[135] M. Wooldridge and N. Jennings. The cooperative problem solving process: A formal model. Technical report, Department of Computing, Manchester Metropolitan University, Chester St., Manchester M1 5GD, UK, 1994.

[136] M. Wooldridge and N. R. Jennings. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94)*, pages 403–417, Lake Quinalt, WA, July 1994.

[137] M. Wooldridge and N. R. Jennings. Towards a theory of cooperative problem solving. In *Proceedings of the Sixth European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-94)*, pages 15–26, August 1994.

[138] M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 1–39. Springer-Verlag: Heidelberg, Germany, January 1995.

[125] P. Wavish and M. Graham. Roles, skills, and behaviour: a situated action approach to organising systems of interacting agents. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 371–385. Springer-Verlag: Heidelberg, Germany, January 1995.

[126] D. Weerasooriya, A. Rao, and K. Ramamohanarao. Design of a concurrent agent-oriented language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 386–402. Springer-Verlag: Heidelberg, Germany, January 1995.

[127] E. Werner. A formal computational semantics and pragmatics of speech acts. In *Proceedings COLING–88*, pages 744–749, 1988.

[128] E. Werner. Toward a theory of communication and cooperation for multiagent planning. In M. Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 129–144. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

[129] E. Werner. Cooperating agents: A unified theory of communication and social structure. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 3–36. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[130] E. Werner. What can agents do together: A semantics of co-operative ability. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 694–701, Stockholm, Sweden, 1990.

[131] M. Wooldridge. An approach to reasoning about multi-agent systems. In *Proceedings of the Third UK Workshop on Belief Representation and Agent Architecture (BRAA-92)*, University of Durham, UK, June 1992.

[132] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992. (Also available as Technical Report MMU–DOC–94–01, Department of Computing, Manchester Metropolitan University, Chester St., Manchester, UK).

[133] M. Wooldridge. Coherent social action. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)*, pages 279–283, Amsterdam, The Netherlands, 1994.

[134] M. Wooldridge and M. Fisher. A first-order branching time logic of multi-agent systems. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 234–238, Vienna, Austria, 1992.

[114] M. P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications (LNAI Volume 799)*. Springer-Verlag: Heidelberg, Germany, 1994.

[115] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[116] Stephen Smith and Peng Ow. The use of multiple problem decompositions in time constrained planning tasks. In *Proceedings 9th International Joint conference on Artificial Intelligence*, August 95.

[117] Barry Smyth and Padraig Cunningham. A blackboard based, recursive, case-based reasoning system for software development. In Kevin Ryan and Richard Sutcliffe, editors, *AI and Cognitive Science '92*, pages 179–194. Springer-Verlag, 92.

[118] K. Sycara. *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods*. PhD thesis, School of Information and Computer Science Georgia Institute of Technology, Atlanta, GA, 1987.

[119] S. Tamura, Y. Okataju, and T. Seki. Development of intellectual distributed processing system. In *IFAC 10th Triennial World Congress*, pages 39–44. IFAC, 87.

[120] S. R. Thomas. *PLACA, an Agent Oriented Programming Language*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, August 1993. (Available as technical report STAN–CS–93–1487).

[121] S. R. Thomas. The PLACA agent programming language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 355–369. Springer-Verlag: Heidelberg, Germany, January 1995.

[122] S. R. Thomas, Y. Shoham, A. Schwartz, and S. Kraus. Preliminary thoughts on an agent description language. *International Journal of Intelligent Systems*, 6:497–508, 1991.

[123] M. V. Nagendra Prasad Time Oates and Vctor R. Lesser. Networked information retrieval as distributed problem solving. In *Third International Conference on Information and Knowledge Management, Workshop on Intelligent Information Agents*, Gaithersburg, Maryland, December 1994.

[124] Tony Veale and Padraig Cunningham. Competitive hypothesis resolution in twig, a blackboard-driven text understanding system. In *ECAI92 10th European Conference on Artificial Intelligence*, pages 561–563. ECAI, August 92.

[101] S. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[102] S.A.R. Scrivener, E.A. Edmonds, and L.A. Thomas. Improving image generation and structuring using raster graphics. In *Proceedings of CAD78*, pages 223–229. IPC Science and Technology Press, 78.

[103] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press: Cambridge, England, 1969.

[104] J. R. Searle. *Expression and Meaning.* Cambridge University Press: Cambridge, England, 1979.

[105] J. R. Searle. Collective intentions and actions. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 401–416. The MIT Press: Cambridge, MA, 1990.

[106] Y. Shoham. Time for action: on the relation between time, knowledge and action. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 954–959, Detroit, MI, 1989.

[107] Y. Shoham. Agent-oriented programming. Technical Report STAN–CS–1335–90, Computer Science Department, Stanford University, Stanford, CA 94305, 1990.

[108] Y. Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings of the National Conference on Artificial Intelligence*, pages 704–709, 1991.

[109] Y. Shoham and D. McDermott. Problems in formal temporal reasoning. *Artificial Intelligence*, 36, 1988.

[110] Y. Shoham and Y. Moses. Belief as defeasible knowledge. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1168–1172, Detroit, MI, 1989.

[111] M. P. Singh. Group intentions. In *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence (IWDAI-90)*, 1990.

[112] M. P. Singh. Group ability and structure. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-90)*, pages 127–146. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[113] M. P. Singh. Towards a formal theory of communication for multi-agent systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 69–74, Sydney, Australia, 1991.

[89] C. L. Mason and R. R. Johnson. DATMS: A framework for distributed assumption based reasoning. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 293–318. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[90] James Mayfield, Yannis Labrou, and Tim Finin. Desiderata for agent communication languages. In *Procedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford University, March 1995.

[91] F. G. M^{c}Cabe and K. L. Clark. `April` — agent process interaction language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 324–340. Springer-Verlag: Heidelberg, Germany, January 1995.

[92] Stegen Ketchpel Michael Genesereth. Software agents. *Communications of the ACM*, 37(7), July 1994.

[93] Marvin Minsky. *Society of Mind*. Simon and Schuster, 1985.

[94] H. Penny Nii. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53, Summer 1986.

[95] Peter W. Pachowicz. Semi-autonomous evolution of object models for adaptive object recognition. *SMC*, 24(8), August 94.

[96] Kevin M. Passino. Intelligent control for autonomous systems. *IEEE Spectrum*, pages 55–62, June 95.

[97] Francois Charpillet Philippe Lalanda and Jean-Paul Haton. A real time blackboard based architecture. In *ECAI92 10th European Conference on Artificial Intelligence*, pages 262–266. ECAI, August 92.

