REFERENCES

Acronymics Inc. 2004. *AgentBuilder*. http://www.agentbuilder.com/. (accessed March 17, 2003).

Agent Lab. 2000. *Multi-Agent Modeling Language*. http://www.maml.hu/. (accessed February 25, 2003).

Agent Oriented Software. 2004. *JACK^{TM} Intelligent Agents Manual*. http://www.agent-software.com/shared/demosNdocs/JACK_Manual.pdf. (accessed May 21, 2004).

Ambros-Ingerson, J. and S. Steel. 1988. Integrating planning, execution and monitoring. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), St. Paul, USA*.

Anton, A.I., and C. Potts. 1998. The use of goals to surface requirements for evolving systems. In *Proceedings of 20thInternational Conference on Software Engineering (ICSE'98)*, *Kyoto, Japan*, 157-166.

Anton, A.I., J. H. Dempster, and D. F. Siege. 1996. Deriving Goals from a Use Case Based Requirements Specification for an Electronic Commerce System. In *Proceedings of the 6th International Workshop on Requirements Engineering: Foundations for Software Quality, Stockholm, Sweden*.

Anton, A.I., W.M. McCracken, and C. Potts. 1994. Goal decomposition and scenario analysis in business process reengineering. In *Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CaiSE'94), Utrecht, The Netherlands*, 136-144.

Arkin, R., and T. Balch. 1997. AuRA: Principles and Practice in Review. *Journal of Experimental and Theoretical Artificial Intelligence* 2-3: 175-189.

Awad E M 1985. *Systems Analysis and Design*. Illinois: Richard D. Irwin.

Baader, F., D.L. Calvanese, D. McGuinness, D. Nardi and P.F. Patel-Schneider, eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. New York: Cambridge University Press.

Bandini, S., S. Manzoni, and G. Vizzari. 2004. A Spatially Dependent Communication Model for Ubiquitous Systems. In *Proceedings of the 1st International Workshop on Environments for Multiagent Systems, New York, USA*.

Bauer, B. 2001a. UML Class Diagrams revisited in the context of agent-based systems. In *Proceedings of Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada*, 1-8.

Bauer, B. 2001b. UML Class Diagrams and Agent-Based Systems. In *Proceedings Autonomous Agents 2001, Montreal, Canada*, 104-105.

Bauer, B., J.P. Muller, and J. Odell. 2000. Agent UML: A Formalism for Specifying Multiagent Software Systems. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick*, Ireland, 91-103.

Bayardo, R.J., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. 1997. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of data, Tucson, USA*, 195-206.

Bechhofer, S., C. Goble, and I. Horrocks. 2001. DAML+OIL is not enough. In *Proceedings of the 1st Semantic Web Working Symposium*, *Stanford, USA*, 151-159.

Benjamins, R. 1995. Problem solving methods for diagnosis and their role in knowledge acquisition. *International Journal of Expert Systems: Research and Applications* 2(8): 93-120.

Benjamins, R., L.N. de Barros, and A. Valente. 1996. Constructing planners through problem-solving methods. In *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*, *Banff, Canada.*

Bergenti, F., and A. Ricci. 2002. Three approaches to the coordination of multiagent systems. In *Proceedings of the 2002 ACM symposium on applied computing*, *Madrid, Spain*, 367-372.

Bergenti, F., and A. Poggi. 2001. Agent-oriented Software Construction with UML. In *Handbook of Software Engineering and Knowledge Engineering* Vol 2, ed. S.K. Chang, 757-770. Singapore: World Scientific Publishing Co.

Bergenti, F., and A. Poggi. 2002. Supporting Agent-Oriented Modelling with UML. *International Journal of Software Engineering and Knowledge Engineering* 12(6): 605-618.

Berners-Lee, T., J. Hendler, and O. Lassila. 2001. *The Semantic Web*. http://www.sciam.com/2001/0501issue/0501berners-lee.html. (accessed February 26, 2002).

Bernon, C., M.P. Gleizes, G. Picard, and P. Glize. 2002a. The ADELFE methodology for an intranet system design. In *Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), Toronto, Canada.*

Bernon, C., M.P. Gleizes, S. Peyruqueou, and G. Picard. 2002b. ADELFE, a methodology for Adaptive Multi-Agent Systems Engineering. In *Proceedings of the 3rd International Workshop on Engineering Societies in the Agents World (ESAW-2002), Madrid, Spain.*

Beydoun, G., G. Low, C. Conzalez-Perez, and B. Henderson-Sellers. 2005. Synthesis of a Generic MAS Metamodel. In *Proceedings of the 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05) St. Louis, USA*, 27-31.

Biegel, G. 2002. Cooperation through the Environment: Stigmergy in CORTEX. In *CORTEX: Preliminary Definition of the Interaction Model*, ed. J. Kaiser, 31-38. http://cortex.di.fc.ul.pt/Deliverables/WP2-D3.pdf. (accessed January 16, 2005)

Bleyer, M. 1998. *Multi-Agent Systems for Information Retrieval on the World Wide Web*. Master Thesis, University of Ulm, Germany.

Boer, F. S. 2000. *Methodology for Agent-Oriented Software Design*. http://www.cs.uu.nl/people/frankb/nwo.html. (accessed November 15, 2003).

Boicu, M., G. Tecuci, M. Bowman, D. Marcu, S.W. Lee, and K. Wright. 1999. A Problem-Oriented Approach to Ontology Development. In *Proceedings of the 16th National Conference on Artificial Intelligence Workshop on Ontology Management, Orlando, Florida.*

Bonabeau, E., F. Henaux, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz. 1998. Routing in Telecommunications Networks with "Smart" Ant-Like Agents. In *Proceedings of the 2nd International Workshop on Agents in Telecommunications Applications (IATA '98), Paris, France.*

Booch, G. 1994. *Object-oriented Analysis and Design*. 2nd ed. Massachusetts: Addison-Wesley.

Bordart, F., A. Flory, M. Leonard, A. Rochefeld, C. Rolland, and H. Tardieu. 1983. Evaluation of CRIS 1 I.S. Developments Methods Using a Three Cycles. In

Information Systems Design Methodologies - A Feature Analysis, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Borgida, A., R.J. Brachman, D.L. McGuinness, and L.A. Resnick. 1989. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, USA*, 59-67.

Brachman, R.J., and J.G. Schmolze. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9(2): 191-216.

Brandt, I. 1983. A Comparative Study of Information Systems Design Methodologies. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Bratman, M.E., D.J. Israel, and M.E. Pollack. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4:349-355.

Bresciani, P., P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. 2004. TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems* 8(3): 203-236.

Brueckner, S. 2000. *Return from the Ant.* PhD thesis, Humboldt-Universität zu Berlin, Germany.

British Telecommunications. 2002. *Zeus*. http://more.btexact.com/projects/agents/zeus/index.htm. (accessed June 5, 2003).

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* RA-2: 14-23.

Burrafato, P., and M. Cossentino. 2002. Designing a multi-agent solution for a bookstore with the PASSI methodology. In *Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), Toronto, Canada*.

Cabri, G., L. Leonardi, and F. Zambonellli. 2000. XML Dataspaces for mobile agent coordination. In *Proceedings of the 2000 ACM symposium on Applied computing, Como, Italy*, 181-188.

Cairo, O. and J.C. Alvarez. 2004. The KAMET II Approach for Knowledge-Based system Construction. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2004), Wellington, New Zealand*, 1227-1234.

Calvanese, D., G. De Giacomo, and M. Lenzerini. 2001. A framework for ontology integration. In *Proceedings of the 1st International Semantic Web Working Symposium, Stanford, USA*, 303–317.

Carbonell, J.G., C.A. Knoblock, and S. Minton. 1991. PRODIGY: An Integrated Architecture for Prodigy. In *Architectures for Intelligence*, ed. K. VanLehn, 241-278. New Jersey: Lawrence Erlbaum Associates.

Cardelli, L. 1994. *Obliq: A Language with Distributed Scope*. Technical report, Digital Equipment Corp, Systems Research Center, California, USA.

Carver, N., and V. Lesser. 1995. The DRESUN Testbed for research in FA/C Distributed situation assessment: extensions to the model of external evidence. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, *San Francisco, USA*, 33-40.

Castro, J., M. Kolp, and J. Mylopoulos. 2001. A Requirements-Driven Development Methodology. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering CAiSE 01, Interlaken, Switzerland.*

Castro, J., M. Kolp, and J. Mylopoulos. 2002. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems* 27: 365-389.

Ceccaroni, L. 2001. What if a wastewater treatment plant were a town of agents. In *Proceedings of the workshop Autonomous Agents 2001 - W03: Ontologies in Agent Systems, Montréal, Canada.*

Cernuzzi, L., and G. Rossi. 2002. On the Evaluation of Agent-Oriented Modelling Methods. In *Proceedings of the OOPSLA Workshop on Agent-Oriented Methodologies*, *Seattle, USA*, 21-33.

Chandrasekaran, B., J.R. Josephson, and V.R. Benjamins. 1999. What are ontologies, and why do we need them? *IEEE Intelligent Agents* 14(1): 20-26.

Chapman, D., and P. Agre. 1986. Abstract reasoning as emergent from concrete activity. In *Proceedings of the 1986 Workshop on Reasoning About Actions &Plans, Los Altos, USA*, 411-424.

Chatley, R. n.d. *Hybrid Deliberative/Reactive Agents*. http://www.iis.ee.ic.ac.uk/~frank/surp99/article2/rbc97/. (accessed May 21, 2002).

Cheikes, B.A. 1995. GIA: An Agent-Based Architecture for Intelligent Tutoring Systems. In *Proceedings of the CIKM'95 Workshop on Intelligent Information Agents, Baltimore, USA*.

Chelberg, D., L. Welch, A. Lakshmikumar, G. Matthew, and Q. Zhou. 2001. Meta-Reasoning For a Distributed Agent Architecture. In *Proceedings of the South-Eastern Symposium on System Theory, Ohio, USA,* 377-381.

CHI Software Inc. 2003. *iGEN The Cognitive Agent Software Toolkit*. http://www.cognitiveagent.com/. (accessed April 28, 2002).

Ciancarini, P. 1996. Coordination models and languages as software integrators. *ACM Computing Surveys* 28(2): 300-302.

Ciancarini, P., O. Nierstrasz, and R. Tolksdorf. 1998. *A case study in coordination: Conference Management on the Internet*. ftp://cs.unibo.it/pub/cianca/ coordina.ps.gz. (accessed April 20, 2004).

Ciancarini, P., A. Omicini, and F. Zambonelli. 1999. Multiagent System Engineering: the Coordination Viewpoint. In *Proceedings of the 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL), Orlando, USA*, 250-259.

Collinot, A., and A. Drogoul. 1998. Using the Cassiopeia Method to Design a Soccer Robot Team. *Applied Artificial Intelligence Journal* 12(2-3): 127-147.

Collinot, A., A. Drogoul, and P. Benhamou. 1996. Agent Oriented Design of a Soccer Robot Team. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96), Kyoto, Japan*, 41-47.

Conri, S.E., R.A. Meyer, and V.R. Lesser. 1988. Multistage negotiation in distributed planning. In *Distributed Artificial Intelligence*, ed. A.H. Bond and L. Gasser, 367-384. California: Morgan Kaufmann Publishers Inc.

Cossentino, M. 2002. Different perspectives in designing multi-agent systems. In *Proceedings of Agent Technology and Software Engineering Workshop (AGES '02), Erfurt, Germany*.

Cossentino, M., and M. Potts. 2002. A CASE tool supported methodology for the design of multi-agent systems. In *Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP'02)*.

Cranefield, S., and M. Purvis. 1999. UML as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden*.

Cranefield, S., S. Hausteiny, and M. Purvis. 2001. UML-based ontology modelling for software agents. In *Proceedings of Ontologies in Agent Systems Workshop*, 21-28.

Cranefield, S., M. Purvis, M. Nowostawski, and P. Hwang. 2002. Ontologies for interaction protocols. In *Proceedings of the 2nd International Workshop on Ontologies in Agent Systems, Bologna, Italy*.

Cremonini, M., A. Omicini, and F. Zambonelli. 1999. Multi-agent systems on the Internet: Extending the scope of coordination towards security and topology. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'99), Valencia, Spain*, 77-88.

Cuesta, P., A. Gómez, J.C. González, and F.J. Rodríguez. 2002. The MESMA approach for AOSE. In *Proceedings of 4th Iberoamerican Workshop on Multi-Agent Systems (Iberagents'2002), Málaga, Spain*.

Cuppari, A., P.L. Guida, M. Martelli, V. Mascardi, and F. Zini. 1999. Prototyping Freight Trains Traffic Management Using Multi-Agent Systems. In *Proceedings of IEEE International Conference on Information, Intelligence and Systems, Bethesda, USA*, 646-653.

Dardenne, A., A. van Lamsweerde, and S. Fickas. 1993. Goal-Directed Requirements Acquisition. *Science of Computer Programming* 20: 3-50.

Davis, D. 1995. *A design for the robot crèche scenario*. http://citeseer.nj.nec.com/davis95design.html. (accessed October 25, 2002).

de Bruijn, J. 2003. *Using Ontologies. Enabling Knowledge Sharing and Reuse on the Semantic Web*. Technical Report, Digital Enterprise Research Institute, Ireland.

Decker, K.S., V.R. Lesser, M.V. Nagendra-Prasad, and T. Wagner. 1995. MACRON: an architecture for multi-agent cooperative information gathering. In *Proceedings of the CIKM-95 Workshop on Intelligent Information Agents, Baltimore, USA*.

Decker, S., M. Erdmann, D. Fensel., and R. Studer. 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In *Proceedings of the IFIP TC2/WG2.6 8th Working Conference on Database Semantics-Semantic Issues in Multimedia Systems, New Zealand*, 351-369.

Degirmenciyan, I., F. Marc, and A. ElFallah-Seghrouchni. 2003. Modeling multi-agent plans with hybrid automata. In *Proceedings of the workshop FAMAS 03 ETAPS 03, Warsaw, Poland.*

DeLoach, S.A. 1999. Multiagent Systems Engineering: A methodology and language for designing agent systems. In *Proceedings of Agent-Oriented Information Systems (AOIS'99), Seattle, USA*, 45-57.

DeLoach, S.A. 2005. Multiagent Systems Engineering of Organization-based Multiagent Systems. In *Proceedings of the 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05) St. Louis, USA*, 5-11.

DeMarco, T. 1978. *Structured Analysis and System Design*. New York: Yourdon Press.

Demazeau, Y., and A.C.R. Costa. 1996. Populations and organizations in open multi-agent systems. In *Proceedings of the 1st National Symposium on Parallel and Distributed AI, Hyderabad, India*.

Dennis, A., and B. Wixom. 2003. *Systems analysis design*. 2nd ed. New York: J. Wiley.

Denti, E., and A. Omicini. 2001. LuCe: A tuple-based coordination infrastructure for Prolog and Java agents. *Autonomous Agents and Multi-Agent Systems* 4(1/2):139–141.

Denti, E., A. Natali, and A. Omicini. 1998. On the expressive power of a language for programming coordination media. In *Proceedings of the 1998 ACM Symposium on Applied Computing, Atlanta, USA*, 169-177.

desJardins, M.E., E.H. Durfee, C.L. Ortiz, and M.J. Wolverton. 2000. A Survey of Research in Distributed, Continual Planning. *AI Magazine* 4: 13-22.

DiLeo, J., T. Jacobs, and S. DeLoach. 2002. Integrating Ontologies into Multiagent Systems Engineering. In *Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), Bologna, Italy*.

Ding, Y. 2001. IR and AI: The role of ontology. In *Proceedings of the 4th International Conference of Asian Digital Libraries (ICADL 2001), Bangalore, India*.

d'Inverno, M., D. Kinny, M. Luck, and M. Wooldridge. 1997. A formal specification of dMARS. In *Proceedings of the 4th International Workshop on Agent Theories, Architectures and Languages, Providence, USA*, 155-176.

Durfee, E.H. 1999. Distributed Problem Solving and Planning. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. G. Weiss, 121-164. London: The MIT Press.

Duursma, C. 1993. *Task Model definition and Task Analysis process*. ESPRIT Project P5248 KADS-II KADS-II/M5/VUB/RR/004/1.1c, Vrije Universiteit Brussel.

Ehrig, M., and Y. Sure. 2004. Ontology Mapping an Integrated Approach. In *Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece*.

Elammari, M., and W. Lalonde. 1999. An Agent-Oriented Methodology: High-Level and Intermediate Models. In *Proceedings of the 1st Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS'99)*, *Heidelberg, Germany*.

Eliason, A.L. 1990. Syst*ems development: analysis, design and implementation*. 2nd ed. Glenview: Scott, Foresman/Little, Brown Higher Education.

Ephrati, E., and J.S. Rosenschein. 1991. The Clarke Tax as a consensus mechanism among automated agents. In *Proceedings of the 9th National Conference on Artificial Intelligence, San Jose, USA*, 173-178.

Erdur, R.C., O. Dikenelli, and H. Sengonca. 1999. A Multiagent System for Searching and Retrieving Reusable Software Components. In *Proceedings of 14th International Conference on Computer and Information Sciences - ISCIS XIV, Kusadasi, Turkey*.

Eurescom. 2001a. *MESSAGE: Methodology for Engineering Systems of Software Agents – Final Guidelines for the Identification of Relevant Problem Areas where Agent Technology is Appropriate*. http://www.eurescom.de/public/projectresults/P900-series/907d2.asp. (accessed Oct 7, 2003).

Eurescom. 2001b. *Methodology for Agent-Oriented Software Engineering*. http://www.eurescom.de/public/projectresults/P900-series/907ti1.asp. (accessed March 21, 2004).

Fabio, B., G. Caire, T. Trucco, and G. Rimassa. 2004. *JADE Programmer's Guide*. http://jade.tilab.com/doc/programmersguide.pdf. (accessed Dec 14, 2004).

Falasconi, S., G. Lanzola, and M. Stefanelli. 1996. Using Ontologies in Multi-Agent Systems. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*, *Banff, Canada*.

Falbo, R.A., C.S. Menezes, and A.R.C. Rocha. 1998. A Systematic Approach for Building Ontologies. In *Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, 349-360.

Falbo, R.A., G. Giancarlo, and K.C. Duarte. 2002. An Ontological Approach to Domain Engineering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy*, 351-358.

Falkenberg, E., G.M. Nijssen, A. Adams, L. Bradley, P. Bugeia, A.L. Campbell, M. Carkeet, G. Lehmann, and A. Shoesmith. 1983. Feature Analysis of ACM/PCM, CIAM, ISAC and NIAM. In *Information Systems Design Methodologies - A*

Feature Analysis, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Fan, X. 2000. Towards a building methodology for software agents. In *Proceedings of the 6th International Conference on Object-Oriented Information Systems, London, UK*, 45-53.

Farquhar, A., R. Fikes, and J. Rice. 1996. The Ontolingua Server: A tool for collaborative ontology construction. Technical Report 926-26, Knowledge Systems Laboratory, Stanford University.

Fensel, D. 1997. An Ontology-Based Broker: Making Problem-Solving Method Reuse Work. In *Proceedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems held in conjunction with the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), Nagoya, Japan,* 23-29.

Fensel, D. 2001. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Berlin: Springer-Verlag.

Fensel, D., E. Motta, S. Decker, and Z. Zdrahal. 1997. Using Ontologies For Defining Tasks, Problem-Solving Methods and Their Mapping. In *Proceedings of 10th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW-97), Heidelberg, Germany*, 113-128.

Ferber, J., and O. Gutknecht. 1998. A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. In *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98), Paris, France*, 128-135.

Ferguson, I. A. 1992. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, University of Cambridge, UK.

Fernandez, M., A. Gomez-Perez, and N. Juristo. 1997. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Proceedings of the AAAI Spring Symposium on Ontological Engineering, California, USA*, 33-40.

Fikes, R.E., and N. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 5(2): 189-208.

Finkelstein, A. 1998. Interoperable Systems: An introduction. In *Information Systems Interoperability*, ed. B.J. Kramer, M.P. Papazoglou and H.-W. Schmidt, 1-9. England: Research studies press.

FIPA. n.d.a. *FIPA Agent Communication Language Specifications*. http://www.fipa.org/repository/aclspecs.html. (accessed January 6, 2003).

FIPA. n.d.b. *The Foundation for Intelligent Physical Agents*. http://www.fipa.org/. (accessed February 24, 2002).

FIPA. 2001a. *FIPA Agent Software Integration Specification*. http://www.fipa.org/ specs/fipa00079/XC00079B.html. (accessed June 25, 2004).

FIPA. 2001b. *FIPA Ontology Service Specification*. http://www.fipa.org/specs/ fipa00086/XC00086D.html. (accessed June 10, 2002).

FIPA.2001c. *FIPA Interaction Protocol Library Specification*. http://www.fipa.org/ specs/fipa00025/XC00025E.html. (accessed January 21, 2003).

FIPA. 2002. *FIPA Interaction Protocols Specifications*. http://www.fipa.org/repository/ ips.php3. (accessed September 3, 2002).

FIPA. 2003. *FIPA Modeling Area: Deployment and Mobility*. http://www.auml.org/ auml/documents/DeploymentMobility.zip. (accessed August 24, 2003).

FIPA. 2004. *FIPA Agent Management Specification*. http://www.fipa.org/specs/ fipa00023/SC00023K.html. (accessed May 19, 2003).

Firby, R.J. 1989. *Adaptive Execution in Dynamic Domains*. PhD thesis, Yale University, Computer Science Department, USA.

Firesmith, D.G., and Henderson-Sellers, B. 2002. *The OPEN Process Framework: An Introduction*. London: Addison-Wesley.

Fisher, M., J. Muller, M. Schroeder, G. Staniford, and G. Wagner. 1997. Methodological Foundations for Agent-Based Systems. *The Knowledge Engineering Review* 12(3): 323-329.

Flores-Mendez, R.A. 1999. Towards a Standardization of Multi-Agent System Frameworks. *ACM Crossroads Student Magazine* 5(4). http://www.acm.org/ crossroads/xrds5-4/multiagent.html. (accessed January 8, 2002).

Franklin, S. and A. Graesser. 1996. Is it an agent or just a program?: A Taxonomy for Intelligent Agents. In *Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages, Budapest, Hungary*, 21-35.

Franzén, T., S. Haridi, and S. Janson. 1992. An Overview of the Andorra Kernel Language. In *Proceedings of the 2nd Workshop on Extensions to Logic Programming, Stockholm, Sweden*, 163-180.

Gamper, J., W. Nejdl, and M. Wolpers. 1999. Combining Ontologies and Terminologies in Information Systems. In *Proceedings of the 5th International Congress on Terminology and Knowledge Engineering, Innsbruck, Austria*, 152-168.

Gat, E. 1991. Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots. *SIGART Bulletin 2*: 70-74.

General Magic Inc. 1995. *The Telescript Language Reference*. http://www.science.gmu.edu/~mchacko/Telescript/docs/telescript.html. (accessed September 23, 2002).

Genesereth, M.R., and N.J. Nilsson. 1987. *Logical Foundation of Artificial Intelligence*. California: Morgan Kaufmann.

Genesereth, M.R., and R.E. Fikes. 1992. Knowledge Interchange Format Version 3.0 Reference Manual. Logic Group Technical Report Logic-92-1, Stanford University Logic Group, Standford University, USA.

Gennari, J.H., S.W. Tu, T.E. Rotenfluh, and M.A. Musen. 1994. Mapping domains to methods in support of reuse. *International Journal of Human-Computer Studies*, 41: 399-424.

Georgeff, M.P. 1994. *Distributed multi-agent reasoning systems (dMARS)*. Technical report, Australian Artificial Intelligence Institution, Melbourne, Australia.

Georgeff, M. P., and A.L. Lansky. 1987. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, *Seattle, USA*, 677-682.

Girardi, R. and C.G. de Faria. 2004. An Ontology-Based Technique for the Specification of Domain and User Models in Multi-Agent Domain Engineering. *Clei Electronic Journal* 7(1).

Girardi, R., C.G. de Faria, and L. Balby. 2004. Ontology-based Domain Modeling of Multi-Agent Systems. In *Proceedings of the 3rd International Workshop on Agent-Oriented Methodologies at OOPSLA 2004*, Vancouver, Canada, 51-62.

Glaser, N. 1996. *Contribution to Knowledge Acquisition and Modelling in a Multi-Agent Framework (the CoMoMAS Approach)*. PhD Thesis, University of Navy 1, France.

Glaser, N. 1997a. The CoMoMAS Approach: From Conceptual Models to Executable Code. In *Proceeding of the 8th European Workshop On Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-97), Ronneby, Sweden*.

Glaser, N. 1997b. The CoMoMAS Methodology and Environment for Multi-Agent System Development. In *Multi-Agent Systems – Methodologies and Applications*, ed. C. Zhang and D. Lukose, 1-16. Berlin: Springer-Verlag.

Glass, G. 1998. ObjectSpace Voyager – The Agent ORB for Java. In *Proceedings of the 2nd International Conference on Worldwide Computing and Its Applications, Tsukuba, Japan*, 38-55.

Goldin, D., and D. Keil. 2004. Toward Domain-Independent Formalization of Indirect Interaction. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04), University of Modena and Reggio Emilia, Italy,* 393-394.

Gray, R.S. 1995. Agent Tcl: A transportable agent system. In *Proceedings of the CIKM Workshop on Intelligent Information Agents, 4th International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore, USA.

Gruber, T. 1993a. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies* 43(5-6): 907-928.

Gruber, T. 1993b. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5(2):199-220.

Grüninger, M., and M.S. Fox. 1995. Methodology for the Design and Evaluation of Ontologies. In *Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada.*

Guarino, N. 1997. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, ed. N. Guarino, 139-170. Berlin: Springer Verlag.

Guarino, N. 1998. Formal Ontology and Information Systems. In *Proceedings of the 1st International Conference on Formal Ontology in* I*nformation Systems, Trento, Italy*.

Guarino, N., and P. Giaretta. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, ed. N. Mars, 25-32. Amsterdam: IOS Press.

Guessoum, Z., and J.P. Briot. 1999. From active objects to autonomous agents. *IEEE Concurrency* 7(3): 68-76.

Guilfoyle, C., and E. Warner. 1994. *Intelligent agents: The new revolution in software*. Ovum Report.

Gutierrez-Casorran, C., J.T. Fernandez-Breis, and R. Martinez-Bejar. 2001. Ontological modelling of natural categories-based agents: an ant colony. In *Proceedings of the*

Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal, Canada.

Haumer, P., K. Pohl, and K. Weidenhaupt. 1998. Requirements Elicitation and Validation with Real World Scenes. *IEEE Transactions on Software Engineering* 24(12): 1036-1054.

Henderson-Sellers, B., A. Simons, and H. Younessi. 1998. *The OPEN Toolbox of Techniques*. England: Addison Wesley Longman Ltd.

Herrero, P. and de Antonio, A. 2002. A Human Based Perception Model for Cooperative Intelligent Virtual Agents. In *Proceedings of the 10th International Conference on Cooperative Information Systems (CoopIS 2002)*, California. USA, 195-212.

Hevner, A.R., S.T. March, J. Park, J., and S. Ram. 2004. Design science in information systems research. *MIS Quarterly*, 28(1): 75-106.

Honavar, V. 1999. *Intelligent Agents and Multi Agent Systems - Tutorial at IEEE CEC*. http://www.cs.iastate.edu/%7Ehonavar/agent99.pdf. (accessed October 10, 2002).

Horlait, E. 2003. *Mobile Agents for Telecommunication Applications (Innovative Technology Series: Information Systems and Networks)*. England: Kogan Page Science.

Horrocks, I., and F. van Harmelen. 2001. *Reference Description of the DAML+OIL Ontology Markup Language*. Technical report. http://www.daml.org/2001/03/reference.html. (accessed May 16, 2004).

Huget, M.P., B. Bernhard, J. Odell, R. Levy, P. Turci, R. Cervenka, M. Nodine, S. Cranefield, and H. Zhu. 2003. *FIPA Modeling: Agent Class Diagrams*. http://www.auml.org/auml/documents/main.shtml. (accessed September 5, 2003).

Huhns, M.N. and L.M. Stephens. 1999. Multiagent Systems and Societies of Agents. Journal of Applied Artificial Intelligence. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. G. Weiss, 79-120. London: The MIT Press.

Huhns, M.N., and M.P. Singh. 1997. Ontologies for agents. *IEEE Internet Computing* 1(6): 81-83.

Huhns, M.N., and M.P. Singh, eds. 1998. *Readings in Agents*. California: Morgan Kaufmann Publishers Inc.

Humphreys, B.L, and D.A. Lindberg. 1993. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of Medical Library Association* 81(2):170-177.

Hwang, C.H. 1999. *Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information*. Technical report, Microelectronics and Computer Technology Corporation, Texas, USA.

IBM. 2002. *Aglets*. http://www.trl.ibm.com/aglets/. (accessed April 25, 2003).

IEEE – Institute of Electrical and Electronics Engineers. 1990. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York.

Iglesias, C.A., M. Garijo, and J.C. Gonzalez. 1999. A survey of agent-oriented methodologies. In *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98), Paris, France*, 317-330.

Iglesias, C.A., M. Garijo, J.C. Gonzalez, and J.R. Velasco. 1996. A Methodological Proposal for Multi-Agent Systems Development Extending CommonKADS. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada.*

Iglesias, C.A., M. Garijo, J.C. Gonzalez, and J.R. Velasco. 1998. Analysis and Design of Multi-Agent Systems using MAS-CommonKADS. In *Intelligent Agents IV (LNAI Volume 1365)*, ed. M.P. Singh, A. Rao, and M. Wooldridge, 313-326. Berlin: Springer-Verlag.

Iivari, J. and P. Kerola. 1983. A Sociocybernetic Framework for the Feature Analysis of Information Systems Design Methodologies. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Institut de Recherche en Informatique de Toulouse. n.d. *ADELFE: Atelier de Développement de Logiciels à Fonctionnalité Emergente*. http://www.irit.fr/ ADELFE/. (accessed July 25, 2002).

Ioannidis, Y.E., and T.K. Sellis. 1989. Conflict Resolution of rules assigning values to virtual attributes. In *Proceedings of ACM SIGMOD 1989 International Conference on Management of Data, Portland, USA*, 205-214.

JAFMAS Java-Based Framework for Multi-Agent Systems. Ohio: University of Cincinnati. http://www.ececs.uc.edu/~abaker/JAFMAS/. (accessed April 25, 2003).

Jayaratna, N. 1994. *Understanding and Evaluating Methodologies - NIMSAD A Systematic Framework*. England: McGraw-Hill.

Jennings, N. R. 1993. 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.

Jennings, N.R. 2001. Building complex, distributed systems: the case for an agent-based approach. *Communications of the ACM* 44(4): 35-41.

Jennings, N.R., and M. Wooldridge. 1995. Applying Agent Technology. *Applied Artificial Intelligence* 9 (4): 351-359.

Jennings, N.R., and M. Wooldridge. 1998. Applications of Intelligent Agents. In *Agent Technology: Foundations, Applications, and Markets*, ed. N. Jennings, and M. Wooldridge, 3-28. Berlin: Springer-Verlag.

