KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 201–208

MULTI-AGENT DISTRIBUTED COMPUTING

CAMELIA CHIRA$^{(1)}$

Abstract. The paper takes a multi-agent approach to distributed cooperative virtual environments. Distributed computing systems aim to connect human and information resources in a transparent, open and scalable way. Multi-agent systems are investigated with the aim of compiling a successful solution for distributed computing problems. An agent-based architecture is proposed to support distributed computing by enabling interoperation among distributed resources and knowledge integration. The multi-agent approach should address load balancing and resource management problems. It is envisaged to improve the proposed architecture using stigmergic optimization techniques, particularly ant-based procedures for generating load balancing mechanisms.

1. Introduction

Intelligent agents are regarded as the natural metaphor to address distribution of data or control, legacy systems and open systems [13, 22]. Composed of several interacting agents, multi-agent systems (MAS) have 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, 14, 15, 17].

Current approach investigates MAS with the purpose of identifying a suitable architecture to support distributed computing. The proposed *Multi-Agent Knowledge Management and Support System (MAKS)* enables distributed collaboration, supports interoperation of heterogeneous resources and facilitates knowledge sharing, reuse and integration in a virtual environment. Performance and efficiency of the MAKS multi-agent architecture are evaluated for a distributed design environment. Future work emphasizes possible extensions of the proposed architecture to better address load balancing problems.

2000 *Mathematics Subject Classification.* 68M14, 68T30.
Key words and phrases. Multi-Agent Systems, Distributed Computing, Ontologies.

2. MULTI-AGENT SYSTEMS AND ONTOLOGIES

An *agent* refers to a system situated in an environment being able to perceive that environment and to act autonomously in order to accomplish a set of objectives [3, 10, 14, 17]. Agents receive inputs about the state of their environment through sensors and they can perform actions through effectors [13]. The main properties of an agent are autonomy, reactivity, pro-activeness, cooperation, learning and mobility [3, 10, 17].

A *multi-agent* approach to developing complex systems involves the employment of several agents capable of interacting with each other to achieve objectives [6]. The benefits of such an approach include the ability to solve large and complex problems, interconnection and interoperation of multiple existing legacy systems and the capability to handle domains in which the expertise is distributed [14, 18]. A MAS is composed of several autonomous and possibly heterogeneous agents [14]. 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 [14].

The interoperation among autonomous agents of 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. *Coordination* is necessary in MAS because agents have different and limited capabilities and expertise [16]. The foremost techniques to address coordination in MAS include organisational structuring, Contract Net Protocol, multi-agent planning, social laws and computational market-based mechanisms [3, 16]. *Negotiation* is essential within MAS for conflict resolution and can be regarded as a significant aspect of the coordination process among autonomous agents [14, 16, 19]. Agents within MAS need to *communicate* in order to exchange information and knowledge or to request the performance of a task as they only have a partial view over their environment [14]. Considering the complexity of the information resources exchanged, agents should communicate through an agent communication language (ACL) [9, 17] such as the Knowledge Query and Manipulation Language [9] and FIPA ACL [12].

A meaningful communication process among agents requires 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 [7, 20]. "Ontologies are explicit formal specification of a shared conceptualization" [21]. Ontologies 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 [1, 11].

3. MULTI-AGENT KNOWLEDGE MANAGEMENT AND SUPPORT FOR DISTRIBUTED COMPUTING

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 [3].

The proposed *Multi-Agent Knowledge Management and Support System (MAKS) architecture* employs multi-agent systems to manage human and information resources in distributed computing environments. Content-related support is ensured by the use of ontologies that handle the information circulated in the environment. The MAKS architecture is composed of the following four major classes of agents (see Figure 1): *(1) User Agents* represent the interface between the system and the end user; *(2) Application Agents* integrate heterogeneous tools by making the application-specific information globally available; *(3) Ontology Agents* manage the information resources of the distributed environment; and *(4) Interoperation Agents* supervise the functionality of the system ensuring agents are meaningfully interconnected and allocation of resources is appropriate.

