Agent-based Software Engineering

Chang-Hyun Jo
Department of Computer Science
California State University Fullerton
jo@ecs.fullerton.edu
http://jo.ecs.fullerton.edu

Outline

- Agent-based Programming
- Agent-based Software Engineering
- BDI Agent Model
- Agent-based Modeling Technique (AMT)
- Examples
- Conclusions

Agent-Based Programming

- Agent-based programming
  - A new programming paradigm
  - No programming language to well support agent programming naturally
  - No proper modeling methodology for agent software development

Basic Idea in Agent Computing

- Object
  - A thing that combines the related data and the associated operations on its data
- Agent
  - A *concurrent*, *autonomous*, *intelligent* and *self-contained* object
    - Self-containing
      - An agent *describes its behavior by itself through the goal* to achieve and *behavior* to implement the goal based on the current *environment*.

Basic Idea in Agent Computing

- Autonomous
- Perceptive [Petrie 96]
  - *Senses the environment*, and *acts* on it based on its own *agenda*
- Reactive, *sensing and acting* [Franklin and Graesser 96]
- Pro-active**,** *goal-directed behavior* [Franklin and Graesser 96]
- Leaning properties [Franklin and Graesser 96]
  - ***Socially able***
  - ***Adaptive***
- Cooperative, perceptive, and pro-active [DeLoach 99]
- Mobile

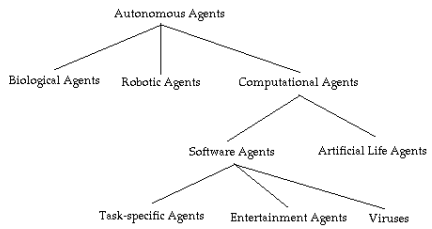
Properties of Agents

[Franklin and Graesser 1996]

Property	Other Names	Meaning
Reactive	Sensing & acting	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions
Goal-oriented	Pro-active Purposeful	Does not simply act in response to the environment
Temporally continuous		Is a continuously running process
Communicative	Socially able	Communicate with other agents, perhaps including people
Learning	Adaptive	Changes its behavior based on its previous experience
Mobile		Able to transport itself from one machine to another
Flexible		Actions are not scripted.
Character		Believable personality and emotional state

Classification of Agents

- Franklin and Graesser [1996]



BDI-Agent Model

- Some agent-based systems imitate *reasoning behavior* according to the theoretical model of human reasoning.
  - Belief Desire Intention (BDI) agent model [Rao and Georgeff 1991]
  - Intention, Plans, and Practical Reason [Bratman 1987]

BDI-Agent Model

- Agents have explicit ***goals*** to achieve or events to handle (***desires***).
- A set of ***plans*** (***intentions***) is used to describe how agents achieve their *goals*.
- Each *plan* describes how to achieve a *goal* under varying ***environments*** (***belief***).
  - A set of data called *belief* describes the *state* of the *environment*. (local & global)

BDI Agents

- Belief-Desire-Intention (BDI) agents
  - Theoretical foundations of BDI agents
    - Bratman et al. 1987, Rao and Georgeff 1991, Shoham 1993, Cohen and Levesque 1990, Jennings 1992, Kinny et al. 1994
  - Implementations of BDI agents
    - Burmeister and Sundermeyer 1992, Georgeff and Lansky 1986, Shoham 1993
  - Building of large-scale applications based on BDI agents
    - Ingrand et al. 1992, Rao et al. 1992, OASIS (agent-oriented air-traffic management system tested at Sydney airport)

Applications

- Procedural Reasoning Systems (PRS) [Georgeff and Lansky 1986] [Ingrand et al. 1992]
  - Agent-oriented systems based on the BDI architecture
- dMARS (distributed MultiAgent Reasoning System)
  - Its successor
- Space shuttle diagnosis [Ingrand et al. 1992]
- Telecommunications network management [Ingrand et al. 1992]
- Air-combat modelling [Rao et al. 1992]
- Business process management
- Agent-oriented approach is useful for building complex distributed systems involving resource-bounded decision-making.