[98] A. Poggi. DAISY: An object-oriented system for distributed artificial intelligence. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 341–354. Springer-Verlag: Heidelberg, Germany, January 1995.

[99] Alex Repenning. Bending icons: Syntactic and semantic transformations of icons. *IEEE Symposium on Visual Languages*, 1994.

[100] J. S. Rosenschein, M. Ginsberg, and M. R. Genesereth. Cooperation without communication. In *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

[76] N. R. Jennings, L. Z. Varga, R. P. Aarnts, J. Fuchs, and P. Skarek. Transforming standalone expert systems into a community of cooperating agents. *International Journal of Engineering Applications of Artificial Intelligence*, 6(4):317–331, 1993.

[77] M. J. Katz and J. S. Rosenschein. Plans for multiple agents. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 197–228. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[78] K. Konolige. *A Deduction Model of Belief*. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, San Mateo, CA., 1986.

[79] H Laasri, B. Maitre, T. Mondot, F. Charpillet, and J.P. Haton. Atome: A blackboard architectrue with temopral and hypothetical reasoning. In *ECAI88 Proceedings of the 8th European Conference on Artificial Intelligence*, pages 5–10. ECAI, August 88.

[80] Yannis Labrou and Tim Finin. A semantics approach for kqml - a general purpose communication language for software agents. In *Procedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, Nonvember 1994.

[81] Donna M. Lamberti and John M. Prager. Advice-giving using reason: An intelligent assistant for interactive computing. In *Seventh IEEE Conference on AI Applications*, pages 428–434. IEEE, IEEE Computer Society Press, 91.

[82] D. M. Lane, M. J. Chantler, E. W. Robertson, and A. G. McFadzean. A distributed problem solving architecture for knowledge based vision. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 433–462. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[83] P. Maes. The dynamics of action selection. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 991–997, Detroit, MI, 1989.

[84] P. Maes, editor. *Designing Autonomous Agents*. The MIT Press: Cambridge, MA, 1990.

[85] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents*, pages 49–70. The MIT Press: Cambridge, MA, 1990.

[86] P. Maes. The agent network architecture (ANA). *SIGART Bulletin*, 2(4):115–120, 1991.

[87] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), July 1994.

[88] Christian Martin and Klemens Waldhor. Basar- a blackboard based software architecture. In Bernd Radig, editor, *ECAI88 proceedings of the 8th European Conference on Artificial Intelligence*, pages 2–4, August 88.

[63] Autonomous Agents Group. Autonomous agents group project list. World Wide Web Home Page, 1995.

[64] B. Hayes-Roth. Architectural foundations for real-time performance in intelligent agents. *The Journal of Real-Time Systems*, 2:99–125, 1990.

[65] B. Hayes-Roth. An integrated architecture for intelligent agents. *SIGART Bulletin*, 2(4):79–81, 1991.

[66] B. Hayes-Roth, K. Pfleger, P. Lalanda, P. Morignot, and M. Balabanovic. A domain-specific software architecture for adaptive intelligent systems. *IEEE Transactions on Software Engineering*, 21(4):288–301, 1995.

[67] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, July 1985. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 503–540, Morgan Kaufmann, 1988.).

[68] Barbara Hayes-Roth, Micheal Hewett, Richard Washington, Rattikorn Hewett, and Adam Seiver. Distributing intelligence within an individual. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence Volume II*, volume 2 of *Research Notes in Artificial Intelligence*, pages 385–412. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[69] J. Hintikka. *Knowledge and Belief.* Cornell University Press: Ithaca, NY, 1962.

[70] Colin Hopkins. Deplan:enabling agents to produce plans that achieve cooperative problem solving. In Bernd Radig, editor, *ECAI88*, pages 421–425, 88.

[71] H. Raghav Rao James C. Moore. Multi-agent resource allocation: An incomplete information perspective. *SMC*, 24(8):1208–1218, August 94.

[72] N. R. Jennings. On being responsible. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 93–102. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[73] N. R. Jennings. Towards a cooperation knowledge level for collaborative problem solving. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 224–228, Vienna, Austria, 1992.

[74] N. R. Jennings. Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving. *Journal of Intelligent and Cooperative Information Systems*, 2(3):289–318, 1993.

[75] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 74(2), 1995. (To appear).

[52] Tim Finin, Jay Weber, Gio Weiderhold, Michael Geneserth, Richard Fritzson, Donald McKay, James McGuire, Richard Pelavin, Stuart Shapiro, and Chris Beck. Draft spceification of the kqml agent-communication language. Technical report, The DARPA Knowledge Sharing Intiative, External Interfaces Working Group, June 1993.

[53] M. Fisher. A survey of Concurrent MetateM — the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag: Heidelberg, Germany, July 1994.

[54] M. Fisher. Representing and executing agent-based systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 307–323. Springer-Verlag: Heidelberg, Germany, January 1995.

[55] M. Fisher and H. Barringer. Concurrent MetateM Processes — a language for distributed AI. In *European Simulation Multiconference*, Copenhagen, Denmark, June 1991.

[56] M. Fisher and M. Wooldridge. Specifying and verifying distributed intelligent systems. In M. Filgueiras and L. Damas, editors, *Progress in Artificial Intelligence — Sixth Portuguese Conference on Artificial Intelligence (LNAI Volume 727)*, pages 13–28. Springer-Verlag: Heidelberg, Germany, October 1993.

[57] Bill fulkerson. Alife in manufacturing. Draft - Convention Chaos Network Presentation, September 94.

[58] J. R. Galliers. A strategic framework for multi-agent cooperative dialogue. In *Proceedings of the Eighth European Conference on Artificial Intelligence (ECAI-88)*, pages 415–420, Munich, Federal Republic of Germany, 1988.

[59] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, July 1994.

[60] Genesereth, M.R., Ginsberg, M.L. and J. S. Rosenschein. Cooperation without communication. In *Proceedings of The fifth national conference of Artificial Intelligence*, Philadelphia, PA., 1986.

[61] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 677–682, Seattle, WA, 1987.

[62] Khaled Ghedira and Gerard Verfaillie. A multi-agent model for the resource allocation problem: A reactive approach. In B. Neumann, editor, *ECAI92*, pages 252–254. John Wiley & Sons, 1992.

[42] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, November 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 268–284, Morgan Kaufmann, 1988.).

[43] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperation through communication in a distributed problem solving network. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 2, pages 29–58. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

[44] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, March 1989.

[45] E. A. Edmonds, L. Candy, R. Jones, and B. Soufi. Support for collaborative design: Agents and emergence. *Communications of the ACM*, 37(7):41–47, July 1994.

