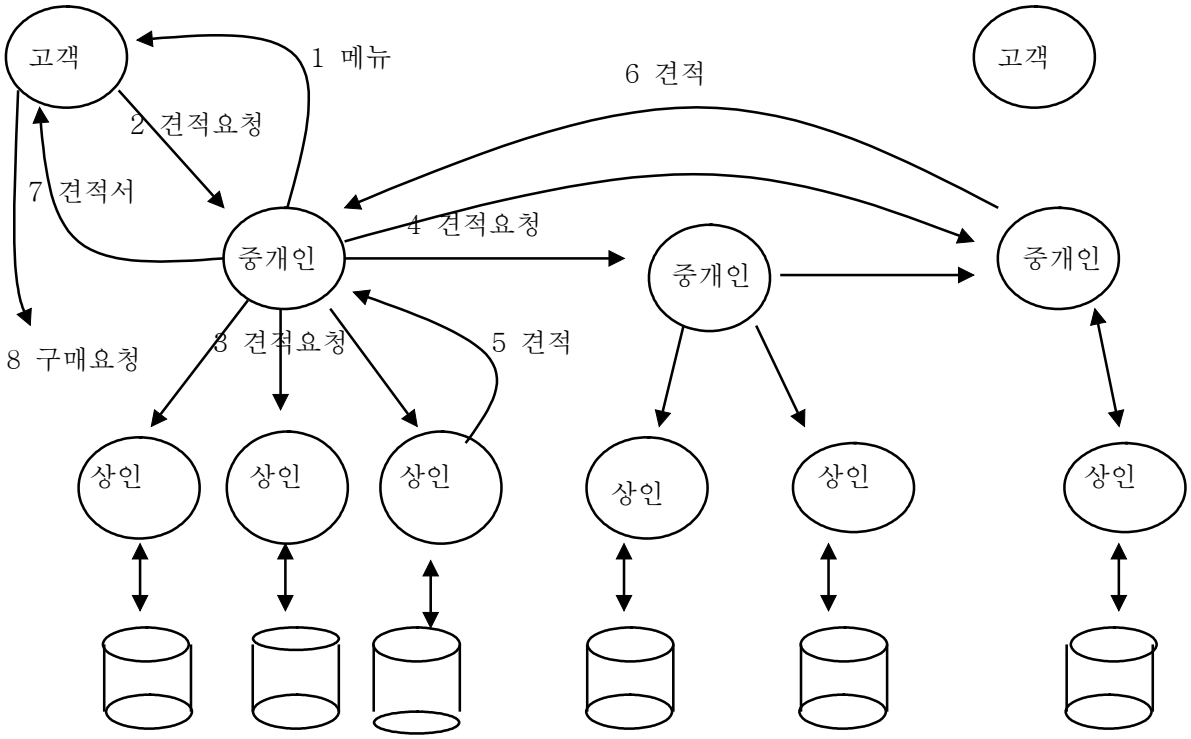

그림 1 전자상거래 모델

고, 또한 다른 중개인들에게도 견적을 요청한다 (3 견적요청과 4 견적요청). 견적의 요청을 받은 중개인들은 자신의 단골 상인들에게 견적을 요청한다 (3. 견적요청). 견적을 요청 받은 상인들은 자신의 재고관리 데이터베이스를 참조하여 견적서를 만들어 견적을 요청한 중개인에게 회신한다 (5 견적). 이때, 재고가 없는 상인은 그러한 물품을 취급하지 않는다고 회신한다. 견적서에 본 견적서를 작성한 상인이 누구인지 명기됨은 물론이다.

견적을 받은 중개인은 견적서를 검토하여 제일 좋은 견적서를 선택한다. 이때, 제일 좋은 견적서는 유일하지 않을 수 있다. 즉, 고객이 요구한 사양을 만족하는 품목이 다수일 때 이러한 품목을 모두 수용한다. 그러나 사양이 동일하면서 가격에 차이가 있는 경우에는 저가인 경우만 선택한다. 중개인은 선택된 견적서를 중개인1에게 전송한다 (그림1의 6 견적).

중개인1은 모든 견적서를 취합하여 또 다시 제일 좋은 것을 선택한 다음, 고객에게 보낸다 (그림1의 7 견적서). 고객은 견적서들을 검토하여 새로운 사양의 물품에 대한 견적을 요청할 수 있다. 그러면 지금까지의 작업이 다시 반복된다. 견적서 중 만족할 만한 사양이 있으면 고객은 견적서에 명시된 상인에게 구매 요청을 한다.

