Specification of Multi-Agent Systems in the Gamma Language

Lin, H.; Yang, Chunsheng

Proceedings of the IEEE 19th Annual Canadian Conference on Electrical and Computer Engineering (CCECE05), 2006

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Specification of Multi-Agent Systems in the Gamma Language **

Lin, H., and Yang, C.
May 2006

* published in the Proceedings of the IEEE 19th Annual Canadian Conference on Electrical and Computer Engineering (CCECE05). May 7-10, 2006. Ottawa, Ontario, Canada. NRC 48476.

Canada

SPECIFICATION OF MULTI-AGENT SYSTEMS IN THE GAMMA LANGUAGE

Hong Lin
Department of Computer & Mathematical Sciences
University of Houston-Downtown
linh@uhd.edu

Chunsheng Yang
Integrated Reasoning Group
Institute for Information Technology
National Research Council of Canada
Chunsheng.Yang@nrc.ca

Abstract

The suitability of using Chemical Reaction Metaphor (CRM) to model multi-agent systems (MASs) is justified by CRM's capacity in specifying dynamic features of multi-agent systems. This paper presents a case study that demonstrates the applications of CRM in specifying multi-agent systems. The case study is the specification of a navigation training system. Given the dynamic and concurrent nature of multi-agent systems, we find that the chemical reaction metaphor provides a mechanism for describing the overall architecture of the distributed multi-agent systems precisely and concisely, while giving the design of the real system a solid starting point and allowing step-by-step refinement of the system using transformational methods.

Keywords: *multi-agent systems, the chemical reaction models, program specification, distributed systems, software architecture*

1. Introduction

Agent-oriented design has become one of the most active areas in the field of software engineering. The agent concept provides a focal point for accountability and responsibility for coping with the complexity of software systems during both design and execution [1]. It is deemed that software engineering challenges in developing large-scale distributed systems can be overcome by an agent-based approach [2]. In this approach, a distributed system can be modeled as a set of autonomous, cooperating agents that communicate intelligently with one another, automate or semi-automate functional operations, and interact with human users at the right time with the right information. Such a model should be general enough to address common architectural issues and not be specific to design issues of a particular system. A direct benefit of such a model is expressiveness and extensibility --- changes in the domain knowledge would not require an intensive system-wide modification to alter the information and objects that initiate actions based on that changing information.

The modeling issue in the abstract computing machine level has been studied in [3], where the chemical reaction model [4-7] is used to model an autonomic system. Given the dynamic and concurrent nature of multi-agent systems, we find that the chemical reaction metaphor provides a mechanism for describing the overall architecture of the distributed multi-agent systems precisely and concisely, while giving the design of the real system a solid starting point and allowing step-by-step refinement of the system using transformational methods [8, 9].

In Section 2, we present a brief description of the ontology of modeling MASs in Chemical Reaction Metaphor; In Section 3, we present a case study in the navigation training system. Conclusions are drawn in Section 4.

2. The Modeling Ontology

Although a precise definition of an agent system is yet to be given, features of an agent system have been summarized in the literatures. According to Griss and Pour [10], an agent shows a combination of a number of the following characteristics: autonomy, adaptability, knowledge, mobility, collaboration, and persistence. These features exist in different types of agent systems such as collaborative agents, interface agents, reactive agents, mobile agents, information agents, heterogeneous agents, and economic agents. The concurrency and automation of agents require that the modeling language does not have any sequential bias and global control structure. In addition, the dynamic nature and non-determinism of interaction between an agent and its environment are suited to a computation model with a loose mechanism for specifying the underlying data structure. For example, data, which move around the Internet, can be well modeled by chemical solution; and mobile agents, which are created dynamically and transferred from clients to servers, can be represented as molecules containing γ-abstractions that transfer among solutions. This provides a mechanism for describing inter-agent communications and agent migration in a single framework. Interested readers are referred to [8], where a sequence of case studies show that features of those different agent systems can be grasped by the Gamma language succinctly, and a comprehensive case study in the design of an e-learning system; and [9], which presents a comprehensive method for architectural design of MASs and discussions on the implementation in the higher-order Gamma framework. It is worthwhile to note that CRM specifications separate architectures from nonessential features of the system effectively. For example, it catches the way program units interact with one another and leaves nonessential specifications, such as the number of program units, connection links for communications, and organizations of data, to the subsequent design phases.

In summary, the benefits of using CRM include: (1) The architectural design of the system can be separated from the design of individual units that have to deal with proprietary features of the underlying computing resources, because CRM allows us to treat each node in the distributed networking systems as an element of a multi-set data structure, which in-turn can be an active program to be defined in a lower level of the program structure. (2) Parallelism can be easily achieved without extra efforts in designing communication and synchronization mechanism because CRM express them implicitly. (3) Concurrency and dynamic nature of MAS can be easily reflected by CRM's non-determinism feature. (4) Autonomy can be expressed naturally by CRM's locality of reaction feature. (5) It provides a framework for combining different programming technologies because no assumptions are made about the way for implementing each node in the system hierarchy. (6) The reusability of the agent systems can be promoted by higher-order CRM languages because the existing agents can be combined by using higher-order operations defined in those languages.

