Multi-Agent Systems for Distributed Collaboration

Article · August 2006

CITATIONS
6

READS
739

2 authors:

Camelia Chira
Babeş-Bolyai University
145 PUBLICATIONS **871** CITATIONS
SEE PROFILE

Dan Dumitrescu
Babeş-Bolyai University
246 PUBLICATIONS **1,860** CITATIONS
SEE PROFILE

Computational Intelligence Reports (CIR)

Department of Computer Science
"Babes-Bolyai" University of Cluj-Napoca,
Kogalniceanu Street, no. 1
Romania

http://www.cir.cs.ubbcluj.ro

Multi-Agent Systems for Distributed Collaboration

Camelia Chira, D. Dumitrescu

CIR Report No. 032006
May, 2006
ISSN 1841 – 995X

Multi-Agent Systems for Distributed Collaboration

CAMELIA CHIRA, D. DUMITRESCU
Department of Computer Science
"Babes-Bolyai" University
1B M. Kogalniceanu Street, Cluj-Napoca, 400084
ROMANIA
cchira@cs.ubbcluj.ro, ddumitr@cs.ubbcluj.ro

Abstract: Multi-agent systems and ontologies can potentially enable resource interoperation and knowledge sharing in virtual collaboration environments where information and people are inherently distributed. A review of agents, multi-agent systems and ontologies in terms of definition, properties, architectures, methodologies, languages and environments is presented. An agent architecture is proposed to address virtual collaboration in a distributed environment. Based on multi-agent systems and ontologies, the proposed architecture has the potential of optimizing the flow of information in communication networks.

Key-Words: agents, multi-agent systems, ontologies, cooperation, distributed environments

1 Introduction

Agents and Multi-Agent Systems (MAS) represent an important and fast growing area of Artificial Intelligence (AI) with the potential to play a crucial role in a large number of application domains including ambient intelligence, computing, electronic business, semantic web, bioinformatics and computational biology [3, 18, 19, 21]. Research shows that intelligent agents are the natural metaphor to address distribution of data or control, legacy systems and open systems [16, 31].

MAS researchers study the group behaviour of autonomous agents, which are working together towards a common goal [16, 19, 31]. MAS are ideal for solving complex real world problems with multiple problem solving methods, multiple perspectives and/or multiple problem solving entities [17]. Addressing complex systems development in distributed environments [22], MAS approach to building computational systems promotes conceptual clarity and simplicity of design [16, 30].

The potential benefits of software agents are exemplified by presenting a multi-agent architecture for distributed collaboration. Proposed architecture employs MAS and ontologies to support distributed users who have to cooperate in a computer-based environment in order to solve problems.

2 Agents

Over the last years, autonomous agents have been the focus of researchers and developers from disciplines such as AI, object-oriented programming, concurrent object-based systems and human-computer interface design [1, 4, 17].

2.1 Agent definition

A literature review in the area of agents and agent-based systems offers many and diverse definitions for the notion of agency [4].

Over a decade ago, Shoham [26] defined an agent as "an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments".

Russell and Norvig [25] believe that an agent is "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors".

Nwana [21] indicates, "when we really have to, we define an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user".

Franklin and Graesser [11] define the term "autonomous agent" as "a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future".

Jennings, Wooldridge and Sycara [17, 30] define an agent as "a computer system that is situated in some environment, and that is capable of flexible autonomous action in this environment in order to meet its design objectives".

More recently, the Foundation for Intelligent Physical Agents (FIPA) [33] indicates that "an agent is an encapsulated software entity with its own state, behaviour, thread of control, and an ability to

interact and communicate with other entities – including people, other agents, and legacy systems".

Although there is no universally accepted agent definition, researchers and scientists generally agree that an agent: (i) acts on behalf of its user; (ii) is situated in an environment and is able to perceive that environment; (iii) has a set of objectives and takes actions so as to accomplish these objectives; and (iv) is autonomous i.e. an agent can take decisions without the intervention of humans or other systems [4, 11, 17, 21, 30].