MAKS Ontology Library composes the machine-enabled framework in which the system's information resources are circulated and stored. The aim is to establish a joint terminology between members of the distributed environment (either humans or agents) by defining concepts, relations and inference rules [3]. MAKS Multi-Agent plane of the proposed architecture specifies the types and behaviours of the software agents required to enable the system's functionality (see [4]). User Agents 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). 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. 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 (Ontology Reader, Ontology Broker, Ontology Reviser, Component Receiver agents), 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 (Ontology Manager agents). The fourth class of agents refers to Interconnection Agents that supervise and support the interoperation

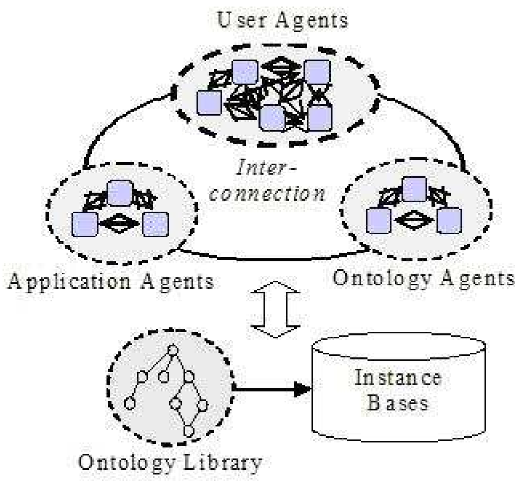

FIGURE 1. MAKS Architecture for Distributed Computing Environments

process among the other agents. Examples of Interconnection Agents include System Manager agents that supervise the overall functionality of the multi-agent system and Directory Facilitator agents that facilitate service discovery.

The MAKS architecture manages resources by employing Application Agents and Ontology Agents to deal with distributed information and knowledge and User Agents to make knowledge easily accessible and shared among dispersed computers. Interoperation Agents address load balancing problems by distributing processing activities across a computer network (creating and activating agents in different locations). Furthermore, mobile Ontology Agents can move around the network to spread information.

4. MAKS EVALUATION

The proposed MAKS architecture has been implemented for the distributed design domain. Designers are able to access information using the proposed multi-agent system in a web format (*MAKS Agent Web Portal*) or based on graphical user interfaces (*MAKS Agent Interface*).

All MAKS agents have been implemented using a Java-based environment for MAS development and are able to take the initiative (i.e. pro-activeness) and interoperate (i.e. cooperation) with other agents in order to achieve their objectives. Moreover, some of the MAKS agents (e.g. User Profile Manager, Application Controller, Ontology Manager, System Manager) should be able to operate on their own without the intervention of users or other agents. Figure 2 presents a possible deployment of the MAKS agents in a distributed engineering design environment.

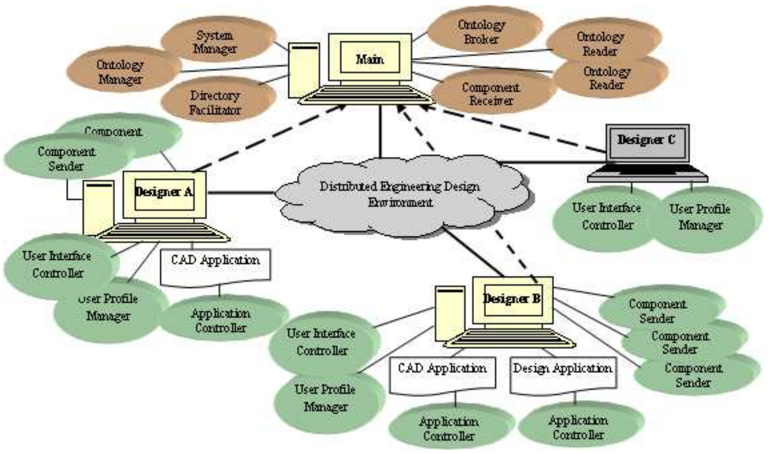

FIGURE 2. MAKS Agents deployment in a distributed design environment

The computer labelled *'Main'* in Figure 2 represents the main platform containing the MAKS manager agents e.g. System Manager, Ontology Manager that supervise the entire agent interoperation process. The other computers in the network are used by different designers each served by a User Interface Controller agent and a User Profile Manager agent (that will have to register with the System Manager and optionally with the Directory Facilitator). These User agents can be accessed through either MAKS Agent Interface or MAKS Agent Web Portal. Furthermore, some designers have one or more Application Controller and Component Sender agents active depending on the number of software applications integrated in MAKS (e.g. the information handled by Designer A using a CAD application is also organized by an Application Controller agent).