[46] O. Etzioni and D. Weld. The first law of robotics. In O. Etzioni, editor, *Software Agents — Papers from the 1994 Spring Symposium (Technical Report SS–94–03)*, pages 17–23. AAAI Press, March 1994.

[47] I. A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Clare Hall, University of Cambridge, UK, November 1992. (Also available as Technical Report No. 273, University of Cambridge Computer Laboratory).

[48] I. A. Ferguson. Towards an architecture for adaptive, rational, mobile agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 249–262. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[49] I. A. Ferguson. Integrated control and coordinated behaviour: A case for agent models. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 203–218. Springer-Verlag: Heidelberg, Germany, January 1995.

[50] Tim Finin, Rich Fritzson, and Don McKay. A language and protocol to support intelligent agent interoperability. In *Proceedings of the CE and CALS*, Washington, June 1992.

[51] Tim Finin, Rich Fritzson, Don McKay, and Robin McEntire. Kqml as an agent communication language. In *Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland, December 1994.

[32] B. Dunin-Keplicz and J. Treur. Compositional formal specification of multi-agent systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 102–117. Springer-Verlag: Heidelberg, Germany, January 1995.

[33] E. H. Durfee. *A Unified Approach to Dynamic Coordination: Planning Actions and Interactions in a Distributed Problem Solving Network*. PhD thesis, COINS, University of Massachusetts, Amherst, MA., 1987.

[34] Edmund H. Durfee. An approach to cooperation: Planning and communication in a distributed problem solving network. Technical Report 86-09, Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, March 1986.

[35] Edmund H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, 1988.

[36] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a blackboard-based problem solver. In *Proceedings of the National Conference on Artificial Intelligence*, pages 58–64, Philadelphia, PA, August 1986.

[37] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 875–883, Milan, Italy, August 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 285–293, Morgan Kaufmann, 1988.).

[38] Edmund H. Durfee and Victor R. Lesser. Negotiation through partial global planning. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[39] Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence Volume II*, volume 2 of *Research Notes in Artificial Intelligence*, pages 229–243. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[40] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Increasing coherence in a distributed problem solving network. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1025–1030, Los Angeles, CA, August 1985.

[41] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Making coherent communication decisions in a distributed problem solving network. In *Proceedings of the Workshop on AI and Distributed Problem Solving*, pages 59–75. National Academy of Sciences, May 1985.

[19] H.-D. Burkhard. Agent-oriented programming for open systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 291–306. Springer-Verlag: Heidelberg, Germany, January 1995.

[20] M. H. Coen. SodaBot: A software agent environment and construction system. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994.

[21] P. R. Cohen and H. J. Levesque. Persistence, intention, and commitment. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 33–70. The MIT Press: Cambridge, MA, 1990.

[22] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. The MIT Press: Cambridge, MA, 1990.

[23] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *International Conference on Multi-Agent Systems*, pages 65–72, San Francisco, California, June 1995.

[24] P. R. Cohen, J. Morgan, and M. E. Pollack, editors. *Intentions in Communication.* The MIT Press: Cambridge, MA, 1990.

[25] Winton H E Davies and Peter Edwards. Agent-k: An integration of aop and kqml. Integrating with KQML.

[26] Randall Davis. Models of problem solving: Why cooperate? *SIGART Newsletter*, (73):42–43, October 1980.

[27] D. C. Dennett. *Brainstorms.* The MIT Press: Cambridge, MA, 1978.

[28] D. C. Dennett. *The Intentional Stance.* The MIT Press: Cambridge, MA, 1987.

[29] Joachim Diederich, Elizabeth M. Gurrie, and Markus Wasserchaff. Computers that learn vs. users that learn: Experiments with adaptive e-mail agents. German National Research Center for Computer Science.

[30] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[31] A. Drogoul and J. Ferber. Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, pages 3–23. Springer-Verlag: Heidelberg, Germany, 1994.

[7] AMR. Best-in-class manufacturing information management strategies and trends. *AMR CIM Strategies*, II(6), June 94.

[8] J. L. Austin. *How to Do Things With Words.* Oxford University Press: Oxford, England, 1962.

[9] Albert Baker, Van Parunak, and Perry Alexander. Simcim: A prototype demonstration platform for autonomous agents. Industrial Technologies Institute, 93.

[10] M. Barbuceanu and M. S. Fox. Cool: A language for describing coordination in multi-agent systems. In *International Conference on Multi-Agent Systems*, pages 17–24, San Francisco, California, June 1995.

[11] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: A framework for programming in temporal logic. In *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS Volume 430)*, pages 94–129. Springer-Verlag: Heidelberg, Germany, June 1989.

[12] H. Barringer, M. Fisher, D. Gabbay, and A. Hunter. Meta-reasoning in executable temporal logic. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91).* Morgan Kaufmann Publishers: San Mateo, CA, April 1991.

[13] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the Thirteenth ACM Symposium on the Principles of Programming Languages*, pages 173–183, 1986.

[14] Nina M. Berry and Soundar R. T. Kumara. Intelligent software agents for non-traditionsl information sharing environments. In *Third International Conference on Information and Knowledge Management, Workshop on Intelligent Information Agents*, Gaithersburg, Maryland, December 1994.

[15] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[16] R. A. Brooks. Elephants don't play chess. In P. Maes, editor, *Designing Autonomous Agents*, pages 3–15. The MIT Press: Cambridge, MA, 1990.

[17] R. A. Brooks. Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991.

[18] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

vironment agents are the providers and consumers of goods in the electronic marketplace such as the Internet. Telescript agents are mobile software processes that can pack state information and move across a network [137].

McCabe and Clark created the Agent PRocess Interaction Language (APRIL) was developed as part of an ESPRIT project - Imageing. This is not a language for multi-agent systems but is oriented to the implementation of multi-agent systems. APRIL is a process oriented symbolic language that allows programmer to define process and for processes to communicate with each other in a distributed environment [90].

Poggi created DAISY, which is a system for for distributed Artificial Intelligence Systems and Algorithms. DAISY consists of two levels. An object level, or infrastructure, that takes care of the low level communication. The top level is the agent level. This level consists of high level communication based on speech act theory. Using this system for modeling an airline reservation and a manufacturing plant scenario are given by Poggi [97].

Wavish and Graham developed an Agent Behavior Language (ABLE). ABLE represented the behaviors and the world the agent interacted using symbolic logic and it also used some rules of behavior, called licenses, to dictate the agents behavior. Later, they created a production rule language called RTA. This language can be broken into three levels. The top level consists of agents performing roles . The middle layer gives the skills each agent needs to perform a role. The bottom layer lists the behaviors the agents need to have a skill. RTA is essentially a fast version of ABLE which can be used for simulating and controlling the behavior of a single agent unlike ABLE which was for simulating and controlling the behavior of a society of agents [124].