본 논문은 mobile agent와 관련된 부분만 다룬다. 따라서, 고객이 상인을 방문하여 물품의 대금을 지불하면, 상인이 물품을 탁송하고, 고객 관리를 위하여 수취여부를 확인하는 등의 실제 거래에 대한 모델은 본 논문에서는 고려하지 않는다.

4. 시스템 설계

그림1에서 보는 바와 같이 본 시스템은 고객 interface, menu 관리자, 고객의 견적 요청자, 중개인, 상인에게 견적 요청자, 중개인에게 견적 요청자, 상인 견적, 중개인 견적, 상인, 견적서 등으로 구성된다.

고객 interface는 고객의 요구에 의하여 고객 사이트에서 수행된다. 그럼으로 고객 interface를 구성하는 class는 server에 위치하여 불특정 다수인 고객들의 요구에 응하여 주어야 한다. 고객 interface는 무엇보다 먼저 메뉴를 출력하여 주어야 함으로 첫 단계에서 menu 관리자를 create 하여 중개인1 사이트로 dispatch한다. 메뉴관리자로부터 메뉴가 전송되면 고객 interface는 “고객의 견적 요청자”를 생성하여 사고 싶은 품목 정보와 함께 중개인1 사이트로 dispatch 한다. "고객의 견적 요청자“로부터 견적서가 전송되면 고객의 의지에 따라 다시 반복되든지 종료한다.

고객 interface에서 create된 “menu 관리자”는 중개인1 사이트에서 수행된다. 중개인1 사이트에는

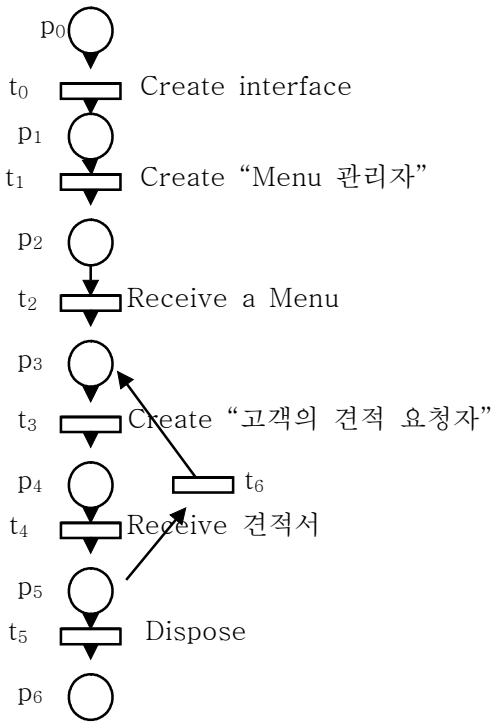

그림 2 고객 Interface의 Petri net 모형

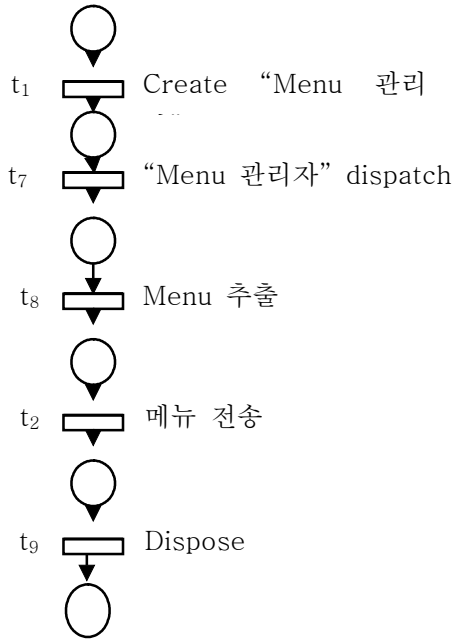

그림 3 Menu 관리자의 Petri net 모형

중개인이 관리하는 품목 데이터베이스가 있다.