2.2 Agent properties

The main properties that should characterize an agent can be summarised as follows [4, 11, 21, 30]:

- *Autonomy:* The ability to operate on its own without the intervention of humans or other systems.
- *Reactivity:* The ability to perceive its environment and to respond to changes that occur in it.
- *Pro-activeness:* The ability to take the initiative in order to pursue its individual goals (goal-directed behaviour).
- *Cooperation (or social ability):* The capability of interacting with other agents and possibly humans via an agent-communication language. Involves the ability of an agent to dynamically negotiate and coordinate.
- *Learning:* The ability to learn while acting and reacting in its environment. Learning can increase performance of an agent over time.
- *Mobility:* The ability to move around a network in a self-directed way.

Furthermore, some researchers identify more properties associated with the notion of agency as follows [3, 4, 11, 19, 21, 32]:

- *Temporal continuity:* The actions of an agent are performed through a continuous running process (over long periods of time).
- *Personality:* A believable character and emotional state.
- *Veracity:* An agent should not knowingly communicate false information.
- *Benevolence:* Agents should not have conflicting goals and every agent should always try to accomplish its objective.
- *Rationality:* An agent should act so as to achieve its goals and not to prevent its goals from being achieved.

Autonomy, reactivity, pro-activeness and cooperation are the basic properties used by Wooldridge and Ciancarini [31] to define an agent. Jennings, Sycara and Wooldridge [17] include three key concepts in their definition of an agent: situatedness, autonomy and flexibility. The latter concept is defined using three properties as follows: reactivity (an agent should be responsive), pro-activeness and social ability. Nwana [21] considers that a truly intelligent agent (the ideal agent) should equally be characterised by three primary attributes i.e. autonomy, cooperation and learning while any system that does not exhibit these three properties (more or less emphasized) should not be considered an agent at all.

2.3 Agent typologies

The most straightforward classification of an agent would be along one of their properties such as [21]:

- *Mobility:* static or mobile agents.
- *Reactivity:* deliberative or reactive agents.

A well known agent taxonomy is Nwana's primary attribute dimension typology [21] which uses autonomy, cooperation and learning to classify agents in four categories i.e. *collaborative agents, collaborative learning agents, interface agents* and *smart agents*.

Franklin and Graesser [11] classify autonomous agents in three classes i.e. biological agents, robotic agents and computational agents (*the kingdom level*). Furthermore, computational agents can be divided in software agents and artificial life agents (*the phylum level*) and software agents can be classified in task-specific agents, entertainment agents and viruses (*the class level*). A further taxonomy can be performed using schemes such as classification via the agent's control structures (e.g. regulation, planning and adaptive), via environments (e.g. database, file system, network, internet), via languages (in which the agent is written) and via applications.

From an architectural point of view, Wooldridge [30] identifies logic-based agents, reactive agents, BDI (Belief-Desire-Intention) agents and layered architectures.

2.4 Agent architectures

Agent architectures address the issues of designing and creating computer-based systems that satisfy agent properties. "An agent architecture is essentially a map of the internals of an agent – its data structures, the operations that may be performed on these data structures, and the control flow between these data structures" [30]. Wooldridge and Jennings [30, 32] identify three classes of agent architectures i.e. deliberative, reactive and hybrid.

Deliberative architectures adopt the traditional AI approach to designing intelligent systems by viewing them as a type of knowledge-based system. The agent-based system that has to be designed receives a symbolic representation of its environment and its desired behaviour, which can be syntactically manipulated. The disadvantages associated with deliberative architectures refer to the transduction problem (it is time consuming to translate information into its symbolic representation) and the representation/reasoning problem [4, 30]. Much of the research and development work on deliberative agents has focused on the agent-oriented programming paradigm. The state of an agent is characterised in terms of its mental attitudes of *belief*, *desire* and *intention* [24]. Agent-oriented programming uses these intentional notions to directly program agents. Shoham developed an experimental language called AGENT0 [26] in order to demonstrate the agent-oriented programming paradigm.

Inspired by the philosophical tradition of understanding practical reasoning, BDI architectures have become very popular over the last years [24, 30]. The BDI architecture represents an agent in terms of its beliefs, desires (or goals) and intentions. The basic components of a BDI agent are data structures (that represent beliefs, desires and intentions) and functions for representing and reasoning about them.