References

[1] Alex Repenning ad Wayne Citrin. Agentsheets: Applying grid-based spatial reasoning to human-computer interaction. *IEEE Workshop on Visual Languages*, 1993.

[2] G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. The MIT Press: Cambridge, MA, 1986.

[3] J. F. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning*. Morgan Kaufmann Publishers: San Mateo, CA, 1990.

[4] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[5] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[6] James F. Allen and Johannes A. Koomen. Planning using a temporal world model. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 741–747, August 1983.

Later in a DARPA project, the Knowledge Query and Manipulation Language was created to handle information passing between intelligent agents [50]. Agents using KQML can communicate attitudes, beliefs, desires as well as conduct queries, subscribe and offer information to other agents in the system. It is based on speech act theory. Each message in KQML is a performative, meaning each message is intended to perform some action. The semantics and syntactics are laid out by Finin [52]

KQML works through communication facilitator agents which maintain a registry of service names, forwards messages to named services, routes messages based on content and provides information for its clients. Experiments with KQML for integrated planning and scheduling for military transportation with each portion (planning agent, scheduler, knowledge base and case based reasoning tool written in different languages are described by Finin [51] On the bottom level of the an agent communication language is the vocabulary that the language uses. The next level is the Knowledge Interchange Format (KIF). This is a pre-fix version of the language of first-order predicate calculus. On top of the KIF level is KQML. KQML supports dialogue between agents on top of the underlying KIF statements [91]. A semantic description for KQML based on ideas from speech act theory and the use of cognitive states such as know, want and intend of the agents as part of the semantics are presented by Labrou [79].

Cohen identifies ambiquity and vagueness in some KQML performatives, misidentified performatives and missing performatives. He also made suggestions for resolution of these problems with KQML [?].

Coordination Language (COOL) deals with the coordination between agents in communication and not just the information content which KIF and KQML concentrates on. Barbuceanu and Fox list three levels of Agent Interaction. Information content, which KIF handles, intentions, which KQML handles, and conventions. COOL handles agents sharing conventions which enable them to coordinate [11].

Experiments with KQML for integrated planning and scheduling for military transportation with each portion (planning agent, scheduler, knowledge base and case based reasoning tool written in different languages are described by Finin in [51] Mayfield, Labrou and Finin suggest criteria that an Agent Communication Language should be evaluated on: form, content, semantics, implementation, networking, environment, and reliability. How KQML stands up versus this criteria is listed in their [89].

Davies and Edwards integrated AOP and KQML creating agents that are created using AOP that can communicate with each other using KQML. This implementation was called Agent-K. The Agent-K agents could communicate with other Agent-K agents and KQML agents across a network. The major drawback of building on AOP was that it limited agent communication to Prolog based agents. Details on the integration problems of AOP and KQML are listed by Davies and Edwards in [25].

There are also many other Agent Communication Languages other than KQML and KQML based Languaes. Below is a list of other Agent Communication Languages and a description of each.

Telescript by General Magic is an environment for building agent societies. In this en-

Weerasooriya, Rao and Ramamohanarao developed AgentSpeak. AgentSpeak Agents are organized into families, which offer certain services to other agents. Also, each agent has its own database. AgentSpeak is similar to Agent0 and PLACA but the mental state of AgentSpeak agents consist of services and relations while the mental state of the Agent0 and PLACA agents consist of beliefs, desires and intentions [125].

Burkhard looked at the work of Shoham and Hewitt and other ideas on Agent Oriented Programming and suggested how these ideas can be applied to use Agent Oriented Programming for Open Systems. Open Systems are defined as having continuous availability, extensibility, De-central control, asynchrony, inconsistent information and arms length relationships [20]. The conclusion was that a single Agent Oriented Language would restrict the use of Agent Oriented Programming because different language are better for certain applications [20].

There are also many other programming languages that have been used to program agents not based on Shoham's Agent0. Below is a list of these languages and a description of each.

Rosenschein and Kaelbing provided a method of specifying the behavior of an agent in terms of a state machine or *situated automata* [100] . The model is then compiled down (possibly to hardware) for efficient execution of the behavior.

Coen believes agents should be able to be created by specifying their abstract behavior. He introduced Sodabot which is a software agent environment and construction system. Sodabot consists of an agent operating system, the Sodabot Agent Programming Language with agents design based on human-level descriptions of agent activity, a graphical user interface and agents distributed across the Internet [21].

Repenning developed Agentsheets, which is a visual programming environment that bridges the gap of traditional high level graphical building blocks in visual program to the low level of conventional programming. Agentsheets are composed of many individual agents with definable behavior. This work was based on the theory of cellular automata because Agentsheets define the global behavior as a collection of simple local relations between agents [2, 98].

5.2 Agent Communication Language

Programming agents involve a deep understanding of the internals of an agent. An Agent Communication Language deals with how these agents communicate with each other.

Rosenschein and Genesereth laid a foundation for information passing strategies between intelligent agents as well as analyzed the effect of passing false information [60].

More groundwork for an Agent Communication Language was laid by Galliers. He describes a theoretical framework for systems to negotiate and resolve differences [58]. Galliers work was based on the formal theory of rational interaction by Cohen and Levesque and work done by Rosenschein and Genesereth work concerning changing and unpredictable environments that agents need to interact [99]. Galliers's framework suggests agents need to understand the nature of the conflict and dialogue actions to manipulate the mental states of other agents in order to resolve conflicts [58].

Dunin-Keplitz and Treur [32] describe a frame work for construction agent architectures by interconnecting sub-task components with semantic links.

5 Languages

Agent Languages facilitate agent creation and communication. Four issues have to be resolved in order to support these: how agents send and receive messages, what the individual messages mean, how agents structure conversations and how to connect systems of agents.

5.1 Agent Programming Languages

Agent programming languages evolved out of concurrent object languages. A good background on concurrent programming languages is given by Agha and Hewitt [3]. The earliest concurrent object framework was Hewitt's Actor Model. An actor, in the Actor Model, is a computational agent which responds to messages sent to it by other computational agents in the system. An actor can pass messages to itself or other actors, create more actors and specify replacement actors to handle subsequent messages [3]. The Actor Model and other concurrent object frameworks are still used to program agents.

An extension of concurrent object framework for programming agents was Shoham's Agent Oriented Programming [106]. In Agent Oriented Programming, each object is an agent. This type of programming would have a system for defining the mental state of each agent with some representation of time, an interpreted language for programming agents and a process for compiling agents in low level executable code [106].