“menu 관리자”는 중개인으로부터 품목 리스트를 받아 “고객 interface”에게 전송하고 종료한다. “menu 관리자”는 중개인1 사이트에서 수행되며 중개인 사이트의 데이터베이스를 사용해야 함으로, “menu 관리자”를 구성하는 class들은 중개인 사이

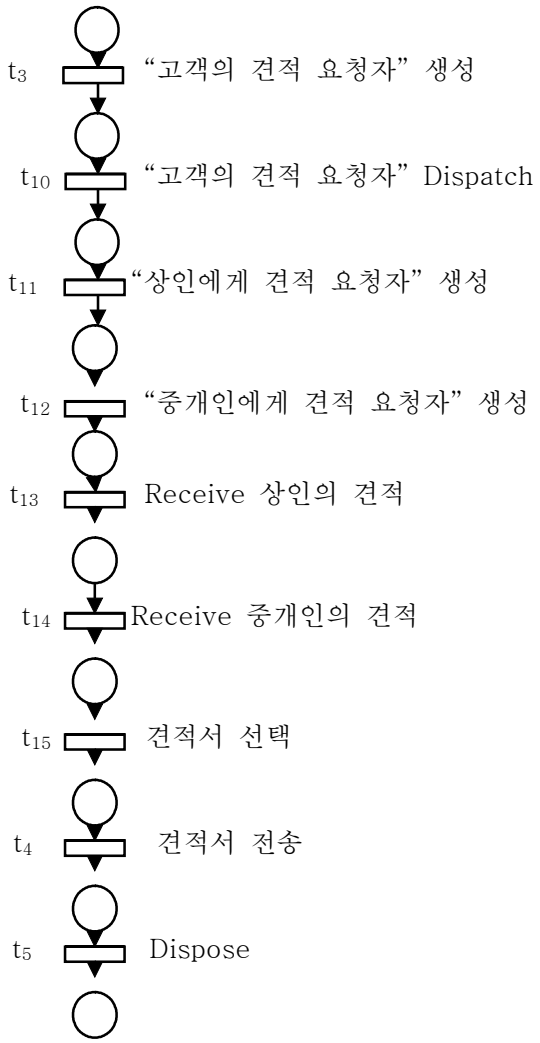

그림 4. “고객의 견적 요청자”의 Petri net 모형

트에 위치하는 것이 바람직하다.

“고객의 견적 요청자”는 중개인1 사이트의 DB를 참조하여 모든 단골 상인에 대하여 “상인에게 견적 요청자”를 생성하여 각각의 상인 사이트로 dispatch 한다. 그리고 다른 중개인들에게도 “중개인에게 견적 요청자”를 생성하여 물품 정보와 함께 dispatch 한다. “고객의 견적 요청자”는 상인과 다른 중개인으로부터 견적서를 전송 받으면 이를 분석하여 제일 바람직한 견적을 고객 interface로 전송한다.

“상인에게 견적 요청자”는 “고객의 견적 요청자”에 의하여 생성되어 상인 사이트로 dispatch된다. “상인에게 견적 요청자”는 상인 사이트에서 DB를 참조하여 고객이 원하는 물품의 견적을 작성하여 “고객의 견적 요청자”에게 전송한다.