Jennings, N.R., and M. Wooldridge. 2001. Agent-Oriented Software Engineering. In *Handbook of Agent Technology*, ed. J. Bradshaw. USA: AAAI/MIT Press.

Jennings, N.R., S. Sycara, and M. Wooldridge. 1998. A Roadmap of Agent Research and Development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1): 7-38.

Kaelbling, L.P. 1991. A situated automata approach to the design of embedded agents. *SIGART Bulletin* 2(4): 85-88.

Kalfoglou, Y., M. Schorlemmer. 2003. IF-Map: an ontology mapping method based on Information Flow theory. *Journal on Data Semantics* 1(1): 98-127.

Kankaanpää, T. 1999. *Design and Implementation of a Conceptual Network And Ontology*. Master thesis, Helsinki University of Technology, Finland.

Karp, P. D., V.K. Chaudhri, and J. Thomere. 1999. *XOL: An XML-Based Ontology Exchange Language*. Technical report version 3. ftp://smi.stanford.edu/pub/bio-ontology/xol.doc. (accessed May 17, 2004)

Kendall, E.A. 1999. Role modelling for agent system analysis, design, and implementation. In *Proceedings of the 1st International Symposium on Agent Systems and Applications, Palm Springs, California*, 204-218.

Kendall, E.A. 2000. Agent Software Engineering with Role Modelling. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 163-170.

Kendall, K. E., and J. E. Kendall. 2002. *Systems Analysis and Design*, 5th ed. New Jersey: Prentice Hall.

Kendall, E.A., and L. Zhao. 1998. Capturing and Structuring Goals. In *Proceedings of Conference on Object-Oriented Programming, Systems, Languages, and Applications OOPSLA'98, Vancouver, Canada.*

Kendall, E.A., M.T. Malkoun, and C. Jiang. 1995. A methodology for developing Agent based Systems for Enterprise Integration. In *Proceedings of the 1st Australian Workshop on Distributed Artificial Intelligence: Architecture and Modelling, Canberra, Australia*, 85-99.

Khan, L.R. 2000. *Ontology-based information selection*. PhD thesis, University of South California, USA.

Kifer, M., G. Lausen, and J. Wu. 1995. Logical foundations of Object-Oriented and Frame-Based Languages. *Journal of ACM* 42(4): 741 - 843.

Kinny, D., and M. Georgeff. 1996. Modelling and Design of Multi-agent Systems. In *Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages (ATAL'96), Budapest, Hungary*, 1-20.

Kinny, D., M. Georgeff, and A. Rao. 1996. A Methodology and Modelling Technique for Systems of BDI Agents. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), Eindhoven, The Netherlands*, 56-71.

Klampanos, I.A., J.J. Barnes, and J.M. Jose. 2003. Evaluating Peer-to-Peer Networking for Information Retrieval within the Context of Meta-Searching. In *Proceedings of the 2003 European Colloquium on IR Research*, *Pisa, Italy*, 528-536.

Klampanos, I.A., and J.M. Jose. 2003. An Architecture for Peer-to-Peer Information Retrieval. *SIGIR'03*, 401-402.

Knoblock, C.A., A. Arens, and C.N. Hsu. 1994. Cooperating Agents for Information Retrieval. In *Proceedings of the 2nd International Conference on Cooperative Information Systems, Toronto, Canada.*

Knowledge Based Systems Inc. 1994. *IDEF5 Method Report*. http://www.idef.com/Downloads/pdf/Idef5.pdf. (accessed October 16, 2001).

Kolp, M. , Castro, J. and Mylopoulos, J. 2001. A social organization perspective on software architectures. In *Proceedings of the 1st International Workshop from Software Requirements to Architectures (STRAW'01), Toronto, Canada*, 5-12.

Kotonya, G, and I. Sommerville. 1998. *Requirements Engineering: Processes and techniques*. John Wiley & Sons.

Kung, C.H. 1983. An Analysis of Three Conceptual Models with Time Perspective. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Leach, C. 1979. *Introduction to statistics: a nonparametric approach for the social sciences*. New York: Wiley.

Lenat, D.B., and R.V. Guha. 1990. *Building large knowledge-based systems: Representation and inference in the CYC project*. USA: Addison-Wesley.

Lesser, V.R. 1996. Cooperative Multi-agent systems: A personal View of the State of the Art. *IEEE Transactions on Knowledge and Data Engineering* 11(1): 133 - 142.

Lind, J. 1999. *MASSIVE: Software Engineering for Multiagent Systems*. PhD Thesis, University of Saarbrucken, Germany.

Lind, J. 2000a. The MASSIVE development method for Multiagent Systems. In *Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agents (PAAM2000), Manchester, UK.*

Lind, J. 2000b. Issues in Agent-Oriented Software Engineering. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 45-58.

Luck, M., P. McBurney, and C. Preist. 2003. *Agent Technology: Enabling Next Generation Computing: A roadmap for Agent Based Computing*. AgentLink

Lueg, C., and Salomon, R. 1997. A New AI Perspective on Software Agents: Preliminary Report. In *Proceedings of the 2nd German Workshop on Artificial Life (GWAL 97), Dortmund, Germany*, 59-60.

Macaulay, L. 1996. *Requirements Engineering*. Berlin: Springer-Verlag.

Madhavan, J., P.A. Bernstein, P. Domingos, and A.Y. Halevy. 2002. Representing and reasoning about mappings between domain models. In *Proceedings of the 18th National Conference on Artificial Intelligence, Alberta, Canada*, 80 – 86.

MADKIT. 2002. http://www.madkit.org/. (accessed July 20, 2002).

Maes, P. 1991. The agent network architecture. *SIGART Bulletin* 2(4): 115-120.

Mahalingam, K., and M.N. Huhns. 1997. An ontology tool for query formulation in an agent-based context. In *Proceedings of the 2nd IFCIS International Conference on Cooperative Information Systems (CoopIS '97), Kiawah Island, USA*, 170-178.

Malucelli, A., and E. Oliveira. 2004. Ontology-Services Agent to Help in the Structural and Semantic Heterogeneity. In *Proceedings of PRO-VE'04 - 5th IFIP Working Conference on Virtual Enterprises*, *Toulouse, France*.

Mamei, M., and F.Zambonelli. 2004. Motion Coordination in the Quake 3 Arena Environment: A Field-based Approach. In *Proceedings of the 1st International Workshop on Environments for Multiagent Systems, New York, USA*.

March, S. and G.F. Smith. 1995. Design and natural science research on information technology. *Decision Support Systems*, 15: 251-266.

Mars, N.J.I., W.G. Ter Stal, H. De Jong, P.E. Van Der Vet, and P.-H. Speel. 1994. Semi-automatic Knowledge Acquisition in Plinius: An Engineering Approach. In *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-based Systems Workshop, Banff, Canada*, 4.1-4.15.

Mason, C.L., and R.R. Johnson. 1989. DATMS: a framework for distributed assumption based reasoning. In *Distributed Artificial Intelligence II*, ed. L. Gasser and M.N. Huhns. London: Pitman/Morgan Kaufman

Masuoka, R., and A. Sato. 1999. *Agent Description Ontology - Version 2 (CFP6 014).* http://www.flacp.fujitsulabs.com/~rmasuoka/papers/fipa99-nice-proposal.pdf. (accessed July 10, 2003).

Mena, E., A. Illarramendi, V. Kashyap, and A. Sheth. 2000. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. International Journal on Distributed and Parallel Databases 8(2): 223--271.

Miles, S., M. Joy, and M. Luck. 2000. Designing Agent-Oriented Systems by Analysing Agent Interactions. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 171-184.

Mine, T., D. Matsumo, A. Kogo, and M. Amamiya. 2004. Design and Implementation of Agent Community Based Peer-to-Peer Information Retrieval Method. In *Proceedings of the 8th International Workshop on Cooperative Information Agents, Erfurt, Germany*, 31-46.

Mishra, S., and P. Xie. 2003. Interagent Communication and Synchronization Support in the DaAgent Mobile Agent-Based Computing System. *IEEE Transactions on Parallel and Distributed Systems* 14(3): 290-306.

Mitsubishi Electric Research Laboratories. 2004. *Concordia.* http://www.merl.com/projects/concordia/. (accessed April 15, 2003).

Moukas, A., and P. Maes. 1998. Amalthaea: an evolving multi-agent information filtering and discovery system for the WWW. *Autonomous Agents and Multi-agent Systems* 1(1): 59-88.

Mountzia, M.A. 1996. An Intelligent-Agent based Framework for Distributed Systems Management. In *Proceedings of the 3rd HP OVUA Workshop*, *Toulouse, France*.

Mueller, H.J. 1997. Towards agent systems engineering. *International Journal on Data and Knowledge Engineering (Special Issue on Distributed Expertise)* 23: 217-245.

Mukherjee, R., P.S. Dutta, and S. Sen. 2000. Analysis of domain specific ontologies for agent-oriented information retrieval. In *Working notes of the AAAI-2000 Workshop on Agent-Oriented Information Systems*.

Muller, J.P. 1999. Architectures and applications of intelligent agents: A survey. *Knowledge Engineering Review* 13(4):353-380.

Muller, J.P., and M. Pischel. 1993. *The Agent Architecture InteRRaP: Concept and Application*. Technical Report RR-93-26, German Research Center for Artificial Intelligence, Saarbrucken, Germany.

Myers, K.L. 1997. *User Guide for the Procedural Reasoning System*. Technical Report, Artificial Intelligence Center, California, USA.

Nareyek, A. 2001. EXCALIBUR: Adaptive Constraint-Based Agents in Artificial Environments. http://www.ai-center.com/projects/excalibur/documentation/ (accessed December 10, 2004)

Newell, A. 1990. *Unified Theories of Cognition.* Massachusetts: Harvard University Press.

Newell, A. and H. Simon. 1963. GPS: A program that simulates human thought. In *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Nissen, H.E. 1983. Subject Matter Separability in Information Systems Design Methods. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Nodine, M. H., and A. Unruh. 1997. Facilitating open communication in agent systems: the InfoSleuth infrastructure. In *Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Language*s, 281-295.

Noy, N.F., and D.L. McGuinness. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical report KSL-01-05, Stanford Knowledge Systems Laboratory.

Nwana, H., and M. Wooldridge. 1996. Software agent technologies. *BT Technology Journal* 14(4): 68-79.

Object Agency Inc. 1995. *A Comparison of Object-Oriented Development methodologies*. http://www.toa.com/smnn?mcr.html. (accessed November 9, 2002)

Object Management Group. 2000. *Agent Technology Green Paper version 1*. http://www.jamesodell.com/ec2000-08-01.pdf. (accessed May 27, 2003).

Object Management Group. 2003. *OMG Unified Modeling Language Specification*. http://www.omg.org/technology/documents/formal/uml.htm. (accessed October 20, 2004).

O'Brien, P.D., and R.C. Nicol. 1998. FIPA - Towards a Standard for Software Agents. *BT Technology Journal* 16(3): 51-59.

Odell, J., and M.-P. Huget. 2003. *FIPA Modeling: Interaction Diagrams*. http://www.auml.org/auml/documents/ID-03-07-02.pdf. (accessed December 17, 2003).

Odell, J., H.V.D Parunak, and B. Bauer. 2000a. Extending UML for Agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, Austin, USA*, 3-17.

Odell, J., H.V.D Parunak, and B. Bauer. 2000b. Representing agent interaction protocols in UML. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 121-140.

Odell, J., H.V.D. Parunak, S. Brueckner, and J. Sauter. 2003b. Temporal aspects of dynamic role assignment. In *Proceedings of the 4th International Workshop on Agent-Oriented Software Engineering (AOSE 2003), Melbourne, Australia.*

Olive, A. 1983. Analysis of Conceptual and Logical Models in Information Systems Design Methodologies. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Olle, T.W., H.G. Sol, and C.J. Tully, eds. 1983. *Information Systems Design Methodologies - A Feature Analysis*. Amsterdam: Elsevier Science Publishers.

O'Malley, S.A., and S.A. DeLoach. 2001. Determining When to Use an Agent-Oriented Software Engineering Paradigm. In *Proceedings of the 2nd Workshop on Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada*, 188-205.

Omicini, A. 2000. SODA: Societies and Infrastructure in the Analysis and Design of Agent-Based Systems. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 185-194.

Omicini, A., and E. Denti. 2001. From tuple spaces to tuple centres. *Science of Computer Programming* 41(3): 277-294.

Omicini, A., and F. Zambonelli. 1999. Coordination for Internet Application Development. *Autonomous Agents and Multi-Agent Systems* 2(3): 251-269.

Omicini, A., E. Denti, and A. Natali. 1995. Agent coordination and control through logic theories. In *Proceedings of the 4th Congress of the Italian Association for Artificial Intelligence on Topics in Artificial Intelligence, Florence, Italy*, 439 - 450.

Padgham, L., and M. Winikoff. 2002a. Prometheus: A methodology for developing intelligent agents. In *Proceedings of 3rd International Workshop on Agent-Oriented Software Engineering (AOSE-2002), Bologna, Italy.*

Padgham, L., and M. Winikoff. 2002b. Prometheus: A pragmatic methodology for engineering intelligent agents. In *Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA*, 97-108.

Papadopoulos, G.A. 2001. Models and technologies for the coordination of Internet agents: A survey. In *Coordination of Internet Agents: Models, Technologies, and Applications*, ed. A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, 25-56. London: Springer-Verlag.

Papadopoulos, G.A., and F. Arbab. 1998. Coordination models and languages. *Advances in Computers* 46: 329-400.

Parent, C., and S. Spaccapietra. 1998. Issues and approaches of database integration. *Communications of the ACM* 41(5): 166-178.

Parrott, L, R. Lacroix, and K. M. Wade. 2003. Design considerations for the implementation of multi-agent systems in the dairy industry. *Computers and Electronics in Agriculture* 38(2): 79-98.

Pavon, J., and J. Gomez-Sanz. 2003. Agent Oriented Software Engineering with INGENIAS. In *Proceedings of 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic,* 394-403.

Pavon, J., J. Gomez-Sanz, and R. Fuentes. 2005. The INGENIAS Methodology and Tools. In *Agent-Oriented Methodologies*, ed. B. Henderson-Sellers and P. Giorgini. Pennsylvania: Idea Group Publishing (in press).

Pazzaglia, J-C. R., and S.M. Embury. 1998. Bottom-up Integration of Ontologies in a Database Context. In *Proceedings of the 5thInternational Workshop on Innovative Application Programming and Query Interfaces, Seattle, USA*, 7.1-7.7.

Pednault, E. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Canada*, 324-332.

Picco, G.P., A.L. Murphy, and G.-C. Roman. 1999. LIME: Linda meets mobility. In *Proceedings of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, USA*, 368-377.

Poggi, A., G. Rimassa, and P. Turci. 2002. Engineering CoMMA Multiagent System with Agent UML. In *Proceedings of the Workshop from Objects to Agents*, *Milan, Italy*.

Potts, C. 1999. ScenIC: A Strategy for Inquiry-Driven Requirements Determination. In *Proceedings of the 4th IEEE International Symposium on Requirements Engineering, Limerick, Ireland*, 58-65.

Rao, A.S., and M.P. Georgeff. 1991. Modelling rational agents within a BDI architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, *Cambridge, USA*, 473-484.

Rao, A.S., and M.P. Georgeff. 1995. BDI agents: from theory to practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA*, 312-319.

Richards, D. 2000. The Reuse of Knowledge: A User-Centered Approach. *International Journal of Human Computer Studies* 52(3): 553-579.

Robinson, D.J. 2000. *A Component Based Approach to Agent Specification*. Master thesis, Air Force Institute of Technology, USA.

Rolland, C., C. Souveyet, and C.B. Achour. 1998. Guiding Goal Modeling Using Scenarios. *IEEE Transactions on Software Engineering* 24(12): 1055-1071.

Russell, S., and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*. 2nd ed. New Jersey: Prentice Hall.

Russell, S., and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach*. 2nd ed. New Jersey: Prentice Hall.

Sabas, A., M. Badri, and S. Delisle. 2002. A Multidimensional Framework for the Evaluation of Multiagent System Methodologies. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI-2002), Orlando, USA*, 211-216.

Sargent, P. 1992. Back to school for a brand new ABC. *The Guardian*, March 12: 28.

Sathi, A., and M.S. Fox. 1989. Constraint-directed negotiation of resource reallocations. In *Distributed Artificial Intelligence II*, ed. L. Gasser and M.N. Huhns. London: Pitman/Morgan Kaufman.

Schreiber, A.T., B. J. Wielinga, R. de Hoog, J. M Akkermans, and W. Van de Velde. 1994. CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert* 9(6): 28-37.

Shave, M.J.R. 1997. Ontological Structures for Knowledge Sharing. *New Review of Information Networking* 3: 125-133.

Shehory, O., and A. Sturm. 2001. Evaluation of modeling techniques for agent-based systems. In *Proceedings of the 5th International Conference on Autonomous agents, Montreal, Canada*, 624-631.

Shen, W., and D.H. Norrie. 1999. Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems* 1(2): 129-156.

Shen, W., D.H. Norrie, and R. Kremer. 1999. Towards an Infrastructure for Internet Enabled Collaborative Agent Systems. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Canada.

Sheth, A.P., and J.A. Larson. 1990. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys* 22(3): 183-236.

Shoham, Y. 1993. Agent-oriented programming. *AI* 60(1): 139-159.

Shoham, Y., and M. Tennenholtax. 1992. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the 10th National Conference on Artificial Intelligence*, Menlo Park, California, 276-281.

Shoham, Y., and S.B. Cousins. 1994. Logics of mental attitudes in AI: A very preliminary survey. In *Foundations of Knowledge Representation and Reasoning*, ed. G. Lakemeyer and B. Nebel, 296-309. Berlin, Heidelberg: Springer Verlag.

Siau, K., and M. Rossi. 1998. Evaluation of Information Modeling Methods – A Review. In *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, *Hawaii, USA,* 314-322.

Silva, V., A. Garcia, A. Brandao, C. Chavez, C. Lucena, and P. Alencar. 2003. Taming Agents and Objects in Software Engineering. In *Software Engineering for Large-Scale Multi-Agent Systems – Lecture Notes in Computer Science*, ed. A. Garcia, C. Lucena, J. Castro, A. Omicini, and F. Zambonelli, 1-26. Springer Verlag.

Silva, V.T.D., and C.J.P.D. Lucena. 2004. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language. *Autonomous Agents and Multi-Agent Systems* 8: 1-45.

Singh, D. 2000. *An agent based architecture for query planning and cost modelling of web sources*. Master Thesis, University of Georgia.

Sowa, J.F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove: Brooks/Cole Thompson Learning.

Standards Australia. 2004. Standard metamodel for software development methodologies (AS-4651-2004). http://www.standards.com.au/.

Steep, R., S. Cammarata, F.A. Hayes-Roth, P.W. Thorndyke, and R.B. Wesson. 1981. *Architectures for distributed intelligence for air fleet control*. Technical report R-2728-ARPA, Rand Corporation, Santa Monica, California.

Studer, R., H. Eriksson, J. H. Gennari, S. Tu, D. Fensel, and M. Musen. 1996. Ontologies and the Configuration of Problem-Solving Methods. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada*.

Stone, P., and M. Veloso. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots* 8(1): 345-383.

Stumme, G., and A. Maedche. 2001. Ontology merging for federated ontologies on the semantic web. In *Proceedings of Workshop on Ontologies and Information Sharing (IJCAI' 01), Seattle, USA*.

Sugumaran, V., and V.C. Storey. 2001. Creating and Managing Domain Ontologies for Database Design. In *Proceedings of the 6th International Workshop on Applications of Natural Language to Information Systems*, *Madrid, Spain*, 17-26.

Sundsted, T. 1998. An introduction to agents. *Javaworld*, June 6. http://www.javaworld.com/javaworld/jw-06-1998/jw-06-howto.html. (accessed August 25, 2002).

Sycara, K. 1998a. Resolving goal conflict via negotiation. In *Proceedings of the 7th National Conference on Artificial Intelligence, Minnesota, USA*, 245-250.

Sycara, K. 1998b. Multiagent Systems. *AI Magazine* 19(2): 79-92.

TACOMA – Tromso and Cornell Moving Agents. Tromso: University of Tromso. http://www.tacoma.cs.uit.no/. (accessed July 10, 2003).

Tahara, Y., A. Ohsuga, and S. Honiden. 1999. Agent system development based on agent patterns. In *Proceedings of the 21st International Conference on Software Engineering, California, USA*, 356–367.

Tamma, V., M. Wooldridge, and I. Dickinson. 2002a. An ontology for automated negotiation. In *Proceedings of the International Workshop on Ontologies in Agent Systems* (OAS'02), Bologna, Italy.

Tamma, V., M. Wooldridge, and I. Dickinson. 2002b. An ontology based approach to automated negotiation. In *Proceedings of the 4th Workshop on Agent Mediated Electronic Commerce (AMEC IV)*, Bologna, Italy.

Tamma, V., S. Phelps, I. Dickinson, and M. Wooldridge. 2005. Ontologies for supporting negotiation in e-commerce. *Engineering Applications of Artificial Intelligence* 18: 223-236.

Taveter, K. 1998. Intelligent Information Retrieval Based on Interconnected Concepts and Classes of Retrieval Domains. In *Proceedings of the 8th DELOS Workshop User Interface in Digital Libraries, Stockholm, Sweden*, 39-43.

Telecom Italia Lab. 2004. *Java Agent Development Framework – an Open Source platform for peer-to-peer agent based applications*. http://jade.tilab.com/. (accessed August 14, 2002).

Thangarajah, J., L. Padgham, and J. Harland. 2002. Representation and Reasoning for Goals in BDI Agents. In *Proceedings of the twenty-fifth Australasian conference on Computer science - Volume 4, Melbourne, Australia,* 259–265.

Tolkdorf, R. 1997. Berlinda: an object-oriented platform for implementing coordination languages in Java. In *Proceedings of the 2nd International Conference on Coordination Languages and Models, Berlin, Germany*, 430-433

Tout, H. 2001. An Informational Model for Cooperative Information Gathering. In *Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal, Canada*.

Tran, Q.N., G. Low, and M.A. Williams. 2003. A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies. In *Proceedings of the 14th International Symposium on Methodologies for Intelligent Systems ISMIS'03, Maebashi, Japan*, 613-617.

Tran, Q.N., G. Low, and M.A. Williams. 2004. A Preliminary Comparative Feature Analysis of Multi-agent Systems Development Methodologies. In *Proceedings of the 6th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2004), Riga, Latvia*, 386-397.

Tran, Q.N., and G. Low. 2005. Comparison of Methodologies. In *Agent-Oriented Methodologies*, ed. B. Henderson-Sellers and P. Giorgini. Pennsylvania: Idea Group Publishing (in print).

Tveit, A. 2001. A survey of Agent-Oriented Software Engineering. In *Proceedings of the 1st NTNU Computer Science Graduate Student Conference, University of Science and Technology, Norway*.

UMBC Lab for Advanced Information Technology. n.d.a. *UMBC KQML Web*. http://www.cs.umbc.edu/kqml/. (accessed January 6, 2003).

UMBC Lab for Advanced Information Technology. n.d.b. *UMBC AgentNews*. http://agents.umbc.edu/agentnews/ (accessed April 22, 2003).

UMBC Lab for Advanced Information Technology. n.d.c. *The Software Agents Mailing List*. http://www.csee.umbc.edu/agentslist/ (accessed May 5, 2003).

Uschold, M., and M. Gruninger. 1996. Ontologies: principles, methods, and applications. *Knowledge Engineering Review* 11(2): 93-155.

Uschold, M., and M. King. 1995. Towards A Methodology for Building Ontologies. In *Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada*.

Valckenaers, P., H. Van Brussel, H. Karuna, M. Kollingbaum, O. Bochmann. 2002. Multi-Agent Manufacturing Control Using Stigmergy. In *Proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain*.

van Breemen, A.J.N. 2002. Integrating Agents in Software Applications. In *Proceedings of Workshops at Net.Objectdays, Erfurt, The Netherlands*, 278-289.

van Heijst, G., A. Schreiber, and B. Wielinga. 1997. Using Explicit Ontologies in KBS Development. *International Journal of Human computer studies* 46: 183-292.

van Lamsweerde, A. 2001. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, *Toronto, Canada*, 249-263.

Vere, S., and T. Bickmore. 1990. A Basic Agent. *Computational Intelligence* 6: 41-60.

Vidal, J.M., P.A. Buhler, and M.N. Huhns. 2001. Inside an Agent. IEEE Internet Computing 5(1): 82-86.

Vlado, K. 1998. *Multi-agent systems for Internet information retrieval using natural language processing*. Master thesis., University of Waterloo.

Wache, H., U. Visser, U., and T. Scholz. 2001. Ontology construction – an iterative and dynamic task. In Proceedings *of the 15th International Florida Artificial Intelligence Research Society Conference, Florida, USA*, 445-449.

Wasserman, A.I., Freeman, P. and M. Porcella. 1983. Characteristics of Software Development Methodologies. In *Information Systems Design Methodologies - A Feature Analysis*, ed. T.W. Olle, H.G. Sol and C.J. Tully, 63-85. Amsterdam: Elsevier Science Publishers.

Wegner, P. 1996. Coordination as constrained interaction. In *Proceedings of the 1st International Conference on Coordination Languages and Models (COORDINATION '96)*, *Cesena, Italy,* 28-33.

Weiss, G., eds. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Massachusetts: MIT Press.

Weyns, D., and T. Holvoet. 2003. Synchronous versus Asynchronous Collaboration in Situated Multi-agent Systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia*, 1156-1157.

Weyns, D., H.V.D. Parunak, F. Michel, T. Holvoet, and J. Ferber. 2004. Environments for Multiagent Systems State-of-the-Art and Research Challenges. In *Proceedings of the 1st International Workshop on Environments for Multiagent Systems, New York, USA*.

Wiegers, K. 2003. *Software Requirements*. 2nd ed. Washington: Microsoft Press.

Winikoff, M., and L. Padgham. 2004. The Prometheus Methodology. In *Methodologies and Software Engineering for Agent Systems*. *The Agent-Oriented Software*

Engineering handbook, ed. F. Bergenti, M.P. Gleizes, and F. Zambonelli, Chapter 11. Kluwer Academic Publishers.

Winikoff, M., L. Padgham, and J. Harland. 2001. Simplifying the Development of Intelligent Agents. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, *Adelaide, Australia*, 557-568.

Wood, S. 1993. *Planning and Decision Making in Dynamic Domains*. Chchester: Ellis Horwood.

Wood, M. F. 2000. *Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems*. Master thesis, Air Force Institute of Technology, USA.

Wood, M., and S.A. DeLoach. 2000a. An Overview of the Multiagent Systems Engineering Methodology. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland*, 207-221.

Wood, M., and S.A. DeLoach. 2000b. Developing Multiagent Systems with agentTool. In *Proceedings of the 7th International Workshop on Agent Theories, Architectures, and Languages, Boston, USA*, 46-60.

Wood, B., R. Pethia, L.R. Gold, and R. Firth. 1988. *A Guide to the Assessment of Software Development Methods*. Technical Report CMUSEI-88-TR-8, SEI, Software Engineering Institute, Carnegie Mellon University.

Wooldridge, M. 1997. Agent-based software engineering. *IEEE Proceedings on Software Engineering* 144(1): 26-37.

Wooldridge, M. 1999. Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. G. Weiss, 27-77. London: The MIT Press.

Wooldridge, M. 2002. *An Introduction to MultiAgent Systems*. Chichester: John Wiley & Sons.

Wooldridge, M., and N.R. Jennings. 1995. Agent Theories, Architectures and Languages: a survey. In *Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents, Amsterdam, The Netherlands*, 1-39.

Wooldridge, M., and N.R. Jennings. 1998. Pitfalls of Agent-Oriented Development. In *Proceedings of the 2nd International Conference on Autonomous Agents*, *Minneapolis, USA*, 385-391.

Wooldridge, M., and P. Ciancarini. 2000. Agent-Oriented Software Engineering: The State of the Art. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000)*, *Limerick, Ireland*, 1-28.

Wooldridge, M., N.R. Jennings, and D. Kinny. 1999. A Methodology for Agent-Oriented Analysis and Design. In *Proceedings of the 3rd International Conference on Autonomous Agents (Agents '99), Seattle, Washington*, 69-76.

Wooldridge, M., N.R. Jennings, and D. Kinny. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems* 3(3): 285-312.

Wray, R., R. Chong, J. Phillips, S. Rogers, and B. Walsh. n.d. *A Survey of Cognitive and Agent Architectures*. http://ai.eecs.umich.edu/cogarch0/index.html. (accessed April 20, 2002)

Wu, X., and A.C. Esterline. 1999. Representing Multi-agent Plans Using Statecharts with Explicit Aggregation. In *Proceedings of ADMI-99 Minority Institutions Computing Conference, Duluth, USA*.

Xu, D., R. Volz, T. Ioerger, and J. Yen. 2002. Modeling and verifying multi-agent behaviors using predicate/transition nets. In *Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering, Ischia, Italy*, 193-200.

Yan, Q., L.J. Shan, and X.J. Mao. 2003. *RoMAS: A Role-Based Modeling Method for Multi-Agent System*. In *Proceedings of International Conference on Active Media Technology, Chongqing, China.*

Yourdon, E. 1989. *Modern Structured Analysis*. Englewood Cliffs, New Jersey: Yourdon Press.

Yu, E. 1995. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Canada.

Yuan, S.T. 1999. Ontology-Based Agent Community for Information Gathering and Integration. *Proceedings of the National Science Council: Physical Science and Engineering (Part A)* 23(6): 766-780.

Zambonelli, F. 2000. Organisational Abstractions for the Analysis and Design of Multi-Agent Systems. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, *Limerick, Ireland*, 127-141.

Zambonelli, F., N. Jennings, and M. Wooldridge. 2001a. Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems. *International Journal of Software Engineering and Knowledge Engineering* 11(3): 303-328.

Zambonelli F., N. Jennings, and M. Wooldridge. 2003. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology* 12(3): 317-370.

Zambonelli, F., N.R. Jennings, A. Omicini, and M. Wooldridge. 2001b. Agent-oriented software engineering for Internet applications. In *Coordination of Internet Agents*, ed. A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, 326-346. New York: Springer-Verlag.

Zhou, J., M. Zhou, and Q. Wu. 2000. An Agent Framework Based on Distributed Object. In *Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Asia'00), Xi'an, China*, 188-194.

APPENDIX A

ADVERTISEMENT FOR SURVEY RECRUITMENT

Dedicated Agent Researchers and Developers Needed!