Agent Oriented Programming built off of James Allen work on temporal logic and representing knowledge of temporal intervals [5, 7, 6]. Shoham applied this temporal reasoning to Artificial Intelligence entities and also uncovered some problems with formal temporal reasoning [108]. He later published on the relationship between time, knowledge and action [105] and how to represent beliefs from knowledge with some representation of time [109].

Shoham later developed the Agent0 programming language using the Agent Oriented Programming Paradigm. In Agent0 an agent is defined in terms of a set of capabilities, a set of initial beliefs and commitments and commitment rules. [107] Shoham then worked with Thomas on expanding the Agent0 language to allow the agents to plan and communicate using high level goals [121]. Later Thomas developed the Planning Communicating Agents (PLACA) Language that would do such [119, 120].

Fisher discovered that Agent0 and PLACA used logic to define the mental states of agents but neither Agent0 nor PLACA truly executed the logic. Building off of his work with Wooldridge and his work with Barringer on programming in temporal logic and executing temporal logic [12, 13] and building off of Barringer's work on concurrent models using temporal logic [14], Fisher developed Concurrent MetateM [55, 53]. This language has concurrently executing agents that communicate by broadcasting messages to the other agents in the system. Each agent is defined using temporal logic specification. This specification is used to direct how an agent behaves [56, 54].

solves the task or breaks it into sub-tasks, which it in turn offers to other contractors. When no more decomposition is necessary, the solutions are assembled into the overall solution for the original problem (or task). Contract net uses a common inter-node language for communication purposes.

Smith [115] proposes the use of multiple problem decompositions in time constrained planning tasks. The application of this method is to aid in the synthesis of a collection of plans accomplishing distinct goals. Smith argues that the ability to reason from both resource-based and agent-based perspectives is essential to appropriate consideration of all of the relevant constraints.

Tamura [118] suggests the development of an intellectual distributed processing system IDPS). The IDPS system elements find their roles autonomously in order to complete objectives. These elements are able to resolve conflicts among objectives without any supervisory element by changing information between one another.

Oates [122] suggests using networked information retrieval as distributed problem solving. He says that the task of information retrieval in a distributed setting can be viewed in general terms as either distributed processing or distributed problem solving, but specifically he argues that in complex networked environments information retrieval is by its very nature an instance of distributed problem solving.

Mason and Johnson [88] describe an architecture of an assumption based reasoning agent. This work is based on Truth Maintenance Systems [30].

4.2.3 Hierarchical Architectures

Hierarchical system architectures have different classes of agents working on tasks at different levels of abstraction. For example, tasks requiring real-time response would be handled by a reactive agent (an agent with a reactive agent-architecture) while a more complex task of determining future actions would be handled by a deliberative planning agent.

In the Intelligent Individual Architecture described by Barbara Hayes-Roth et.al. [65], the system consists of a hierarchy of agents, performing perception, action and reasoning functions under the supervision of a control agent.

Lane et.al. [81] describe a hierarchical architecture applied to vision recognition. The agents (called rational cells) work on more abstract vision tasks at higher levels in the hierarchy.

4.2.4 Mobil-Agent Frameworks

Mobil-agent frameworks provide an infrastructure for executable agents to be shipped across a network and executed on host machines. To facilitate portability over heterogeneous networks, the agents are often written in interpreted scripting languages. An evaluation of the benefits of mobile agent technology are discussed and IBM research report [**?**]. These system architectures are closely related to work done in distributed operating systems and heterogeneous process migration.

to solve the problem, but in unison, by writing their ideas on a common blackboard at the front of the classroom, they can solve the problem together.

Smyth [116] suggests using blackboards systems with dedicated control agents to guide the reasoning process. The goal of this system is to create a blackboard based, recursive reasoning system for software development. Smyth discusses the Deja Vu software, which is meant to general plant control software for controlling autonomous vehicles in a steel mill environment involved in the loading and unloading of metal coils during the milling process. Deja Vu's model consists of two key element: knowledge agents and panels (or blackboard panels).

Martin [87] proposes to use the blackboard paradigm to integrate standard and knowledge based applications. The architecture must support advanced user interfaces and parallel or distributed processing. The main components of Martin's architecture include a scheduler, a dialogue manager, and a communications manager. Martin succeeded in producing a prototype of the a;architecture, called BASCAR, which stands for BAsic Software ARchitecture. BASCAR uses the blackboard framework primarily for managing cooperation, interaction, and scheduling issues in complex software systems.

Maitre [78] discusses the use of a blackboard architecture with temporal and hypothetical reasoning. The reason for the selection of blackboards for this task was that they allow opportunistic efficiency while controlling multiple knowledge sources. Also, it allows for the easy integration of the temporal and hypothetical reasoning to deal with applications evolving in time, and manipulation noisy or errant information.

Veale [123] proposes a blackboard driven text understanding system called TWIG. TWIG uses the integration of both lexical and encyclopedic knowledge into a unified representation which affords a uniform hypothesis format, thus facilitating communication between various and diverse knowledge agents, all acting on the blackboard.

Lalanda [96] has proposed a real-time blackboard based architecture which is based on the multi-agent blackboard model. This model integrates planning capabilities, which facilitates real-time control by integrating opportunistic response to unanticipated events and carefully planned sequences of action.

Katz and Rosenschein describe the use of plans (directed acyclic graphs of actions) for agent coordination [76].

4.2.2 Distributed Control Architectures

The contract net protocol [114] was developed as a high level protocol for communication among the nodes in a distributed problem solver. Its purpose is to facilitate the distributed control of cooperative task execution with efficient node to node communication. This application consists of a variety of nodes which act as a manager, a contractor or both. The manager is responsible for monitoring the execution of a task, and the contractor is responsible for the actual execution of the task. When a problem is posed, it is decomposed into a number of tasks by the manager. These tasks are announced, and the contractors bid on the job. A contractor (node) is awarded the task based on its qualifications, and then either

action. The activities are arranged in a hierarchy that is used to govern which activities have control over the agents behavior at a given time.

In related work, Maes defines the *agent network architecture* [82, 83, 84, 85] as a way to determine which behaviors should be active in a reactive system. The behaviors are represented by *competence modules* and placed in a neural-network-like configuration and a spreading activation is used to determine the active behaviors.

4.1.3 Hybrid Agents

The Procedural Reasoning System (PRS) described by Georgeff and Lansky [61] combines planning with reactive behavior.

Ferguson's Touring Machines [48, 47, 49] contains reactive, planning and modeling layers. "Each control layer is designed to model the agent's world at a different level of abstraction and each is endowed with different task-oriented capabilities" [49].

