멀티 에이전트 설계 시스템 (I)

지식기반설계는 전문가로부터 고도로 특화된 다양한 지식을 완전히 입수하여 설계를 종합화(synthesize)하고 개선하는 것이다. 지식기반설계 시스템은 이러한 지식을 찾는 과정을 자동화한다. 그러나 표준과 기술 그리고 역동적인 시장의 끊임없는 변화는 전문적 지식과 그 지식을 적용하는 과정에서 높은 수준의 적응성을 필요로 한다. 다양하고 고도로 전문화되며 급변하는 기술과 지식 덕분에 멀티 에이전트 파라다임이 특히 지식 기반시스템에 적절하게 되었다.

연구자들이 멀티 에이전트 기법을 설계영역에 적용할 경우에 생기는 어려움은 다양한 플랫폼에서 사용되는 이종 에이전트 간의 작동을 지원하거나, 설계과정을 조화롭게 하거나, 또는 그 과정에서 발생한 분쟁을 처리하는 것들이다. 이 논문은 프로젝트와 예제를 통하여 멀티 에이전트 시스템에 있어서의 최신기술을 조사하고, 제안된 멀티 에이전트 설계 프로그램의 필요조건과 목적을 도출하는데 있어서 발생하는 문제점을 제기하고자 한다. 이 조사 결과에 대한 필요조건을 고찰해 보면 우리는 이 기술이 매우 복잡한 분야와 유망한 기술이 조화된 상태라는 것을 볼 수 있다. 합작된 전문에이전트의 조직은 설계에 필요한 유연성과 한계를 제공하지만, 효과적인 합작을 위한 기술을 개발하는 것이 멀티 에이전트 설계 기법이 새로운 기술이 되는 주요 인자가 될 것이다.

1. 에이전트 기술

소프트웨어 에이전트는 최신의 전산 기술을 말하며 아직까지도 정의가 제대로 이루어지지 않은 분야이다. 실제로 에이전트를 연구하는 연구자들 사이에 가장 맹렬히 토론되는 내용은 바로 에이전트의 정의에 대한 것이다. 에이전트가 무엇인가에 대한 명확한 정의는 아직 없지만 이 단어는 매개체라는 말로 써 받아들여지고 있으며 또한 모든 문제에 대한 해결책으로 제안되고 있다. 이렇게 주목받는 것은 마치 과거에 AI가 그랬듯이 비현실적인 기대를 불러일으킨다는 측면에서 바람직하지 않은 면도 있다.

에이전트와 관련 있는 프로젝트의 예나 또 에이전트 기법을 탐구하는 회사나 학교의 연구집단에 대한 Web site를 볼 수 있는데 그것을 AgentWeb(University of Maryland at Baltimore의 Tim Finin이 관리한다. http://www.cs.umbc.edu/agents)이나 AI in Design Webliography(Worcester Polytechnic Institute의 David Brown에 의해 관리된다. http://www.cs.wpi.edu./Research/aidg/AIinD-hotlist.html)을 통하여 볼 수 있다. 에이전트 연구는 대체적으로 공동 에이전트시스템, 개인용 조수 또는 이동 에이전트로 분류할 수 있다.

⊙ 공동 에이전트 시스템(Collaborative Agent Systems)

에이전트 기술 중에서 설계에 특히 중요한 분야는 이종적이고 반자동화된 지식기반 소프트웨어 요소를 통합된 어플리케이션으로 만드는 것이다. 수치계산에 기초한 설계분야에서의 현재의 연구와 개발은 주로 설계과정의 특정 부분을 자동화시키는데 그 주안점을 둔다. 수많은 설계 기법(예를 들어 솔리드 모델링, 시뮬레이션, 시각화, 그리고 해석 도구 등)은 광범위하게 사용되지만 이러한 기법은 수작업에 의해 통합되는 환경에서는 '고립된 자동화(Islands of Automation)'로 묘사될 것이다.