Reactive architectures are an alternative to the symbolic AI paradigm. They involve developing and combining individual behaviours of reactive agents situated in some environment [30]. Reactive agents have a very simple representation of the world but provide tight coupling of perception and action. The behaviour-based paradigm informs the reactive approach to building agents. Each individual behaviour continually maps perceptual input to action output. In the reactive approach, intelligent behaviour emerges from the interaction of various simpler behaviours as well as from the interaction between an agent and its environment. The main disadvantage of this architecture relates to the fact that agents do not employ models of their environment. This means that they need a great deal of local information to determine an acceptable action. Decision making is realised in the agent's local environment without necessarily taking into account non-local information [30].

Hybrid architectures combine the deliberative and reactive approaches [4]. An agent consists of several subsystems that manifest characteristics of both deliberative (subsystems develop plans and make decisions using symbolic reasoning) and reactive (subsystems are able to react quickly to events without complex reasoning) approaches.

A popular approach to the design of hybrid agents is the use of *layered architectures* [32]. The various subsystems of the architecture are arranged into a hierarchy of interacting layers each of which is reasoning about the environment at different levels of abstraction.

3 Multi-agent systems

A multi-agent approach to developing complex software applications involves the employment of several agents capable of interacting with each other to achieve objectives [6]. The benefits of such an approach can be summarized as follows [3, 17, 22]: (i) ability to solve large and complex problems as opposed to a single centralised agent that might fail the same task; (ii) interconnection and interoperation of multiple existing legacy systems; (iii) ability to provide solutions to efficiently manage domains in which the information resources are spatially distributed; and (iv) ability to handle domains in which the expertise is distributed.

3.1 MAS definition

A MAS is a "loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [17]. The problem solvers from this definition are autonomous and possibly heterogeneous agents able to interact with each other in order to reach an overall goal. Moreover, each agent within the MAS has a limited set of capabilities or incomplete information to solve the problem. The MAS approach implies that there is no global system control, data is decentralized and computation is asynchronous [17].

The interoperation among autonomous agents of a MAS is essential for the successful location of a solution to a given problem. Agent-oriented interactions span from simple information interchanges to planning of interdependent activities for which cooperation, coordination and negotiation are fundamental. Jennings notes that these agent interactions differ from those that occur in other computational models from two perspectives [16]: (1) an agent knows which goals should be followed and, therefore, agent-oriented interactions are taking place at the knowledge level, and (2) agents are flexible entities in an environment over which they have only partial control and, therefore, they have to make run-time decisions about their interactions.

3.2 Coordination in MAS

Agents have to coordinate their activities in order to determine the organizational structure in a group of agents and to allocate tasks and resources. Coordination has been defined as "a process in which agents engage in order to ensure a community of individual agents acts in a coherent manner" [21]. Agents may have to *communicate* in order to achieve the necessary coordination.

Coordination is necessary in a MAS because agents have different and limited capabilities and expertise. Furthermore, interdependent activities require coordination (the action of one agent might depend on the completion of a task for which another agent is responsible). Coordination prevents anarchy or chaos during conflicts [20].

The foremost techniques to address coordination in MAS include organisational structuring, Contract Net Protocol (CNP), multi-agent planning, social laws and computational market-based mechanisms [4, 20].

3.3 Negotiation in MAS

Representing the focus of many research studies, negotiation is essential within a MAS for conflict resolution and can be regarded as a significant aspect of the coordination process among autonomous agents [17, 20].

Fatima, Wooldridge and Jennings [8] view negotiation as "a means for agents to communicate and compromise to reach mutually beneficial agreements". The main characteristics of negotiation include the existence of a conflict, the need to resolve the conflict in a decentralised manner by self-interested agents, bounded rationality and incomplete information [17].

Research shows that an effective negotiation process may be achieved by having agents reasoning about the beliefs, desires and intentions of other agents [20, 24]. This approach motivates the interest in other research areas such as logic, case-based reasoning, belief revisions, distributed truth maintenance, model-based reasoning, optimisation and game theory.

3.4 Communication in MAS

In order to achieve a beneficial agent interoperation, communication in a MAS is a requirement because agents need to exchange information and knowledge or to request the performance of a task since they only have a partial view over their environment [17].