Barbara Hayes-Roth has been involved in the development of several agent architectures [66, 67, 63, 64, 65].

4.2 System Architectures

Early Multi-agent System Architectures could be distinguished by the presence (or lack) of a centralized control mechanism [76]. There are additional of features of system architectures such as providing an infrastructure for agent mobility.

4.2.1 Centralized Control Architectures

In an agent-based system architecture with centralized control, there is some component of the system (possibly an agent) which facilitates cooperation among agents via some form of planning or negotiation. As defined by Durfee, "Negotiation is the term used in distributed problem-solving research to denote the process by which autonomous nodes coordinate their views of the world and act and interact to achieve their goals." [39] Durfee was involved in early work on coordinating distributed problem solvers [33, 40, 41, 34, 36, 43, 42, 38, 35, 37, 44].

The blackboard model originally arose from the HEARSAY-II speech understanding systems which were developed between 1971 and 1976 [93]. The HEARSAY-II project developed a speech recognition for queries made of a database. HASP, an application of ocean surveillance software for passive sonar, was next in the development scheme. These applications both were formative in the blackboard concept.

The basic blackboard model involves knowledge sources, which are independent entities with unique knowledge and experiences. The blackboard data structure is the global database where the problem solving state data is kept. Knowledge sources supply their knowledge to the blackboard to solve a problem. The idea of blackboard models owes its name partly to the concept of a group of professors (i.e. the knowledge sources) sitting in a classroom trying to solve a difficult problem. None of them has enough knowledge alone

The existence of these two levels of abstraction is not always made clear in the literature. They are related in the following manner: Agent architectures are often designed with the interface the agent provides to the system in mind. Therefore, certain agent architectures may be more suited to providing the interface for certain system architectures. The distinction between the two levels of architecture becomes confusing in the case where individual agents are implemented as agent-based systems.

4.1 Agent Architectures

Agents architectures can be classified along the following lines [137]

- *Deliberative* - the agents have an internal model of the world and perform symbolic reasoning. This work stems from the classical approaches to artificial intelligence.
- *Reactive* - the agents have no model of the world and perform no complex reasoning. This approach to agent implementation is described by Maes as "Behavior-Based AI" [85] because often hierarchies of basic behaviors are combined to form a system with high levels of functioning This work is a direct offshoot of Minsky's work in intelligent systems of agents 2.2.2.
- *Hybrid* - a combination of deliberative and reactive behavior. One method for implementing this type of agent is as a multi-agent system composed of deliberative and reactive agents.

4.1.1 Deliberative Agents

Dunin-Keplitz and Treur [32] describe a framework for constructing agent architectures at a high level of abstraction. The behavior of an agent is defined by interconnecting sub-task components via semantic links. This could also be viewed as a high-level agent programming language.

Jennings describes the GRATE* architecture to work within the Responsibility Model (see Section 2) which supports joint intentions. The architecture is broken into two layers: one to handle cooperation and control and the other to deal with the problem domain. The architecture appears to be modularized enough to be adapted to other cooperation mechanisms.

4.1.2 Reactive Agents

The motivation behind Brooks' subsumption architecture [16, 17, 19, 18] is to avoid the unrealistic assumption that an agent can make the abstractions about its environment that are necessary to provide input to symbolic (deliberative) reasoning. Instead of partitioning systems into perception and reasoning subsystems, he partitions in terms of *activities* or behavior producing systems. These activities are direct connections between perception and

game that allows a user who is technically inexperienced configure and run a job shop using agent-based techniques. This demonstration serves as an "intuition builder" which will potentially allow those in the manufacturing community to become more aware of the potential of agents.

3.6 Control Problem Solving Agents

A more general application of agents is in the area of control problem solving. Davis [26] works with agents implemented using "meta-rules" for retrieval, refinement, and execution of control in the control problem solving problem.

Passino [95] discusses the control problem in more depth, identifying three layers, including an execution layer, a coordination layer, and a management layer. The execution layer connects to the process under control with sensors, the coordination level tunes, schedules, supervises, and redesigns the algorithms, and the management level supervises the lower level functions and manages the interfaces with humans.

Hopkins [69] has outlined a method of cooperative problem solving, where an agent in a multi-agent environment can model the actions of the other agents. Since communications between agents is vital, in such environments the behavior of one will affect the decisions of the other agents. As a result, prediction of the behavior of other agents has been tried by Hopkins to make achievement of the individuals goals easier.

3.7 Object recognition agents

Object recognition agents are discussed by Pachowicz [94]for variable perceptual conditions. His approach assumes that the system has to recognize objects on separate images of a sequence and that the objects show the variability that they will have in appearances in the real world.

4 Architectures

Software architecture, in general, deals with the implementation of software systems as a collection of interconnected components (or modules). With regard to agent-based systems, there are two levels of software architecture that are of interest:

- *System Architectures* where (some of) the system components are agents interconnected by some communication mechanism. In this case, we are interested in the interface (outward appearance) of the agents, not their internal structure, i.e. the agents are black-boxes.

- *Agent Architectures* describe ways to implement the individual agents themselves. Here we, the components are the different different structures that represent the agent's knowledge, either implicitly or explicitly.

model ecological and sociological systems, the process of emergence is important. The agent implementation allows for the testing of hypotheses, building new theories, and integrating different partial theories from a variety of disciplines into a coherent framework.

3.5 Manufacturing Agents

Agents can be used to represent various objects on the shop floor, including machines, tools, raw materials, fixtures, and even workers. In the manufacturing arena, agents have several applications. Interface agents are being developed for workers in all parts of todays factories, from shop-floor personnel to foremen to engineers to order entry personnel. In addition to such interfaces, scheduling agents are a topic of much interest.

Fulkerson [57] has been working on a general purpose simulation of Artificial Life named Swarm. Swarm refers to groups of interacting agents. The Swarm simulation is a means to develop distributed algorithms for systems optimization and control for manufacturing systems. Fulkerson has applied agents to scheduling in three main areas, including assembly line scheduling, paint shop scheduling, and shop floor scheduling. Assembly line scheduling uses strong and weak constraints with the agents to determine the optimal schedule, avoiding bottlenecks. Paint shop scheduling uses agents as modules, which bid on the next paint job to be processed. The "dispatcher assigns the job to the winning agent," where the winning agent is determined by factors including which color they paint and how busy they are. The shop floor scheduling uses agents to match customer needs with supplier availability on machines to determine the next operation on the fly. Each agent (work station) has up to date information and automatic lot tracking.