BDI Agent-based Modeling Research

- Explore how existing OO modeling techniques can be extended to apply to BDI agent systems
  - Kinny and Georgeff (1995)
  - Kinny et al. (1995)

BDI-Agent Modeling Technique

- BDI Agent-based Modeling Technique (AMT)
- BDI Agent-based Modeling Language (AML)
- BDI Agent-based Programming Language (APL)

AMT

- Agent-based Modeling Technique (AMT)
  - ***A seamless approach to develop agent-based software*** through the phases of analysis, design, and implementation.
    - AML is used to model the agent-based system at the stage of analysis and design, and
    - APL is used to implement the agent-based application based on the models constructed by using AML.
  - AMT defines models, steps, and activities to develop agent-based systems by using AML and APL.

Process, Model and Techniques

- Agent-based Modeling Techniques (AMT)
  - A technique to provide a framework from which software engineers can design agent-based systems systematically
  - With the Agent-based Process Model (APM), AMT defines who are involved in, which tasks each participant should do, and what kinds of artifacts would be produced.

Decomposition

- Decomposition level in agent-based computing is:
  - An agent to achieve an *independent goal*
  - Agents to achieve the *same goal* independently
  - Agents to achieve the *common goal* cooperatively
- Dynamic interactions among agents are unpredictable at design time and compile time [Jennings and Wooldridge 00].
  - A *self-reflective agent system*

Agent Decomposition

D_g: global desire
D_1, B_1, I_1
D_2, B_2, I_2
:

Figure 2. Desire Lists

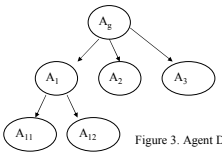
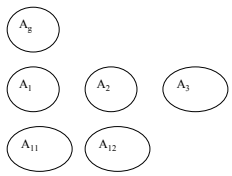


Figure 3. Agent Decomposition

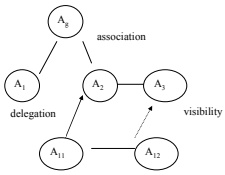
Identify Agents and Concepts

- Agent diagrams are used to describe all participating agents in the system.



Identify Relationships among Agents

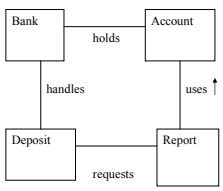
- Some drafts of relationship diagrams are used to show the relationships among the involving agents.



Example
Identify Agents

⌘*Customer, Bank, Deposit, Accounts, Reporting, Marketing, Credits, Service, …*

Example
Identify Relationships among Agents



Analysis

- Requirements analysis
  - Gather the customer's requirements
  - Identify the goal, users and system behavior through the analysis modeling
- Construct ***Belief-Desire-Intention (BDI) cards***.
  - A ***BDI card*** summarizes the requirements of an agent system and its BDI.

A Template for BDI Card

Desire	A goal or a set of goals to achieve with this agent			
General Belief	General belief on this agent – A set of states to describe current environment surrounding this agent			
Stimuli (List a set of events)	Intention (A set of plan to solve the problem initiated from this stimulus)	Belief (This is belief applicable to this stimulus.)	Response (List responses to react this stimulus.)	Collaborator (List collaborators to achieve the goal by reacting this stimulus.)
Event-i	…	…	…	…
Event-j	…	…	…	…

Figure 1. A Template of BDI Cards

Example
BDI Card for Deposit in Banking Application

Agent	Deposit			
Desire	Deposit_Checking Deposit_Saving			
General Belief	Banking DB, Personal Info DB, Credit Info DB, Checking DB, Saving DB			
Stimuli	Intention	Belief	Response	Collaborator
Plan-1	Create Checking	Checking DB	Checking Creation	Account
Plan-2	Deposit Checking	Checking DB	Checking Deposit	Report
Plan-3	Create Saving	Saving DB	Saving Ac Creation	Account
Plan-4	Deposit Saving	Saving DB	Saving Deposit	Report
…	…	…	…	…

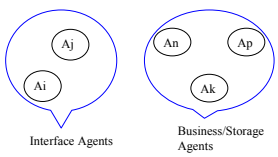
Build Scenarios

- A scenario can describe a process of an agent, all the participating agents, or partial behavior of an agent.
  - For each stimuli of an agent, one or more scenarios can be defined.
  - A scenario describes what happened in the agent system by describing a sequence of steps to be performed by each participating agent.
  - The scenario describes a client agent, a server agent, and the corresponding current goal to achieve.