Considering the complexity of the information resources exchanged, agents should communicate through an *agent communication language* (ACL) [10, 21]. Standard ACLs designed to support interactions among intelligent software agents include the Knowledge Query and Manipulation Language (KQML) proposed by the Knowledge Sharing Effort consortium [10] and FIPA ACL defined by the FIPA organization [33]. Both KQML and FIPA ACLs are designed to be independent of particular application vocabularies [4].

Furthermore, a meaningful communication process among agents requires, besides an ACL, a common understanding of all the concepts exchanged by agents. Ontologies represent one of the most significant technologies to support this requirement being capable of semantically managing the knowledge from various domains.

4 Ontologies for MAS

Many researchers emphasize the need to use ontologies for domain knowledge representation to meaningfully support agent interoperation [7].

4.1 Ontology definition

Ontologies enable content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology/ontologies by the means of ontological commitments [27]. They describe concepts and relations assumed to be always true independent from a particular domain by a community of humans and/or agents that commit to that view of the world [15]. Being generic and task-independent, ontologies differ from traditional database schemas from the following perspectives [7, 9]: (i) ontologies are syntactically and semantically richer than common databases; (ii) the ontological information contains semi-structured natural language text; (iii) ontologies must be shared and consensual; and (iv) ontologies provide domain theory.

A merge of Gruber [14] and Borst, Akkermans and Top [2] definitions is generally accepted by researchers, as follows: "Ontologies are *explicit formal* specification of a *shared conceptualization*" [28], where *explicit* means that "the type of concepts used, and the constraints on their use are explicitly defined", *formal* means that "the ontology should be machine readable, which excludes natural language", *shared* "reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group" and *conceptualization* emphasizes the "abstract model of some phenomenon in the world

by having identified the relevant concepts of that phenomenon".

Most definitions and interpretations of ontologies use consensus and formality as the key characteristics. However, only the *consensus* property is generally accepted to support the representation of knowledge from an ontology in a consensual manner [7]. Regarding the *formality* requirement, Uschold [29] allows ontologies to be expressed in a restricted and structured form of natural language, while Gruber [14] enforces a well-defined logical model for ontologies. However, the general vision is that ontologies should be machine-enabled and, if not directly human-readable, they should at least contain plain text notices or explanations of concepts and relations for the human user [2, 15, 29].

4.2 Ontology methodologies

Methodologies for building ontologies represent the focus of active ongoing research within the AI community. The pioneer methodologies include those developed within the Enterprise Ontology project and the TOVE project over ten years ago (see [4]).

Recommended by FIPA, the Methontology approach [13] to building ontologies is the most valued methodology for ontology construction within the AI community [7]. Based on an IEEE standard, the Methontology framework includes three main processes i.e. identification of the ontology development process, ontology life cycle based on evolving prototypes and particular techniques for carrying out each activity [13].

5 Agent-oriented methodologies

A systematic methodology for the analysis and design of agent-based applications is a crucial requirement for the success of agent-oriented software engineering [16, 31].

The main two approaches to the development of agent-based systems refer to (1) methodologies that extend or adapt object-oriented methodologies e.g. AAII, Gaia, MaSE, AUML and (2) methodologies that adapt knowledge engineering models or other techniques e.g. DESIRE (see [4]).

With many agent theories, languages and architectures available, more work is needed in the area of agent-oriented methodologies to assist the developer in all the phases of the life cycle of an agent-based application [4].

6 Agent languages and environments

The growing interest in the area of software agents and MAS motivates the development of languages that facilitate the design and construction of agent-based applications.

Although several languages and platforms have been created by different research groups and companies to support the development of agent-based applications, traditional languages are still used to construct agent applications [4]. Languages created for object-oriented programming (e.g. Java, C++) are extensively employed for the implementation of agent applications [6]. The reason for this is that agents and objects share some properties such as encapsulation, inheritance and message passing. However, objects may respond to the same message in different ways (polymorphism) while agents must have a common ACL. In object-oriented programming, an object can decide to invoke a method of another object but when an agent wants to do the same thing (to request an action from another agent) the decision lies with the agent that receives the request [17].