If you have knowledge and/or experience in agent-oriented software engineering, please take your time to complete a Multi-Agent Development Methodology Survey, which is available at

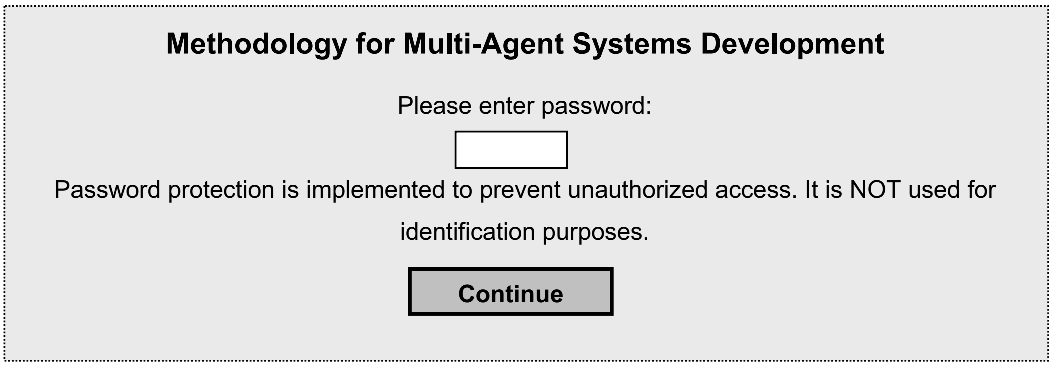
http://129.94.244.146/personal/numi+tran/surveyq.nsf/survey/.
Access password: MAS

The survey is part of a doctoral research project and its purpose is to gather your professional opinions and suggestions on what generic features, process steps and modelling concepts should be part of a methodology for developing Multi-Agent Systems. The features, steps and modelling concepts must be ranked and rated with regard to their importance. The survey can be completed in stages and will take approximately 30-40 minutes to finish. The Closing Date is 31 Jan 2003.

The survey is demanding, but the acquired information will prove to be invaluable to the Agent community. Participation is completely voluntary and you can remain anonymous. Contact: Quynh Nhu Numi Tran mailto:numitran@unsw.edu.au.

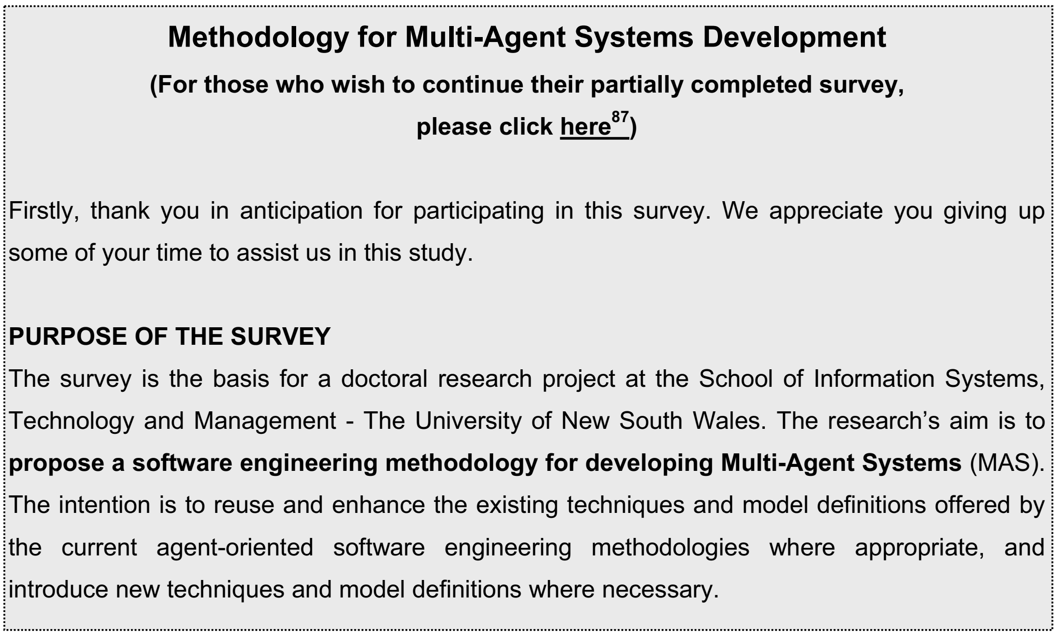
APPENDIX B
ONLINE SURVEY QUESTIONNAIRE

START-UP PAGE



Password protection is implemented to prevent unauthorized access. It is NOT used for identification purposes.

WELCOME PAGE[86]



Methodology for Multi-Agent Systems Development

(For those who wish to continue their partially completed survey, please click here[87])

Firstly, thank you in anticipation for participating in this survey. We appreciate you giving up some of your time to assist us in this study.

PURPOSE OF THE SURVEY

The survey is the basis for a doctoral research project at the School of Information Systems, Technology and Management - The University of New South Wales. The research's aim is to **propose a software engineering methodology for developing Multi-Agent Systems** (MAS). The intention is to reuse and enhance the existing techniques and model definitions offered by the current agent-oriented software engineering methodologies where appropriate, and introduce new techniques and model definitions where necessary.

[86] This page is loaded when button "Continue" in "Start-up page" is clicked.
[87] This is a hyperlink which when clicked will load "Survey Return page".

The survey aims to gather your professional opinions and suggestions on what **features**, **steps** and **modelling concepts** should be supported by an Agent-Oriented Software Engineering (AOSE) methodology for developing MAS.

If you wish, the findings of the survey and the final results of the research will be forwarded to you when available.

WHO SHOULD PARTICIPATE IN THE SURVEY

- Project managers, system analysts, system designers or system developers who have been involved in developing at least one MAS.
- Researchers/academics whose area of interest is MAS development.

RESEARCH CONTACTS

If you have any questions on the survey, please contact:

Miss **Quynh-Nhu (Numi) Tran**
School of Information Systems, Technology and Management
The University of New South Wales
numitran@unsw.edu.au

Prof. **Graham Low**
Head of School
School of Information Systems, Technology and Management
The University of New South Wales
g.low@unsw.edu.au

Continue

INSTRUCTIONS PAGE[88]

Methodology for Multi-Agent Systems Development

GUIDELINES FOR QUESTIONNAIRE

The survey questionnaire consists of 5 parts.

- **Part 1** collects your demographic and background information.
- **Part 2** gathers your opinions on a list of *features* in terms of how important these features are to a “standard” MAS development methodology.
- **Part 3** seeks your opinions on a list of *steps* with regard to how important these steps are to a “standard” MAS development process.

[88] This page is loaded when button “Continue” on “Welcome page” is clicked.

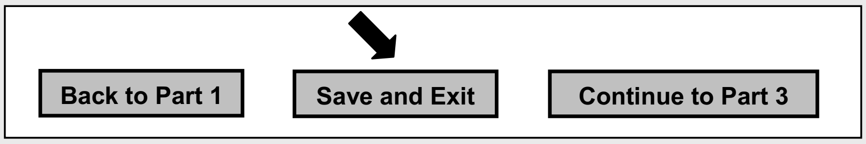
- **Part 4** obtains your opinions on a list of *concepts* with respect to how important these concepts are to models of a "standard" MAS development methodology.
- **Part 5** asks for your recommendations on various issues relating to the construction of a MAS development methodology.

The whole questionnaire takes approximately 30-40 minutes in total to complete.

IMPORTANT NOTES

1. You do NOT have to complete the whole questionnaire in one go. After starting the survey, you can leave the questionnaire at any point and come back later for further completion.

To save a partially completed questionnaire, you just need to click on the button "**Save and Exit Survey**" at the end of each part (as shown on the picture below).



When you leave a partially completed survey, you will be given an ID Number which allows you to return to the questionnaire later. You can save and go back to your partially completed questionnaire as many times as you like until you finish the survey.

2. It is required to **enable Javascript** on your browser to allow the questionnaire to function.

3. Please navigate between the questionnaire parts using the **navigation buttons at the end of each part**, and ***not*** the browser's "Back" and "Forward".

Start

SURVEY PART 1 PAGE[89]

Methodology for Multi-Agent Systems Development

PART 1

This part of the survey questionnaire aims to gather some background information about you and your experience with Multi-Agent Systems (MAS) and Multi-Agent System development. You can remain anonymous if you wish.

[89] This page is loaded when button "Start" on "Instructions page" is clicked.

If you would like to remain anonymous, please tick below:

☐ Anonymous

1. Name: ☐
2. Organisation: ☐
3. Department: ☐
4. Email: ☐
   (required if wish to receive feedback on the survey's findings and/or final result of the research)
5. Please tick if you would like to be informed of

   ☐ Survey's findings ☐ Research's final result (i.e. documentation of the proposed MAS development methodology

Please provide the following information even if you wish to remain anonymous.

6. What is your IT role of work? (multiple choices are allowed)

   ☐ Project manager ☐ Programmer
   ☐ System analyst ☐ Researcher/Academic
   ☐ System developer/developer ☐ Other. Please specify below ☐

7. How would you describe your current theoretical knowledge of Multi-Agent Systems (MAS)?

   Low ○1 ○2 ○3 ○4 ○5 ○6 ○7 Extensive

8. How would you describe your current industrial experience with MASs?

   Low ○1 ○2 ○3 ○4 ○5 ○6 ○7 Extensive

9. How would you describe your current theoretical knowledge of MAS development?

   Low ○1 ○2 ○3 ○4 ○5 ○6 ○7 Extensive

10. How would you describe your current industrial experience with MAS development?

   Low ○1 ○2 ○3 ○4 ○5 ○6 ○7 Extensive

11. Have you been involved in developing any MAS? ○ Yes ○ No

   If Yes,

   11a. How many MAS development projects have you been involved with?

   ☐

   11b. What is the number of agents in these MASs? (multiple choices are allowed)

   ☐ Fewer than 10 ☐ 10-50 ☐ 51-99 ☐ 100 or more

   11c. How do you perceive the average level of complexity of these MASs? (e.g. in terms of agent cognitive ability and intelligence, agent interactions and system dynamics)

   Very low ○1 ○2 ○3 ○4 ○5 ○6 ○7 Very high

   11d. What is/are the application area(s) of these MASs? (multiple choices are allowed)

   ☐ Personal Assistance ☐ Information Gathering & Management
   ☐ Electronic Commerce ☐ Simulation
   ☐ Automated Control/Monitoring ☐ System and Network Management
   ☐ Others. Please specify ☐

11e. Did you follow any system development methodology in any of these projects?

○ Yes ○ No

If yes, what is/are the methodology(ies)? (Please provide as many details as you can on the methodology's names, authors and/or references)

11f. Have you been involved in developing Ontology-Based MASs (i.e. MASs whose design specification explicitly includes ontologies, and ontologies are used by agents at run-time to facilitate the operation of MAS)

○ Yes ○ No

Save and Exit **Continue to Part 2**

(4 more parts to go)

SURVEY PART 2 PAGE[90]

Methodology for Multi-Agent Systems Development

PART 2

This part of the survey questionnaire aims to gather your opinions and suggestions on *what generic features should be offered* by an AOSE methodology for MAS development. (From here on, the term "AOSE methodology" will be used to mean "AOSE methodology for MAS development").

INSTRUCTIONS

Below you will find a list of features that may be offered by an AOSE methodology. Although all of them are important, some of the features are more important to be supported by an AOSE methodology than the others. Thus, we ask for your opinion on this prioritisation.

Desirable features of an AOSE process

Please order rank the following features in terms of their importance to be provided by an AOSE process. Please also indicate the rating of their importance.

Note: Please try to give each feature a unique ranking. But if you cannot differentiate, features can be ranked equally.

[90] This page is loaded when button "Continue to Part 2" on "Survey Part 1 page" is clicked.

	Most important 1st	2nd	3rd	4th	5th	*Least important* 6th	*Rating of importance*
1. Specification of a system development lifecycle	○	○	○	○	○	○	
2. Support for verification and validation (more)[92]	○	○	○	○	○	○	
3. Specification of steps for the development process	○	○	○	○	○	○	
4. Specification of models and/or notational components to be generated from each process step	○	○	○	○	○	○	
5. Specification of techniques and heuristics for performing each process step and for producing each model	○	○	○	○	○	○	
6. Support for refinability (more)	○	○	○	○	○	○	

Desirable features of AOSE model definitions

Please order rank the following features in terms of their importance to be provided by AOSE model definitions. Please also indicate the rating of their importance.

Note: Please try to give each feature a unique ranking. But if you cannot differentiate, features can be ranked equally.

	Most important 1st	2nd	3rd	4th	5th	6th	7th	*Least important* 8th	*Rating of importance*
1. High degree of completeness/expressiveness (more)	○	○	○	○	○	○	○	○	
2. High degree of formalisation/preciseness (more)	○	○	○	○	○	○	○	○	
3. Provision of guidelines/logics for model derivation (more)	○	○	○	○	○	○	○	○	
4. Guarantee of consistency (more)	○	○	○	○	○	○	○	○	
5. Support for modularity (more)	○	○	○	○	○	○	○	○	
6. Manageable number of concepts in each model and each notational component	○	○	○	○	○	○	○	○	
7. Models expressed at various levels of abstraction and detail	○	○	○	○	○	○	○	○	
8. Support for reuse	○	○	○	○	○	○	○	○	

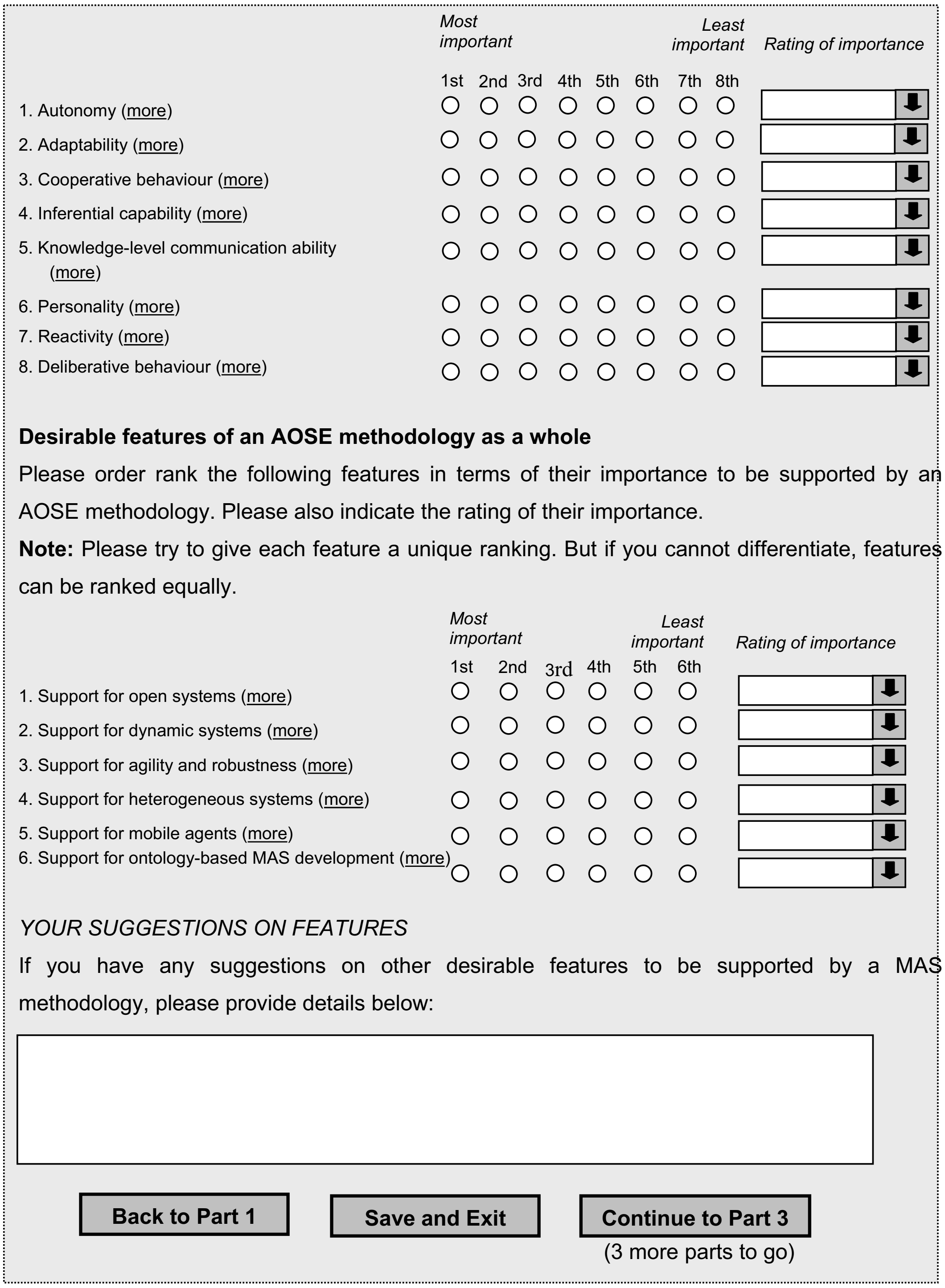
Agent properties desirable to be captured/represented by AOSE models

Please order rank the following agent properties in terms of their importance to be captured/represented by AOSE models. Please also indicate the rating of their importance.

Note: Please try to give each agent property a unique ranking. But if you cannot differentiate, agent properties can be ranked equally.

[91] This is a combo box which contains 5 possible ratings: "Very High", "High", "Medium", "Low" and "Very Low".

[92] "more" is a programmed hyperlink which when clicked will open a small pop-up screen to show more explanation about a particular feature.



	Most important							*Least important*	*Rating of importance*
	1st	2nd	3rd	4th	5th	6th	7th	8th	
1. Autonomy (more)	○	○	○	○	○	○	○	○	
2. Adaptability (more)	○	○	○	○	○	○	○	○	
3. Cooperative behaviour (more)	○	○	○	○	○	○	○	○	
4. Inferential capability (more)	○	○	○	○	○	○	○	○	
5. Knowledge-level communication ability (more)	○	○	○	○	○	○	○	○	
6. Personality (more)	○	○	○	○	○	○	○	○	
7. Reactivity (more)	○	○	○	○	○	○	○	○	
8. Deliberative behaviour (more)	○	○	○	○	○	○	○	○	

Desirable features of an AOSE methodology as a whole

Please order rank the following features in terms of their importance to be supported by an AOSE methodology. Please also indicate the rating of their importance.

Note: Please try to give each feature a unique ranking. But if you cannot differentiate, features can be ranked equally.

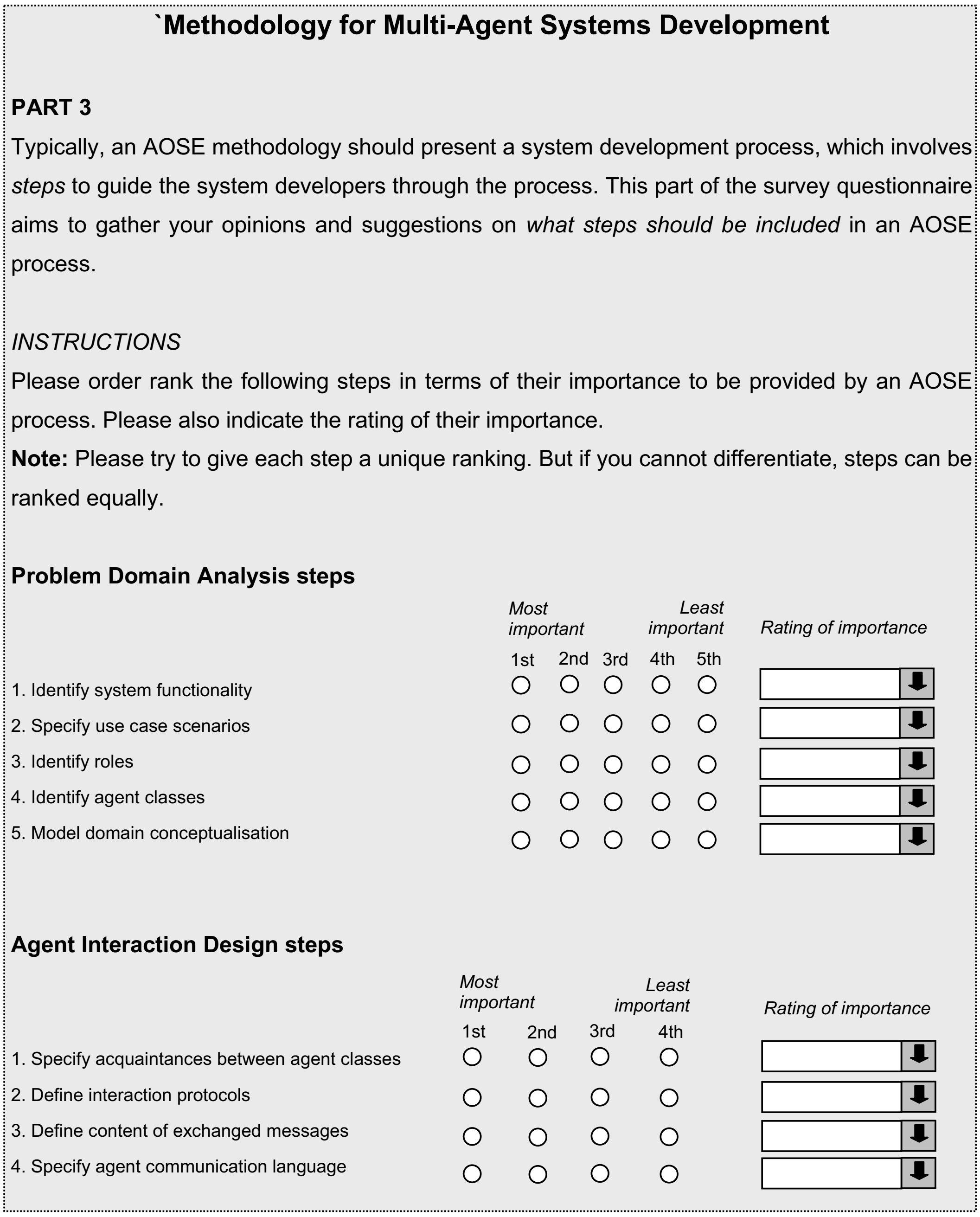
	Most important					*Least important*	*Rating of importance*
	1st	2nd	3rd	4th	5th	6th	
1. Support for open systems (more)	○	○	○	○	○	○	
2. Support for dynamic systems (more)	○	○	○	○	○	○	
3. Support for agility and robustness (more)	○	○	○	○	○	○	
4. Support for heterogeneous systems (more)	○	○	○	○	○	○	
5. Support for mobile agents (more)	○	○	○	○	○	○	
6. Support for ontology-based MAS development (more)	○	○	○	○	○	○	

YOUR SUGGESTIONS ON FEATURES

If you have any suggestions on other desirable features to be supported by a MAS methodology, please provide details below:

(3 more parts to go)

SURVEY PART 3 PAGE[93]



`Methodology for Multi-Agent Systems Development

PART 3

Typically, an AOSE methodology should present a system development process, which involves *steps* to guide the system developers through the process. This part of the survey questionnaire aims to gather your opinions and suggestions on *what steps should be included* in an AOSE process.

INSTRUCTIONS

Please order rank the following steps in terms of their importance to be provided by an AOSE process. Please also indicate the rating of their importance.

Note: Please try to give each step a unique ranking. But if you cannot differentiate, steps can be ranked equally.

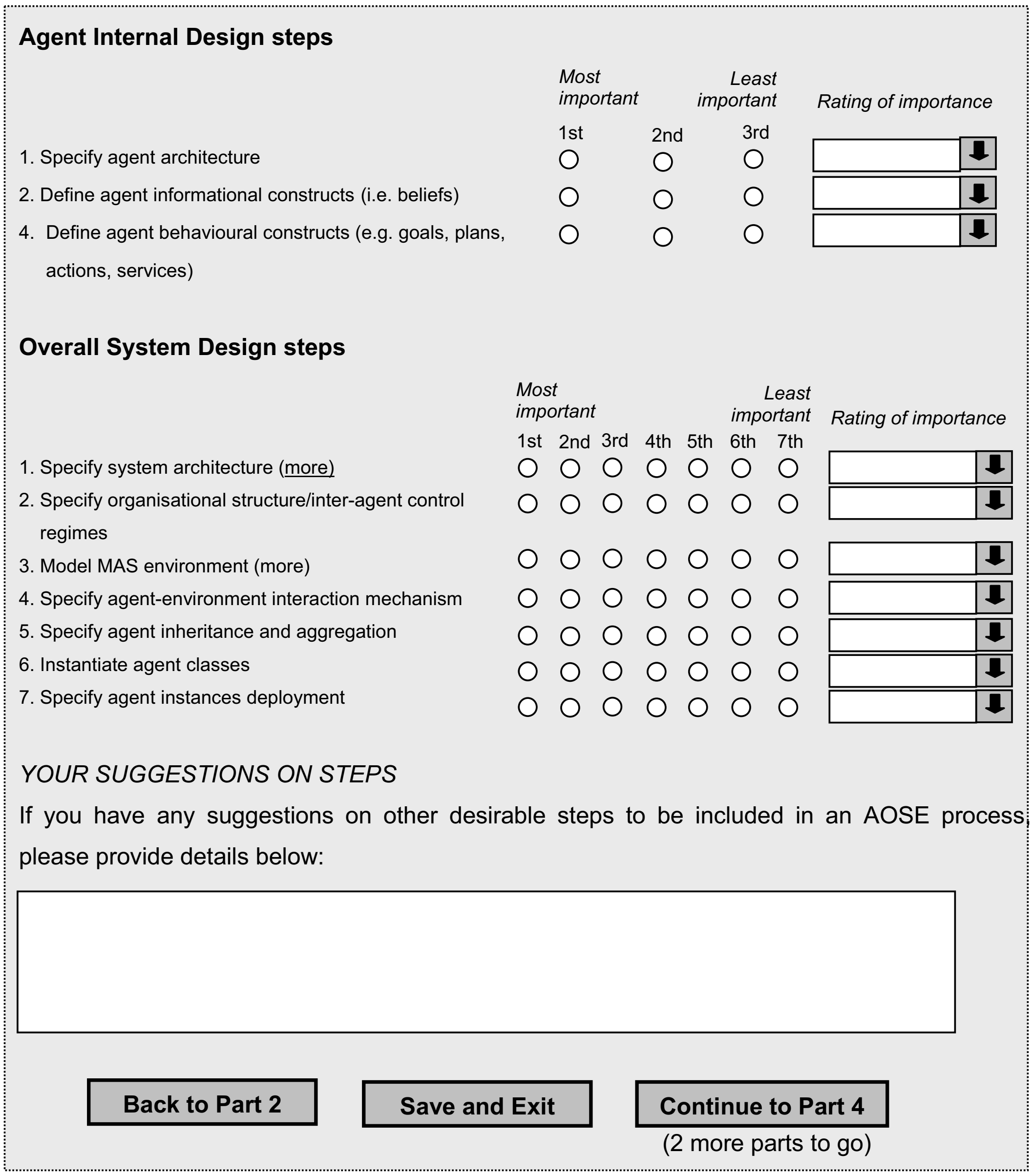
Problem Domain Analysis steps

	Most important				*Least important*	*Rating of importance*
	1st	2nd	3rd	4th	5th	
1. Identify system functionality	○	○	○	○	○	⬇
2. Specify use case scenarios	○	○	○	○	○	⬇
3. Identify roles	○	○	○	○	○	⬇
4. Identify agent classes	○	○	○	○	○	⬇
5. Model domain conceptualisation	○	○	○	○	○	⬇

Agent Interaction Design steps

	Most important			*Least important*	*Rating of importance*
	1st	2nd	3rd	4th	
1. Specify acquaintances between agent classes	○	○	○	○	⬇
2. Define interaction protocols	○	○	○	○	⬇
3. Define content of exchanged messages	○	○	○	○	⬇
4. Specify agent communication language	○	○	○	○	⬇

[93] This page is loaded when button “Continue to Part 3” on “Survey Part 2 page” is clicked.



Agent Internal Design steps

	Most important 1st	2nd	*Least important* 3rd	*Rating of importance*
1. Specify agent architecture	○	○	○	⬇
2. Define agent informational constructs (i.e. beliefs)	○	○	○	⬇
4. Define agent behavioural constructs (e.g. goals, plans, actions, services)	○	○	○	⬇

Overall System Design steps

	Most important 1st	2nd	3rd	4th	5th	6th	*Least important* 7th	*Rating of importance*
1. Specify system architecture (more)	○	○	○	○	○	○	○	⬇
2. Specify organisational structure/inter-agent control regimes	○	○	○	○	○	○	○	⬇
3. Model MAS environment (more)	○	○	○	○	○	○	○	⬇
4. Specify agent-environment interaction mechanism	○	○	○	○	○	○	○	⬇
5. Specify agent inheritance and aggregation	○	○	○	○	○	○	○	⬇
6. Instantiate agent classes	○	○	○	○	○	○	○	⬇
7. Specify agent instances deployment	○	○	○	○	○	○	○	⬇

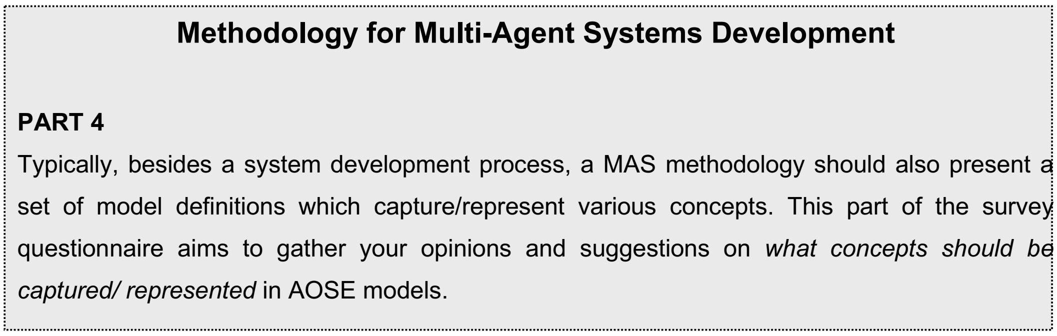
YOUR SUGGESTIONS ON STEPS

If you have any suggestions on other desirable steps to be included in an AOSE process, please provide details below:

Back to Part 2 **Save and Exit** **Continue to Part 4**

(2 more parts to go)

SURVEY PART 4 PAGE[94]



Methodology for Multi-Agent Systems Development

PART 4

Typically, besides a system development process, a MAS methodology should also present a set of model definitions which capture/represent various concepts. This part of the survey questionnaire aims to gather your opinions and suggestions on *what concepts should be captured/ represented* in AOSE models.

[94] This page is loaded when button "Continue to Part 4" on "Survey Part 3 page" is clicked.

INSTRUCTIONS

Please order rank the following concepts in terms of their importance to be captured/ represented in AOSE models. Please also indicate the rating of their importance.

Note: Please try to give each step a unique ranking. But if you cannot differentiate, concepts can be ranked equally.