Example Scenarios

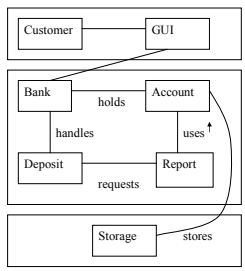
Deposit a checking account or saving account		
Client	Server	Plan
Customer	Bank/Deposit	Open_Checking
Customer	Bank/Deposit	Deposit_Checking

Agent Boundary



- Agent boundary draws the limits of a set of agents.
- Agent boundary can be used as a basis for implementation packages in the design and implementation phases.

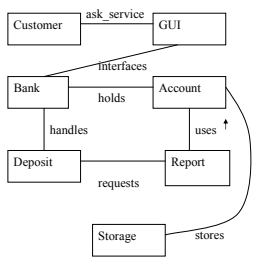
Example Agent Boundary



Design Relationship Diagrams

- Relationship diagrams show the relationship among agents.
  - It shows several kinds of relationships such as inheritance, dependency, visibility and logically and physically structured organization.
  - Relationship diagrams built at the analysis phase are refined precisely on consideration of implementation.
  - Components for agents in the relationship diagrams include agent's name, functions, belief, desire, and intention.
  - Components for agents in the relationship diagrams can be separately and precisely described in detail in the agent diagrams.

Example Relationship Diagrams

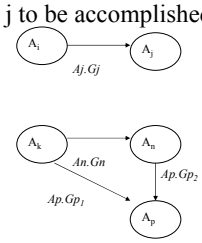


Interaction diagrams

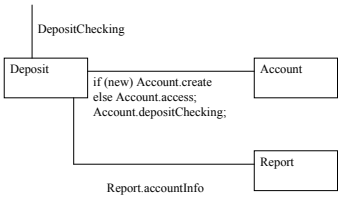
- Interaction diagrams show the interactions among several agents.
- Interaction diagrams are in the different kinds of levels such as local, partial and global.
  - Local interaction diagrams show the interactions among agents in a local system.
  - Global interaction diagrams show all the interactions among all agents in the several different nodes logically or physically distributed on the network.
  - Partial interaction diagrams show partial interactions among distributed agents on some specific nodes. Interaction diagrams are related to the scenario.
- Interactions shown on a scenario are described on one or several interaction diagrams.

Interaction Diagrams

- Ai represents an agent i, and Gj represents the current goal j to be accomplished.

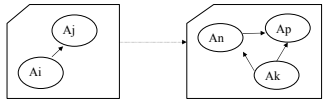


Example Interaction Diagrams

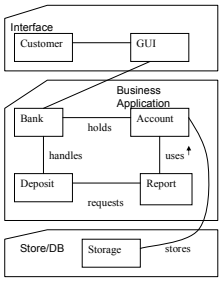


Component diagrams

- Component diagrams show the closely related components for a group of agents to be implemented and grouped together.
- Component diagrams indicates the implementation group as packages in most programming languages.



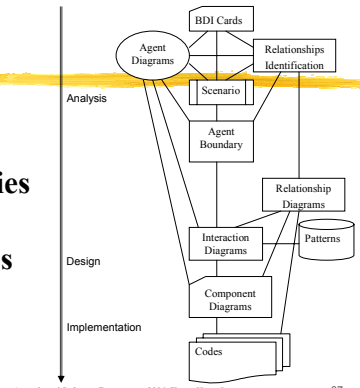
Example Component Diagrams



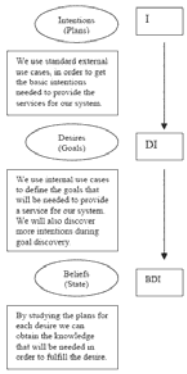
Dependencies among Deliverables

- There are several artifacts delivered through the modeling process.
  - There may be dependencies among those artifacts.
  - It means that those artifacts can be delivered from the information involved in the other models.

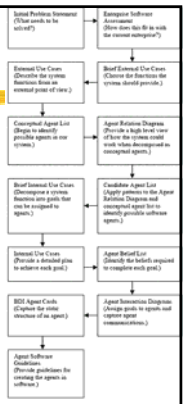
Dependencies among Deliverables



BDI Discovery



BDI Agent Process



Implementation of Models

- One to one mapping from the models to the codes may not be possible.
  - Ideally, there is no component in the code, which is not identified from the models.
  - However, if we find new components in the following phases, the models in the previous phases should be synchronized in each iterative phase.