6.1 Agent-oriented programming

In the early 90s, Shoham [26] proposed a new programming paradigm called agent-oriented programming (AOP) that "promotes a societal view of computation". A specialization of object-oriented programming, AOP allows the direct programming of agents in terms of their mental state (consisting of components such as beliefs, decisions, capabilities and obligations). Agent programs control agents and include communication primitives such as informing, requesting and offering (based on the speech act theory).

Created by Rao [23], AgentSpeak(L) is a programming language that allows the formalization of BDI agents. It is based on restricted first-order language with events and actions and consists of a set of base beliefs and a set of context-sensitive plans allowing hierarchical decomposition of goals (see [4]).

Extending a previous version, ConGolog (Concurrent Golog) is a concurrent programming language for process specification and agent programming [12]. It handles concurrent processes with possibly different priorities, high-level interrupts and arbitrary exogenous actions (see [4]).

6.2 Agent platforms

Over the last years, a large number of toolkits and developing environments have been created to

support the agent developer in the task of implementing agent-based systems.

Having mentioned that it is impossible to be exhaustive, the most significant agent toolkits available include *ZEUS* – an open-source software developed by BT Laboratories, the *Java Agent DEvelopment Framework (JADE*) – a FIPA-compliant platform for MAS development, *JACK Intelligent Agents* – a Java framework developed by the Agent Oriented Software group that supports different agent models including BDI architectures, the *Open Agent Architecture (OAA)* – a domain independent framework for constructing agent-based systems and *JATLite* – a collection of Java objects and class libraries developed by Stanford University (see [4]).

7 A multi-agent ontological approach to distributed support. Applications in communication systems

A multi-agent and ontological architecture to support distributed cooperation and optimise information flows in communication networks is proposed.

7.1 Problem statement

Emerging enterprise models involve multiple users distributed in a virtual environment who have to cooperate using the software tools available in order to solve problems. Being highly heterogeneous, these users (or teams of people) can be geographically, temporally, functionally and semantically distributed over the enterprise [6]. A computer-based communication network is the work environment where interoperation has to take place [4].

Computational support is needed for communications and accessibility to knowledge, past records and histories [6]. Any software infrastructure intended to support distributed collaboration should address the following issues [4]:

- Efficient management of the information circulated in a distributed environment by providing content related support.
- Cooperation support through an effective use of communication, co-location, coordination and collaboration processes.
- Integration of the heterogeneous software tools used in the distributed environment enabling the flow of information.

The proposed architecture employs multi-agent systems for interoperation among distributed resources and ontologies for knowledge sharing, reuse and integration.

7.2 Proposed multi-agent architecture

From a high-level view, the proposed architecture consists of an Ontological Plane and a Multi-Agent Plane (see Fig. 1).

The *Ontological Plane* specifies the hierarchy of ontologies that define concepts, relations and inference rules. These ontologies compose the machine-enabled framework in which the system's information resources are circulated and stored. It also includes specific knowledge of the domain instantiated according to the rules specified by the Ontology Library. The scope of the Ontology Library is to create a common shared understanding of the application domain so that information and knowledge can be shared among the members of the distributed environment. These members can be humans or software agents. The ontology aims to establish a joint terminology between these members [4].

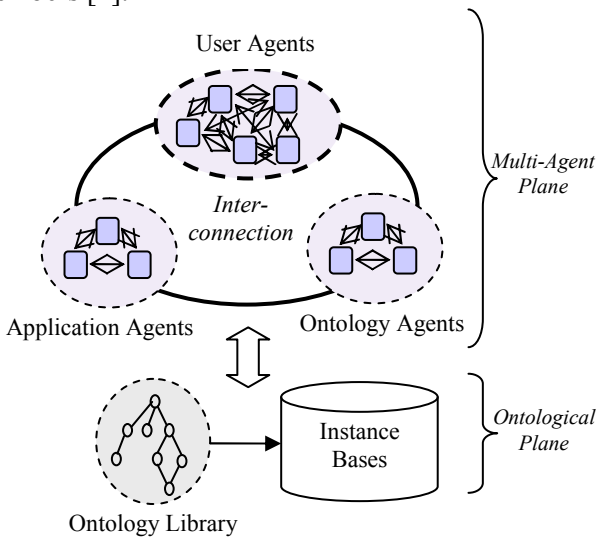

Fig.1 A high-level view of the proposed architecture

The *Multi-Agent Plane* specifies the types and behaviours of the software agents required to enable the system's functionality (see [5]). It facilitates the access, retrieval, exchange and presentation of information to distributed teams through various agent systems (e.g. user agents, application agents, ontology agents and interconnection agents). Therefore, the flow of information in the environment can be potentially optimised.