이와 같은 '고립된 자동화'의 개념을 벗어나 완전히 통합되고 조합된 애플리케이션으로 발전하는 것이 현재 연구의 방향이다. 이러한 연구는 이종적인 기계, 연산용 컴퓨터, 프로그래밍 언어 그리고 데이터와 프로세스의 표현방식간에 정보가 매끄럽게 유통될 수 있는 기준과 기법을 개발하는 것이다. 공동 에이전트 시스템에 대한 현재의 연구는 협력적인 분산 문제 해결, 컴퓨터를 이용한 협력작업, 동시기법

엔지니어링, 제한 조건에 기반을 둔 추론과 같이 다양한 영역에 기초하고 있다.

⊙ 개인 조수(Personal Assistants)

개인 조수란 사용자가 정보나 작업을 운용하도록 해주는 소프트웨어 에이전트를 말한다. 여기에는 사용자가 어떤 작업을 수행하도록 하는 적응 인터페이스가 포함되는데 그 작업이란 스케쥴링, 구매 또는 다른 일에서 사용자를 자동적으로 대변하는 소프트웨어 요소, 사용자에게 필요한 정보를 찾아주는 Web상의 검색엔진, 그리고 메시지를 발송하거나 여과하는 메일 에이전트 등이다.

⊙ 이동 에이전트(Mobile Agents)

이동 에이전트는 외부의 소스로부터 정보를 수집하여 지역 호스트에서 처리하는 대신에, 정보를 처리하기 위해서 자신이 원거리에 위치한 호스트로 움직인다. 이동되어야 하는 데이터의 양이 추론을 수행하는데 필요한 코드의 양보다 더 크거나, 또는 지역 호스트에 할당된 프로세싱 시간의 양이 제한될 경우에는 위와 같은 과정이 분명히 유리하게 된다. 보안성을 보장하거나 연산의 효율을 제고하는 것과 같은 몇 가지 문제점이 있지만 에이전트의 이동 가능성은 미래의 컴퓨터 기술에 큰 영향을 미칠 것이다.

본 논문에서는 에이전트에 대하여 매우 포괄적인 정의를 내린다. 즉, 인간이나 다른 소프트웨어 요소와 비교적 자동화된 소프트웨어 요소나 설계 과정에 어떤 정보를 제공할 수 있는 것은 그 요소가 사용자를 대변하거나, 사용자를 도와주거나, 혹은 설계 작업을 자동적으로 수행하던지 간에 에이전트로서 가능성이 있다.

그러나 이 기사의 본래의 목적은 에이전트에 대한 것이 아니라 멀티 에이전트 시스템에 관한 것이다.

2. 멀티 에이전트 설계 시스템

소프트웨어 에이전트를 통합하는 설계 어플리케이션이 멀티 에이전트 설계 시스템(MADS)이다. 복잡한 설계 어플리케이션은 일반적으로 자동화된 소프트웨어 요소와 의사를 결정하는 인간을 결합함으로써, 설계 과정에 있어서 인간과 컴퓨터 모두에게 없어서는 안될 역할을 수행한다. 사적 보조 에이전트는 인간이 설계의 루프안에 있게끔 해주며 공동 에이전트 시스템 기술은 다양한 소스와 정보 그리고 추론을 연결하는 매력적인 틀을 제공한다. 물론 어려운 점은 매력적인 틀로부터 효과적인 어플리케이션으로 발전하는 것이다. MADS를 사용하면 어떤 점이 가장 이로울 것인가? 설계 도메인의 어떤 면이 특히 멀티 에이전트의 적용 예로 적당한가? MADS 기술에 있어서 각광받는 것 중 하나는 그를 통하여 소프트웨어의 유지나 재사용이 향상될 가능성이 있다는 것이다.

⊙ 소프트웨어의 유지와 재사용

소프트웨어 유지는 설계와 같이 복잡하고 동적인 영역에 속하는 어려운 문제이다. 전통적으로 소프트웨어의 유지는 어플리케이션의 전 수명 비용의 50%에서 80%를 차지한다. 또 공학 설계에 있어서 새로운 설계 소프트웨어는 항상 개발되고 있으며, 소프트웨어간의 상호연관작업은 점점 더 늘고 있고, 신제품의 시장 진출에 걸리는 시간을 단축해야 한다는 압력이 커지고 있다.