To make the design knowledge manageable by MAKS agents, the distributed design domain has been mapped to a library of ontologies defining concepts such as product, property, material, resource and process. The key concept defined in the engineering design ontology is that of a Product considered the final outcome of the design process. Figure 3 shows the UML-based ontology diagram describing the concept of a Product. Each product is viewed as a hierarchy of assemblies and parts, with each assembly being made-up of further assemblies and parts defined in terms of their characteristics (e.g. name, mass, version) and relations (has_author, has_manager, has_feature, has_material).

The testing phase of MAKS for distributed design uses the *protocol analysis (PA)* technique to evaluate the proposed system when used by a single designer or by a team of designers in a distributed environment to perform a given set of tasks. The subjects were videotaped while using the system (MAKS Agent Web Portal

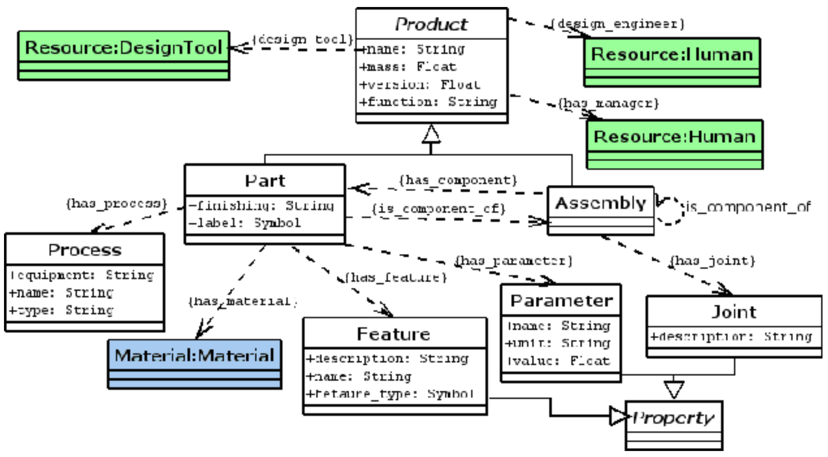

FIGURE 3. UML-based ontology diagram of a product

and MAKS Agent Interface) and verbalizing their thoughts or communicating with other designers (depending on the task). Besides the proposed multi-agent system, subjects were asked to use traditional groupware technologies to complete similar tasks. The intention was to evaluate the proposed system itself using the PA approach and furthermore to compare it with groupware technologies currently used by designers (in a best-case real scenario) to share information in a distributed design environment. The groupware technology selected for this reason is Lotus Sametime Document Repository, which allows logged users to manage documents through a web-based interface.

The transcripts of each PA session were designed to support the capture and analysis of the subject's exact verbalization, the observer's notes and the records of user's actions. The segmentation of episodes is based on the steps and different screens used by the subject in order to complete the given tasks. Figure 4 shows the episode times for each subject in a PA session where the given tasks refer to the retrieval of specific information about a component in a given assembly.

Each subject used the Sametime Document Repository and the proposed multi-agent system (through the MAKS Agent Interface and the MAKS Agent Web Portal) to retrieve the requested information. It is clear that the groupware technology (see Figure 4) was more difficult to be used whereas the MAKS Agent Interface and Web Portal have about the same amount of time allocated.