“중개인에게 견적 요청자”는 “고객의 견적 요청자”에 의하여 생성되어 품목 정보와 함께 중개인으

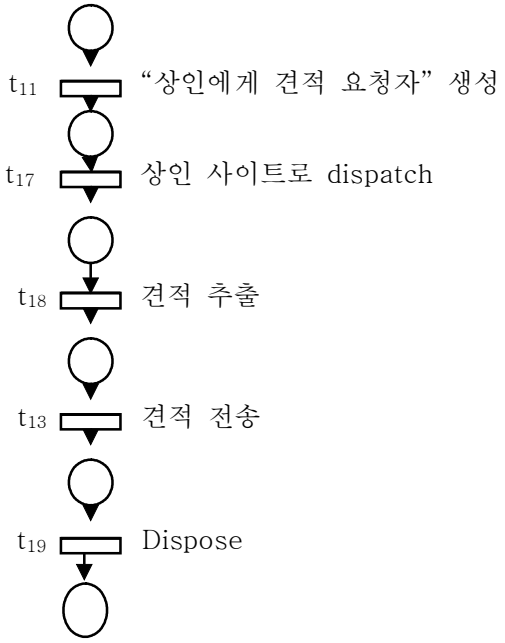

그림 5. “상인에게 견적 요청자”의 Petri net 모형

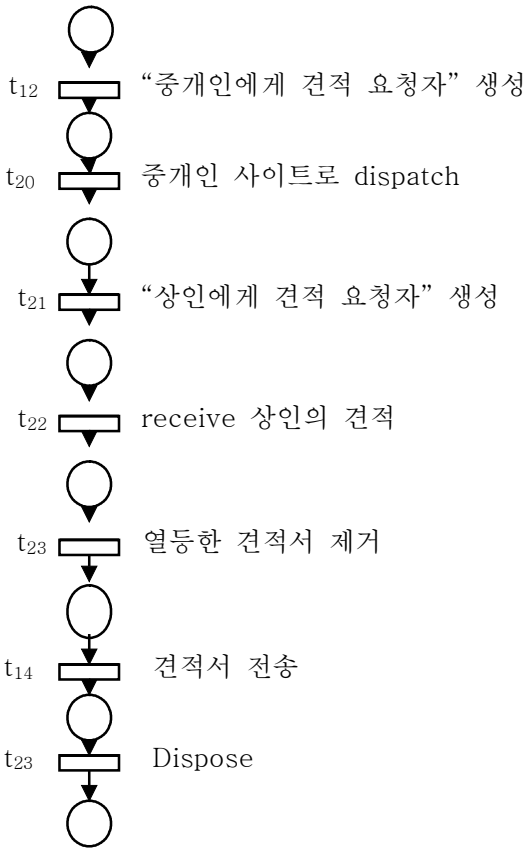

그림 6. “중개인에게 견적 요청자”의 Petri net 모형

로 dispatch된다. Dispatch된 “중개인에게 견적 요청자”는 중개인 사이트의 DB를 참조하여 모든 단골 상인에 대하여 “상인에게 견적 요청자”를 생성하여 dispatch한다.

고객 interface의 Petri net 모형은 그림2와 같다. 그림2에서 t_1은 “Menu 관리자”의 생성 트랜지션과 같은 것이며, t_2는 “Menu 관리자”의 “send a menu”와 같은 것이다. 마찬가지 방법으로 t_3와 t_4는 “고객의 견적 요청자”와 공유된다.

“Menu 관리자”는 “고객 Interface”에 의하여 생성되며 중개인1 사이트로 dispatch된다. 또한 중개인1 사이트의 DB를 참조함으로 “Menu 관리자”는 중개인1 사이트에 정의되어 있는 것이 바람직하다. “Menu 관리자”의 Petri net 모형은 그림3과 같다. 그림3에서 t_1은 “고객 Interface”의 t_1과 같은 트랜지션이며, t_2는 “고객 Interface”의 t_2 (receive menu)와 동일한 트랜지션이다.

“고객의 견적 요청자”는 “고객 Interface”가 생성하여 중개인1에게 dispatch된다. 이것 역시 중개인 사이트에 존재하는 DB를 참조하여 단골 상인 사이트와 다른 중개인에 대한 정보를 이용하여야 함으로 중개인1 사이트에 정의되어 있는 것이 바람직하다. “고객의 견적 요청자”의 Petri net 모형은 그림4와 같다. Petri net 모형에서 t_3는 “고객 Interface”의 t_3와 같다. t_{11}과 t_{13}은 “상인에게 견적 요청자”와 공유되는 트랜지션이다. 비슷한 이유로 t_{12}와 t_{14}은 “중개인에게 견적 요청자”와 공유되는 트랜지션이다. 트랜지션 t_4는 “고객 Interface”와 공유된다.

“상인에게 견적 요청자”는 “고객의 견적 요청자”에 의하여 생성되어 상인 사이트로 dispatch 된다. 상인 사이트의 DB로부터 사양을 충족하는 모든 물품에 대한 견적서를 작성하여 “고객의 견적 요청자”에게 전송한다. 따라서 본 agent는 상인 사이트에 정의되어 있는 것이 바람직하다. “상인에게 견적 요청자”의 Petri net 모형은 그림 5와 같다. 이 Petri net 모형에서 t_{11}과 t_{13}은 “고객의 견적 요청자”와 공유된다.

“중개인에게 견적 요청자”는 “고객의 견적 요청자”에 의하여 생성되어 다른 중개인 사이트로 dispatch 된다. 중개인 사이트의 DB로부터 단골 상인의 명단을 추출하여 이들 각각에 대하여 “상인에게 견적 요청자”를 생성한다. “상인에게 견적 요청자”로부터 상인의 견적을 받으면 그 중 제일 좋은 것을 선택하여 “고객의 견적 요청자”에게 전송한다. 따라서 본 agent는 중개인 사이트에 정의되어 있는 것이 바람직하다. “중개인에게 견적 요청자”의

Petri net 모형은 그림 6과 같다. 이 Petri net 모형에서 t_{12}과 t_{14}는 "고객의 견적 요청자"와 공유된다.