The *User Agents* form the interface between the system and the user. They provide different services to the user and respond to queries and events initiated by the user (or on behalf of the user) with

the help of the ontological agents. Examples of User Agents include a User Profile Manager agent (which should act autonomously to manage the profile of the user and should learn user preferences over time) and a User Interface Controller agent (which should provide a customizable graphical user interface based on the user profile).

The *Application Agents* are in charge of retrieving information from the software applications called by the user and forward it for storage to the ontological agents. They should be integrated in the software tools regularly used in the specific distributed domain and act autonomously pursuing their objective (i.e. information retrieval).

The *Ontology Agents* provide ontology management services in communication networks. They are able to access, retrieve, add, modify and delete information from the Ontology Library. Besides the agents that can read, write and update information, the ontology agent society should contain agents that are able to supervise the ontology management process ensuring the consistency of the ontology and the delivery of the requested ontology-related services.

The agents from the *interconnection society* supervise and support the interoperation process among the other agents. The main objective of this agent society is to ensure that agents are meaningfully interconnected. This can be achieved through a *System Manager* agent that supervises the overall functionality of the multi-agent system and a *Directory Facilitator* agent that helps agents to find other agents that provide a requested service. Based on the FIPA specifications, the *System Manager* must be able to perform functions such as register, deregister, modify, search and get-description. Furthermore, the System Manager agent has the capability to execute the actions such as suspending an agent, terminating an agent, creating an agent, resuming agent execution, invoking an agent, executing an agent and managing resources. Being FIPA compliant, the *Directory Facilitator* provides a Yellow Pages service to the agent community. Any agent can use the Directory Facilitator to find other agents providing required services for achieving internal objectives. For example, when a User Interface Controller agent needs to display information regarding a specific concept in a graphical format, the Directory Facilitator can be used to retrieve the agent identifier of the specific Ontology agent(s) that can read the requested information from the Ontology Library.

The agent interactions within the proposed system are vital for a successful and constructive support provided to distributed users. It is proposed that the agents are FIPA [33] compliant and communicate by exchanging ACL messages. The FIPA agent management ontology is part of each agent expertise to enable meaningful agent interoperation [4].

The proposed system exploits agent properties such as autonomy, cooperation, learning and pro-activeness in a semantic approach to support a process that involves dispersed heterogeneous resources and multidisciplinary people (see [4]).

8 Conclusions and Future Work

Agents and multi-agent systems have the potential to manage the complexity inherent in distributed software systems and therefore forming an important new agent-oriented software engineering paradigm [16, 31]. However, software agent research still lacks in universally accepted concepts from definitions, architectures, methodologies and languages to protocols for coordination, negotiation and communication. Ongoing research focuses on the development of agent-oriented methodologies and languages, the study of interoperation and trust models as well as the establishment of agent standards.

The potential of the multi-agent approach is demonstrated by presenting a MAS architecture for the support of distributed collaboration over a computer network. The proposed multi-agent architecture aims to optimise the flow of information in communication networks by enabling distributed resource interoperation and knowledge exchange. Future research focuses on further development of multi-agent ontological architectures for the support of emerging communication systems.

Acknowledgements

This research is supported by the Grant "Natural Computing. New Paradigms and Applications" funded by the Ministry of Education and Research, Romania.

References:

[1] H. Ali, *Security & Trust in Agent-Enabled E-Commerce: Survey*, 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, 2005.