The PA test results show that agent properties such as autonomy, pro-activeness, cooperation and mobility are highly beneficial to the distributed designer during

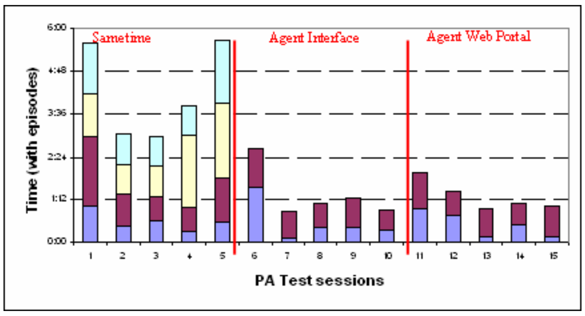

FIGURE 4. MAKS evaluation results: PA episodes for each subject

the information-intensive problem solving process of design. Compared to traditional groupware technologies, the multi-agent approach has clear potential benefits including reliability, robustness and faster access to required information.

5. CONCLUSIONS AND FUTURE WORK

Distributed computing systems aim to connect resources in a transparent, open and scalable way. The agent-based architecture proposed in the current paper employs multi-agent systems for interoperation among distributed resources and ontologies for knowledge sharing, reuse and integration. 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. The main issues addressed by the agent-based architecture include resource management and information flow in communication networks.

Future work focuses on the improvement of MAKS using emerging metaheuristics such as Ant Colony Systems and Evolutionary Computing techniques. Ongoing research focuses on Ant Colony Optimization [5, 8] algorithms and their employment for solving load balancing problems in the enhanced extension of the MAKS system.

REFERENCES

[1] 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.

[2] J.M. Bradshow, An Introduction to Software Agents, in Software Agents, J.M. Bradshow, MIT Press, 1997.

[3] C. Chira, The Development of a Multi-Agent Design Information Management and Support System, PhD Thesis, Galway-Mayo Institute of Technology, 2005.

[4] 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.

[5] C. Chira, C-M. Pintea, D. Dumitrescu, Stigmergic Agent Optimization, Romanian Academy Journal of Information Science and Technology, vol.9, no.3, pp.175-183, 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, 2004.

[8] M. Dorigo, G.D. Caro, The Ant Colony Optimization Meta-Heuristic, ed; New Ideas in Optimization; ed. D. Corne, M. Dorigo. F. Glover; 1999.

[9] T. Finin, Y. Labrou, J. Mayfield, Kqml as an Agent Communication Language, in Software Agents, B.M. Jeffrey, Ed., MIT Press, 1997.

[10] 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.

[11] N. Guarino, Formal Ontology and Information Systems, Formal Ontology in Information Systems. FOIS'98, 6-8 June 1998., Trento, IOS Press,; 1998.

[12] http://www.fipa.org, Foundation for Intelligent Physical Agents.

[13] N.R. Jennings, On Agent-Based Software Engineering, Artificial Intelligence,, 2000.

[14] 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.

[15] M. Luck, P. McBurney, C. Preist, Agent Technology: Enabling Next Generation Computing, AgentLink, ISBN 0854 327886, 2003.

[16] H. Nwana, L. Lee, N. Jennings, Coordination in Software Agent Systems, BT Technology Journal, Vol. 14, No. 4, 1996, pp. 79-88.

[17] H.S. Nwana, Software Agents: An Overview, Knowledge Engineering Review, Vol. 11, No. 3, 1996, pp. 1-40.

[18] S. Park, V. Sugumaran, Designing Multi-Agent Systems: A Framework and Application, Expert Systems with Applications, Vol. 28, No., 2005, pp. 259-271.

[19] A.S. Rao, M.P. Georgeff, Bdi Agents: From Theory to Practice, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, 1995.

[20] P. Spyns, R. Meersman, M. Jarrar, Data Modelling Versus Ontology Engineering. ACM SIGMOD Record, 2002.

[21] 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.

[22] M. Wooldridge, P. Ciancarini, Agent-Oriented Software Engineering: The State of the Art, ed; Agent-Oriented Software Engineering; ed. P. Ciancarini., M. Wooldridge; AI Volume 1957; 2001.

[1] Department of Computer Science Babes-Bolyai University Cluj-Napoca 1B M. Kogalniceanu, 400084, Romania

E-mail address: `cchira@cs.ubbcluj.ro`