Problem Domain concepts

	Most important			*Least important*	
	1st	2nd	3rd	4th	*Rating of importance*
1. System functionality	○	○	○	○	⬇
2. Use case scenario	○	○	○	○	⬇
3. Role	○	○	○	○	⬇
4. Domain conceptualisation	○	○	○	○	⬇

Agent concepts

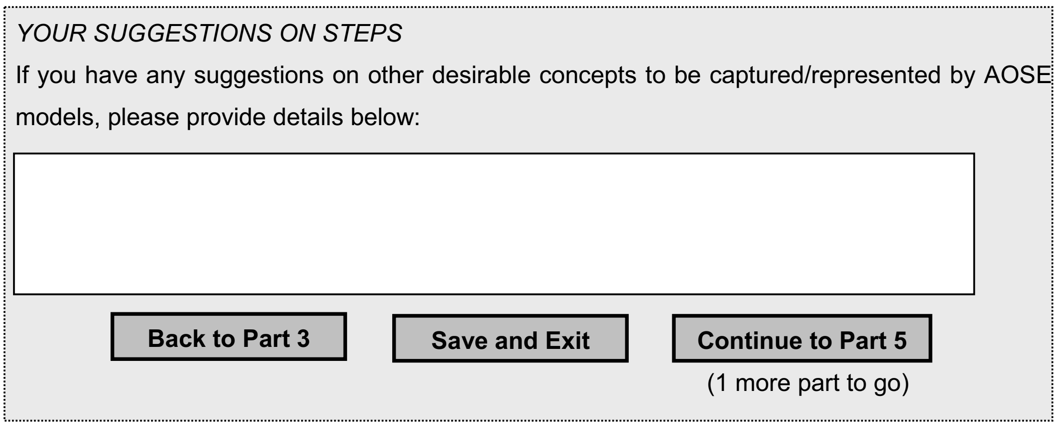
	Most important						*Least important*	
	1st	2nd	3rd	4th	5th	6th	7th	*Rating of importance*
1. Agent-role assignment	○	○	○	○	○	○	○	⬇
2. Agent goal/task	○	○	○	○	○	○	○	⬇
3. Agent belief/knowledge	○	○	○	○	○	○	○	⬇
4. Agent plan/reasoning rule/problem solving method	○	○	○	○	○	○	○	⬇
5. Agent capability/service	○	○	○	○	○	○	○	⬇
6. Agent percept/event	○	○	○	○	○	○	○	⬇
7. Agent architecture	○	○	○	○	○	○	○	⬇

Agent Interaction concepts

	Most important		*Least important*	
	1st	2nd	3rd	*Rating of importance*
1. Agent acquaintance	○	○	○	⬇
2. Interaction protocol	○	○	○	⬇
3. Content of exchanged message	○	○	○	⬇

Overall System Design concepts

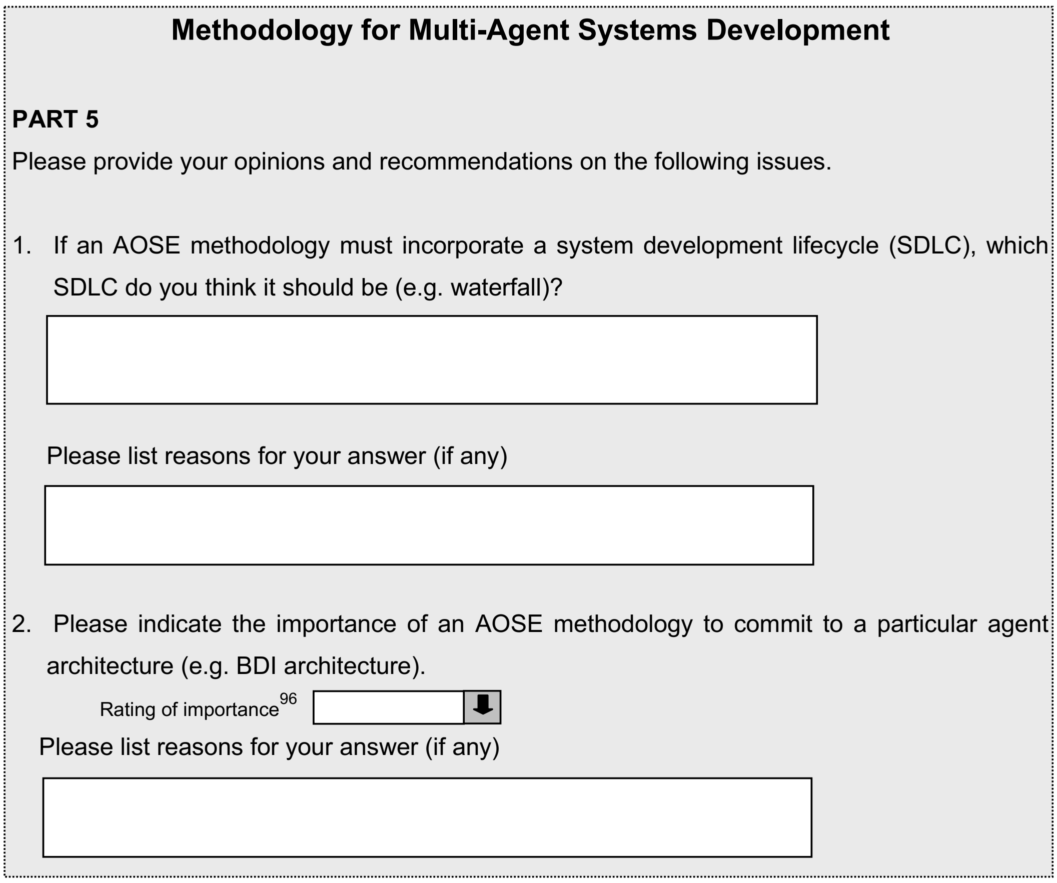
	Most important							*Least important*	
	1st	2nd	3rd	4th	5th	6th	7th	8th	*Rating of importance*
1. System architecture	○	○	○	○	○	○	○	○	⬇
2. Organisational structure/ inter-agent control regimes	○	○	○	○	○	○	○	○	⬇
3. Environment resource/facility	○	○	○	○	○	○	○	○	⬇
4. Agent aggregation relationship	○	○	○	○	○	○	○	○	⬇
5. Agent inheritance relationship	○	○	○	○	○	○	○	○	⬇
6. Agent instantiation	○	○	○	○	○	○	○	○	⬇
7. Agent instance deployment	○	○	○	○	○	○	○	○	⬇



YOUR SUGGESTIONS ON STEPS

If you have any suggestions on other desirable concepts to be captured/represented by AOSE models, please provide details below:

SURVEY PART 5 PAGE[95]



Methodology for Multi-Agent Systems Development

PART 5

Please provide your opinions and recommendations on the following issues.

1. If an AOSE methodology must incorporate a system development lifecycle (SDLC), which SDLC do you think it should be (e.g. waterfall)?

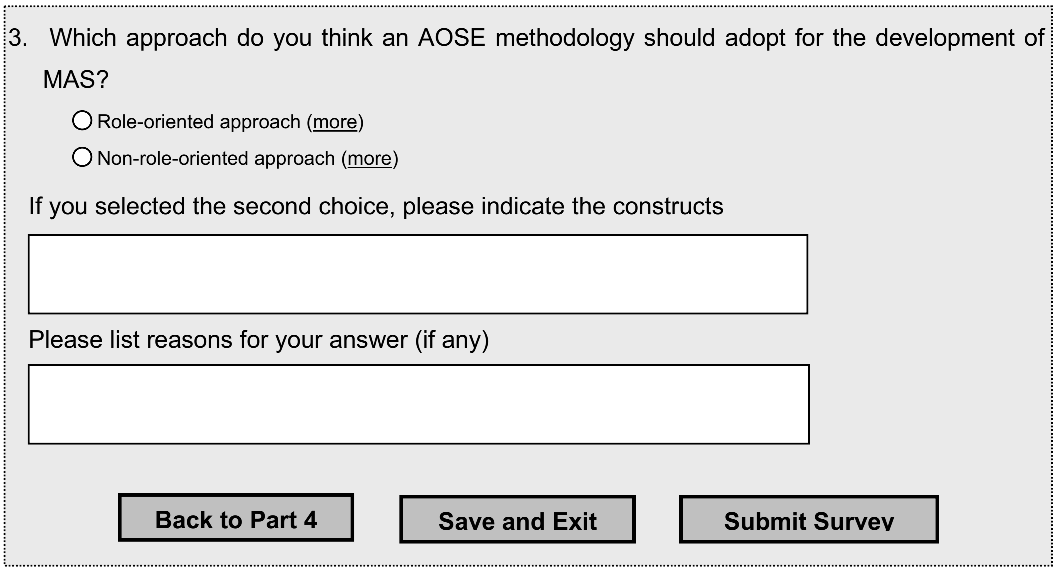
   Please list reasons for your answer (if any)

2. Please indicate the importance of an AOSE methodology to commit to a particular agent architecture (e.g. BDI architecture).

   Rating of importance[96]

   Please list reasons for your answer (if any)

[95] This page is loaded when button "Continue to Part 5" on "Survey Part 4 page" is clicked.
[96] This is a combo box which contains 5 possible ratings: "Very High", "High", "Medium", "Low" and "Very Low".


3. Which approach do you think an AOSE methodology should adopt for the development of MAS?

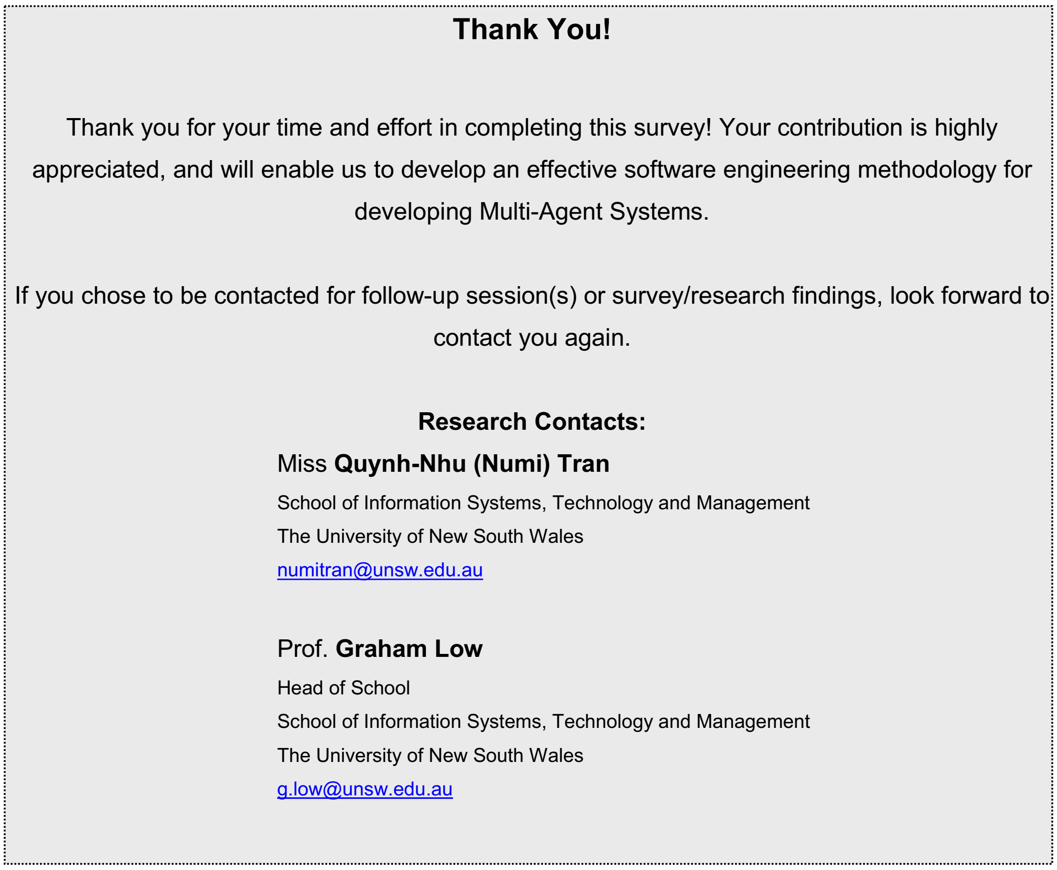
○ Role-oriented approach (more)

○ Non-role-oriented approach (more)

If you selected the second choice, please indicate the constructs

Please list reasons for your answer (if any)

THANK YOU PAGE[97]


Thank You!

Thank you for your time and effort in completing this survey! Your contribution is highly appreciated, and will enable us to develop an effective software engineering methodology for developing Multi-Agent Systems.

If you chose to be contacted for follow-up session(s) or survey/research findings, look forward to contact you again.

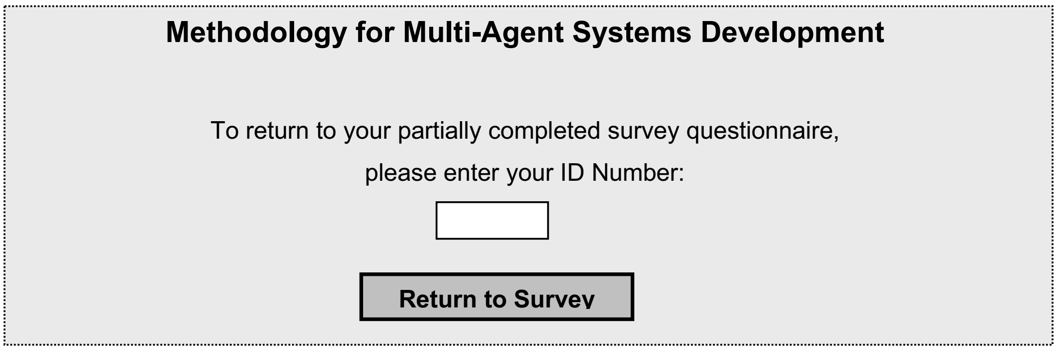
Research Contacts:

Miss **Quynh-Nhu (Numi) Tran**
School of Information Systems, Technology and Management
The University of New South Wales
numitran@unsw.edu.au

Prof. **Graham Low**
Head of School
School of Information Systems, Technology and Management
The University of New South Wales
g.low@unsw.edu.au

[97] This page is loaded when button "Submit Survey" on "Survey Part 5 page" is clicked.

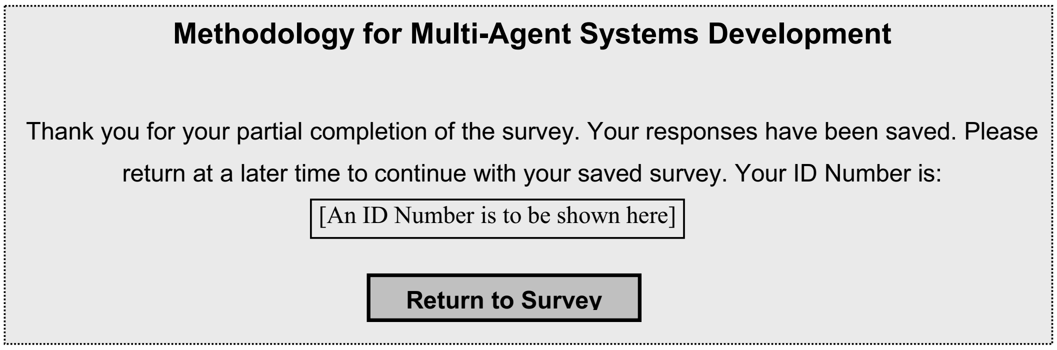
SURVEY RETURN PAGE[98]


Methodology for Multi-Agent Systems Development

To return to your partially completed survey questionnaire,

please enter your ID Number:

SURVEY SAVE AND EXIT PAGE[99]


Methodology for Multi-Agent Systems Development

Thank you for your partial completion of the survey. Your responses have been saved. Please return at a later time to continue with your saved survey. Your ID Number is:

[An ID Number is to be shown here]

[98] This page is loaded when hyperlink "here" on "Welcome page" is clicked. It will direct the respondent back to the survey part which he had partially completed.
[99] This page is loaded whenever button "Save and Exit" on other pages is clicked.

APPENDIX C

DEMOGRAPHIC AND PROFESSIONAL CHARACTERISTICS OF SURVEY RESPONDENTS

This appendix presents the descriptive statistics of seven variables that pertained to the demographic and professional characteristics of survey respondents. The other four demographic variables, namely "*Theoretical knowledge of MAS*", "*Theoretical knowledge of MAS development*", "*Industrial experience with MAS*" and "*Industrial experience with MAS development*", are analysed in Section 5.3.4.1.

Variable "Field of work"

A majority of the respondents worked in the field of research/academia (Figure AppendixC.1). Four respondents were involved in multiple fields, including two who worked as both researcher and system developer/developer, one who worked as both researcher and project manager, and two who worked concurrently as researcher, programmer and system developer/developer.

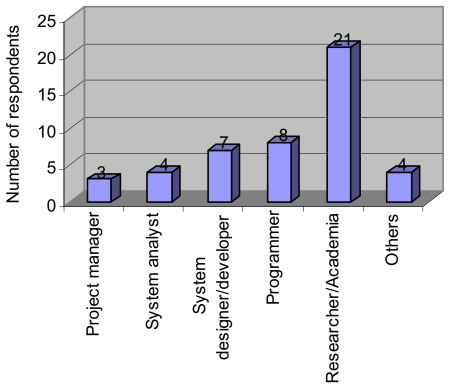


Figure AppendixC.1 – Survey respondents' field of work

Variable “Involvement in MAS development projects”

A high proportion of the respondents (33 out of 41) had participated in at least one MAS development project (Figure AppendixC.2). Out of these respondents, twenty-six had engaged in 1-5 projects, while two had participated in more than 10 projects.

An exploratory analysis was conducted to discover if “*Involvement in MAS development projects*” had any significant impact on the respondents’ “rating of importance” and “order ranking” of features, steps and modelling concepts in Parts 2, 3 and 4 of the survey. For this analysis, Mann-Whitney Tests[100] were performed to compare the “ratings of importance” and “order ranks” from two different groups of respondents – those who had participated in at least one MAS project and those who had not. The comparisons were carried out for all features, steps and modelling concepts. However, no significant difference was detected between the two groups at a significance level of 5%. Consequently, the data obtained from both groups of respondents was combined to form the final survey data set.

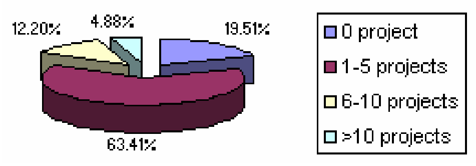


Figure AppendixC.2 – Survey respondents’ involvement in MAS development projects

Variable “Size of past MAS projects”

This variable, as well as the succeeding four variables, were collected from respondents who had been involved in at least one MAS development project (i.e. 33 respondents). Most respondents had developed small-sized to medium-sized MASs, i.e. MASs with less than 10 agents and from 10 to 50 agents respectively (Figure AppendixC.3). Six respondents were involved in multiple MAS projects of different sizes.

[100] Mann-Whitney Test was chosen because it is a well-known test for comparing two independent samples with continuous ordinal data (Leach 1979). The two samples in this case were the rating (or order ranking) data of the respondents who have involved in MAS projects and those who have not, with regard to a particular feature, step or concept. The samples are thus independent and the collected data is ordinal and continuous.

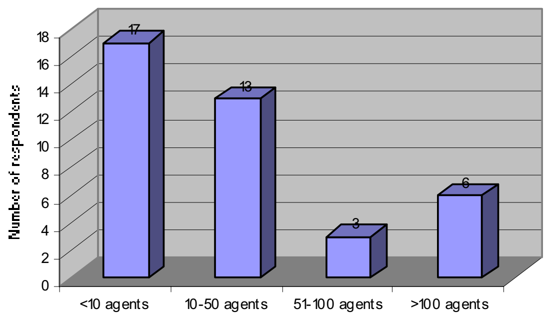


Figure AppendixC.3 – Size of past MAS projects

Variable “Level of complexity of past MAS projects”

The *median* complexity of the past MAS development projects that the respondents had been involved in was “5” (on a 7-point Likert scale ranging from “Very low” to “Very high”), indicating an average level of medium complexity for the involved MAS development projects (Figure AppendixC.4).

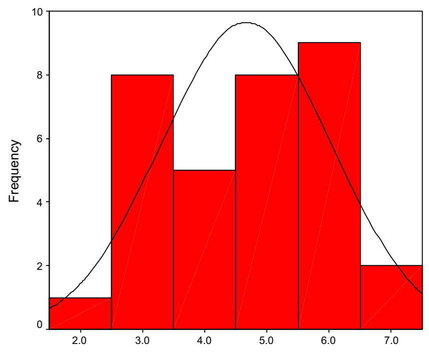


Figure AppendixC.4 – Level of complexity of involved MAS projects

Variable “Application areas of past MAS projects”

A large number of involved MAS projects were in the areas of Information Gathering/Management, Simulation and Personal Assistant (Figure AppendixC.5). Nine respondents had been involved in projects of two different application areas, while six respondents were involved in three different application areas.

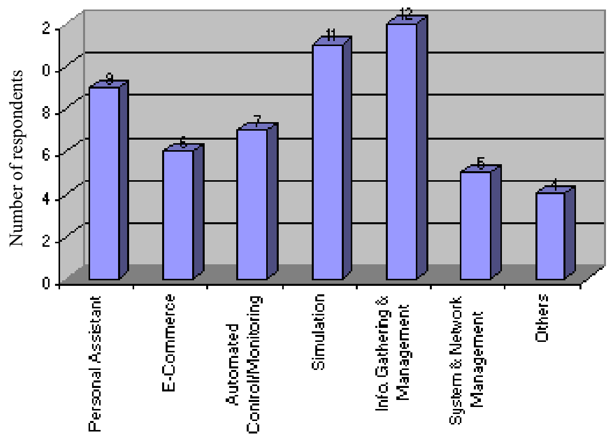


Figure AppendixC.5 – Application areas of involved MAS projects

Variable "Adoption of AOSE methodologies in past MAS projects"

A large proportion of the respondents (26 out of 33[101]) did not follow any AOSE methodology or framework in their past MAS projects. Of the respondents that did follow a methodology or framework, the listed AOSE methodologies and frameworks were:

- PROMETHEUS (Padgham and Winikoff 2002a; Padgham and Winikoff 2002b);
- GAIA (Wooldridge et al. 1999; Wooldridge et al. 2000);
- INGENIAS (Pavon and Gomez-Sanz 2003; Pavon et al. 2005);
- RoMAS (Yan et al. 2003);
- MESMA (Cuesta et al 2002);
- JADE framework (Telecom Italia Lab 2004);
- FIPA specifications (FIPA n.d.b); and
- Adapted OO frameworks and techniques for agent-oriented development, including Rational Unified Process, UML and design patterns.

Variable "Involvement in Ontology-Based MAS development projects"

Of the respondents who had been involved in MAS development projects, only a small proportion had experienced the construction of Ontology-Based MASs (15 out of 33).

[101] The proportion was calculated out of the respondents who had been involved in at least one MAS development project, i.e. 33 respondents.

APPENDIX D

EVALUATION OF EXISTING MAS DEVELOPMENT METHODOLOGIES

This appendix presents the evaluation of 16 AOSE methodologies according to criterion "*Support for steps"* (cf. Table 5.22) and six other criteria relating to the AOSE process, namely (cf. Table 5.21):

- "*Specification of model kinds and/or notational components*";
- "*Definition of inputs and outputs of steps*";
- "*Specification of techniques and heuristics*";
- "*Ease of understanding of techniques*";
- "*Usability of techniques*"; and
- "*Provision of examples for techniques*".

All of these criteria used the list of steps in Table 5.22 as yardsticks.

If a methodology was found to provide "*support for [a particular] step*", this support was evaluated as "explicit" ("E") or "implicit" ("I") (cf. Tables 8.9a to 8.9p). The former applies if the methodology specifies the step as a distinct activity in its development process. The latter incurs when the step is implicitly fulfilled as part of another step, or only briefly mentioned by the methodology. If a step is specified as part of, or in conjunction with, another step, this other step is indicated in the square brackets [].

Evaluation for "*Definition of inputs and outputs of steps*" is denoted as "I" if only inputs are specified, "O" if only outputs are specified, and "B" if both inputs and outputs are defined. Criterion "*Specification of techniques and heuristics*" was assessed in two parts: "*Techniques used to perform each step*" and "*Techniques used to produce each model or notation component*" (cf. Table 5.21 in Section 5.4.1). "*Ease of understanding of techniques*" and "*Usability of techniques*" were evaluated as either "high" ("H"), "medium" ("M") or "low" ("L").

Table AppendixD.1 – Support for steps of MASE

MASE								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Goal hierarchy diagram	B	Analyze initial system specifications, e.g. technical documents, user stories, and se cases. Identify tasks that each role should perform to achieve goals	Hierarchically organise goals in the order of importance. All sub-goals should relate functionally to their parent. Attach tasks to roles in Extended role diagram	H	H	Y
2. Specify use case scenarios	E	Use case diagrams	B	Conventional OO techniques	Conventional OO techniques	H	H	Y
3. Identify roles	E	Role diagram	B	Typically one-to-one mapping between goals and roles	Show roles, their related goals, and communication paths between roles	H	H	Y
4. Identify agent classes	E	Agent class diagram	B	Group roles into agent classes	Show agent classes, related roles, and acquaintances between agents	H	H	Y
5. Model domain conceptualisation	E		B	Define purpose and scope of the ontology, collect data from the information domain, form the initial ontology, and finally refine the ontology into a complete version		H	H	Y
6. Specify acquaintances between agent classes	I [5]	Agent class diagram	B	Any communication paths between 2 roles indicate acquaintances between their respective agent classes		H	H	Y
7. Define interaction protocols	E	Communication class diagrams	B	Specify conversations between agents by analyzing inter-role interactions in use cases, and task descriptions in Task state diagrams	Produce a Communication class diagram (which is a finite state machine) for each participant in the conversation	H	H	Y
8. Define content of exchanged messages	E[8]	Communication class diagrams	B	Analyze inter-role interactions in use cases, and task descriptions in Task state diagrams	Model messages as transitions between states in Communication class diagram. Specify performatives and parameters	H	H	Y
9. Specify ACL								
10.Specify agent architecture	E	Agent class architecture diagram	O	Refer to the work of (Robinson 2000)	Refer to the work of (Robinson 2000)	H	M	Y
11.Define agent informational constructs (i.e. beliefs)	I[4]	Task state diagram	B	Specify how a role/agent can fulfill a task with a structured set of activities and communications. This implicitly represents an agent's plan for achieving tasks.	Depict the task processing as a finite state machine	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes	E	Deployment diagram	B	Similar to instantiating objects from object classes	Show numbers and types of agents	H	H	Y
19.Specify agent instances deployment	E [18]	Deployment diagram	B	Consider message traffic between agents, and processing power available on particular machines and required by particular agents	Show locations of agents (e.g. hostname and address)	H	H	Y

Table AppendixD.2 – Support for steps of MASSIVE

MASSIVE								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Task View	B	Analyze the intended workflow to specify what to be done. The functional decomposition of tasks can be supported by Structured Analysis	Construct the Task Tree following hierarchical decomposition. The granularity of decomposition depends on the specific problem, but should not become a specification of a particular algorithm	H	H	Y
2. Specify use case scenarios								
3. Identify roles	E	Role View	B	Group atomic activities (from Task View) into roles while satisfying the physical constraints of the operational environment		H	H	Y
4. Identify agent classes	I[3]	Role View, Architectural View	B			L	L	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	I[7]	Interaction View				L	L	Y
7. Define interaction protocols	E	Interaction View	B	Characterise the nature of agent interactions, thereby choosing appropriate interaction scheme and protocols		H	H	Y
8. Define content of exchanged messages								
9. Specify ACL	I[7]	Interaction View	O	Recommend KQML as a de-factor standard for ACL				Y
10.Specify agent architecture	E	Architectural View	B	Characterise the requirements of the agent architecture, thereby selecting a suitable architecture		H	H	Y
11.Define agent informational constructs (i.e. beliefs)								
12.Define agent behavioural constructs (e.g. goals, plans, actions services)								
13.Specify system architecture (i.e. overview of all system components and their connections)	E	Architectural View	B	Examine the overall nature of the system, then choose an architectural patterns that firs		H	M	Y
14.Specify organisational structure/inter-agent authority relationships	E	Society View	B	Characterise the target system society and design/choose an optimal social structure accordingly		H	H	Y
15.Model MAS environment	E	Environment View	B	Characterise MAS' environment from both the perspectives of the developer and of the system.	Characterise organisational context (e.g. accessible or inaccessible, deterministic or non-deterministic, episodic or non-episodic, static or dynamic) and runtime environment (programming model, programming language, and communication mode)	H	H	Y
16.Specify agent-environment interaction mechanism	E [15]	Environment View	O	Determine a generic model of sensors + effectors that allows agents to interact with the environment.		H	M	Y
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.3 – Support for steps of SODA

SODA								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	I[3]	Role Model	O		Specify each task in terms of its responsibilities, competences and required resources. Classify each task to either "individual" task or "social" task	H	L	N
2. Specify use case scenarios								
3. Identify roles	E	Role Model	B	Associate each "individual" task to an "individual" role, each "social" task to a group of "social" roles.	Define each role/role-group in terms of its individual and/or social tasks, permissions to resources (which are identified in Resource Model), and interaction protocols and rules (which are defined in Interaction Model).	H	M	N
4. Identify agent classes	E	Agent Model	B	Groups roles into agent classes	Define each agent by its individual/social tasks, permissions to resources, interaction protocols associated with its roles, cardinality, location and source (i.e. from inside or outside the system).	H	M	N
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	I[7]	Interaction Model						
7. Define interaction protocols	E	Interaction Model	O	Define the interaction protocols for roles and for resources, as well as interaction rules for role-groups	An interaction protocol specifies the information required/provided by a role to accomplish its tasks, or by a resource to invoke its services. An interaction rule for a role-group governs the interactions among social roles and resources so as to make the group accomplish its social task	H	H	N
8. Define content of exchanged messages								
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)								
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment	E	Resource Model, Environment Model	O	Identify resources offered by environment. Map these resources onto infrastructure classes. Specify the topological model of environment	Define resource in terms of services, access modes and permissions granted to roles and role-groups. Describe each infrastructure class is described in terms of services, interfaces, location, owner and cardinality	H	M	N
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes	I[4]	Agent Model						
19.Specify agent instances deployment	I[4]	Agent Model						

Table AppendixD.4 – Support for steps of GAIA

GAIA								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	I [3]	Role model	O	Specified as "liveness responsibilities" of roles	Each liveness responsibility is made up of "actions" and/or "protocols"	H	M	Y
2. Specify use case scenarios								
3. Identify roles	E	Role model	B	Identify roles from individuals, departments/offices, or sub-organisations in the target system	Model each role by its "responsibilities" (including "liveness" and "safety") and "permissions"	H	H	Y
4. Identify agent classes	E	Agent model	B	Typically one-to-one mapping between roles and agent classes	Show identifier of agent classes and their respective roles	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Acquaintance model	B	Identify acquaintances from Role, Agent and (inter-role) Interaction models	Show agent classes and communication paths between them	H	M	Y
7. Define interaction protocols	E	Interaction model	O	Only specifies protocols for inter-role interactions. Each protocol defines an institutionalized pattern of interaction with no detailed sequences of exchanged messages	Specify purpose, initiator, responder, inputs, outputs, and (informal) processing description for each inter-role protocol	H	H	Y
8. Define content of exchanged messages								
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)								
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E	Service model	B	Identify agents' services by analyzing their roles' responsibilities, actions, and protocols.	Show inputs, outputs, pre-condition, and post-condition for each agent's service	H	H	Y
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships	E	Organisational structure model	B	Choose a structure that optimizes the organisational efficiency and simplicity, respects organisational rules, and reflects the structure of real world organisation.	Specify organisational dependencies between roles	H	H	Y
15.Model MAS environment	E	Environmental model	B	Identify abstract computational resources (e.g. tuples/variables) that are available to agents for sensing, effecting or consuming	Specify a symbolic name, types of actions permitted on each environmental resource, and their textual/structural descriptions	H	M	Y
16.Specify agent-environment interaction mechanism	I			Implicitly indicates that agents interact with environment via sensors and affectuators. No additional information provided				
17.Specify agent inheritance and aggregation	I[4]	Agent model	B	Aggregation occurs when an agent class is composed of the roles that make up other agent classes. Does not consider inheritance	Show an aggregate agent class as the parent of the children classes in the tree structure of Agent model	H	H	N
18.Instantiate agent classes	E [4]	Agent model	O		Specify numbers of instances for each agent class by annotating the class with qualifiers from Fusion	H	H	Y
19.Specify agent instances deployment								