[2] P. Borst, H. Akkermans, J. Top, *Engineering Ontologies*, International Journal of Human-Computer Studies, Vol. 46, No. Special Issue on

Using Explicit Ontologies in KBS Development, 1997, pp. 365-406.
[3] J.M. Bradshow, *An Introduction to Software Agents*, in Software Agents, 1997.
[4] C. Chira, *The Development of a Multi-Agent Design Information Management and Support System*, PhD Thesis, Galway-Mayo Institute of Technology: Galway, 2005.
[5] C. Chira, O. Chira, *A Multi-Agent System for Design Information Management and Support*, International Conference on Computers, Communications and Control (ICCCC 2006), Baile Felix Spa Oradea, Romania, 2006.
[6] O. Chira, C. Chira, D. Tormey, A. Brennan, T. Roche, *An Agent-Based Approach to Knowledge Management in Distributed Design*, Special issue on E-Manufacturing and web-based technology for intelligent manufacturing and networked enterprise interoperability, Journal of Intelligent Manufacturing, Vol. 17, No. 6, 2006.
[7] V.O. Chira, *Towards a Machine Enabled Semantic Framework for Distributed Engineering Design*, PhD Thesis, Galway-Mayo Institute of Technology: Galway, 2004.
[8] S.S. Fatima, M. Wooldridge, N.R. Jennings, *An Agenda-Based Framework for Multi-Issue Negotiation*, Artificial Intelligence, Vol. 152, 2004, pp. 1-45.
[9] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce,* 2000.
[10] T. Finin, Y. Labrou, J. Mayfield, *KQML as an Agent Communication Language*, in Software Agents, B.M. Jeffrey, 1997.
[11] S. Franklin, A. Graesser, *Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996, Berlin, Germany, 1996.
[12] G.D. Giacomo, Y. Lespérance, H.J. Levesque, *Congolog, a Concurrent Programming Language Based on the Situation Calculus*, Artificial Intelligence, 2000.
[13] A. Gomez-Perez, *Knowledge Sharing and Reuse*, in The Handbook on Expert Systems, Liebowitz, 1998.
[14] T.R. Gruber, *A Translation Approach to Portable Ontology Specification*, Knowledge Aquisition, Vol. 5, No. 2, 1993, pp. 199-220.
[15] N. Guarino, *Formal Ontology and Information Systems*, Formal Ontology in Information Systems, 1998.
[16] N.R. Jennings, *On Agent-Based Software Engineering*, Artificial Intelligence, 2000.
[17] N.R. Jennings, K.P. Sycara, M. Wooldridge, *A Roadmap of Agent Research and Development*, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1, No. 1, 1998, pp. 7-36.
[18] R. Kotina, F.P. Maturana, D. Carnahan, *Multi-Agent Control System for a Municipal Water System*, 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, 2006.
[19] M. Luck, P. McBurney, C. Preist, *Agent Technology: Enabling Next Generation Computing*, AgentLink, 2003.
[20] H. Nwana, L. Lee, N. Jennings, *Coordination in Software Agent Systems*, BT Technology Journal, Vol. 14, No. 4, 1996, pp. 79-88.
[21] H.S. Nwana, *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, No. 3, 1996, pp. 1-40.
[22] S. Park, V. Sugumaran, *Designing Multi-Agent Systems: A Framework and Application*, Expert Systems with Applications, Vol. 28, No., 2005, pp. 259-271.
[23] A.S. Rao, *Agentspeak(L): Bdi Agents Speak out in a Logical Computable Language*, Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 1996.
[24] A.S. Rao, M.P. Georgeff, *Bdi Agents: From Theory to Practice*, Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, USA, 1995.
[25] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach, 2/E,* 2003.
[26] Y. Shoham, *Agent-Oriented Programming*, in Readings in Agents,
[27] P. Spyns, R. Meersman, M. Jarrar, *Data Modelling Versus Ontology Engineering*. ACM SIGMOD Record2002.
[28] R. Studer, V.R. Benjamins, D. Fensel, *Knowledge Engineering: Principles and Methods*, Data and Knowledge Engineering, Vol. 25, No. 1-2, 1998, pp. 161-197.
[29] M. Uschold, *Knowledge Level Modelling: Concepts and Terminology*, The Knowledge Engineering Review, Vol.13, No.1, 1998.
[30] M. Wooldridge, *Intelligent Agents,* An Introduction to Multiagent Systems; ed. G. Weiss; 1999.
[31] M. Wooldridge, P. Ciancarini, *Agent-Oriented Software Engineering: The State of the Art*, in Agent-Oriented Software Engineering, 2001.
[32] M. Wooldridge, N.R. Jennings, *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review, Vol. 10, No. 2, 1995.
[33] http://www.fipa.org