Agent Definition for Deposit Agent

```
agent Deposit extends BDI {
  belief Deposit_B;
  desire Deposit_D;
  intention Deposit_I;
  main() { … }
   // main body to execute on this agent
}
```


Belief Definition for Deposit Agent

```
belief Deposit_B extends BDI {
  CheckingDB;
  SavingDB;
  :    // some other belief declarations
  :    // also goal completion flags defined
  Accessor4CheckingDB();  // accessor for belief
  Modifier4CheckingDB();  // modifier for belief
  :
}
```

Desire Definition for Deposit Agent

```
desire Deposit_D extends BDI {
  DepositChecking() { … }
  DepositSaving() { … }
  :
}
```


Intention Definition for Deposit Agent

```
intention Deposit_I extends BDI {
  createAccount();
  accessAccount();
  depositChecking();
  depositSaving();
  :
}
```


Refined Plan in the Intention Definition for Deposit

```
intention Deposit_I extends BDI {
  createAccount();
  accessAccount();
  depositChecking() {   // refined plan
    :
    if (Account is new) checkingAcc = Account.create;
    else checkingAcc = Account.access;
    checkingAcc.depositChecking(amount);
    :
    reportAg.accountInfo(checkingAcc);
    :
  }
```

```
:
  depositSaving();
  :
}
```


Package Definitions for the Banking Agent System

```
package UserInterface;
   agent Customer;
   :   // BDIs for this agent
   agent GUI;
   :
package BusinessApplication;
   agent Bank;
   agent Account;
   agent Deposit;  // the exemplified agent
   agent Report;
   :
```

```
:
package StoreDB;
   agent StorageDB;
   :
```


CONCLUSION

- A novel idea on agent-based modeling techniques with the useful associated schemes such as modeling language and programming languages, which are mostly useful in agent-based software engineering
- A seamless approach from the agent-based modeling to the programming codes that are supposed to run
- The agent-based programming language, APL, can be used to implement the models constructed through the modeling phases.

CONCLUSION

- Potential research
  - Agent-based software development processes
  - Agent-based programming languages
  - CASE Tools
  - Validation and Verification
  - Quantitative approach

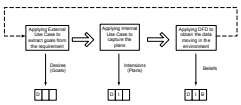
ACKNOWLEDGMENTS

- The speaker thanks his *students* for helping him to implement this idea.
  - *Allen J. Arnold (APL Interpreter)*
  - *Xin Feng (Testing the BDI-Agent concept)*
  - *Jeffery M. Einhorn (Extending AMT using UC)*
  - *Wei Zhao (APL Compiler)*
  - *Guobin Chen (Extending AMT with a diff. app.)*

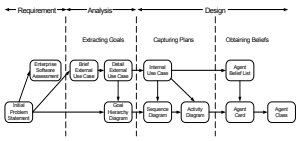
ACKNOWLEDGMENTS

- (Cont'd)
  - Willy Tanimihardja (BDI Agent CASE Tool)
  - Dongshi Zhang (A Compiler Design for the Agent-based Programming Language)
  - Tianzhi Zheng (Modeling Language for the BDI Agent Software Development)
  - Sujatha H. Thippeswamy (Collaborative CASE Tool for Agent Based Software Development Process)
  - Sujito Lie (A Secured Mobile Agent for E-Banking)

BDI Agent Software Development Process

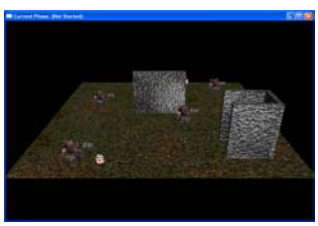


- MS Student: Guobin Chen, Fall 2003
- This research proposes an agent-based software development process based on Belief-Desire-Intention (BDI) agent model as a new software development process.



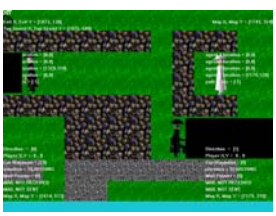
Pure Instincts

- MS Student: Kelvin Yong, Spring 2004
- The Pure Instincts algorithm provides an efficient method for the exploration and navigation of unknown environments, which can ultimately be used in time critical environments such as robots, military applications and games.



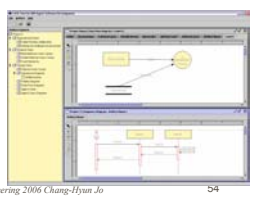
A Flexible Agent Coordination Language

- MS Student: Andrew Hess, Spring 2004
- The purpose of this project is to present a new way for agents to coordinate their conversations.
- In order to test the strength of our new coordination language, a video game was created where characters in the game are agents and use the object-oriented coordination features described.