Table AppendixD.5 – Support for steps of MESSAGE

MESSAGE								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Goal/Task view	B	Analyze organisation chart, company goals description, and business processes to identify goals. Identify services that can be performed by roles to satisfy these goals, and tasks that can be implemented to fulfill these services	Show a hierarchy of goal decomposition in Goal diagram. Describe the flow of tasks to achieve a service in a Workflow diagram	H	H	Y
2. Specify use case scenarios								
3. Identify roles	E	Agent /Role view	B		Associate roles to goals in Delegation structure diagram, and to services and tasks in Workflow diagrams. Describe each role with Role Schema	H	M	Y
4. Identify agent classes	E	Agent /Role view	B	Assign roles to agents based on the developer's experience and heuristics	Describe each agent with an Agent Schema and an Agent diagram (which shows the associated roles, goals, tasks and assessed data sources)	H	M	Y
5. Model domain conceptualisation	E	Domain view	B	Incrementally add domain concepts and relations to Domain view as they are needed in other views	Specify concepts as classes in UML class diagram	H	M	Y
6. Specify acquaintances between agent classes	E	Organisation View	B		Specify acquaintances between roles/agents in Organisation view.	H	M	Y
7. Define interaction protocols	E	Interaction view	B	Incrementally built from Analysis to Design. In Analysis phase, only need to highlight which, why and when roles communicate. In Design phase, elaborate each interaction considering the assignment of roles to agents and implementation of services in terms of tasks	May model each protocol with AUML protocol diagrams or UML statechart. Can model the behaviour of each agent/role in a protocol with statecharts	H	H	Y
8. Define content of exchanged messages	I[7]	Interaction view	O		Define each message in appropriate ACL in protocol diagrams.	H	L	N
9. Specify ACL	E		O	In *Agent-Platform Driven* design approach, the developers should choose an ACL and content language to use, e.g. KQML/KIF or FIPA-ACL/SL		H	M	N
10.Specify agent architecture	E		B	Select an architecture that suits the functional requirements of agents (e.g. cognitive versus reactive)	Specify architecture components/layers, depending on the chosen architecture	H	H	Y
11.Define agent informational constructs (i.e. beliefs)	E		B	Depending on the agent architecture, various categories of knowledge may need to be specified, including domain knowledge, social knowledge, and behavioural knowledge. These can be determined by analyzing the Domain view, Organisation view, and Goal/Task view.	Represent domain entities (for domain knowledge), social constraints (for social knowledge), and rules, objectives, and tasks (for behavioural knowledge) probably using UML notation	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)	E [14]	System architecture diagram	B	Derive system architecture from Organisation Model	Show all system components as a package structure	H	H	Y
14.Specify organisational structure/inter-agent authority relationships	E	Organisation view	B	Incrementally built from Analysis to Design. Start by analyzing organisation chart and business process documentation	Show stakeholders/users, agents/roles, resources, sub-organisations, and relationships bet them (e.g. power/peer-to-peer organisational relationships, acquaintances)	H	H	Y
15.Model MAS environment	I [14]	Organisation view	O	Identify resources that agents use, control or receive input from (e.g. databases, computational resources)	Show resources and their relationships with agents in Organisation view	H	L	N
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes	I[4]			Mentioned but no techniques/model kinds provided		L	L	N
19.Specify agent instances deployment								

Table AppendixD.6 – Support for steps of INGENIAS

INGENIAS								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Goals and Tasks model	B	Identify goals from system requirements or objectives associable to agents. Derive tasks from system requirements or from goals	Show goals, goal-subgoals dependencies, tasks, tasks' pre-conditions, post-conditions, and goals-tasks associations	H	H	Y
2. Specify use case scenarios	E	Use case diagrams	O	Incrementally identified and refined	Conventional OO techniques	H	H	Y
3. Identify roles	I[4,15, 7]	Agent model, Organisation model, Interaction model	B	Identify roles from the analysis of workflows and tasks in Organisation model	Show roles as actors of workflows/tasks in Organisation model, as participants in Interaction model, and associated to agents in Agent model	H	M	Y
4. Identify agent classes	E	Agent model	B	Apply "rationality principle" on system components to identify agents.	Describe each agent in terms of its roles, goals, tasks, mental states, and control structure/process.	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Interaction model	B	In Analysis phase, identify significant interactions between actors (i.e. agents/roles) and initial schemes of exchanged info.	Show participants (agents/roles) and goals pursued by each interaction	H	H	Y
7. Define interaction protocols	E[6]	Interaction model	B	In Design phase, elaborate each interaction with detailed description of exchanged elements (e.g. messages, tuples, method calls)	Specify exchanged elements and order of their execution (e.g. iteration, concurrency, branching)	H	H	Y
8. Define content of exchanged messages	E[7]	Interaction model	B		For each message, show name of operation, parameters, guards, and annotation of sequence	H	M	Y
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)	E	Agent model	B	Determine agent's "mental states" from analysis of goals, tasks, and interactions. Define the "control" of agent to assure desired transitions between Its mental states	Represent mental states in terms of goals, tasks, facts, or any other entity that helps in state description. Agent goals can be modelled as initial state. Can model agent control as algorithms or complex deliberative process	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)	I[14]	Organisation model	B		Implicitly reflected in the Organisation model where all system components and their connections are shown	H	H	Y
14.Specify organisational structure/inter-agent authority relationships	E	Organisation model	B	Incrementally identify and refine the org. structure in terms of system components (i.e. agents, roles, resources, and applications), and social dependencies among them	Show how system components are grouped, their social dependencies (e.g. subordination and client-server relations), task/workflow assignment, and resources used/produced.	H	H	Y
15.Model MAS environment	E	Environment model	B	Identify resources and applications in the environment by analyzing system requirements and agent requirements	Model resources and applications as objects. Specify internal states and operations for applications, and initial states, category, and limit of consumption for resources.	H	H	Y
16.Specify agent-environment interaction mechanism	E[15]	Environment model	B	Determine how agents perceive outputs of applications in the environment. Possible perception mechanisms: sampling or notification	Represent agent's perception mechanism as a type of association relationship between the agent and an application in Environment model	H	H	Y
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.7 – Support for steps of BDIM

BDIM								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality								
2. Specify use case scenarios								
3. Identify roles	E		B	Roles can be organisational or functional; can be domain dependent or required by system implementation		M	L	Y
4. Identify agent classes	E	Agent Model	B	Group roles (that have common lifetime and intense interactions) into a draft agent hierarchy. Refine the hierarchy to introduce new abstract agent classes, new concrete agent classes and agent instances	Produce Agent Class Diagram and Agent Instance Diagram (may be combined into a single diagram if the number of agents is small)	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Interaction Model	B	Identify interactions between agents by analyzing the provision of services among agents.	Offer no modelling notation for Interaction Model Developers can use any notation that fits	M	L	N
7. Define interaction protocols								
8. Define content of exchanged messages	E [6]	Interaction Model		For each interaction, identify the speech acts required for the messages and the messages' information content.	Offer no modelling notation	M	L	N
9. Specify ACL								
10.Specify agent architecture	I			BDIM adopts a BDI agent architecture				
11.Define agent informational constructs (i.e. beliefs)	E	Goal Model, Belief Model, Plan Model	B	Identify agent goals from agent's service. For each goal, identify means for achieving the goal (i.e. plans) and the context, conditions, inputs and outputs of goals and plans (i.e. beliefs)	A Goal/Belief Model consists of 1 Goal/Belief Set and many Goal/Belief States. A Plan Model contains a set of plan diagrams	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E		B	For each agent, identify its responsibilities and the services provided/used to fulfill these responsibilities		H	M	Y
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation	E	Agent Model	B	Identify inheritance and aggregation relationships by examining similarity in lifetime, services and interaction interfaces of agents	Inheritance allows an agent to override/extend the Goal/Belief/Plan Model of its superclass(es). Aggregation allows for agents with independent Goal/Belief/Plan Models to be combined into an aggregate class	H	H	Y
18.Instantiate agent classes	E	Agent Model	B		Capture instantiation information (e.g. instance identification and cardinality) either in Agent Class Diagram or separately in Agent Instance Diagram	M	M	Y
19.Specify agent instances deployment								

Table AppendixD.8 – Support for steps of HLIM

HLIM								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E [2]	High Level Model	B	Identify system tasks as "responsibilities" appearing in use case scenarios	Each path in the UCM connects responsibilities, indicated by named points along paths	H	H	Y
2. Specify use case scenarios	E	High Level Model	B		Develop a Use Case Map (UCM) for each use case scenario, where the UCM's path traces a scenario from a start to finish	H	H	Y
3. Identify roles	E [2]	High Level Model	O		Roles are represented by "slots" (boxes with dashed lines) along UCM's paths	H	M	Y
4. Identify agent classes	E [2]	High Level Model	B	Identify agents as carrier of responsibilities in UCMs. Initial agents can be extracted from essential and active entities that exist in the problem domain.	Represent agents as boxes incorporating responsibilities along UCM's paths.	H	M	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	I[7]	Conversation Model	B	Derive necessary agent interactions from Agent Relationship Model and Internal Agent Model		M	M	Y
7. Define interaction protocols	E	Conversation Model	B	Each type of dependency relationships and jurisdictional relationships has a predefined interaction protocol associated with it	Express a protocol as a set of performatives that are specified in the exchanged messages	H	H	Y
8. Define content of exchanged messages	E	Conversation Model	B	Identify what messages are required to fulfill the dependency and jurisdictional relationships. The content of messages is determined by examining plans that satisfy the dependencies.	Use tabular format. A table for each agent. 3 columns: receive, send and comment. Each message contains a performative and parameters.	H	M	Y
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)	E	Internal Agent Model	B	Derive agents' goals, tasks, beliefs and plans directly from UCM's components in High Level Model	Use tabular template, where agent goals, tasks and beliefs are specified in columns. Plans are combinations of goals, tasks and beliefs (i.e. rows)	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E	Contract Model	O		Services provided by each agent are captured in its contracts with other agents	M	L	Y
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships	E	Agent Relationship Model	B	Identify organisational relationships by analysis of path segments responsibilities in UCMs.	Model organisational relationships as Dependency and Jurisdictional relationships. Each type is captured in Dependency Diagram and Jurisdictional Diagram respectively	H	H	Y
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.9 – Support for steps of MEI

MEI								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	IDEF/CIMOSA Function Model	O	Borrow techniques from enterprise modelling	Borrow techniques from enterprise modelling	H	H	Y
2. Specify use case scenarios	E	Use Case Model	O	Borrow techniques from OOSE	Borrow techniques from OOSE, including use case extension and inheritance	H	H	Y
3. Identify roles								
4. Identify agent classes	E		B	Identify agents from actors in use cases and resources/mechanism in CIMOSA/IDEF function model		H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E		B	Agent collaboration exists if there is more than 1 actor per use case or more than 1 resource per enterprise function		H	H	Y
7. Define interaction protocols	E	Coordination Protocol Script	B	Derive protocols from event trace of use cases and information exchanges between resources	Use State Diagrams to model protocol scripts for each agent	H	H	Y
8. Define content of exchanged messages								
9. Specify ACL								
10.Specify agent architecture	I			MEI adopts a BDI-like model of agency, where each agent is composed of goals, plans and beliefs				
11.Define agent informational constructs (i.e. beliefs)	E	Goal-Plan Diagram, Plan State Diagrams	B	Derive agents' Goals and Plans from use cases and enterprise functions with control outputs. Context/ invocation conditions of plans can be derived from control inputs of enterprise functions, or input from actor and entity objects. Agent Beliefs correspond to domain objects in use cases and entities in IDEF Information Model.	An agent's goals and plans can be depicted as a tree structure, where goals are the root nodes and plans are leaves. Each plan can be further defined by a state diagram	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment								
16.Specify agent-environment interaction mechanism	I		O	Agents interact with environment via sensor and effector objects, which communicate with co-existing objects or sensor/effector objects of other agents	Each agent may have many sensor/effector objects	H	H	Y
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.10 – Support for steps of PROMETHEUS

PROMETHEUS								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Functionality descriptor	B	Identify a set of functionalities by considering groupings of goals.	Describe each functionality in terms of its goals, actions, percepts/events and potential data read/written	H	H	Y
2. Specify use case scenarios	E	Use case descriptor	B	Identify sequences of steps that describe how the system achieves a goal or responds to an event	Annotate each step with associated functionality and data read/written	H	H	Y
3. Identify roles								
4. Identify agent classes	E	Agent class descriptor	B	Assign functionality to agent class based on the criteria of strong coherence and loose coupling	Describe each agent class in terms of its functionality, goals, events, actions, and data read/written, cardinality, agent lifetime).	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Interaction diagrams	B	Whenever there's a step in a use case that involves functionality from a new agent, there must be an interaction pathway from a previously involved agent and this new agent	Show the core interaction channels between agents using sequence diagrams	H	H	Y
7. Define interaction protocols	E [6]	Interaction protocols	B	Elaborate each complex interaction with protocol by analyzing use case scenarios	Show all variations of interaction sequences that are valid in the system. Each protocol may be split into smaller chunks	H	H	Y
8. Define content of exchanged messages	I [6, 7]	Interaction diagrams and protocols	B	Analyze use case scenarios		H	L	Y
9. Specify ACL								
10.Specify agent architecture	E	Agent overview diagram	B		Show the top-level view of an agent's internals, including top-level capabilities, events connecting these capabilities, and data objects internal to the agent	H	H	Y
11.Define agent informational constructs (i.e. beliefs)	E	Plan descriptor, Data descriptor	B	Recursively decompose each agent's capability into plans, events connecting plans, data read/written by plans, and sub-capabilities.	Describe each agent "plan" in terms of input/output events, actions, and messages. Describe each "data object" used by the agent with fields and methods.	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E	Capability diagram	B	Identify agent "capabilities" by analyzing functionalities assigned to the agent	Describe each capability in terms of sub-capabilities, plans, events, and data read/written in Capability diagram	H	H	Y
13.Specify system architecture (i.e. overview of all system components and their connections)	E	System overview diagram	B	Describe how the system as a whole will function	Show an overview of all agent classes in the system, events connecting classes, and shared data objects	H	H	Y
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment	E	System overview diagram	O	There may be data objects existing in the environment that must be shared among agents (e.g. databases)	Show and link shared data objects to agents in System overview diagram	H	M	Y
16.Specify agent-environment interaction mechanism	E	Percepts descriptor, Actions descriptor	B	Specify raw data available to the system as "percepts", and activities performed by the system on the environment as "actions"	Specify percepts and actions for each system functionality in Percepts and Actions descriptors	H	H	Y
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes	I[4]	Agent class descriptor	O		Specify cardinality for each agent in its Agent class descriptor	H	L	N
19.Specify agent instances deployment								

Table AppendixD.11 – Support for steps of PASSI

PASSI								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	System requirements model	O	Follow standard requirements elicitation techniques in OO, or scenario-based teleological methods such as GBRAM	Specify functionality as use cases in Use case diagrams	H	H	Y
2. Specify use case scenarios	E [1]	System requirements model	B		Develop a hierarchical series of use case diagrams. The uppermost serves as a context diagram	H	H	Y
3. Identify roles	E	System requirements model, Agent society model	B	Identify roles for each agent by exploring all the possible scenarios of inter-agent interaction (captured in Agent identification diagram – step 5)	For each inter-agent interaction scenario, develop a Role identification diagram to specify the roles that agents play during the interaction. Describe roles with a Role description diagram, which shows their agents, role changes within an agent, roles' tasks, roles' interactions and dependencies.	H	H	Y
4. Identify agent classes	E	System requirement model	B	Package use cases into agents	Show agents, their respective use cases, and interaction paths between use cases in Agent identification diagram	H	M	Y
5. Model domain conceptualisation	E	Agent society model	O	Specify concepts/entities that define the domain's knowledge.	Represent domain ontology as an XML schema or class diagram in Domain ontology diagram	H	M	Y
6. Specify acquaintances between agent classes	I[4]	System requirement model	B		Agent acquaintances are reflected via the interaction paths between use cases in Agent identification diagram	H	H	Y
7. Define interaction protocols	E	Agent society model	B	Select and refine protocol for each agent acquaintance by consulting e.g. FIPA library. Should also specify the ontology used with the protocol	Document each protocol in Protocol description diagram (which may be AUML sequence diagram). Specify identifier of protocol and ontology for each agent acquaintance in Communication ontology diagram.	H	H	Y
8. Define content of exchanged messages	I	Agent implementation model	B	Specify messages' performatives as required by the interaction protocol and messages' contents by using concepts defined in Domain ontology diagram.	Model exchanged messages (including their performatives and contents) as transitions between agents in the Multi-agent behavior description diagram	H	H	Y
9. Specify ACL								
10.Specify agent architecture	E	Agent implementation model	B	Define agent structure as being composed of one main agent class and a set of inner classes, each representing a task of the agent	Specify data structures and methods of the agent and its tasks in the main agent class and task classes respectively.	H	H	Y
11.Define agent informational constructs (i.e. beliefs)	I [10]	Agent implementation model	B	Specify agent "knowledge" by using the concepts defined in Domain ontology diagram	Model knowledge as agent data structures in Single-agent structure definition diagram	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E	System requirements model, Agent implementation model	B	Agent's capabilities are represented by its tasks, which can be identified by analyzing its roles and interactions described in Role identification diagrams	Show all tasks of an agent in a Task specification diagram. Further describe each method required to achieve each task in Single-agent behavior description (using flow charts, state diagrams, or text description)	H	H	Y
13.Specify system architecture	E	Agent implementation model	B		Show all agent classes, their knowledge, tasks, and connections with external actors in Multi-agent structure diagram	H	H	Y
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment	E	Deployment model	O		Show processing units, agents in each unit, agent movements, and units/agents connections in Deployment configuration diagram.	H	L	N

Table AppendixD.12 – Support for steps of ADELFE

ADELFE								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E [2]	Use case model	O	Identify the different functionalities the system has to carry out	Express each functionality as a use case	H	H	Y
2. Specify use case scenarios	E	Use case model	B	Apart from identifying use cases, need to also highlight the possible cooperation failures in the identified use cases	Conventional OO techniques	H	H	Y
3. Identify roles								
4. Identify agent classes	E	Software architecture document	B	First, decompose system into entities. Determine which entities fit to be agents, i.e. whether they are autonomous, goal-directed, dynamic, and need to deal with unpredictable events. If an agent needs to be adaptive/evolving, it should be decomposed into a collective of sub-agents.	Show entities and their relationships in Preliminary class diagram. Update this diagram to indicate which classes are agents	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Software architecture document	B	Identify potential interaction relationships between agents, and also between agents and non-agent active/passive entities	Model interaction relationships with sequence diagrams or collaboration diagrams	H	M	Y
7. Define interaction protocols	E	Interaction languages document	B	Analyze each use case and interaction scenarios	Elaborate each interaction relationship with a protocol diagram that specifies information exchanges between agents and between agents and non-entities	H	M	Y
8. Define content of exchanged messages	E [7]	Interaction languages document	B	As above. Also select the communication languages for specifying the messages		H	M	Y
9. Specify ACL	E [8]	Interaction languages document	O		ACL used to implement the exchanged messages is documented in Interaction Languages document	H	M	N
10.Specify agent architecture	E	Detailed architecture document	O	Agent architecture should contain 5 components: representations, social attitudes, interaction languages, aptitudes, and skills	Model each agent in terms of the 5 listed components	H	H	Y
11.Define agent informational constructs (i.e. beliefs)	E	Detailed architecture document	B	"Representations" are agent's beliefs about itself and environment. "Social attitudes" contain rules for dealing with non-cooperative situations. "Interaction languages" involve protocols used by the agent.	Specify attributes and methods for agent's representations. Select protocols for agent's interaction languages from the set defined in step 8. Specify non-cooperative situations and rules for cooperative attitudes.	H	H	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	E	Detailed architecture document	B	"Skills" are capabilities that an agent brings to its collective. "Aptitudes" are agent's capabilities on its knowledge.	Specify methods and/or attributes for agent's "skills" and "aptitudes"	H	M	Y
13.Specify system architecture (i.e. overview of all system components and their connections)	E	Detailed architecture document		Define the system architecture in terms of packages and classes (of agents and objects). Should use design patterns and/or re-usable components	Generate a Class diagram for each package	H	H	Y
14.Specify organisational structure/inter-agent authority relationships								
15.Model MAS environment	E	Environment definition document	B	Identify active and passive entities in the environment; characterise the system's environment as being accessible or not, deterministic or not, static or dynamic, and discrete or continuous		H	H	Y
16.Specify agent-environment interaction mechanism	I [11]	Detailed architecture document	O	Agents interact with environment via percepts and actions, implicitly specified in agents' "skills", "aptitudes" and "interactions".		H	L	N
17.Specify agent inheritance and aggregation	I	Detailed architecture document	O		Show aggregation relationships between agents in Class diagrams	H	M	Y
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.13 – Support for steps of COMOMAS

COMOMAS								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Task Model	O	Identify tasks to be solved by the target MAS and data/control dependencies between them	Develop a task hierarchy, along with each task's details (i.e. input, output and control structure). Can use Conceptual Modelling Language (CML) as notation	H	M	Y
2. Specify use case scenarios								
3. Identify roles								
4. Identify agent classes	E	Agent Model	B	Identify agents by clustering the competencies for solving tasks (Expertise Model) while respecting the design requirements (Design Model)	Model each agent as a composition of knowledge structures obtained from other models. Can use CML as modelling notation	H	M	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes								
7. Define interaction protocols	E	Cooperative Model	O		Specify cooperation protocols, cooperation methods (e.g. data sharing or message exchange) and conflict resolution methods. Can use CML as modelling notation	H	L	Y
8. Define content of exchanged messages								
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)	E	Expertise Model	B	Determine agent competencies required to solve system tasks and social knowledge required to enable it to act smoothly during interaction	Competencies include "task knowledge" (i.e. agents' experience on previously solved tasks), "problem-solving knowledge" and "reactive knowledge" (i.e. agents' reactive responses to stimuli). "Social knowledge" includes roles, association between beliefs, commitments, intentions and goals. Can use CML as modelling notation.	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships	E	System Model	B			M	L	Y
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.14 – Support for steps of MAS-CommonKADS

MAS-CommonKADS								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Task model	O	Decompose tasks following a top-down approach	Show tasks in an and/or tree. Describe each task in terms of inputs, outputs, task structure, required capabilities of performer, and preconditions	H	H	N
2. Specify use case scenarios	E	Use case	B	Perform user-centered analysis during Conceptualization phase to identify potential users and how the system processes a user request	Conventional OO techniques	H	H	Y
3. Identify roles								
4. Identify agent classes	E	Agent model	B	Analyze various sources e.g. use cases, statement problems, heuristics, initial Task and Expertise models	Describe each agent in terms of type, role, position, description, offered/used services, goals, plans, knowledge, collaborates, skills (sensors and effectors), reasoning capabilities, general capabilities norms, preferences and permissions.	H	M	Y
5. Model domain conceptualisation	E [11]	Expertise model	O	Specify domain conceptualisation as agent's domain knowledge	Represent concepts, properties, expressions, and relationships in the domain using e.g. class/object diagrams	H	M	Y
6. Specify acquaintances between agent classes	E	Coordination model	B	Identify prototypical conversations between agents by analyzing the results of techniques used for identifying agents (e.g. use cases, heuristics, task model, and CRC cards).	Model conversations by using Message Sequence Charts and Event flow diagrams	H	H	Y
7. Define interaction protocols	E [6]	Coordination model	B	Identify protocols for complex conversations by consulting existing libraries and reuse protocol definitions	Model protocols using high level Message Sequence Charts. Model the processing states of an agent during a protocol using State transition diagrams	H	H	Y
8. Define content of exchanged messages	E [7]	Coordination model	B	Analyze use cases and Expertise model	Model data interchanged in each interaction in terms of data structures specified in Expertise model	H	H	Y
9. Specify ACL								
10.Specify agent architecture	E	Design model	B	Select an appropriate architecture and map the elements defined in Coordination, Expertise, Agent, and Task models onto modules of the architecture. No techniques or models are discussed		H	L	N
11.Define agent informational constructs (i.e. beliefs)	E	Expertise model	B	Specify domain knowledge, task knowledge, inference knowledge, and problem solving knowledge for each agent	Describe each type of knowledge in Domain knowledge ontology, Inference diagrams, Task knowledge specification, or Problem solving method diagrams/templates.	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)	I	Agent model; Organisation model	O		Identify the services that an agent offers to other agents and document this in Agent Model and/or Organisation Model	H	L	Y
13.Specify system architecture (i.e. overview of all system components and their connections)	E	Organisation model	B		Show all agents, objects and their relationships (e.g. inheritance, association, agent-object relationship)	H	M	Y
14.Specify organisational structure/inter-agent authority relationships	I [13]	Organisation model			Agent organisational relationships are modelled as association relationships annotated with roles (of each involved agent)	H	L	Y
15.Model MAS environment	E	Reaction cases; Design model	B	Perform environment-centered analysis during Conceptualization phase to identify objects in the environment and potential events coming from each object and actions performed by agents on each object. Identify networking, knowledge and coordination facilities.	Describe the reaction cases coming from interaction of agents with objects in the environment	H	M	N
16.Specify agent-environment interaction mechanism								

17.Specify agent inheritance and aggregation	E	Organisation model	B	Specify aggregation relationships for agent groups, and inheritance relationships for agents that inherit from the values of the precedent agents		H	M	Y
18.Instantiate agent classes	I	Organisation model	O	Mentioned but no techniques discussed	Organisation model can be developed for both agent classes and agent instances	H	L	N
19.Specify agent instances deployment								

Table AppendixD.15 – Support for steps of CASSIOPEIA

CASSIOPEIA								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E		B	Employ existing functional or OO analysis techniques	Define system behaviour at a level of abstraction that makes sense to the achievement of the system's collective task	H	H	Y
2. Specify use case scenarios								
3. Identify roles	E	Coupling Graph	B	CASSIOPEIA identifies 3 layers of roles: domain-dependent roles, relational roles and organisational roles. Identify domain-dependent roles by grouping elementary behaviors needed to achieve the task. See steps 7 and 16 for other 2 types of roles		H	M	Y
4. Identify agent classes	E	Coupling Graph	B	Group roles into agents. One agent may play many roles (one of which is active at a point in time) and one role may be played by many agents		H	M	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Coupling Graph	B	Identify dependencies between domain-dependent roles, thereafter deriving dependencies/acquaintances between agents.	Specify "relational roles" for each agent, i.e. the role of an "influencing" agent or an "influenced" agent	H	H	Y
7. Define interaction protocols	E [6]		O	Specify "influence signs" (i.e. interaction messages) between influencing and influenced agents by analyzing the domain-dependent roles that each agent is playing.		H	L	Y
8. Define content of exchanged messages								
9. Specify ACL								
10.Specify agent architecture								
11.Define agent informational constructs (i.e. beliefs)								
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships	E		B	Analyze the agents' dependencies and relational roles to determine their "organisational roles", i.e. role of "group initiator" and "group participant". Also identify the "organisational behaviors" of agents when playing these organisational roles, i.e. group formation behaviour, commitment behaviour and dissolution behaviour.		H	M	Y
15.Model MAS environment								
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

Table AppendixD.16 – Support for steps of TROPOS

TROPOS								
Steps	**Supported?**	**Model kinds/ Notational components?**	**Inputs/ Outputs?**	**Techniques for step**	**Techniques for modelling**	**Ease of understanding**	**Usability**	**Examples**
1. Identify system functionality	E	Actor diagram, Rationale diagram	B	In Early Requirements, identify goals and softgoals of stakeholders, and perform means-end analysis to determine how these goals can be fulfilled. In Late Requirements, focus on the target system and how it can fulfill the assigned goals. Perform means-end analysis to identify tasks to achieve goals during both Early and Late Requirements	Show task dependencies among stakeholders and system in Actor diagram. Show how goals are achieved through tasks in Rationale diagram.	H	H	Y
2. Specify use case scenarios								
3. Identify roles								
4. Identify agent classes	E	Actor diagram	B	Depending on the chosen organisational structure, decompose the system into sub-actors, each of which can be recursively refined into sub-actors (can consult catalogues of agent patterns for this activity). Assign sub-actors to agents.	Show sub-actors within each system actor, and goal/task/resource dependencies among them	H	H	Y
5. Model domain conceptualisation								
6. Specify acquaintances between agent classes	E	Sequence diagrams, Collaboration diagrams	B	Identify interactions between agents to fulfill particular tasks		H	M	Y
7. Define interaction protocols	E [6]	Sequence diagrams	B	Elaborate each inter-agent interaction in greater detail (e.g. by introducing additional or refined exchanged messages)		H	H	Y
8. Define content of exchanged messages	I[6, 7]				Model each message as a communication act in ACL	H	M	Y
9. Specify ACL								
10.Specify agent architecture	I	BDI agent architecture		TROPOS adopts BDI model for agent architecture				
11.Define agent informational constructs (i.e. beliefs)	E	Plan diagrams, Agent class diagram	B	Define agent's "plans" to achieve a goal, perform a task, or respond to a (communicative) event. Identify resource entities that are incorporated in the agent's knowledge base	Specify Plan diagrams at a directly executable level. Represent resource entities as component classes of an agent class in Agent class diagram	H	M	Y
12.Define agent behavioural constructs (e.g. goals, plans, actions, services)								
13.Specify system architecture (i.e. overview of all system components and their connections)								
14.Specify organisational structure/inter-agent authority relationships	E	Non-functional requirement model	B	Select a suitable organisational structure style (e.g. from the set proposed by TROPOS) by evaluating its quality attributes against the system's softgoals.	Specify how well each alternative organisational structure style fulfils the system's softgoals	H	H	Y
15.Model MAS environment	I[1]	Actor diagram	B		Model environment via stakeholders, and their goal/ task/resource dependencies with the system	H	H	Y
16.Specify agent-environment interaction mechanism								
17.Specify agent inheritance and aggregation								
18.Instantiate agent classes								
19.Specify agent instances deployment								

APPENDIX E

MODELLING NOTATION OF MOBMAS

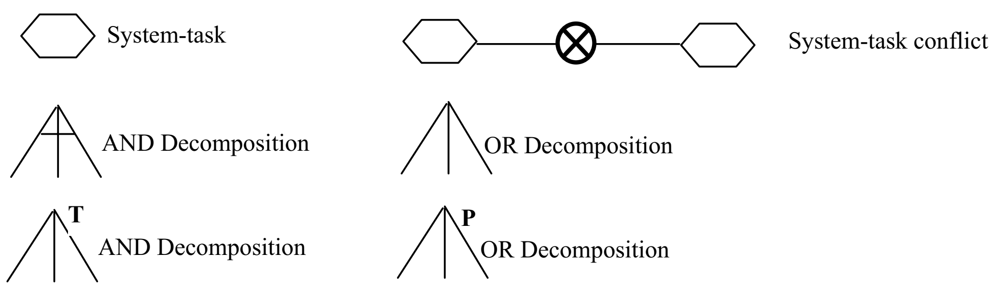
The modelling notation of MOBMAS is mostly reused or adapted from UML, AUML and other sources (such as the existing AOSE methodologies), except for the following notation components that are represented using MOBMAS' own notation:

- Role Diagram;
- Agent Class Diagram;
- Agent Relationship Diagram;
- Agent Plan Template; and
- Resource Diagram.