멀티 에이전트 시스템은 이와 같이 급변하는 영역에 매우 적합하다. 각각의 에이전트는 특별한 기능을 갖고 있으며 일반적으로 잘 정의된 어플리케이션 프로그램 인터페이스(API)가 있는 준 자동화된 개체로 간주된다(API는 프로그래머로 하여금 그 내부 구조에 대한 지식없이 어플리케이션을 사용할 수 있도록, 어플리케이션 프로그램의 입출력 조건이나 특징에 대해 설명한 명세표라 할 수 있다). 에이전트간의 경계에는 잘 정의되고 구분된 인터페이스가 존재하므로 에이전트를 개선하거나 교체할 때에 이러한 변화가 다른 에이전트의 내부 구조에 전달되지 않는다. 그러므로 MADS 기법을 통한 가장 큰 이점은 어플리케이션이 설계 프로젝트, 시장, 그리고 새로운 기술과 도구에 대응하여 성장하고 변화할 때에 시간이 지남에 따라 그 기술이 축적된다는 것이다. 또한 암호화 기술을 이용한 어플리케이션 인터페이스가 개발되고 있다.[1-3]

⊙ 정보 흐름의 관리

이종 소프트웨어간에 데이터와 정보처리 과정을 공유하는 능력과 상호연관작업(Interoperability)은 매우 활발한 연구가 수행되는 부문이다. 이 논문에서

소프트웨어 요소는 어떤 객체나 에이전트를 뜻하게 된다. 상호연관작업을 하는데는 몇 가지 단계의 기능이 요구된다. 첫 번째 단계에서는 프로그래밍 언어와 운영체제의 경계를 넘어서는 정보의 표현과 프로세스의 표현에 대한 기준을 개발한다. 이러한 접근 방법은 PDES/STEP, 정보교환형식(KIF), 그리고 Jinxin Lin, Marx Fox, Taner Bilgic에 의해서 제안된 공학적 요구조건 표현의 개발과 같은 데이터 표현의 표준화 작업을 통하여 더 분명해진다.[4]

이 것 이외에도 상호연관작업시에 메시지가 어디로 전달되고 메시지의 수신자가 어떤 행동을 해야하는가를 명확히 해 줄 수 있는 언어가 필요하다. 인터페이스 명세 언어(예를 들어 Corba의 Interface Definition Language) 또는 에이전트 통신 언어(예를 들어 Knowledge Query나 Manipulation Language) 등을 사용함으로써 통신가능한 표현의 기준을 제공하게 된다.[5]

통신 언어를 완성시킨 후 전달의 상호연관작업이 제대로 이해되려면 시스템 차원에서의 지원이 있어야하며, 이것은 메시지가 그 목적지로 제대로 가거나 소프트웨어 요소가 요구되는 작업을 제대로 하는가를 파악하는 것이다. 그러므로 대부분의 경우에 특정한 정보를 어떤 시스템에 의해 제공된 서비스로 보내던가 혹은 특성화된 시스템 요소로 보내게 된다.

에이전트가 상호 교류해야 하는 다른 에이전트의 정확한 주소값을 모르고서도 통신이 가능한 몇 가지 방법이 있다. 예를 들어 분산 개체 매니저인 Corba, OLE, OpenDoc는 수신자의 물리적 위치를 모르더라도 메시지를 보낼 수 있게 한다.[6]

데이터 표준, 통신 표준, 그리고 시스템 서비스 방법은 상호보완적이며 또 이 셋을 합하여 상호연관작업이 가능하게 된다. 현재 최신의 기술은 어떤 어플리케이션에서의 상호연관작업은 구축 가능하지만 그러기 위하여 보통 어플리케이션에 고유하고 그리고 유일한 방법만이 존재한다고 요약해서 말할 수 있다.

따라서 설계자들이 원하는 목표는 고유한 표현 방식, 인터페이스, 그리고 서비스를 통하여 이러한 과정이 보다 쉽고 덜 특성화되는 것이다.

⊙ 설계과정의 관리

MADS를 구성하는데 필요한 작업의 대부분은 에이전트간에 정보와 데이터를 공유하는데 그 초점을 맞추고 있다. 그러나 에이전트 사이의 협조도 양질의 설계를 하고 자원을 효과적으로 사용하는데 중요한 것이다. MADS는 설계과정에서 전략적인 추론을 지원하며 이 능력은 최고의 설계를 효과적으로 할 수 있게 한다.