Advanced Manufacturing Research [8] , A Boston-based firm refers to several trends in applications of agents in manufacturing, specifically at Allied Signal Automotive in their Safety Restraint Systems plant. Their agent based approach implements each workstation as an agent, with a customer/supplier relationship between them. Each agent knows what is required of it and what it requires, and it is thus able to autonomously select from a number of tasks that it can perform in order to satisfy its customers. The system uses distributed bills of material as well. Thus, instead of running a planning program for two days over the weekend, as is often the case, the planning can be done much quicker on an individual basis.

Both Moore [70] and Ghedira [62] have applied agents to the resource allocation problem. Their agent based approaches make full use of the distributed problem-solving capabilities for optimization processes such as simulated annealing. This application necessitates the consideration of incorporating costs associated with various choices, in order to obtain the optimal solution to the allocation of vital resources.

Berry [15] identifies three main benefits of agents in manufacturing applications, which include the ease of conversion of current computer integrated manufacturing systems to intelligent agents, the resulting "plug and play" architecture permits upgrading and modifications of the system with little work, and the ease of scalability and portability of such agent systems.

Baker, Alexander, and Parunak [10] have worked on SimCIM, which is a demonstration

application. As the name implies, these agents function as a type of electronic secretary for the user.

Other application areas of current interest at MIT [1] include animated autonomous agents which allow human interaction, such as ALIVE. This system allows wireless interaction between human participants and a graphical world of autonomous agents that respond to the participant. This type of interface has possible entertainment applications, such as game playing or movie making.

Henry Lieberman [1] is working on graphical interfaces for software visualization and debugging. This project explores how modern graphical interface techniques and explicit support for the user's problem solving activities can be made more effective with graphical interfaces.

3.2 Interface agents

Interface agents facilitate the intelligent user-assistance, which has been an ongoing problem in human-interface design [80]. All interface agents are types of user interfaces, but this specific classification of agents implies user assistance in an application environment. Lamberti [80] discusses the use of REASON (Real-time Explanation And Suggestion). REASON is an intelligent user-assisted prototype which is designed for a windowed, multi-tasking environment. The users can ask questions in via a natural language interface. User mistakes are dealt with by offering suggestions about what might have been intended based on the context of the interaction.

3.3 Design Agents

Collaborative design is a very complex activity requiring many different skills. In the design process, exploration, analysis, and evaluation are important elements. To create what never was requires a type of emergence: that is, simple elements must be put together in new ways, and conventional rules must be broken to come up with unconventional combinations. In this way, agents are now being applied to the design process.

Edmonds [45] has worked with drawing packages, such as MacDraw, and then applied techniques developed by Scrivener [101] to support collaborative design using intelligent agents. The agents worked to analyze the images presented to them by breaking them down into components and grouping them. This process allows the agent to identify emergent shapes in the original image.

3.4 Modeling agents

,

Modeling agents, as discussed by Drogoul [31], is the application of agents to the concept of modeling societies. While Drogoul applies this type of modeling to ant colonies as an example, the driving idea behind this application is the following: In order to understand and

memorize all of the 'situation-action pairs' [86] generated. For example, if the user places mail from a certain individual in a folder after reading it, a description of the situation and the action taken by the user is recorded. Thus, when a new event occurs, the agent compares the situation with the memorized situations and tries to predict the action the user will take, and suggest it.

Diederich [29] refers to a similar application called NECAS (Neural E-mail Classification and action Selection System), which is able to classify e-mail, extract relevant information from the messages during a training period, and learn responses for the proposal stage, where actions are suggested to the user.

- *Meeting scheduling agents* can be attached to existing scheduling or calendar programs, and the resulting agent can assist a user with scheduling meetings. This type of agent typically interacts with other agents like itself, and the agents will collaborate to schedule, reschedule, etc. meetings.

- *Entertainment selection agents* can help users select books, movies, music, magazines, television and radio programs, etc. that might appeal to their interests. These agents can select the entertainment medium based on the users interests, as observed from past selections. Every selection made by the user is cataloged, and this comprehensive list provides the basis for the agent's suggestions. They take input from the user as to whether the selection was appreciated or not, and as a result the selections become more accurate with time.

  Pattie Maes discusses Ringo, which is a personalized music recommendation system implemented on a Unix platform in Perl. The key of this system is that a person's agent will supply suggestions based on two types of inputs. The first is the user, where the user inputs music that he/she tried and liked or disliked. The second type of input is from other agents. The agents confer with other agents whose users have similar musical tastes. Then, musical selections are compared between the similar agents and any on the other agents list that haven't been read by the user are recommended.

- *News filtering agents*, like entertainment selection agents, can help users select items which might interest them. In this case, the agents are employed selecting news stories which it determines might be relevant to the user. As more and more information becomes available on the network, such filtering agents would help users get the information which is more relevant to them without the difficulty of sifting through all of the other information which is available.

  Maes [86] refers to NewT, which is a system that helps users filter Usenet Netnews. It is implemented in C++, and it allows the user to create several 'news agents' which are trained by the user by supplying examples of news articles of both articles which should be selected and those which should not be selected.

- *Secretary agents*, which basically incorporate scheduling and interface agent ideas, as discussed by Sycara [117], apply many of the above applications to an integrated agent

2.2.2 Intelligent Systems of Agents

The theoretical foundations for explaining intelligent systems as collections of unintelligent agents arise from work done in the philosophy of AI. This work attempts to explain the emergence of an intelligent mind from a collection of unintelligent neurons. In "Society of Mind" [92], Marvin Minsky discusses this 'phenomenon' as well as a range of related topics.

As quote by Fulkerson from the book of Proverbs, [57] "Go to the ant, you sluggard; consider its ways and be wise! It has no commander, no overseer or ruler, yet it stores its provisions in summer and gathers its food at harvest." Similar emergent behavior can be observed in a simulated environment among software agents.

When an entity is constructed from a number of finite elements, and the whole exhibits characteristics which were not present in the individual pieces, this occurence is called emergence. Emergent behavior is the phenomenon in which the whole is more than the sum of the parts. In certain tasks, such as design, emergence is very important. Design is effectively creating what never was from a set of known building blocks, thus assembling pieces to produce behavior not seen in the pieces. Edmonds [45] says that "emergence is associated with different interpretations of the knowledge that is expressed in the form of a given drawing." While design is merely one example of emergence, the concept may be key in the development of agents.

3 Applications

There are a wide range of applications domains that are making use of agent-based systems. Agent applications range from personalized assistants and user interfaces to state of the art manufacturing control.