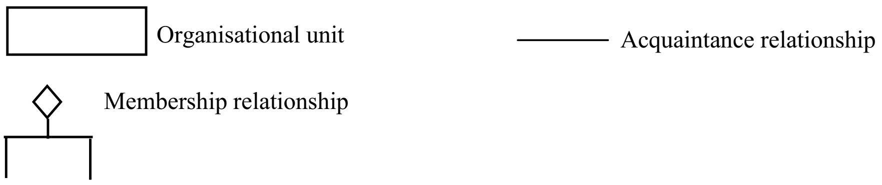
The notation proposed by MOBMAS has a similar syntax to UML. For example, an agent class or a role or a resource is represented as a rectangular box with multiple compartments, each specifying a different property of the target entity. The similarity in syntax between MOBMAS notation and UML is intentional, because it facilitates the use of MOBMAS by developers who are familiar with UML.

A summary of MOBMAS notational syntax is presented below.

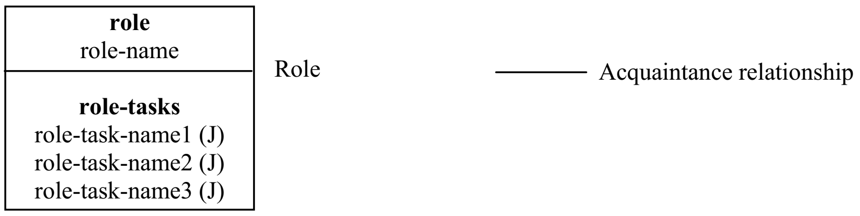
SYSTEM TASK DIAGRAM



ORGANISATION CONTEXT CHART



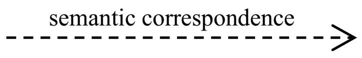
ROLE DIAGRAM



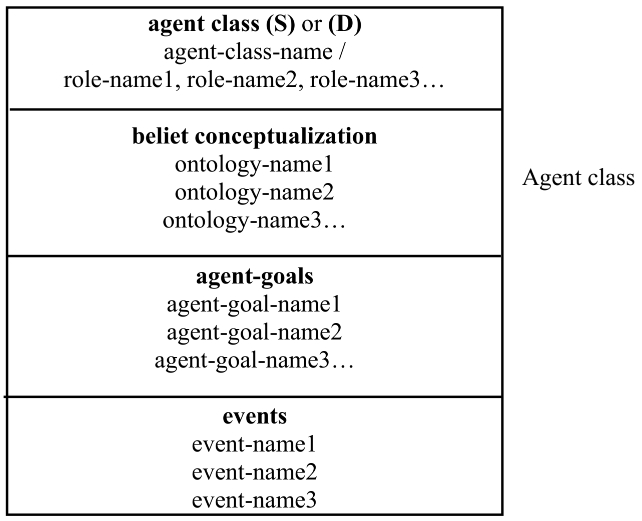
ONTOLOGY DIAGRAM

Adapt the notation of UML Class Diagram (Object Management Group 2003).

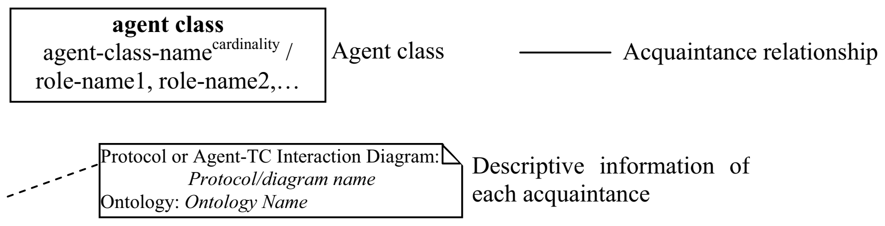
- Ontological concepts are represented as UML classes, attributes or predicates that describe the associations between concepts.
- Relations between ontological concepts are represented as UML relationships between classes, which can be specialisation, aggregation or association.
- Ontological mappings are represented as UML dependency relationships:



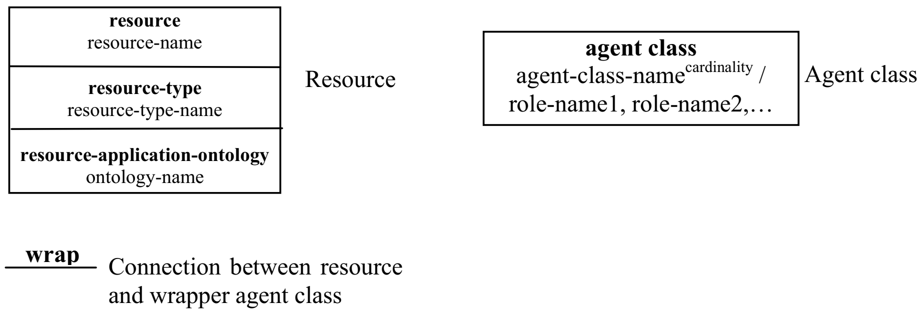
AGENT CLASS DIAGRAM



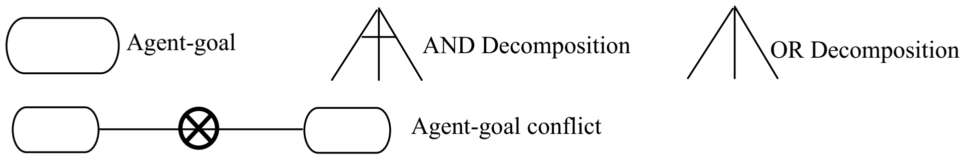
AGENT RELATIONSHIP DIAGRAM


agent class
agent-class-name^{cardinality} /
role-name1, role-name2,…
Agent class
Acquaintance relationship
Protocol or Agent-TC Interaction Diagram:
Protocol/diagram name
Ontology: Ontology Name
Descriptive information of each acquaintance

RESOURCE DIAGRAM



AGENT GOAL DIAGRAM



AGENT PLAN TEMPLATE

Initial state: *state definition*

Target agent-goal: *state definition*

Commitment strategy: e.g. *blind, single-minded* or *open-minded*

List of sub-agent-goals (if any): *state definition and name of the Agent Plan Template that achieves the sub-agent-goal*

List of actions (if any): *action name and parameter list*

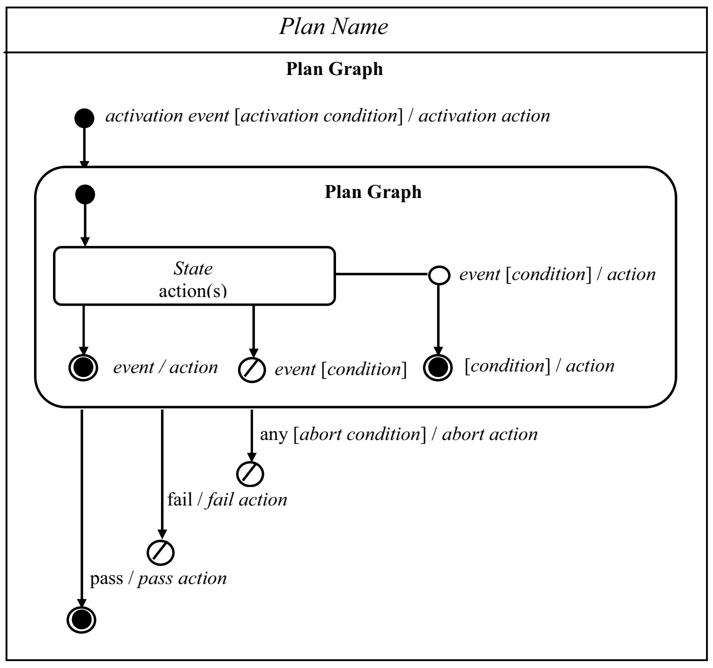
  Pre-condition: *state definition*

  Post-condition: *state definition*

Events: *list of events*

Conflict resolution strategy (if applicable): *strategy name for each agent-goal*

AGENT PLAN DIAGRAM



REFLEXIVE RULE SPECIFICATION

Adapt the notation of UML Activity Diagram (Object Management Group 2003).

- Actions are represented as UML activities.
- Events, internal processing triggers and guard conditions are represented as UML events and guard conditions.

INTERACTION PROTOCOL DIAGRAM / AGENT-TC INTERACTION DIAGRAM

Reuse AUML Interaction Diagrams

TUPLE-CENTRE BEHAVIOUR DIAGRAM

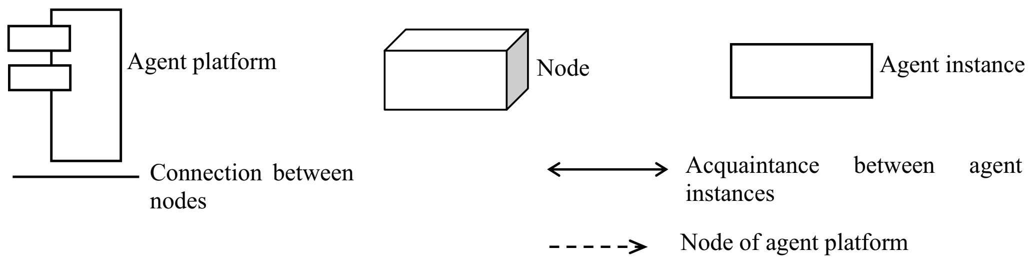
Adapt UML Statechart Diagram.

- Reactions are represented as states.
- Events are represented as transitions between states.

AGENT ARCHITECTURE DIAGRAM

□	Architectural module/layer/subsystem	→	Data input/output

MAS DEPLOYMENT DIAGRAM



APPENDIX F
EXPERT REVIEWS OF MOBMAS

This appendix documents the two expert reviews of MOBMAS which were obtained from Prof. Brian Henderson-Sellers and Prof. Mary-Anne Williams. Each review contained each expert's opinions on the strengths of the methodology, areas for improvement and how to improve these areas. These opinions were recorded informally as comment notes on MOBMAS' documentation which was initially given to each expert.

Expert Review 1

EXPERT: PROF. BRIAN HENDERSON-SELLERS

Strengths of MOBMAS

1. The overall methodology is easy to understand and appears to be easy to follow.
2. The methodology is comprehensive and offers support for diverse aspects of MAS development, covering from analysis to agent internal design to agent interaction design.
3. While the modelling notation of MOBMAS needs to be revised, the steps and techniques are mostly practical and comprehensive.
4. The methodology proposes a clear mapping from roles to agent classes, and from role-tasks to agent-goals and events, thus providing a smooth transition from MAS analysis to agent internal design. The classification of role-tasks into reactive and proactive tasks, thereby identifying agent-goals and events, is also original.

Areas for improvement and suggestions on how to improve these areas

1. Many notational components of MOBMAS are described as *extensions* of UML notational components. However, some extensions are inappropriate because the extended notation is too semantically distant from the original UML notation (namely, MOBMAS Role Diagram, Agent Class Diagram and Resource Diagram). Some other extensions are appropriate but the semantics of the extended modelling notation are not well-documented.

Therefore, it is necessary to determine whether a particular MOBMAS notational component is eligible to be an extension of a UML component. Valid UML extension mechanisms are “stereotypes”, “tagged values” and “constraints”. Thus, if a MOBMAS notational component cannot be mapped to an UML component via these permissible mechanisms, it should be regarded as MOBMAS’ *own* modelling notation. For example, although the Agent Class Diagram of MOBMAS has a similar appearance to UML Class Diagram (with multiple compartments, each modelling a different property of the class), the semantics of each compartment in MOBMAS Agent Class Diagram is very different from the semantics of each compartment in the UML Class Diagram. The difference in semantics is too significant to be expressed by stereotypes, tagged values or constraints.

If a MOBMAS notational component is eligible to be considered as an extension of UML, the methodology should explicitly specify the extension mechanism adopted, and clearly define the semantics of the extended notation.

2. Throughout the development process, MOBMAS employs a variety of modelling concepts, including “system-task”, “role-task”, “agent class”, “agent-goal” and “agent”. Although the semantics of these concepts are defined at their first occurrence in the methodology, it is hard for the readers to recall the meaning of a particular concept, especially after many other concepts have been introduced (for example, concepts “system-task” and “role-task”, or “agent class” and “agent” can be easily confused). Moreover, various modelling concepts in MOBMAS are closely linked (e.g. “system-task” is associated to “role-task”, which is mapped onto “agent-goal”). Even though these linkages are highlighted in the documentation of the respective steps and model kinds, it is difficult for the readers to recall these associations when numerous associations exist.

   A meta-model of the core modelling concepts of MOBMAS should be developed. This meta-model will assist the readers in their understanding and remembering of the semantics and associations of these concepts.

3. The following errors in modelling notation should be fixed:
   - An idle state or a decision point in the tuple-centre Behaviour Diagram (which is basically a UML State Chart) should be represented as a circle ◯ and not a diamond◇, to adhere to UML specification of state charts.
   - The actors in Interaction Diagrams of Agent Coordination Model should be agent instances instead of agent classes. Accordingly, the names of agent classes in the boxes above the lifelines should be preceded with a colon (**:**) to represent instances.

4. The modelling of relationships between ontological concepts in the Ontology Model should include the modelling of "composition" relationship apart from the "aggregation" relationship. The difference in semantics between these two relationships should be highlighted.

Expert Review 2

EXPERT: PROF. MARY-ANNE WILLIAMS

Strengths of MOBMAS

1. The methodology provides extensive support for the openness, heterogeneity and dynamics of MAS.
2. Ontology modelling is tightly incorporated into the analysis and design of MAS, with numerous two-way verification and input linkages between Ontology Model and other MAS analysis and design models (such as Agent Class Model, Agent Behaviour Model and Agent Interaction Model).
3. The methodology provides comprehensive support for MAS development, incorporating diverse analysis and design activities and modelling MAS from diverse aspects (from internal to external).
4. The internal and interaction design of agents are relatively detailed.

Areas for improvement and suggestions on how to improve these areas

1. Regarding steps "Develop System Task Model" and "Analyse Organisational Context" of MOBMAS, the applicable conditions of each step are not appropriate because they may be overlapped. Specifically, for step "Analyse Organisational Context", the recommended applicable condition is that

> "…the target MAS is a processing application system that does not exhibit any specific and apparent human organisational structure"

and for step "Develop System Task Model", the condition is that

> "if the target application exhibits a clear organisational structure, roles can later be identified directly from this structure, thus making the adoption of the Organisation Analysis Approach beneficial".

Accordingly, most MASs are eligible for step "Analyse Organisational Context" because they aim to support a human organisation whose structure is clear. But at the same time, these MASs are also eligible for step "Develop System Task Model" because they do not aim to adopt the human organisational structure.

Therefore, the application conditions of each step should be made clearer and more sensible, avoiding any potential overlaps. The methodology should also give consideration to whether or not the target MAS should adopt the existing human organisational structure.

2. For the naming of system-tasks, MOBMAS adopts the naming format of "*To do something*" (e.g. "*To receive user query"* or "*To get information from resources*"). This naming format makes system-tasks appear like an abstract objective. It may be more appropriate to name system-tasks as phrases starting with *imperatives* to signify activities/actions.

3. MOBMAS offers only one technique for the resolution of conflicts within agents and between agents: "priority conventions". The methodology should consider the adoption, or allow the developer to adopt, other techniques of conflict resolution.

4. When comparing between the direct interaction mechanism via ACL and the tuplespace/tuple-centre interaction mechanism in step "*Select interaction mechanism*", MOBMAS does not consider the issue of security support by the two coordination mechanisms. Since security is an important matter in agent interactions, it should be included as a criterion for the comparison between the two coordination mechanisms.

5. In step "*Develop Agent Interaction Model*", the issue of agent synchronisation is not discussed. This step should be extended to include the specification of agent synchronisation in the design of agent interactions.

6. For step "*Identify agent-environment interface requirements*", the provided techniques are insufficient because:
   - they do not address inter-agent communication; and
   - they only focus on "hardware" sensors and effectors (e.g. camera and wheels).

   The support for this step should be extended to account for the various categories of agent-environment interactions and various types of sensors and effectors.

7. In Ontology Model, MOBMAS considers the adoption of only UML notation for the representation of ontologies. This may limit MOBMAS' applicability because UML notation may not be sufficiently powerful for the modelling of highly complicated ontologies. The developer should be allowed to adopt other notation for Ontology Model if necessary.

8. For System Task Model, MOBMAS should mention the possibility of a complex tree structure where two or more system-tasks share the same sub-system-task(s).

9. In step "*Specify resources*", MOBMAS should distinguish between resources that are available to the agents within the system only and those that are available to other systems. This differentiation will help to clarify the system boundary during design.

10. The AND/OR decomposition of system-tasks and agent-goals should be represented using the well-known notation of AND/OR graphs (Figure AppendixF.1) rather than adopting the uncommon notation introduced by TROPOS (Figure AppendixF.2).

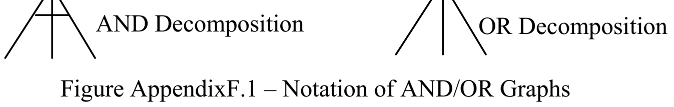


Figure AppendixF.1 – Notation of AND/OR Graphs

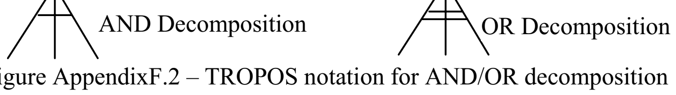


Figure AppendixF.2 – TROPOS notation for AND/OR decomposition

APPENDIX G

EXTERNAL DEVELOPERS' EVALUATION OF MOBMAS

This appendix documents the evaluation of MOBMAS by Dr. Ghassan Beydoun and Dr. Cesar Gonzalez-Perez, who used MOBMAS on a "Peer-to-Peer Information Sharing" application (cf. Appendix H). Each evaluation contained each developer's opinions on the strengths, areas for improvement, how to improve these areas, the ease of understanding and the ease of following of the steps and model kinds of MOBMAS. These opinions were recorded via a specially-designed evaluation form. It should be noted that the forms used by the two developers is slightly different from each other, because they were built upon the two different versions of MOBMAS[102]. Some steps and model kinds listed in the evaluation forms are also different from those specified in the final version of MOBMAS. This is because these forms were based upon the earlier versions of MOBMAS.

Evaluation of Developer 1

DEVELOPER: DR. GHASSAN BEYDOUN

Overall ease of understanding of the development process: High
Overall usability of the development process: High
Overall ease of understanding of model definitions: Medium

[102] Recall that MOBMAS was refined after the evaluation of the first developer.

Evaluation of Analysis Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of step and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Develop System Task Model (optional)	- MOBMAS overlooks the fact that organisational chart can be seen as a result of functional analysis. - Analysis of system functionality should be done for all applications. Analysis of real-world organisational structure can be optional.	High	Medium
STEP 2. "Analyse Organisational Context" (optional)	The real-world organisational structure does not always necessarily correlate with software agent roles.	High	High
STEP 3. Specify roles	MOBMAS does not accommodate dynamic roles.	High	High
STEP 4. Identify Application Ontologies		High	High
STEP 5. Develop Application Ontologies	How the role and task models can be used here is not articulated. There is a lot of focus on notation instead.	Medium	Medium
STEP 6. Identify ontology management roles	This step is described as an option, but there are not any guidelines of when to use this step.	High	Medium

Evaluation of models		
Models	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
System Task Model		High
Real-World Organisational Chart		High
Role Model	Relationships between this and organisational charts are not explicit.	High
Ontology Model	Class diagrams are sufficiently powerful to express ontologies. How some of previous models can be used to develop parts of this model is mentioned briefly, without any specific procedures to follow.	High

Comments on the strengths of Analysis steps and techniques for performing steps:

The first three analysis steps make the transition from the 'system requirement realm' (creating System Task Model) to the agent description realm (Role Model). Development steps offer two layers of abstractions (from system-tasks to roles), getting closer to the agent world design and implementation. The analysis phase also suggests steps from moving from layer 1 (system-tasks) to layer 2 (roles). This layered abstraction view is a useful and intuitive complexity management approach to analyse and implement MAS.

The development steps involved in the above 2-layered abstraction generate (as a by-product) parts of the MAS Application Ontology. Having a stream in the analysis effort focussed on ontology development can be used to verify the completeness of the two earlier models (system-tasks and roles).

Comments on the strength*s* of Analysis models and modelling techniques:

As earlier commented, the two models System Task Model and Role Model are logically related. The notation suggested for the System Task Model is intuitive (And-Or graph) for non programmers to follow (this notation has been used before in AI to represent declarative knowledge). The process of generating the two models produces parts of the MAS Application Ontology. The development of MAS Application Ontology can serve as the completeness verification of the previous models. If developed by a different person, it may also serve as the validation of the previous two models.

Any suggestions for improvements on Analysis steps and techniques for performing these steps?
I wonder if there should be a spiral development between System Task Model and Role Model. From our experience in the Peer2Peer application with MOBMAS, identifying roles may lead to articulating some lower level system-tasks. For example, in P2P experience, identifying the role 'Portal agent' led to some deeper insights into refining the task 'Locating a Portal Agent'.

Any suggestions for improvements on Analysis models and modelling techniques?

The two models (System Task Model and Role Model) have some ontological units and relationships uncovered. Viewing the ontological view of the system, as a refinement of the view articulated by the two models, may allow the refinement and validation of the two models based on the ontological analysis. For example, a developed MAS Application Ontology can lead to the refinement of the System Task Model, and if another person undertakes the development of the ontology, this may serve as a validation. For example, if the notion of 'time' is captured in the ontology of the P2P system, then the task of 'Timeout recovery' may become relevant.

Evaluation of Architectural Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Specify agent classes		High	High
STEP 2. Specify MAS organisational structure	Hybrid structures where peers talk to each other and report to a mediator at the same time are not discussed. Maybe a new structure ('hybrid') is worth considering.	High	Medium
STEP 3. Specify resources	This is for heterogeneous systems only.	High	High
STEP 4. Develop System Overview Diagram	This is too small to be a step. Is it not combining steps 1 and 3? What does this step do in addition to 1 and 3?	High	High
STEP 5. Identify Resource Application	Should this not be merged with step 6?	High	High
STEP 6. Develop Resource Ontologies		High	High

Evaluation of models			
Models		**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Agent Class Model	**Agent Class Diagram**	Too many fields. I wonder whether it is possible to fit everything. With respect to 'ontologies': it is not clear what is meant by this.	Medium
	Agent Relationship Diagram	Association is very general. There are not any specified relationships between classes.	Medium
MAS Organisational Structure Model			High
Environment Model			Medium

Comments on the strengths of Architectural Design steps and techniques for performing steps:

- Considers a wide range of possibilities.
- Has a good focus on open heterogeneous systems.

Comments on the strengths of Architectural Design models and modelling techniques:
NA

Any suggestions for improvements on Architectural Design steps and techniques for performing these steps?
The number of steps should be reduced. Too many steps which are similar confuses. Suggestions to do this: note that the fundamental exercise is one of modelling. The result of this are models. Models can be represented in many ways, diagrams is one way to represent models. For instance, identifying the ontology is a modelling exercise. Why not combine 'identification' and construction of ontologies into one step: 'developing ontology'. An ontology is a model. Diagrammatic representation of the ontology is not a new modelling task. You can use the word 'diagram' instead of model, and combine similar steps into a single step which includes identification (I assume you mean by this identification of the basic units), development (you call this construction) and drawing.

In relation to the above point, you can present organisational structure first, then agent classes (e.g. authority can be mapped to classes relationships in the developed structure); or possibly you can combine these steps into one modelling task which is closely related to the Role Model. Some explicit spiral between the three (agent roles, classes and organisation structure) might be fruitful.

Much of the steps related to resources are related to a specific kind of MAS, heterogeneous systems. I recommend that this point is emphasised more.

Any suggestions for improvements on Architectural Design models and modelling techniques?
See above

Evaluation of Agent Internal Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Specify Agent Belief Set	If the initial belied set is to be modified, it is not clear what MOBMAS recommends (in terms of believed revision).	Medium	High
STEP 2. Specify Agent-goals and Events	It is not clear how an agent would handle conflicting goals. Commitment strategies are not also discussed.	High	Medium
STEP 3. Specify Agent Plans	It is not clear whether the plan is a sequence of sub-goals or a sequence of actions. It ought to be sequence of sub-goals if it is to be general.	High	Medium

Evaluation of models		
Model	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Agent Behaviour Model	See earlier comment regarding agent plans	Medium

Comments on the strength*s* of Agent Internal Design steps and techniques for performing steps:
It relates the earlier analysis of the system-tasks to the internal structure and behaviour of individual agents. It goes a long way towards this.

Comments on the strength*s* of Agent Internal Design models and modelling techniques:
The Agent Behaviour Model is a powerful modelling bridge to link the structure of the internals of agents to system-tasks from the analysis phase.

Any suggestions for improvements on Agent Internal Design steps and techniques for performing these steps?
In complex scenarios, the modelling units of a general plan model can not be the exact actions to be performed by the agents. One way to keep the plan model general is to express it in terms of sub-agent-goals that can be used later on by an off-the-shelf planning language (e.g. STRIPS or ADL). Specifically, these planners will select which actions to perform at run-time, and the sequence of these actions.

Other points which MOBMAS might want to comment on:

- Belief revision and how this or is not related to capabilities of an agent (e.g. learning)
- Commitment strategies (recognition of failed plans)
- Conflict between agent goals (preference and selection of goals)

It is not expected from a single methodology to handle all possible scenarios to deal with those issues. However, an awareness of those issues seems to be in place since the methodology discusses goal modelling to a great degree and learning to a lesser degree.

Any suggestions for improvements on Agent Internal Design models and modelling techniques?
It is not clear how far this phase wants to go into capturing details of an agent. It seems that it is going to very low level (e.g. in verification of ontologies against plans). This probably led to some of the observations above, with respect to how plans should be modelled.

Evaluation of Agent Interaction Design Phase

Evaluation of steps				
Steps		**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Select interaction mechanism			High	High
STEP 2. Develop Agent Interaction Model	**For direct interaction mechanism**	The structure of the organisation of the system can be checked here.	High	High
	For tuplespace/ tuple-centre interaction mechanism	The structure of the organisation of the system can be checked here.	High	High
Evaluation of models				
Model		**Comments on *weaknesses* of models and techniques to produce models**	**Ease of understanding** (High, Medium or Low)	
Agent Interaction Model			High	

Comments on the strengths of Agent Interaction Design steps and techniques for performing steps:
Integrating standard interaction mechanisms with the rest of the methodology makes the methodology much more useable.

Comments on the strengths of Agent Interaction Design models and modelling techniques:
Integrating standard interaction modelling notations (e.g. A-UML Interaction diagrams) with the rest of the methodology makes the methodology much more useable.

Any suggestions for improvements on Agent Interaction Design steps and techniques for performing these steps?
My only concern in this step is where the ‘communicative actions’ are coming from.

Any suggestions for improvements on Agent Interaction Design models and modelling techniques?
The Agent Interaction Model can be used to verify the system organisational structure as well as the Agent Class Model.

Evaluation of Deployment Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Identify agent-environment interface requirements	There is lots of effort on the distinction between physically embedded agents and software embedded agents.	Medium	Medium
STEP 2. Select agent architecture	Some mentioned agent architectures are not 'market' products (e.g. SOAR is not a market product). Some planning languages are out there in the market (e.g. STRIPS but the more commonly superseding language is ADL).	High	High
STEP 3. Specify infrastructure facilities	It assumes distributed processing. How about MAS built for simulations.	High	High
STEP 4. Instantiate agent classes		High	High
STEP 5. Specify deployment configuration	Agent mobility is represented in this phase. However, from earlier discussion on the architecture phase, mobility is excluded in MOBMAS otherwise.	High	Medium

Evaluation of models		
Model	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Environment Model		High
Architecture Model	Mobile agents are discussed only in this model (namely Deployment Diagram).	Medium

Comments on the strengths of Deployment Design steps and techniques for performing steps:
They take into account important issues: number of agents deployed and organisation of external resources.

Comments on the strengths of Deployment Design models and modelling techniques:
It focuses on distributed environment issues, physically embedded agents and external resources planning.

Any suggestions for improvements on Deployment Design steps and techniques for performing these steps?
MAS built for simulations are not accommodated in this phase. In addition, mobile agents seem to appear only in this phase. I have been under the impression that MOBMAS does not deal with mobile agents.

Any suggestions for improvements on Deployment Design models and techniques modelling techniques
Explicitly exclude mobility or qualify and reconcile why it is accommodated here and is not accommodated elsewhere. If the notation for MAS Deployment Diagram accommodates mobile agents, but MOBMAS does not, I suggest that this is explicitly stated e.g. in a footnote. MOBMAS should also address the case of MASs developed for simulation.

Evaluation of Developer 2

DEVELOPER: DR. CESAR GONZALEZ-PEREZ

Overall ease of understanding of the development process: High

Overall usability of the development process: High

Overall ease of understanding of model definitions: High

Overall comments on MOBMAS

MOBMAS is described as being composed of a process, a collection of techniques and *models*. I imagine that you mean *model specifications*. As far as I understand, MOBMAS does not contain models; the models will be built by people using MOBMAS. MOBMAS contains the specifications (or definitions, if you want) of how to build these models. You can use the terms "model specifications", "model definitions" or, even better, "model kinds". Thus you could describe, for example, the Agent Interaction Model Kind, which specifies how developers can build Agent Interaction Models. But these models (which users will create) are not part of MOBMAS; they are an outcome of applying MOBMAS. I would suggest changing the wording wherever it is necessary to clarify this point, because in its current form it results confusing and detracts significantly from the otherwise excellent conceptualisations found in MOBMAS' documentation. You can have a look at the Australian Standard AS 4651 "Standard Metamodel for Software Development Methodologies", which defines concepts such as Model, Model Kind, Model Unit and Model Unit Kind (what you call "modelling concepts"). This could be useful to you.