전통적인 설계기법은 설계작업을 미리 계획된 틀에 맞추어 연속적으로 수행되는 부작업으로 나누는 연속적인 설계 모델링을 이용했다. 그러나 연속적인 설계는 유연하지 못하여 가끔 비용과 시간이 많이 소모되는 단계를 되풀이하는 경우가 있었다. 이러한 문제 때문에 설계상에서 검토되어야 할 대안의 숫자가 제한되게 되었다. 이에 반하여 동시기법 설계는 설계과정의 초기단계에서 하부단계에 발생할 수 있는 제한 조건이나 모순을 파악하거나, 또는 부작업을 가능한 한 정도까지 동시에 수행함으로써 위와 같은 문제점을 제기하려 한다.

그러나 동시기법 설계를 효과적으로 수행하기 위해서는 적당한 부작업 구조를 결정하거나 그 부작업을 에이전트에게 부과하거나, 또는 가능한 정도까지 제한 조건 전파나 동시기법을 둘다 지원하는 작업을 완수할 수 있게 계획을 세우고 개선하는 제어 전문기술이 많이 필요하게 된다. 협조에 관계되는 전문기술은 다음의 사항을 보장한다.

- 에이전트간의 정보 교류가 원활하다.
- 필요한 때와 장소에서 결과를 입수할 수 있다.
- 최소의 노력을 들여서 최고 양질의 설계를 달성할 수 있도록 부작업사이의 관계를 언제라도 이용할 수 있다.

⊙ 모순의 관리

MADS가 갖는 또 하나의 장점은 새로운 기법이 소개되거나 예상치 못했던 조건이 발생할 때에 대응하여 에이전트를 바꾸거나 재구성할 수 있다는 것이다. 그러나 이러한 유연성마저도 지식공학을 통해서 발생한 모순을 없애지는 못한다는 것이다. 이것은 에이전트를 설계하거나 실행시키는 단계에서 모든 다른 가능성 있는 에이전트와 완벽히 호환되게 만들 수 없다는 것이다. 따라서 동적인 모순관리가 MADS에 고유한 필요 조건이다. 모순이 발생할 가능성은 다음과 같은 경우이다.

- 부정확한 정보나 지식

• 모순된 가정을 만드는 불완전한 지식
• 모순된 신념
• 대안으로 제시된 설계에 대한 상이한 판단기준
• 상이한 추론과정
• 상이한 상호 정보교환 조건

모순관리 어플리케이션은 개개의 에이전트나 전체 시스템의 제어기관이나 독립된 모순 관리 에이전트에 있거나 또는 사용자 스스로인 경우도 있다.

모순 회피: 모순을 가장 잘 관리하는 것은 바로 그것을 피하는 것이다. 그렇다고 해서 에이전트가 모순을 피하기 위하여 정당한 조건을 무시하라는 것은 아니다. 그러나 모순은 전역적인 조망이 결여될 때 발생한다. 설계의 초기단계에서 지역제한 조건, 우선순위, 그리고 선호도에 대한 정보를 공유함으로써 전역적으로도 유용하고 지역적으로도 적절한 결정을 반복적인 설계 사이클이 필요없이 에이전트가 내리도록 도와주게 된다.[7] 모순의 회피는 동시기법 공학의 근본적인 원리이다.

모순 분류: Mark Klein은 모순을 1급의 현상으로 다루면서 모순의 종류에 대한 분류법을 제안했다.[8] Klein의 Cooperative Design Engine은 LAN에 대한 전문지식과 모순에 대한 해결책을 제시하는 에이전트를 이용하여 LAN을 설계하는 것이다. 모순이 발생하면 에이전트는 그 형식을 결정하고 그 형식과 관계된 제안을 이용하여 모순을 해결하는 계획을 세우며 이 계획 중의 하나를 선택하여 수행한다.