그림2부터 그림6까지의 Petri net 모형을 통합하여 그림1에 보이는 전자상거래 모델을 나타내는 Petri net 모델을 작성할 수 있다. 또한 통합된 Petri net을 분석하여 모델의 일관성을 검증하는 방법을 고안하는 것도 가능하다.

5. 향후 연구 계획

본 논문은 Mobile agent를 이용한 시스템을 설계하는 예를 보였다. 설계의 배경은 Aglet을 염두에 두었다. 그럼으로 다음 단계로 설계된 시스템을 실제로 구현하는 것이 시급하다. 이를 바탕으로 Mobile Agent 시스템 구축 도구를 개발하는 것이 연구의 최종 목표이다.

본 논문은 말로 묘사된 사용자가 구축하고자 하는 시스템으로부터 필요한 Agent들을 분별하여 내고, 각 Agent들이 수행해야 할 과업을 정확히 분별하기 위하여 Petri net으로 표현하여 보았다. 다양한 예제에 대하여 이러한 작업을 수행하여 봄으로써 Agent를 분별하여 내는 지식을 유추하고, 이를 정형화하여 말로 기술된 시스템 사양으로부터 필요한 agent들을 자동으로 분별해 내는 프로그램을 개발하는 것이 가능하리라 본다.

또한 각 agent가 할 일이 Petri net으로 표현되면 이를 바탕으로 프로그래머가 작성할 프로그램의 골격을 제공해 주는 소프트웨어의 개발도 가능하리라 본다. 이러한 소프트웨어는 프로그래머의 생산성을 향상시킨다.

6. 결론

본 논문은 Mobile agent 시스템의 예로 전자상거래 시스템을 선택하여 전자상거래 소프트웨어 시스템의 설계 예를 보였다. 프로그램 의뢰자는 원하는 전자상거래 시스템의 사양을 말로 설명한다. 시스템 설계의 첫 단계로 시스템 사양으로부터 필요한 사이트를 분별해 낸다.

본 mobile 시스템은 Aglet을 바탕으로 구현한다고 가정하였다. 따라서 개체지향 시스템 설계 지침이 그대로 적용된다. 정보 hiding 원칙을 지키기 위하여 사이트간에 통신되는 정보를 분별하여 이들을 중심으로 필요한 agent를 분별하는 것이 본 시스템 설계의 두 번째 단계이다.

Agent 생성과 dispatch시 요구되는 정보 전송량은 시스템의 성능에 큰 영향을 미친다. 세 번째 단계에서 mobile 시스템의 전송량을 최소화시키도록 Agent의 정의를 배치하였다.

네 번째 단계에서 각 agent의 과업을 Petri net으로 명확하게 규명하였다. 이들을 통합하고 통합된 Petri net을 분석하여 설계된 시스템의 일관성을 검증하고 통신량을 분석하는 것이 향후 연구 과제이다.

나아가서 말로 설명된 시스템 사양으로부터 필요한 agent를 자동으로 판별하고, 각 agent의 과업을 표현하는 Petri net을 자동으로 구축하며, 이를 바탕으로 코드를 자동으로 생성하는 시스템을 구축하는 것이 향후 연구 과제이다.

참고문헌

[1] 최중민, "온라인 전자상거래에서의 멀티에이젼트와 이동에이젼트 응용," http://cse.hanyang.ac.kr

[2] S. Russel and P. Norvig, *"Artificial Intelligence a Modern Approach,"* Prentice Hall 1995년

[3] 박수현, 김태석, "소프트웨어 에이전트 분류 및 기술 동향," 한국멀티미디어학회지 제3권 2호, 1999년 10월 pp. 3-16.

[4] http://www.trl.ibm.co.jp/aglets

[5] J. White, *Telescript Technology, The Foundation of the Electronic Marketplace*, General Magic White Paper, 1994.

[6] J. White, "Mobile Agents White Paper," http://www.genmagic.com/agents, 1996

[7] ObjectSpace, "Voyager: ORB3.0 Developer Guide," ObjectSpace Inc. http://www.objectspace.com/Voyager/, 1999.

[8] G. Glass, "The ObjectSpace Voyager Universal ORB," ObjectSpace Inc. http://www.objectspace.com/Voyager/, 1999.