3.1 Personal Assistant Agents

Personal assistant agents have become an area for great potential for marketability. The applications of personal assistant agents can be classified into several areas [86]:

- *Internet agents* can take many forms, as outlined by Etzioni [46]. These agents can be "tour guides" that help the user navigate the web, they may act as indexing agents that carry out massive autonomous searches of the web (e.g. WebCrawler), they may act as FAQ finders that find answers to the most Frequently Asked Questions, or they may act as expertise finders that locate individuals who are experts in particular fields of study or research.

- *Electronic mail agents* act as filters for personal e-mail. These types of agents learn to prioritize, filter, sort, delete, and save mail messages for the user, based on the users typical response to certain types of e-mail. Maxims [86] is an agent that assists users with e-mail handling. These types of agents function by observing and recording behaviors as the user deals with e-mail. As the user performs actions, the agent will

agent is modeled as having a set of beliefs and a set of inference rules. What the agent actually believes is then defined in terms of what can be inferred.

2.2 System Theories

There are two viewpoints when it comes to the theoretical foundations of agent-based systems. The first considers the interaction of a collection of intelligent agents while the second considers the emergence of intelligent behavior from a collection of unintelligent agents.

2.2.1 Systems of Intelligent Agents

Models for systems of intelligent agents deal with agent communication and cooperation. Because we are modeling at the system level, we are interested in the external characteristics of agents, and not their internal structure. Therefore, formal models of agent systems model agents as intentional systems (having intentions, beliefs, desires, etc). There are many groups working on modeling agent interatcions based on their individual intentions.

The formal basis for most agent communication models has its roots in the speech act theory from the philosophy of language [9, 102, 103, 104]. An overview of speech act theory as it relates to agent communication is presented by Labrou and Finin's in their use of speech act theory is used as a basis for a formal semantics of the KQML agent communication language [79].

Cohen and Levenson developed a formal model of action and intention which is more general than speech act theory [23, 22]. Communication is then formalized in terms of agents understanding the intentions of other agents.

Werner concentrates on communication and cooperation in multi-agent systems [127, 126, 128, 129]. From a formal theory of intention, he develops a theory of communication based on speech acts and finally cooperation and social structure.

The work done by Singh [110, 111, 112, 113] is an attempt to provide a comprehensive formal basis for multi-agent systems. Agents are described as intelligent, intentional entities. Formalizations are provided for concurrent actions, intentions and *know-how* and communications. Singh defines know-how as "the knowledge of how to act, or the knowledge of skills," which we interpret as the knowledge of the requirements and actions necessary to achieve a goal.

Jennings developed the *responsibility model* as a model of agent interaction [71, 72, 73, 75, 74]. The knowledge required for responsible agent interaction is represented explicilty in a *cooperation knowledge level* [72] expending the ideas proposed by Newell citenewell:82a. Wooldridge, with Jennings and others, developed a logic for reasoning about agents and systems of agents [131, 133, 130, 132, 135, 134, 136]. The goal is to provide formalisms for specification and verification of multi-agent systems. This work provides models of both the internal and external workings of agents.

agents are not necessarily considered intelligent, rather the action of the system as a whole exhibits intelligent characteristics.

The two apparently different types of agents are closely related: the latter are sometimes used to implement the former. Some intelligent agents are built by coordinating basic non-intelligent behaviors. Therefore, *agent* may mean different things at various levels of abstraction in a single system.

There are several benefits of using the intelligent agent paradigm for software systems. Agents can provide a high level of abstraction for dealing with intelligent systems and distributed systems. The agent abstraction is a natural extension of object-oriented technology, encapsulating the agents knowledge within an active process and providing a standard interface for intercommunication. This leads to another benefit of agent based systems, interoperability [59]. With a common Agent Communication Language, such as KQML, programs written as agents can communicate and cooperate with other such programs. In other words, an agent-based system architecture provides a consistent interface for intelligent systems to interact with. Finally, agent systems often provide a high-level 'human-like' interface to take the GUI revolution one step further.

2 Theoretical Foundations

The theoretical foundations of agent-based systems consist of formal models for every aspect of agent-based systems. Our intention here is to provide references to the theoretical literature that can provide a basis for understanding the other aspects of agents (architecture, applications, languages).

We have divided the theoretical foundations into theories describing agents and theories describing systems of agents. The former proposes models for the internal representations that agents use to model the external world as well as their beliefs and goals. The system theories focuses on modeling the external appearance of agents as intentional systems [28, 27, 24] and provide models of agent cooperation and communication. The line is blurred between the two types of models because agents sometime contain representations of the intentions of other agents in the system.

2.1 Agent Theories

Theories for modeling intelligent agents are aimed at describing the mental state (beliefs, goals, intentions, etc...) of individual agents. The models used are based from work done in more general areas of symbolic reasoning such as knowledge representation and temporal reasoning and planning [4, 5].

The most common model used or extended is the logical model of knowledge and belief proposed by Hintikka [68]. This work is commonly referred to as the *possible worlds model*. A popular extension of this work is Konolige's deduction model of belief [77]. In Konolige's model takes into account the computational limits on an agent's reasoning abilities. The

An Annotated Bibliography of Software Agent Technology

Benjamin Moore, Mark Noschang, and John Penix
Department of Electrical and Computer Engineering
University of Cincinnati
Cincinnati, Ohio 45221-0030

Abstract

This paper provides an overview of software agent technology. The various aspects of agent technology and research are broken into the theoretical foundations, applications, architectures and languages. The theoretical foundations of agent-based systems consist of formal models for every aspect of agent-based systems. Agent applications range from personalized assistants and user interfaces to state of the art manufacturing control. Agent architectures deal with the structure of multi-agent systems as well as the internal structure of individual agents. Agent languages are used to define the behavior of agents and to facilitate agent communication.

1 Introduction

The term *agent* has become a buzzword. It is used unsparingly to refer to software systems, parts of software systems, and basically anything you can click a mouse on. The use of the term agent is often justified by noting the system has attributes such as intelligence, intentions, perception, complex reasoning, autonomy (physical or causal), conceptual models, or various other attributes that are also ill-defined, ambiguous and over-used terms. As a part of exploring software agent technology, we had hoped to come forth with a precise definition for *agent*. The futility of this goal became apparent quickly.

Historically in AI there have been two standard uses of the term agent. The first use is in describing an intelligent entity (usually hardware) which appears to have its own intentions. This often meant that the system performed some actions to manipulate it's environment (via moving, or communicating) in some coherent manner. These intelligent agents communicate and interact with people and/or other agents to solve problems. More recently, software systems having these characteristics are referred to as agents.

On the other hand, the term agent is used to refer to a different, yet closely related set of entities. In Marvin Minsky's "Society of Mind" [92], he talks of groups of simple agents (agencies) working together to accomplish higher functions. In this model the individual