타협: 이 영역의 기법은 교섭, 재구성, 제한 조건 완화, 조정, 그리고 중재 등이다. 타협은 주로 대립된 경우에 적용되나 협조적인 경우에도 발생할 수 있다. 예를 들어, Keith Werkman은 Design Fabricator Interpreter 시스템을 만들었는데 여기서 에이전트는 설계, 생산, 그리고 조립 전문가인데 철제의 빔이나 칼럼으로 구성된 건축 연결 부재의 집합으로부터 인간이 적당한 것을 설계하도록 도와주는 것이다.[9] DFI 시스템은 에이전트 사이의 모순을 해결하는데 조정이나 중재의 기법을 이용한다.

감사의 글

이 연구는 미 국방성의 Advanced Research Projects Agency(ARPA) Manufacturing Automation and Design Engineering(MADE) RaDeo No. 70NANB6H0074 협약에 의해 부분적인 지원을 받았다.

References

1. M. R. Cutkosky *et al.*, "PACT: An Experiment in Integrating Concurrent Engineering Systems," *Computer*, **26**(1): 28-38 (1993).
2. S. E. Lander, S. M. Staley, and D. D. Corkill, "Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools," Concurrent Engineering: Research and Applications, special issue on the application of multiagent systems to concurrent engineering, **4**(1): 59-72 (1996).
3. C. M. Pancerella, A. J. Hazelton, and H. R. Frost, "An Autonomous Agent for On-Machine Acceptance of Machined Components," Proc. Modeling, Simulation, and Control Technologies for Manufacturing, Int'l Soc. for Optical Eng., Bellingham, Wash., 1995, pp. 146-159.
4. J. Lin, M. S. Fox, and T. Bilgic, "A Requirement Ontology for Engineering Design," Concurrent Engineering: Research and Applications, **4**(3): 279-291 (1996).
5. T. Finin *et al.*, "KQML as an Agent Communication Language," Proc. Third Int'l Conf. Information and Knowledge Management, ACM Press, New York, 1994.
6. T. J. Mowbray and R. Zahavi, The Essential CORBA: Systems Integration Using Distributed Objects, JohnWiley & Sons, New York, 1995.
7. S. E. Lander and V. R. Lesser, "Sharing Meta-Information to Guide Cooperative Search among Heterogeneous Reusable Agents," IEEE Trans. Knowledge and Data Engineering, to appear in 1997.
8. M. Klein, "Supporting Conflict Resolution in Cooperative Design Systems," IEEE Trans. Systems, Man, and Cybernetics, **21**(6): 1379-1390 (1991).
9. K. J. Werkman, "Multiple Agent Cooperative Design Evaluation Using Negotiation," Artificial Intelligence in Design, J. S. Gero and F. Sudweeks, eds., Kluwer Academic Publishers, London, 1992.
10. T. P. Darr and W. P. Birmingham, "Automated Design for Concurrent Engineering," IEEE Expert, **9**(5): 35-42 (1994).
11. W. Shen and J. P. Barthes, "DIDE: A Multiagent

Environment for Engineering Design," Proc. First Int'l Conf. Multiagent Sysems, AAAI Press, Menlo Park, Calif., 1995, pp. 344-351.

12. M. R. Genereth and S. P. Ketchpel, "Software Agents," Comm. ACM. special issue on intelligent agents, 37(7): 48-53 (1994).

Susan E. Lander는 Amherst의 Blackboard Technology의 책임 연구원이다. 그녀의 관심연구분야는 협조적인 에이전트 기반시스템, 에이전트 기반 공학 설계, 분산검색, 지식공유, 그리고 에이전트시스템의 모순관리에 혹관기법을 사용하는 것이다. 그녀는 University of Massachusetts, Amherst의 전산학과에서 학사, 석사, 박사를 취득했다. 그녀는 AAAI, ACM, IEEE의 회원이다. 관심있는 독자들은 다음의 주소를 통하여 Lander와 연결될 수 있다. Blackboard Technology, Amherst, MA 01002; lander@bbtech.com; http://www.bbtech.com/lander/

≪IEEE Expert Intelligent System & Their Applications Vol. 12, No. 2, April 1997≫

본 기사는 경희대의 김영진 편집위원이 "IEEE Expert Intelligent Systems & Their Application"에서 발췌하였으며 출판사인 IEEE Computer Society의 연락처는 다음과 같다.

• http://www.computer.org/pubs/expert/expert.htm