MOBMAS is also described as being composed of "phases". What do you mean by "phase" here? From the text I imagine that you see "phase" as meaning the description of certain work that must be done. For example, the Architectural Design phase is different from the Agent Internal Design phase because it involves work of different a kind. If I am right, the correct term in software engineering is not "phase" but "activity" (OPEN) or "process" (ISO 15504, OOSPICE), or even "discipline" (SPEM). If you say "phase", software engineers will immediately think of temporal concerns. Phases mean giving temporal structure to the work defined by activities. Since you claim that MOBMAS is iterative and incremental (more on this later) and that the described phases do not necessarily have to be performed in order, then they are not phases because they do not address timing concerns. I suggest you drop the term "phase" in this context and adopt "activity" instead. After this, and only if you want to go beyond, you can organise your five activities into phases to give them temporal structure.

MOBMAS identifies system-tasks from system-goals. What if the target system does not have a single or overall goal? This is a criticism that has been made repeatedly to functional decomposition in traditional structured analysis and design approaches. Imagine an operating system. What is the System Goal? I can argue that there is no overall goal, and if you come up with one, I can show that it is a shoehorned example. If you really want to use the concept of System Goal, then you are constraining your methodology to systems that can be successfully described by an overall goal. You need to acknowledge this and accept that MOBMAS will not be usable for other kinds of systems.

You then define some terms such as System Task or Role. I have a couple of comments on the definitions. First, some of them (Role and Agent Class) are not definitions but comments and discussion. Regarding Role, you say that a role is "something similar to…" which is not really a definition. I would suggest that you try to find a proper definition as you have done with other concepts. Secondly, I would try to keep definitions concise and unambiguous. For example, you define System Task as "a concrete, low-level activity, or unit of work, that…". What is the meaning of "or" here? Is a System Task either a low-level activity *or* a unit of work? Is "low-level activity" the same as "unit of work"? You probably want to simplify this definition. Also, when defining Agent Class, you say that "agent class is abstract and…". What do you mean by “abstract”? Finally, when defining Agent Plan Template, you use the concept of “sub-Agent Goal” that has not been introduced.

I also have some comments about the contents of the definitions. First, in Agent Goal, you say that the agent goal implies proactiveness, since agents need to take the initiative to satisfy their goals. I don’t agree with this. For example, I have a phone switch router agent that has the goal “route incoming calls to their destination using the optimal path”. It is a purely reactive agent, which uses a collection of lookup tables to decide how to do the routing. But it still has a goal. For this agent, having a goal does not imply that it is proactive or takes the initiative. I think that having a goal and being or not proactive are completely unrelated things. Secondly, you define Event as something to which an agent reacts. There is a subtle issue here. Imagine an event that happens and no agents react to it. It is still an event, right? Whether or not some agents react to an event is up to the agents, and is not dependent on the event. I suggest you change “to which an agent reacts” to “to which agents may react”. Thirdly, you define Reflexive Rules as “a sequence of actions that an agent performs to react to an event”. I would say “when reacting” rather than “to react”, since the reactive course is part of the reaction itself.

Figure 6.1 shows MOBMAS’ core concepts plus the relationships amongst them. You say that it is a meta-model. I would not use that term, since “meta-model” implies that this is a model of another model. This can be circumstantially true in this case, but is not the focus of the discussion. In my opinion, you are just presenting a *diagram*, not a *meta-model*. If you use the term “meta-model” I would expect a reason why this is so and not just a simple model or even a diagram. In addition, what is the notation being used for Figure 6.1? I mean, what is the meaning of the boxes, lines, arrowheads, diamonds, etc.? You need to include a legend or either make a reference to some well-known notation such as UML. Then you need to make sure that you stick to the standard. Secondly, the diagram does not show cardinalities, which would be very useful. For example, I can see in the diagram that role-tasks are derived from system-tasks, but how many of each? Furthermore, the diagram includes boxes labelled “Action” and “sub-Agent Goal”, which have not been defined or introduced previously. You probably need to check the consistency of this.

Section 6.1.3 describes the notational components for each model. You use the term “composed of” (or a synonym of this) to describe the relationship between a model kind (such as System Task Model) and the notational components that can be used to depict it (System Task Diagram in this case). I don’t think the relationship is of containment or composition. A model does not *contain* a diagram, but *can be depicted by* it. I would rather say that models can be depicted by notational components. This happens all over Section 6.1.3 at least. Moreover, I would expect the notational components to “depict” or “show” (or even “document”) things, but not “define” or “capture” or “specify” anything. Models define, capture and specify; notations show or depict. However, some of the notational components are described as defining, capturing or specifying their contents. I suggest you change these terms to “depict” or “show”. This also happens within other model kinds in this section. You also say that four models in MOBMAS reuse and extend UML notation. If they are models, how can they reuse a notation? You probably want to say that *the notational components* that depict MOBMAS models reuse and extend UML notation. I suggest clarifying this.

In the Organisational Context Model Kind, you use the term “real-world” quite often to refer to the organisational structure. Why “real-world”? From the explanation in the text, I know that you want to emphasise that the modelling is of the organisational context of the system rather than the system itself, but wouldn’t “organisational” be enough? If you add “real-world”, you are introducing two potential problems. First, I could ask “what do you mean by ‘real’?”, or “real to whom?”. This is a complex philosophical issue that I bet you don’t want to get into. Secondly, you are limiting your model to the “real” world (whatever it means) and discarding other “non-real” worlds. And software is plenty of non-real, virtual worlds! For example, if we take “real” as meaning simply the physical world in which we live, then a fictitious organisation that I find in a novel is not part of the real world. Can I still use MOBMAS to model it? I’m sure I can. But your Organisational Context Model is not depicting the real world. I would suggest getting rid of “real-world” in this context.

When describing Agent Behaviour Model, you introduce the notational components named "Agent Plans" and "Reflexive Rules". I don't think these are good names for notational components, because they are just the plural form of modelling concepts. "Agent Plans" really means a collection of agent plans; it does not convey the idea of a diagram or a document. But you want it to mean some other thing, namely a specific notational component that depicts a view of the Agent Behaviour Model. I think that you probably need to name it accordingly. For example, "Agent Plan Diagram" or "Agent Plan Description" or something like that. The same happens with "Reactive Rules".

Evaluation of Analysis phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of step and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Develop System Task Model	- You claim (backed by Fan 2000) that users often have a clearer idea of system goals than of system tasks. In the experience of other authors (me included) this is quite the other way around. - You say that use cases can be used to identify system goals, but do not say how. - You talk about system goals producing and consuming resources. How can a goal produce or consume anything? Goals are states. You need some kind of process, activity or task in order to produce or consume resources. - I would suggest you have a look at "Software Requirements" by Karl Wiegers. Very different approach to the classical references. There is a new second edition now. - You make constant references to the "actors" or "performers" of system goals and tasks. But we haven't determined them yet. We know nothing about them at this stage! - You talk about system tasks being "fulfilled". I imagine you mean "executed".	Medium	High
STEP 2. "Analyse Organisational Context" (optional)	- You start saying that the previous step elicits system functionality. I think it models system functionality. Elicitation is only a part of requirements engineering. - I would avoid using the term "real-world" here. Please	High	High
STEP 3. Develop Role Model	- One of the biggest problems of traditional structured methods is that they rely too much on functional decomposition. You are at risk of falling in the same trap here. You say that "System-tasks that share the same parent […] are typically related strongly in term of functionality, thus being good candidates for being combined into one role.". I strongly disagree. Sharing a common parent is a functional property, completely unrelated to the structural property of who the responsible for such piece of functionality is. Sharing or not sharing a parent is irrelevant as far as role	Medium	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of step and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	assignment is concerned. Only this way you will avoid creating a system organised around a functional decomposition of its requirements. - Furthermore, you say that System Tasks should not be associated to the same role when they are expected to be executed on distributed physical locations or when they interface with distributed or different kinds of resources. Why? Some roles are precisely defined to coordinate tasks like this! - I don't think that the term "social task" is appropriate. The term "social" (systematically misused by the AI community) has strong implications such as the existence of a common set of rules and both subjective and inter-subjective spaces. I don't think you mean this. I would avoid this term and use "joint", "distributed" or "collective" instead. If this is what you mean, why use another word? - The cardinality of the relationship between System Tasks and Role Tasks is not clear. - You use guillemets to label the different sections in a box representing a role. This is extremely confusing since UML uses the same symbol for stereotypes. I would simply remove these symbols.		
STEP 4. Develop Ontology Model	- You state that the Conference Program Management system does not need a Domain ontology because the information to be handled is just "profiles" of the conference papers. First, I don't understand what you mean by "profiles". Secondly, I can see a clear Conference Program domain ontology containing the concepts Paper, Reviewer, etc. - I would say that an application always relates to a particular domain. Actually, I can argue that "application" and "domain" are almost synonyms. Therefore, you cannot say that an application only needs Task ontology without needing Domain ontology. If an ontology defines concepts and their relations (structure), this is orthogonal to any usage (behaviour) that you may do of such concepts. For example, you say that User Query, Keyword and Result List are domain independent and you build a "task ontology" for them because you happen to use them for a particular task. I would adopt a different approach. To me, these concepts belong to a Query Management domain that exists on its own and which	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of step and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	you happen to need for your particular task. - In connection with the previous point, it seems that an ontology that "specialises" from a more general one can do it by (a) adding subtypes of the concepts in the general ontology and (b) incorporating specific instances of such concepts. This is not said. For example, Diabetes is not "subsumes" by Disease, but is an instance of it. Similarly, Hyperglycaemia and Hypoglycaemia are instances of Symptom, and not subtypes. - You need to say what notation you are using for the diagrams. You say earlier that UML is used for ontology diagrams, but the reader will not know that when looking at diagrams before that. - You say that MOBMAS recommends a graphical language for ontology modelling but, if not powerful enough, a textual one can be used. Are you assuming that graphical languages are, in general, less powerful than textual ones? I would differentiate between power of expression and power of communication. The first relates to the abstract syntax and the semantics of the language, which are not related to graphical vs. textual whatsoever. The second relates to concrete syntax or notation, which does relate to graphical vs. textual. Some clarification would be helpful here. - You introduce the concept of generalisation but define specialisation! - You introduce the concepts of aggregation and composition. These concepts are poorly defined by UML, and they can get you in some trouble (see some papers by Henderson-Sellers on whole/part relationships). If you are taking these concepts straight from UML, I would recommend you just point to the definitions in UML and avoid including your own definition. If you want to enhance over UML, then you need to be careful with how you define these terms. See Brian's papers for details. - You say that relationships may be annotated with cardinality indicators. If this is only optional, how are the relationships supposed to be implemented? You need cardinality specifications for every single relationship before you can put them into a computer. - You consider three types of "ontological mapping"		

Evaluation of steps			
Steps	**Comments on *weaknesses* of step and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	relationships: equivalent, subsumes and intersects. I can see two problems with this. First, how is subsumption different from instantiation or specialisation? The definition is not clear; you say that "…one concept includes the other…", but the meaning of this can be either instantiation or specialisation. Secondly, and most importantly, an association in object-oriented modelling (e.g. UML) describes a potential relationship between instances of the involved classes. It does not describe a relationship between the classes themselves. However, from the definitions and description of your three stereotypes, you are trying to characterise relationships between classes. For example, if you say that Disease subsumes Diabetes, that means that instances of Disease may subsume instances of Diabetes. And this is not what you mean; you mean that the class Disease subsumes the class Diabetes. In summary: associations are not the appropriate mechanism to do this. - In Figure 6.13, how can a "Hit" be equivalent to a "Car"?		
STEP 5. Identify ontology management roles	To me, ontology management looks similar to database management in traditional systems. It is an infrastructural service provided to applications, and therefore application development usually does not deal with the modelling of such infrastructure. If this is not the case and ontology management is not an infrastructural service of MAS applications, this must be clearly stated to avoid confusion.	High	High

Evaluation of models		
Models	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
System Task Model		High
Organisation Context		High
Role Model		High
Ontology Model		High

Comments on the strengths of Analysis steps and techniques for performing steps:
A lot of heuristics are given, which is unusual for a text on methodologies and extremely welcome.

Comments on the strengths of Analysis models and modelling techniques:
See above evaluation

Any suggestions for improvements on Analysis steps and techniques for performing these steps?
See above evaluation

Any suggestions for improvements on Analysis models and techniques modelling techniques
See above evaluation

Evaluation of Architectural Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Specify MAS organisational structure	- The phase is called "architectural design" but you talk about specifying the "organisational structure" of the system. What is the meaning of "organisational" here? If you mean architecture, just say "architecture" instead of "organisational structure". - You mention the term "control regimes" a few times. What is it? Control issues in a system are part of the system dynamics, which happen at run-time. On the other hand, architecture is a structural, design-time concept. Architecture cannot specify control, but support it. - All your discussion of organising roles in layers or federated groups seems to be based exclusively in performance reasons ("helps to reduce interaction traffic", "interaction costs are sufficiently low"). This is not the reason that most authors would give for architecting a system in layers or blocks. The main reasons are two: (a) modularity (most important) and (b) non-functional requirement support. Modularity is concerned with separation of concerns, technological isolation and, in general, encapsulation and containment. Non-functional requirement support is related to providing the necessary structural mechanisms so the necessary non-functional requirements (robustness, integrity and performance) are met. As you can see, performance is only a small bit. - You keep using keywords enclosed in guillemets, which closely resemble UML stereotypes. I don't think this is a good idea. - You use an arrow sign embedded in the «control→» keyword. This does not look very intuitive. Furthermore, it ties the model to the graphical representation, which is not a good idea. I would try to replace it by an adorned line, perhaps. - After Figure 11.6, you say that roles should be organised so they highlight the layers or groupings	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	adopted. Well, I think that this works the other way around. Once you organise your roles in a sensible manner, layers or groupings will arise. You cannot force your roles into groups just to make the model look neat.		
STEP 2. Develop Agent Class Model	- You talk about "grouping" roles into agent classes. I would say "associating" roles to agent classes. - The reasons that you give to associate multiple roles to a single agent class look very arguable to me. First, you say that if two roles interact intensively with each other, they should be associated to the same class. I don't think this is right. A client and a server (in whichever domain you want to think of) interact intensively but are, by definition, well separated entities. Second, you say that if two roles share a lot of common data or resources, then they again should be mapped to the same class. Again, I disagree. In my view, the most important issue you need to look at in order to decide whether or not to put two roles together into a single class is semantics. Look at their names, look at whether it makes sense, from a semantic perspective, that a single class plays both roles. You hint at this by talking about coherence and having a single class name with no conjunctions. This is OK; now you need to change your heuristics (currently based on interaction bandwidth and shared data) to match this. - You give some advice on the computational complexity of agent classes. I think this is not realistic at this point. When you develop a software system in the real world, you almost never know what kind of hardware is going to run it, and even if you know it, it will change every 12 or 18 months most probably. So any reasoning based on processor load, at this stage, seems inappropriate to me. - There is a great and ongoing ambiguity between "agent" and "agent class". You sometimes talk about assigning roles to agents, but you have just said that roles are assigned to agent classes. You probably want to revise this whole section and make sure that "agent" and "agent class" are used with rigour. - For the Agent Class Diagram, you say that when an	High	Medium

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	agent class dynamically plays a particular role, this role must be differentiated from other static roles by being annotated with an adornment. I don't think this is useful at all. Structural models show static properties of entities, i.e. properties that are observable at any point in time. When you represent an agent class in the Agent Class Diagram and enumerate its roles besides its name, you are saying that in the system there will be agents of that class and they will be able to play these roles. Whether or not a specific agent is playing a specific role cannot be guaranteed to be observable at any point in time. All roles look the same, no matter whether they are static or dynamic. That is a dynamic concern. You should remove the special annotation that dynamic roles are supposed to have in the current model definition, since it does not make sense in a structural model. - Some of the examples in the text are not very fortunate, in my view. For example, a Search Agent class is assigned roles User Interface and Searcher. The encapsulation of a computation (searching) and a user interaction (user interface) in the same entity violates the n-tier philosophy of separating persistent data from data access from computation (business logic) from user interaction. I am not sure whether this is a characteristic of agent modelling or just an oversight.		
STEP 3. Specify resources	- In your characterisation of resources, you use the term "knowledge source" to refer to databases and web servers. Well, a database can be seen as a data source, but only very arguably as a knowledge source. Why not use a more conventional, all-encompassing term such as "information sources"? Knowledge is a different thing. - You introduce an Environment Model which contains 3 notational components. One component (Resource Diagram) is developed in this step, while the remaining two will be dealt with in Deployment Design. I don't like this because, to me, a model describes its target at a certain level of abstraction and from a certain perspective. Phases as different as Architectural Design and Deployment Design almost certainly vary significantly in abstraction and	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	purpose, so how can they all be major contributors to the same model? - In the modelling of resources, you mention the "communication properties" dimension. This is very low level at this stage. Things like the IP address of a machine are probably irrelevant at the architectural design level (which is what you are doing here). I would remove this bit and move it to Deployment Design, perhaps. - You recommend considering the introduction of a Resource Broker role. This looks to me like something belonging to the infrastructural services of the run-time environment, not something that each application needs to design. It is similar to a name server or a middleware service in traditional services: they are already there for you to use. You don't design a new one for each application you write. - This step uses some not very good examples. For example, role task "Provide yellow page services" and "Display acknowledgement" are at very different levels of granularity. I would try to keep role tasks (or any other model units of the same kind) at the same level of granularity. Furthermore, the Feedback Manager role has been assigned the task "Display acknowledgement", which seems to belong more naturally to the User Interface role. Similarly, the User Interface role contains the task "Extract keywords from user query" which looks like a computation and not a user interface operation.		
STEP 4. Extend Ontology Model to include Resource Application ontology	- You say that for resources that are information sources, the resource ontology is the conceptual schema of the information stored in the resource. This is arguable. Consider levels of abstraction: the information source may store information at a very low level of abstraction, and therefore its conceptual schema would contain all sorts of details that are irrelevant (and even harmful) to you. You need a different ontology, one that maps to this conceptual schema but removes unnecessary detail. In my experience, this happens all the time in real-life projects with databases. So you cannot simply say that the conceptual schema of the information source is equivalent to the resource ontology.	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	- You say that, in some cases, a resource ontology can coincide with (a fragment of) the application ontology, and that, in this case, the application ontology will suffice. I don't agree, because the non-agent resource, by definition, is external to your system, and keeping a separate ontology for it, even if it repeats concepts in you application ontology, is recommendable for the sake of modularity. Imagine that you want to make a change in your application ontology but need to keep the resource ontology untouched: you cannot do it unless you have the two ontologies separate.		

Evaluation of models			
Models		**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Agent Class Model	**Agent Class Diagram**		High
	Agent Relationship Diagram		High
MAS Organisational Structure Model			High
Environment Model			Medium

Comments on the strengths of Architectural Design steps and techniques for performing steps:
NA.

Comments on the strengths of Architectural Design models and modelling techniques:
NA.

Any suggestions for improvements on Architectural Design steps and techniques for performing these steps?
See above evaluation

Any suggestions for improvements on Architectural Design models and techniques modelling techniques
See above evaluation

Evaluation of Agent Internal Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Specify Agent Belief Set	- The name "belief set" is not intuitive. "Belief set" means a set of beliefs, not a formal structure of beliefs. You may want to change this name. - Along the whole chapter, there is an ambiguity of "agent" vs. "agent class". This also happened in earlier chapters. For example, you talk about defining the belief set "for each agent", when you surely mean "for each agent class". - When a belief set changes (because the ontologies change), the belief state must also change accordingly, adjusting itself to the new structure. You don't discuss this. How are agents notified of ontology changes? - A belief set, as defined, is one or more ontologies to which the agent commits. Isn't this overkill? I can think of many cases in which an agent would only need a small bit of an ontology to do its work. Therefore it would be nice if agents could commit to a subset of a given ontology.	High	High
STEP 2. Specify Agent-goals and Events	- Across the whole chapter, it is stated (and assumed) that reactive agents exhibit simple behaviour while pro-active agents exhibit complex behaviour. For example, you give some rules to classify a Role Task of an agent class as pro-active if the task will need deliberation and complex processes. Also, you recommend a reactive architectural style if the agents hold a simple representation of the world, while a pro-active style is suggested if the knowledge involved is more complex. In my view, this is wrong. Two different concerns are being mixed here. The first is pro-activity vs. reactivity and the second is the complexity of behaviour. Pro-activity and reactivity are related only to the particular way in which a piece of behaviour (an action, a goal, whatever you want to call it) is triggered. The degree of complexity of this behaviour is completely orthogonal to how it is triggered. For example, I have a phone switch router agent that is completely reactive: only does something when a call comes in. The behaviour that this triggers, however, is highly complex, and a lot of deliberation with other agents is necessary.	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	- You classify the tasks "Extract keywords from user query" and "Get information from resources" as pro-active tasks. I disagree. You state that they are triggered by some stimuli, so they are a reaction to something. Therefore, they are reactive. You say that they are pro-active because they are highly complex, which is true, but not related to proactivity/reactivity at all. - In connection with the above two paragraphs, I think that the concepts of proactivity and reactivity are useful as abstract ideas to characterise agents, but they are not that useful at a detailed level. In software systems (and in most systems, actually), everything is reactive to a high degree. You keep talking about stimuli that trigger proactive tasks (see Figure 12.11 for example). This is an oxymoron. Only really complex entities (such as human beings) can be truly pro-active, i.e. initiate action without a stimulus. I am aware that this is a deep issue with multiple ramifications, but unfortunately I cannot defend your usage of proactivity/reactivity. - Regarding events, you define them as something significant that happens in the environment. Now, how do you define "environment"? Does the environment include other agents? Does it include all the agents, including self?		
STEP 3. Develop Agent Behaviour Model	- For each action, you define preconditions and effects. Why don't you use the term "postconditions" instead of "effects", for the sake of symmetry? - You mention that two (or more) agents can share the same agent-goal. However, from you previous chapters, I recall that each agent has its own agent-goal, but the definitions of these goals are the same. So, strictly speaking, they do not share a goal, but have equivalent (or identical) agent-goals. This may seem a bit of word play but it is not, as you can see in the next paragraph. - You say that when two (or more) agents "share" the same agent-goal, they will have to interact to achieve that goal and possibly do some distributed planning. I don't think this is necessarily true. Since agents do not actually share a goal but have goals that are identical (see previous paragraph), they may as well pursue their identical goals separately.	Medium	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for step**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	So, distributed planning is only an option. - The differences between blind (or fanatical), single-minded and open-minded strategies are hard to understand. I suggest you use the exact same words to define each one changing only the minimum. - Your usage of "vice versa" is a bit odd. This expression means that the same relationship that happens between A and B also happens between B and A. But in most of the cases where you use it, the relationship between the parties involved is not the same in one way and in the other. - You use attributes from the ontologies as datatypes. For example, you define carModel: Car.Model. This gives no room for using "basic" parameters such as a string or an integer that bear no relationship at all to the ontologies. For example, if you want to display a message to the user or wait for a specific number of seconds, you would need to pass a string or an integer, respectively. Forcing the developer to create a class in the ontology (and perhaps a whole new ontology!) because of this seems inappropriate.		

Evaluation of models		
Model	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Agent Behaviour Model		High

Comments on the strengths of Agent Internal Design steps and techniques for performing steps:
A lot of heuristics are given, which is unusual for a text on methodologies and extremely welcome.

Comments on the strengths of Agent Internal Design models and modelling techniques:
NA.

Any suggestions for improvements on Agent Internal Design steps and techniques for performing these steps?
See above evaluation

Any suggestions for improvements on Agent Internal Design models and techniques modelling techniques
See above evaluation

Evaluation of Agent Interaction Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Select interaction mechanism	- The capitalisation of "tuple centre" is random. You may want to homogenise it. - You frequently use the term "coordination" when you really mean "interaction". Many interactions are not related to coordination at all. I think you should use "interaction" where you say "coordination", except for the (perhaps) few places where you really mean "coordination". - You say that using a tuple centre and using direct messages between agents are the two common agent interaction mechanisms. However you do not say which others exist, or where to find them. In this step, things are explained as if these two mechanisms where the only ones. - You say that "…the ACL interaction mechanism […] infers a strong…". I am not sure what you mean with this sentence, but "to infer" means to deduce, to conclude. Is that what you mean? - Also in the same section, you sometimes use the term "agent" when you really mean "agent class". This is extremely confusing. For example, you say that embedding the constraints that govern agent interaction into the agents themselves can be difficult if the number of agents is large. You mean agent classes, I bet. Otherwise, the sentence does not make any sense. - In a footnote in the same section, you say that if an agent goes down, then another agent of the same class can replace it. I don't think this is the rule. As you well know, agents obtain knowledge during their lives, so two agents of the same class that have lived for a while will probably have different beliefs and intentions. How can then one replace the other? - You say that the ACL Interaction Mechanism is not as "efficient" (I would suggest "appropriate" instead) as using the tuple centre mechanism when the MAS is open and dynamic, contains heterogeneous agents, and when the agents have many shared agent-goals. You say that this happens because there is a strong coupling between "the agents' behaviour and the management of the coordination process". I think you should change the wording here, because the coordination process is part of	Medium	High

Evaluation of steps				
Steps		**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
		agents' behaviour. How an agent interacts with other agents, what for and when, is all part of the agent's behaviour. In addition, I don't think that agents have "shared agent-goals". They may have the same agent-goals, but not shared. I already discussed this in a previous chapter. - You say several times that the tuple centre is capable of some "reasoning", or even "low-level reasoning". Do you mean "processing"? If so, I would use "processing", which has the right technical meaning. "Reasoning" is what humans do.		
STEP 2. Develop Agent Interaction Model	**For direct interaction mechanism**	- In "Specify agent synchronisation" you supposedly describe two synchronisation methods: synchronous and asynchronous. Well, these are not synchronisation methods, especially the latter. If a MAS adopts an asynchronous approach, the agents in it are not synchronised at all, by definition. They will need to use mutexes or some other synchronisation objects to actually synchronise with one another. This would be "synchronisation methods". What you are describing are "synchronisation approaches" or "modes" but not "methods". What's the meaning of "method", by the way? - You define a guard condition as the condition "by which" the message is sent. I can't understand this. Is it the condition that sends the message, that makes sensing the message possible, that triggers the message being sent…? You need a better definition here. - You try to define "sequence-expression" but the words are not a definition. It is only a comparison with UML. - When describing the content of messages, you use datatypes that look very much like coming from an ontology. However, you don't say that. I think it would be good, at this stage, to say explicitly that datatypes map to the some ontology. - You use some "built-in" datatypes such as "Integer". What do you mean by "built-in"? Where are they built-in? Who or what provides these datatypes?	High	High

Evaluation of steps				
Steps		**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
	For tuplespace/ tuple-centre interaction mechanism	- Again, in the message content, you use datatypes that look like coming from an ontology but you say nothing about this. This is puzzling for the reader. - Again, you mention the existence of "built-in" datatypes. Where are they built in? - You describe "synchronisation methods" which are really approaches or modes, not methods. - Figure E.6 shows an "out" arrow coming out of the tuple centre into an agent. I think that is wrong since "out" arrows always go into the tuple centre. - What does an interaction diagram represent? You don't say that anywhere. Is it a specific conversation, or is it a specification of the conversations that may take place?	High	High

Evaluation of models		
Model	**Comments on *weaknesses* of models and techniques to produce models**	**Ease of understanding** (High, Medium or Low)
Agent Interaction Model		High

Comments on the strengths of Agent Interaction Design steps and techniques for performing steps:
A lot of heuristics are given, which is unusual for a text on methodologies and extremely welcome.

Comments on the strengths of Agent Interaction Design models and modelling techniques:
NA.

Any suggestions for improvements on Agent Interaction Design steps and techniques for performing these steps?
See above evaluation

Any suggestions for improvements on Agent Interaction Design models and techniques modelling techniques
See above evaluation

Evaluation of Deployment Design Phase

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 1. Identify agent-environment interface requirements	- Again, you sometimes use the term "coordination" meaning "interaction", like in the previous chapter. - You keep mentioning the implementation phase but it is not discussed in MOBMAS. Since you refer to it in specific terms, I wonder how you conceptualise it. What kind of work is expected to be done? What products are generated? Can you add some information on this to your work? - I would think that designing the agent's interface with its environment is not related to deployment at all but architecture or detailed design. It is an integral part of the agent, not related to the specifics of the run-time infrastructure. - You mention differences between sensors/effectors in hardware and software. In real life situations, software developers do not interact with hardware; hardware is virtualised or wrapped by driver components so other software components interface directly with them. For example, if your agent has an effector to move a robotic arm and you want to make it move the arm, you will talk to a software component that virtualises the physical arm. You don't have to worry about the hardware. So, for the software developer, there is no difference at all between interfacing with software or hardware. - As part of the identification of agent-environment interaction requirements (you use "interaction" here rather than "coordination", which is good), you include issues that have already been dealt with in previous phases, such as deciding whether to use a tuples centre or direct messages. - You explain that the Environment Model contains 3 notational elements, 1 of which is created by the Architectural Design phases. I have already commented on this. But here you can see very clearly how the different notational elements of this model have really nothing to with each other. They are not view on the same model but completely unrelated models. I suggest you decompose the Environment Model into smaller models which will make much more sense.	High	High

Evaluation of steps			
Steps	**Comments on *weaknesses* of steps and techniques for steps**	**Ease of understanding** (High, Medium or Low)	**Ease of following** (High, Medium or Low)
STEP 2. Select agent architecture	- You talk about the Implementation phase, but you have not mentioned it before. Is there one? - You say that there are a number of agent architectures available on the market. What do you mean? Are agent architectures products that you can buy? - You mix the concerns of proactivity/reactivity with the complexity of behaviour, as I said some paragraphs earlier. You probably want to revise this. - Two of the criteria that you mention are "Size of knowledge base" and "Support for scalability". The differences between these two are not clear. - Figure 6.51 shows some little solid black circles that are not defined.	High	High
STEP 3. Specify infrastructure facilities	In my view, everything in this step is actually architecture design, not deployment design. Specifying how a system will interact with other systems, including its infrastructure, is part of architecture. What do you understand by "deployment"?	High	High
STEP 4. Instantiate agent classes	You use a rounded rectangle to represent an agent instance in the Agent Instantiation Diagram. What do you need this for? Agent instance icons bear no information and add no value to the diagram. Can't you get rid of them altogether?	High	High
STEP 5. Specify deployment configuration	You use the word "physical" quite often. For example, you say that agent platforms are the *physical* infrastructure in which agents are deployed. And nodes are *physical* hosts. And they are linked by *physical* connections. I am not sure what you mean by "physical", but if you delete the word from these sentences everything makes sense and, in addition, you are not limited to "physical" entities. For example, nodes do not have to be real computers; they can be virtual machines or other non-physical processors. The same for connections between nodes. Network connections are physical at a very low level, but most application deployment activities take place at high levels where the physical topology of the network is not important, only its logical topology. I would suggest deleting this word unless you have a good reason to keep it.	High	High

Evaluation of models		
Model	**Comments on *weaknesses* of models and modelling techniques**	**Ease of understanding** (High, Medium or Low)
Environment Model		Medium
Architecture Model		High

Comments on the strengths of Deployment Design steps and techniques for performing steps:
NA

Comments on the strengths of Deployment Design models and modelling techniques:
NA.