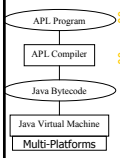


BDI Agent CASE Tool

- MS Student: Willy Tanimihardja, Spring 2004
- The BDI Agent CASE Tool is one of the necessary instruments in supporting modeling and design process of the Belief-Desire-Intention (BDI) Agent-based Software Development.



A Compiler Design for the Agent-based Programming Language



- MS Student: Dongshi Zhang, Spring 2004
- This project is to build a prototype compiler for the BDI Agent-based Programming Language (BDI-APL).

```
01 agent CSLFS {
02   belief <int> Hot_Numbers[] = new <int>[6];
03
04   intention <int>[] RequestHotNumber(){
05      belief <int> i;
06      belief <int> Numbers[];
07
08      agent <HotNumberManager> c = new <HotNumberManager>();
09      Numbers = c.RequestHotNumber();
10          . . . .
11
12      return Hot_Numbers;
13   }
14 }
```

```
01 .source csLFS.apl
02 .class  public synchronized csLFS
03 .super  java/lang/Object
04
05 .field private Hot_Numbers [I
06
07 .method public <init>()V
08     .limit stack 2
09     .limit locals 1
10
11     aload_0
12     invokenonvirtual java/lang/Object/<init>()V
13
14     aload_0
15     bipush 6
16     newarray int
17     putfield csLFS/Hot_Numbers [I
18     return
19 .end method
20
21 .method public RequestHotNumber()[I
22     .limit stack 4
23     .limit locals 4
24
25     new HotNumberManager
26     dup
27     invokenonvirtual HotNumberManager/<init>()V
28     astore_3
29
30     aload_3
31     invokevirtual HotNumberManager/RequestHotNumber()[I
32     astore_2
33     .
34     .
35     .
46     aload_0
47     getfield csLFS/Hot_Numbers [I
48     areturn
49 .end method
```


More Agents

- Jo (2001 – 2005)
  - BDI agent-based software modeling techniques
  - BDI agent-based programming languages
  - Jo, Chang-Hyun, Research related to Agent Computing, http://jo.ecs.fullerton.edu/research, 2006.
  - Lecture notes on agents at http://jo.ecs.fullerton.edu/cpsc589, 2006.

References

- Jo, Chang-Hyun. A Seamless Approach to the Agent Development, ACM 2001 15th Annual Symposium on Applied Computing (ACM SAC'01), Las Vegas, 641-647, (March 2001).
- Jo, Chang-Hyun and Allen J. Arnold, "Agent-based Programming Language: APL", ACM 2002 16th Annual Symposium on Applied Computing (ACM SAC'02), Madrid, Spain, 27-31, (March 2002).
- Jo, Chang-Hyun. "A New Way of Discovery of Belief, Desire and Intention in the BDI Agent-Based Software Modeling", The International Conference and Exhibition on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM 2003), Manila, Pavillon Hotel, March 27-30, 2003. (ISBN# 971-92723-0-9)
- Jo, Chang-Hyun and Einhorn, Jeffery M., "A Process for BDI Agent-Based Software Construction", The 2003 International Multi-Conference in Computer Science and Computer Engineering – The International Conference on Software Engineering, (IMCCSCE – SERP'03), 204-209, Las Vegas, Nevada, June 23-26, 2003.
- Jo, Chang-Hyun, Zhao, Wei and Cong, Bin., "A Design and Implementation of the Belief-Desire-Intention Agent-based Programming Language", ***Information: An International Interdisciplinary Journal***, ISSN 1343-4500 (print), 1344-8994 (electronic), 7(1), International Information Institute, (January 2004), 137-153.