Any suggestions for improvements on Deployment Design steps and techniques for performing these steps?
See above evaluation

Any suggestions for improvements on Deployment Design models and techniques modelling techniques
See above evaluation

APPENDIX H
APPLICATION OF MOBMAS

This appendix documents the "Peer-to-Peer Information Sharing" application on which MOBMAS was used by the two developers, Dr. Ghassan Beydoun and Dr. Cesar Gonzalez-Perez. The appendix also presents the major models produced by each developer to illustrate the design of MAS for the application as a result using MOBMAS.

Problem Description – "Peer-to-Peer Information Sharing"

In recent years, the Peer-to-Peer (P2P) networking paradigm has become one of the most rapidly developing areas of modern computing (Klampanos and Jose 2003). P2P contrasts with the well-known Client-Server networking model in that all nodes in the network are capable of acting as both server and client, that is, each node can serve as both provider and user of services (Klampanos et al. 2003; Mine et al. 2004). The P2P networking model helps to avoid the problems of bottle-neck and heavy traffic that is commonly witnessed in the Client-Server architecture.

In this application, the P2P model is employed for *information sharing*. Information to be shared is files such as HTML, pdf and multimedia (e.g. music or video). The users of the system are distributed "peers" in the information sharing network. Their knowledge bases (i.e. stored files) are enlisted, and the users can communicate directly with each other to exchange this knowledge.

System requirements
Each user possesses a knowledge base containing files that he/she is willing to distribute to other peer users. Each file is identified by its title and type (e.g. HTML, pdf, music or video).

Any user in the network can enter a query to request for files that satisfy his/her query. Each query contains a set of keywords. The system is responsible for identifying those candidate users who may have files that satisfy the query, and sending the query to these users. The answer from each candidate user may either be:

- the titles and types of the files that satisfy the query; or
- a refusal of service (either because no appropriate files are found, or because the user is unwilling to supply the files at the time of request).

When the answers are received from all candidate provider users, the system will combine and refine the results to compose a list of files' titles and types, which is then presented to the user. The user can then select which files he/she wants to download. The system then contacts the respective provider user to carry out the file transfer process. After a successful transfer, the user's knowledge base is updated to contain the new received file(s).

Each user keeps a record of his/her history of information sharing. The history contains:

- a list that records the queries made by the user and their responders; and
- a list that records the queries received by the user and their senders.

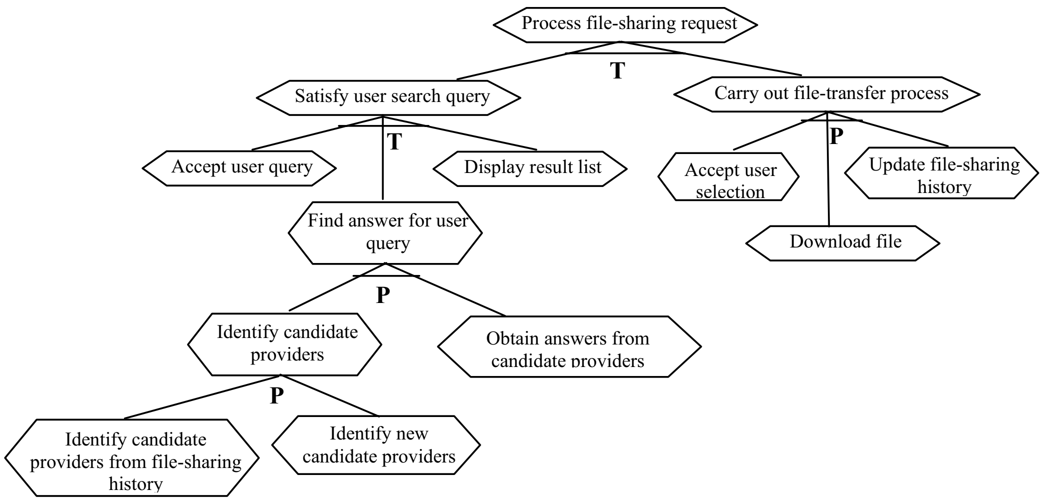
The former needs to be updated every time the user receives a result list from the system, while the latter requires update every time the user replies to a query sent by the system. This history lists help the system to produce *short lists of candidate providers* for future queries, by calculating the similarity between the user's query and a past query (Mine et al. 2004). If no candidate providers can be identified this way, or if all candidate users do not provide the service required, the system will need to broadcast the query to all users in the community, so as to identify new candidate providers. The new providers are eventually added to the history of the user, thereby expanding the user's contact circle.

Although the above schema of information sharing can be applied to any application domain, this research illustrates the use of MOBMAS on the *Movies* domain. An ontology for this domain (written in DAML) is currently available from http://www.cse.dmu.ac.uk/~monika/Pages/Ontologies/CinemaAndMovies.daml.

Major models produced by Developer 1

DEVELOPER: DR. GHASSAN BEYDOUN

System Task Model



Process file-sharing request
T
Satisfy user search query
Carry out file-transfer process
T
P
Accept user query
Display result list
Accept user selection
Update file-sharing history
Download file
Find answer for user query
P
Identify candidate providers
Obtain answers from candidate providers
P
Identify candidate providers from file-sharing history
Identify new candidate providers

Figure AppendixH.1 – System Task Diagram by Developer 1

Ontology Model

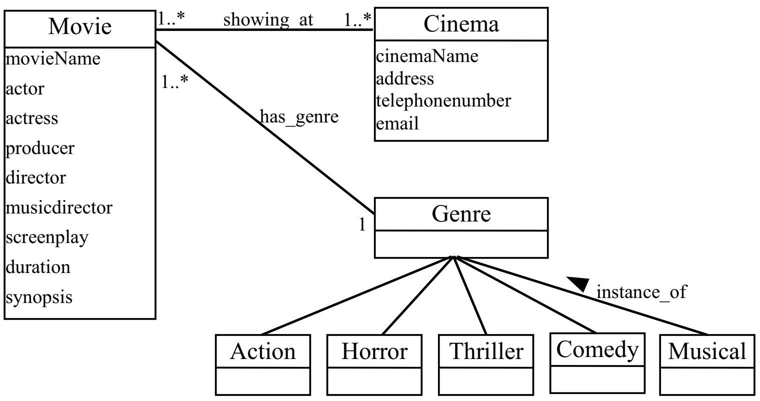


Figure AppendixH.2 – Ontology Diagram for *Movie Ontology* by Developer 1 (based upon DAML ontology at http://www.cse.dmu.ac.uk/~monika/Pages/Ontologies/CinemaAndMovies.daml)

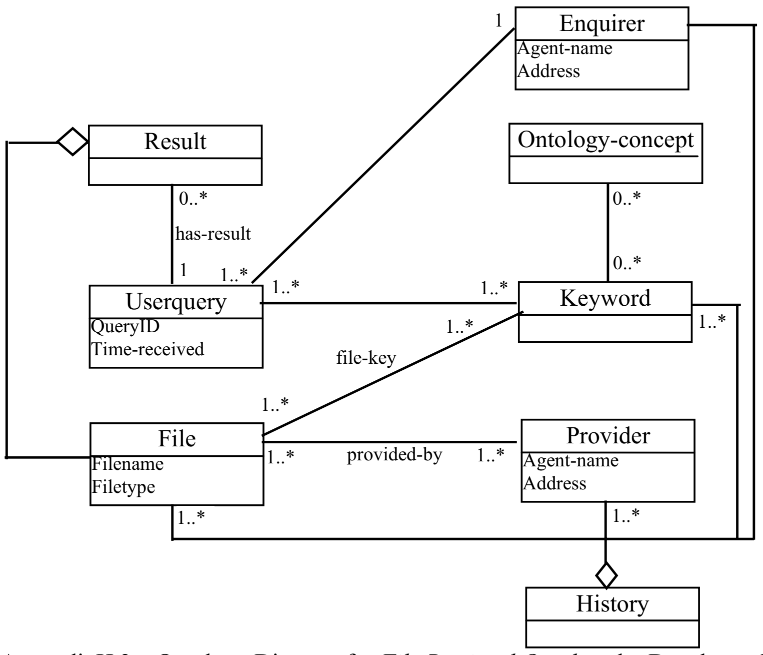
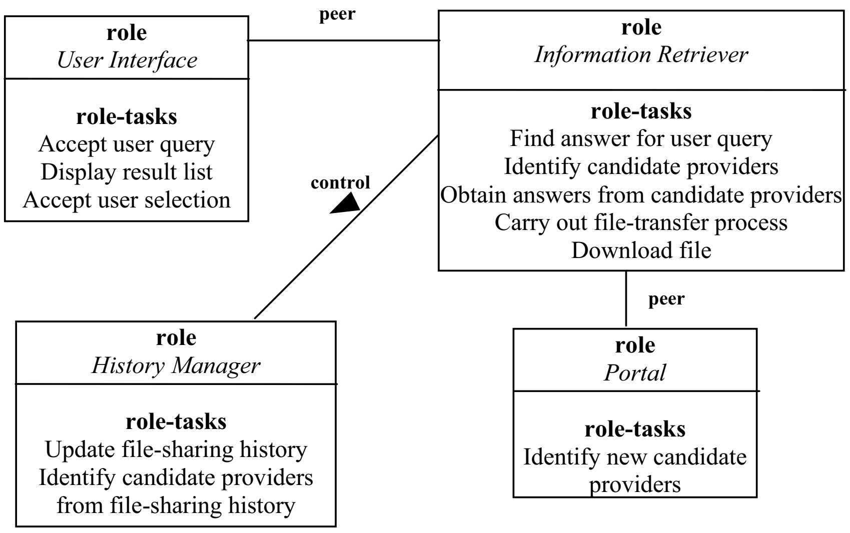


Figure AppendixH.3 – Ontology Diagram for *File Retrieval Ontology* by Developer 1

Role Model



role
User Interface

role-tasks
Accept user query
Display result list
Accept user selection

role
Information Retriever

role-tasks
Find answer for user query
Identify candidate providers
Obtain answers from candidate providers
Carry out file-transfer process
Download file

role
History Manager

role-tasks
Update file-sharing history
Identify candidate providers from file-sharing history

role
Portal

role-tasks
Identify new candidate providers

Figure AppendixH.4 – Role Diagram by Developer 1

Agent Class Model

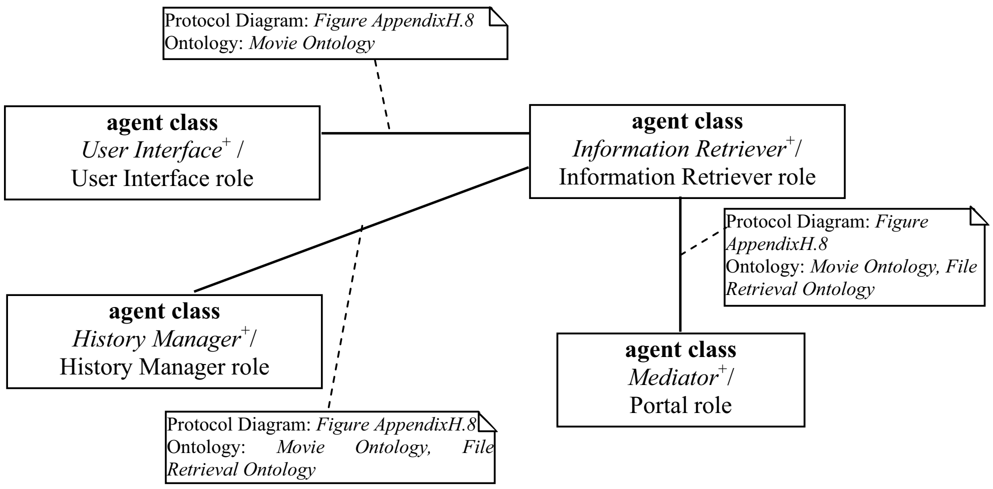

Protocol Diagram: *Figure AppendixH.8*
Ontology: *Movie Ontology*
agent class
User Interface$^+$ /
User Interface role
agent class
Information Retriever$^+$/
Information Retriever role
Protocol Diagram: *Figure AppendixH.8*
Ontology: *Movie Ontology, File Retrieval Ontology*
agent class
History Manager$^+$/
History Manager role
agent class
Mediator$^+$/
Portal role
Protocol Diagram: *Figure AppendixH.8*
Ontology: *Movie Ontology, File Retrieval Ontology*

Figure AppendixH.5 – Agent Relationship Diagram by Developer 1

agent class
Mediator / Portal role

belief conceptualisation
Movie Ontology
File Retrieval Ontology

agent-goals
Address of potential providers are identified given particular keywords

events
Keywords received from Information Retriever agent class

Figure AppendixH.6 – Agent Class Diagram by Developer 1 (for *Mediator* agent class)

Agent Behaviour Model

Initial state: *kw*: Keyword is received from Information Retriever agent
Target agent-goal: *filepointer*: File and *p:Provider* are identified and sent to Information Retriever agent
Commitment strategy: single-minded
List of sub-agent-goals: OntologyConcept_Identified, FileLocated, HistoryUpdated
List of actions:
Action 1: MatchKeyword (*kw:* Keyword, *h:* History)
Pre-condition: *kw* is received from Information Retriever agent
Post-condition: *oc*: Ontology-concept is identified

Action 2: RetrieveFile (*oc*: Ontology-concept, *h:* History)
Pre-condition: *oc* is identified
Post-condition: *filepointer*:File and corresponding *p:Provider* are located

Action 3: UpdateHistory (*oc*: Ontology-concept, *agent_ID*:Enquirer.Agent-name)
Pre-condition: *oc* is identified
Post-condition: History is updated with *agent_ID*

Events: message(*kw*: Keyword) arrives from Information Retriever agent

Figure AppendixH.7 – Agent Plan Template Diagram by Developer 1 (for *History Manager* agent class)

Agent Interaction Model

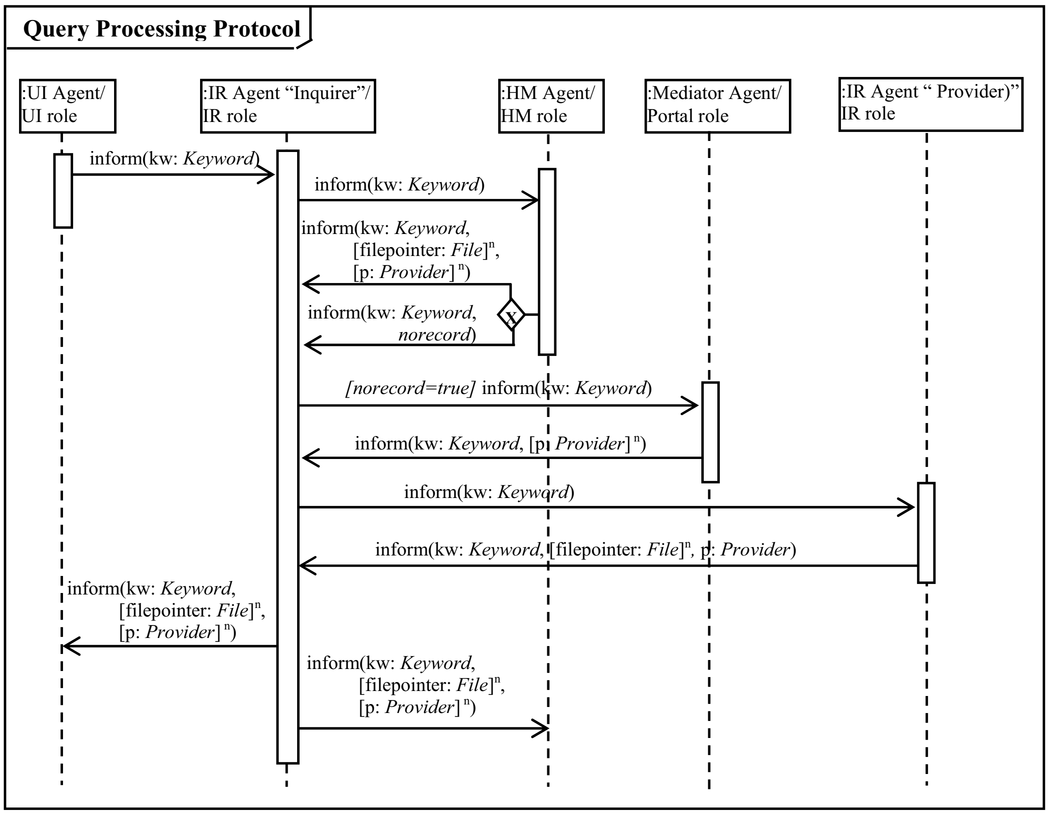

Query Processing Protocol
:UI Agent/ UI role
:IR Agent "Inquirer"/ IR role
:HM Agent/ HM role
:Mediator Agent/ Portal role
:IR Agent " Provider)" IR role
inform(kw: Keyword)
inform(kw: Keyword)
inform(kw: Keyword, [filepointer: File]n, [p: Provider] n)
X
inform(kw: Keyword, norecord)
[norecord=true] inform(kw: Keyword)
inform(kw: Keyword, [p: Provider] n)
inform(kw: Keyword)
inform(kw: Keyword, [filepointer: File]n, p: Provider)
inform(kw: Keyword, [filepointer: File]n, [p: Provider] n)
inform(kw: Keyword, [filepointer: File]n, [p: Provider] n)

Figure AppendixH.8 – Interaction Protocol Diagram by Developer 1

Major models produced by Developer 2

DEVELOPER: DR. CESAR GONZALEZ-PEREZ

System Task Model

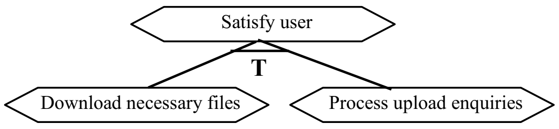


Figure AppendixH.9 – System Task Diagram 1 by Developer 2

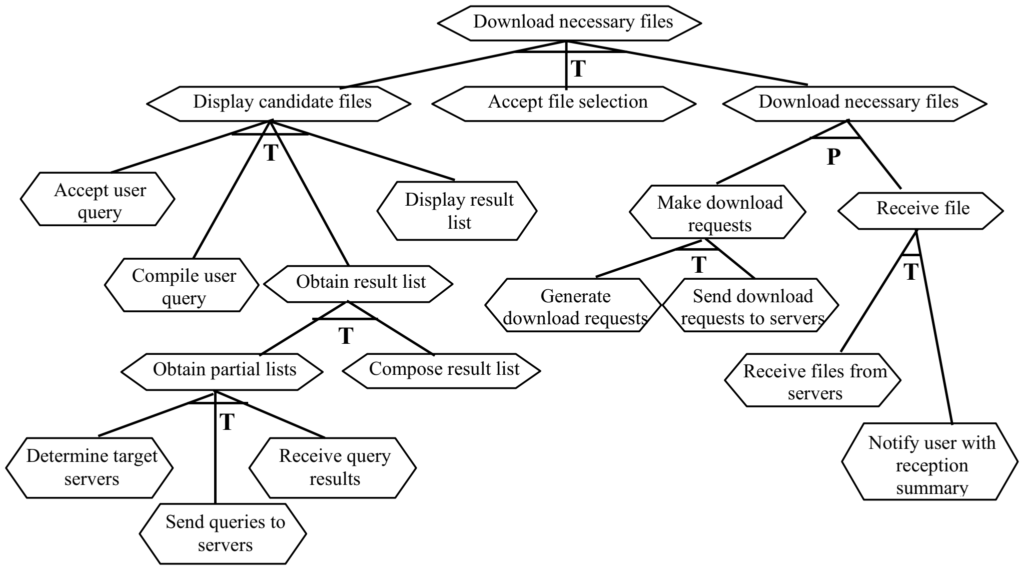
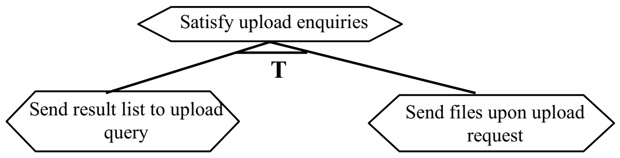


Figure AppendixH.10 – System Task Diagram 2 by Developer 2



Satisfy upload enquiries
T
Send result list to upload query
Send files upon upload request

Figure AppendixH.11 – System Task Diagram 3 by Developer 2

Ontology Model

Note that Ontology Diagram for *Movie Ontology* is reused from that developed by Developer 1 (Figure AppendixH.2).

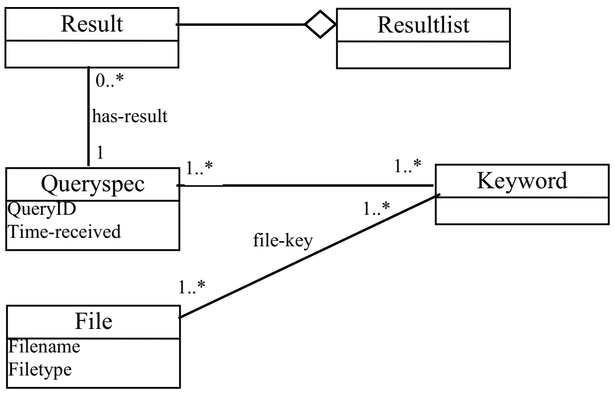


Figure AppendixH.12 – Ontology Diagram for *File Sharing Ontology* by Developer 2

Role Model

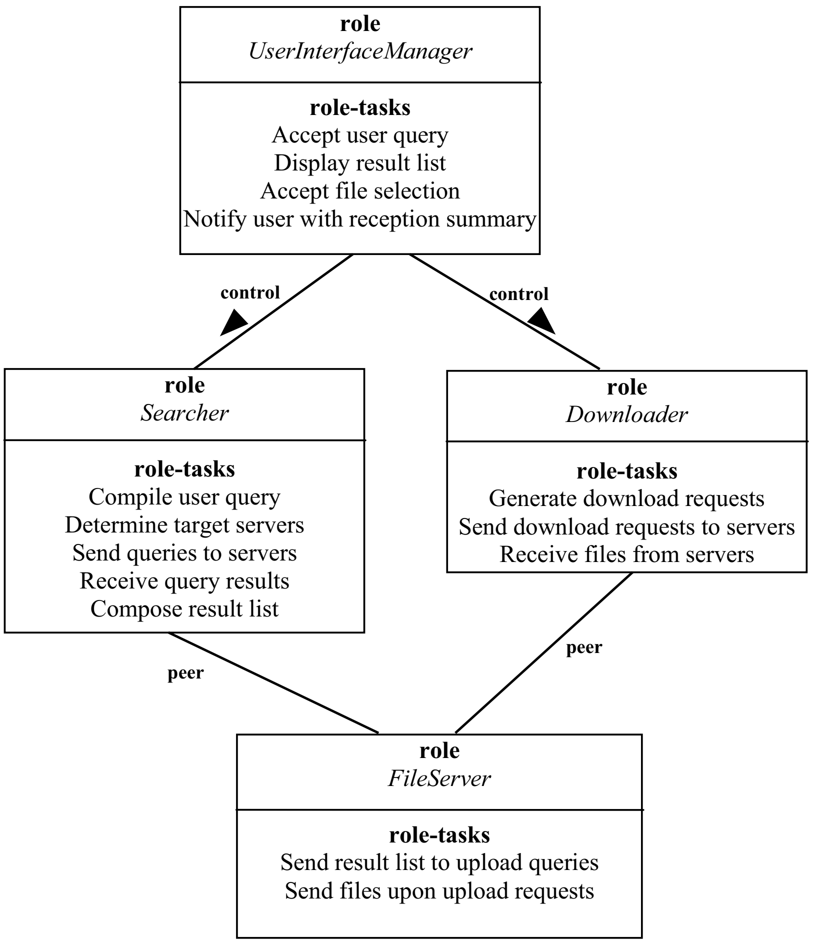
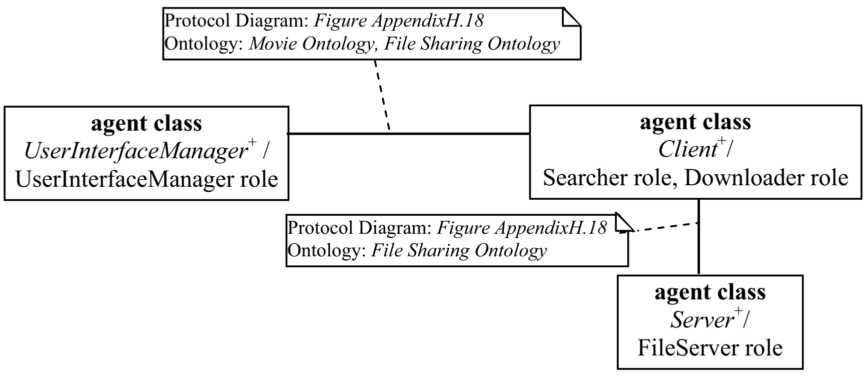


Figure AppendixH.13 – Role Diagram by Developer 2

Agent Class Model



Protocol Diagram: Figure AppendixH.18
Ontology: Movie Ontology, File Sharing Ontology
agent class
UserInterfaceManager+ /
UserInterfaceManager role
agent class
Client+/
Searcher role, Downloader role
Protocol Diagram: Figure AppendixH.18
Ontology: File Sharing Ontology
agent class
Server+/
FileServer role

Figure AppendixH.14 – Agent Relationship Diagram by Developer 2

agent class
Server / FileServer role

belief conceptualisation
Movie Ontology
File Sharing Ontology

agent-goals
Upload query is responded as soon as it is received
Upload request is satisfied as soon as it is received

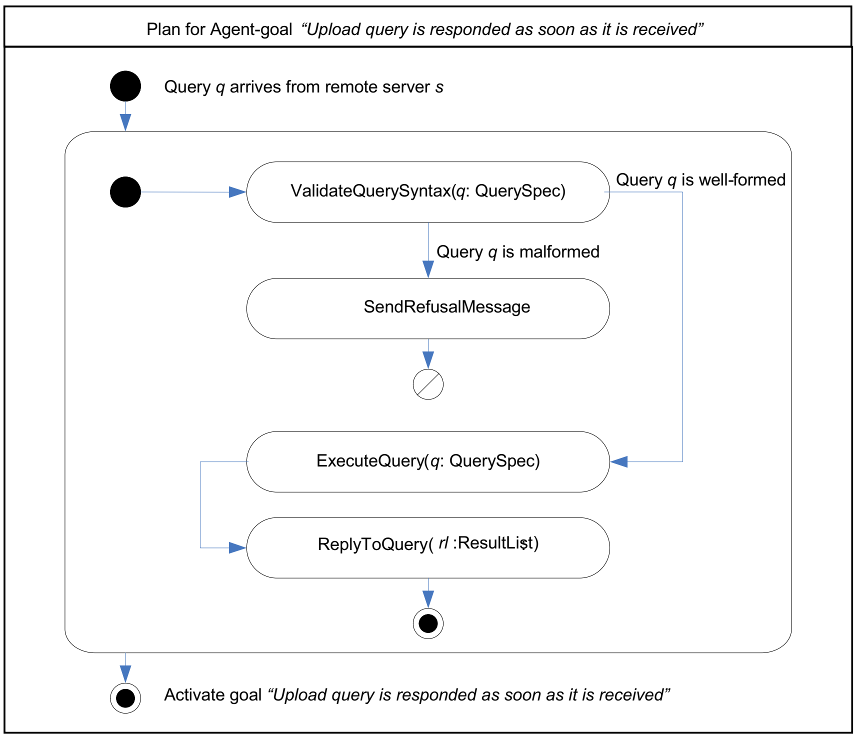
events
Upload query arrives
Upload request is received

Figure AppendixH.15 – Agent Class Diagram by Developer 2 (for *Server* agent class)

Agent Behaviour Model

Initial State: any
Agent Goal: Upload query is responded as soon as it is received
Commitment Strategy: single-minded
Action 1: ValidateQuerySyntax(*q*: QuerySpec)
Pre-condition: true
Post-condition: Query *q* is valid OR refusal message has been replied
Action 2: ExecuteQuery(*q*: QuerySpec)
Pre-condition: *q* is valid
Post-condition: Query *q* has been executed and result list is known
Action 3: ReplyToQuery(*rl*: ResultList)
Pre-condition: true
Post-condition: Result list *rl* has been sent back to remote server *s*
Event 1: Query *q* arrives from remote server *s*

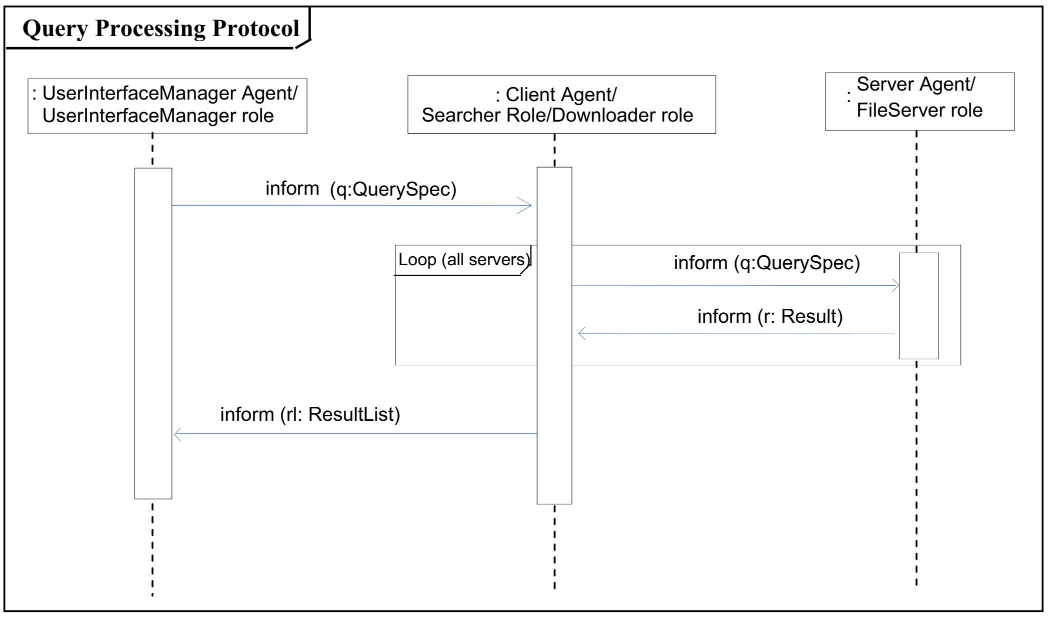
Figure AppendixH.16 – Agent Plan Template by Developer 2 (for *Server* agent class)



Plan for Agent-goal *"Upload query is responded as soon as it is received"*
Query *q* arrives from remote server *s*
ValidateQuerySyntax(*q*: QuerySpec)
Query *q* is well-formed
Query *q* is malformed
SendRefusalMessage
ExecuteQuery(*q*: QuerySpec)
ReplyToQuery(*rl* :ResultList)
Activate goal *"Upload query is responded as soon as it is received"*

Figure AppendixH.17 – Agent Plan Diagram by Developer 2 (for *Server* agent class)

Agent Interaction Model



Query Processing Protocol
: UserInterfaceManager Agent/ UserInterfaceManager role
: Client Agent/ Searcher Role/Downloader role
: Server Agent/ FileServer role
inform (q:QuerySpec)
Loop (all servers)
inform (q:QuerySpec)
inform (r: Result)
inform (rl: ResultList)

Figure AppendixH.18 – Interaction Protocol Diagram by Developer 2