References

- Jo, Chang-Hyun. "A New Way of Discovery of Belief, Desire and Intention in the BDI Agent-Based Software Modeling", ***the International Journal of Advanced Computational Intelligence & Intelligent Informatics (JACIII)***, ISSN 1343-0130, 8(1), 2-6, Jan. 2004.
- Jo, Chang-Hyun, Guobin Chen and James Choi, "A New Approach to the BDI Agent-Based Modeling", ACM SAC 2004, 1541-1545, Nicosia, Cyprus, March 14-17, 2004.
- Jo, Chang-Hyun, Won-Young Kim, Jeong-Min Shim, and Wan Choi. Agent-based Framework for Software On-Demand. Proceedings of the IEEE 7th International Conference on Advanced Communication Technology, Phoenix Park, Korea, (Vol.2), 730-735, Feb. 21-23., 2005. (IEEE Catalog # 05EX1046) (ISBN: 89-5519-123-5)
- Jo, Chang-Hyun and Einhorn, Jeffery M., "A BDI Agent-Based Software Process", ***Journal of Object Technology (JOT)***, Vol.4, No.9, 101-121, November – December 2005. (available at http://www.jot.fm/issues/issue_2005_11/article3)
- Jo, Chang-Hyun, Research related to Agent Computing, http://jo.ecs.fullerton.edu/research, 2006.

References

- Kinny, D., Georgeff, M., and Rao, A. A Methodology and Modelling Techniques for Systems of BDI agents. In Agents Breaking Away: Proc. Of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI Vol. 1038): 56-71, Springer-Verlag, 1996, also at http://citeseer.nj.nec.com/cache/papers/cs/40/ftp:zSzzSzwww.aaii.com.auzSzpubzSzaaii-technoteszSztechnote58.pdf/kinny96methodology.pdf.
- Maamar, Z. and Moulin B. An Overview of Software Agent-Oriented Frameworks, ACM Computing Survey, 32(1es), 1-6, March 2000, also available at http://portal.acm.org/citation.cfm?doid=351936.351955, 2000.
- Maes, Pattie. Autonomous Agents Group, MIT Media Laboratory, http://agents.media.mit.edu/index.html, 2003.
- Maudlin Michael. Julia, Information available from Forner, Lenny's Julia, http://foner.www.media.mit.edu/people/foner/Julia/, 1999.
- OAA Home, The Open Agent Architecture, SRI International, http://www.ai.sri.com/~oaa/, 2001.
- Odell, J., Parunak, H. V. D., and Bauer, B. Extending UML for Agents, Proc. Of the AOIS Workshop at the 17th National Conference on AI (AAAI 2000), 3-17, available at http://www.jamesodell.com/ExtendingUML.pdf, 2000.

References

- Petrie, Charles J. Agent-based Engineering, the Web, and Intelligence, http://www-cdr.stanford.edu/NextLink/Expert.html, 2003, also available from IEEE Expert, Dec. 1996.
- Petrie, Charles, Agent-Based Software Engineering, *Agent-Oriented Software Engineering, Lecture Notes in AI, Springer-Verlag,* 58-76, 2001.
- Rana, O. F., Winikoff, M. Padgham, L., and Harland, J. Applying Conflict Management Strategies in BDI Agents for Resource Management in Computational Grids, Proc. Of the 25th Australian Conference on CS, Vol.4, 205-214, Jan 2002.
- Rao, Anand S. and Georgeff, Michael P., BDI Agents: From Theory to Practice, *Australian Artificial Intelligence Institute*, April, 1995.
- Sycara, K. P. Multiagent Systems, http://www-2.cs.cmu.edu/~softagents/papers/multiagentsystems.PDF, also appears in AI Magazine, 19(2), 79-92, 1998.
- Thangarajah, J., Padgham, L., and Harland, J. Representing and Reasoning for Goals in BDI Agents, Proc. Of the 25th Australian Conference on CS, Vol4, 259-265, Melbourne, Victoria, Australia, Jan. 2002.
- UMBC Agent Web, News and Information on software agent technology, http://agents.umbc.edu/, 2003.

References

- Weiss G., editor, Multi-Agent Systems, *The MIT Press: Cambridge, MA*, 1999.
- Wooldridge, Michael and Nicholas R. Jennings, "Agent Theories, Architectures, and Languages: a Survey," in Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22. 1995.
- Wooldridge, M. and Jennings, N. R., Intelligent Agents: Theory and Practice, *Knowledge Engineering Review, Cambridge Univ. Press*, 10(2), 115-152, June 1995.
- Wooldridge, M. and Jennings, N. R., Software Engineering With Agents: Pitfalls and Pratfalls, *IEEE Internet Computing*, 20-27, May-June 1999.
- Wooldridge, M., Jennings, N. R., and Kinny, D., A Methodology for Agent-Oriented Analysis and Design, *Autonomous Agents 1999, Seattle, WA,* 69-76, 1999.
- Wooldridge, M., Reasoning about Rational Agents, *The MIT Press: Cambridge, MA*, 2000.