3. Specification of Navigation Training Systems

To allow students to access the training systems from different locations and platforms at any time, we propose to use distributed learning environments as the infrastructure for multi-agent-based education systems [11]. As shown in Figure 1, the navigation training systems provide a distributed training environment for students from different locations and at any times to learn how to handle ships in navigation environments. Obviously, this is a multi-agent-based distributed learning environment, which provides intelligent decision-making support as well as a multitude of training course objects for students to chose and experience under various navigation environments.

A multi-agent-based system for navigation training consists of the client portion and the server portion. On the client side it has an html/JSP user interface. On the server side there are Java servlets and a multi-agent platform implemented using JADE (Java Agent Development Framework) [12]. JADE is a framework for developing multiagent systems according to FIPA (The Foundation for Intelligent Physical Agents) standards.

This multi-agent-based distributed infrastructure provides a fundamental learning environment, which consists of the following main agents:

- Information management agent (IMA)
- Intelligent decision-making support agent (IDMS)
- MGIS support agent (MGIS)
- Collaboration agent (CA)
- Student interface agent (SIA)

With these agents, the learning environments support the interactions among agents that may be geographically dispersed on the Internet. This distributed learning environment provides an on-line training environment that enables students to acquire knowledge and improve navigation skills at any time and from anywhere, simply by using a Java-enabled browser. The following is a scenario showing how students use the training systems. When a student logs on a education system through web-based applications, a student interface agent looks at requirements and profile, then chooses a training course object. After the system downloads the assigned training course object, it can run the course object, which will communicate with a collaboration agent. The collaboration agent coordinates with other agents. For example, it will ask an information agent to provide navigation environment data to an intelligent decision-making support agent and an MGIS agent. Once the MGIS agent receives the navigation environment data it provides geographical information to student's learning object and displays an ECDIS (Electronic Chart Display Information Systems) chart on his/her local desktop. An intelligent decision-making agent provides the recommended action for safe navigation with detailed explanations. Therefore, the students could acquire basic navigation knowledge and master navigation skills through the well-designed training course objects.

In the architectural specification of the navigation training system, we avoid involving specific user requests and deduction rules or knowledge used to form the response to the user requests. Instead, we focus on the interaction among agents in the system.

IMA manages the base information needed to support decision-making. The base information includes the information about the global environment. Information managed by IMA is considered static because it is a collection of knowledge and historic facts. MGIS manages specific environment knowledge and performs inferences to acquire information needed for decision-making. It may get information from IMA to support its operations. In addition, MGIS has a rendering component that displays the geographic environment in graphical mode. IDMS receives requests from the SIAs and makes recommendations for the students. IDMS interacts with MGIS to obtain necessary information for decision-making. SIA provides user interface through which the students send requests to IDMS and obtain responses from IDMS. CA is an agent that manages the interactions among other agents. It defines communication protocols and handles the communication channels. In our system, we describe CA as a manager of a pool of messages transmitted among agents.

A request-response cycle includes the following steps: A request is generated by one of the SIAs, and the request is transmitted to IDMS. IDMS finds a set of deduction rules that will be used to form a solution and a set of necessary factors that are needed by the deduction rules. To obtain the factors, a set of requests are sent to MGIS. MGIS finds another set of rules that will be used to obtain the requested factors. It also finds a set of necessary factors that are needed by IDMS and a set of requests to obtain the factors and send them to IMA. IMA then responses with the requested factors, which enable MGIS to find the factors requested by IDMS and in turn enable

IDMS to form the solution. Note that all the communications in the above process are coordinated by CA.

The Gamma specification of the system follows:

```
NT S D E K =
      [CA, MGIS = [P, Knl = D,
                      Req = Ø,
                      Fac = Ø,
                      Sol = Ø
                  ],
            IMA = [Q, Env = E,
                      Req = Ø,
                      Sol = Ø
                  ],
            IDMS = [R, Knl = K,
                       Reg = Ø,
                       Fac = Ø,
                       Sol = Ø
                   ],
            SIA = [T, Stu = S,
                      Req = Ø,
                      Sol = Ø
                  ]
       ] where
      CA = Up1 + Up2 + Up3 +
           Dn1 + Dn2 + Dn3 where
     Up1 = s:SIA, i:IDMS →
           s[r:Req = Ø]:SIA,
           i[r:Req = r+s.r]:IDMS ←
           s.r ≠ Ø
     Up2 = i:IDMS, m:MGIS →
           i[r:Req = Ø]:IDMS,
           m[r:req = r + i.r]:MGIS ←
           i.r ≠ Ø
     Up3 = m:MGIS, i:IMA →
           m[r:req = Ø]:MGIS,
           i[r:req = r + m.r]:IMA ←
           m.r ≠ Ø
     Dn1 = i:IMA, m:MGIS →
           i[s:Sol = Ø]:IMA,
           m[f:Fac = f + i.s]:MGIS ←
           i.s ≠ Ø
     Dn2 = m:MGIS, i:IDMS →
           m[s:Sol = Ø]:MGIS,
           i[f:Fac = f + m.s]:IDMS ←
           m.s ≠ Ø
     Dn3 = i:IDMS, s:SIA →
           i[s:Sol = Ø]:IDMS,
           s[s:Sol = s + i.s]:SIA ←
           i.s ≠ Ø
```

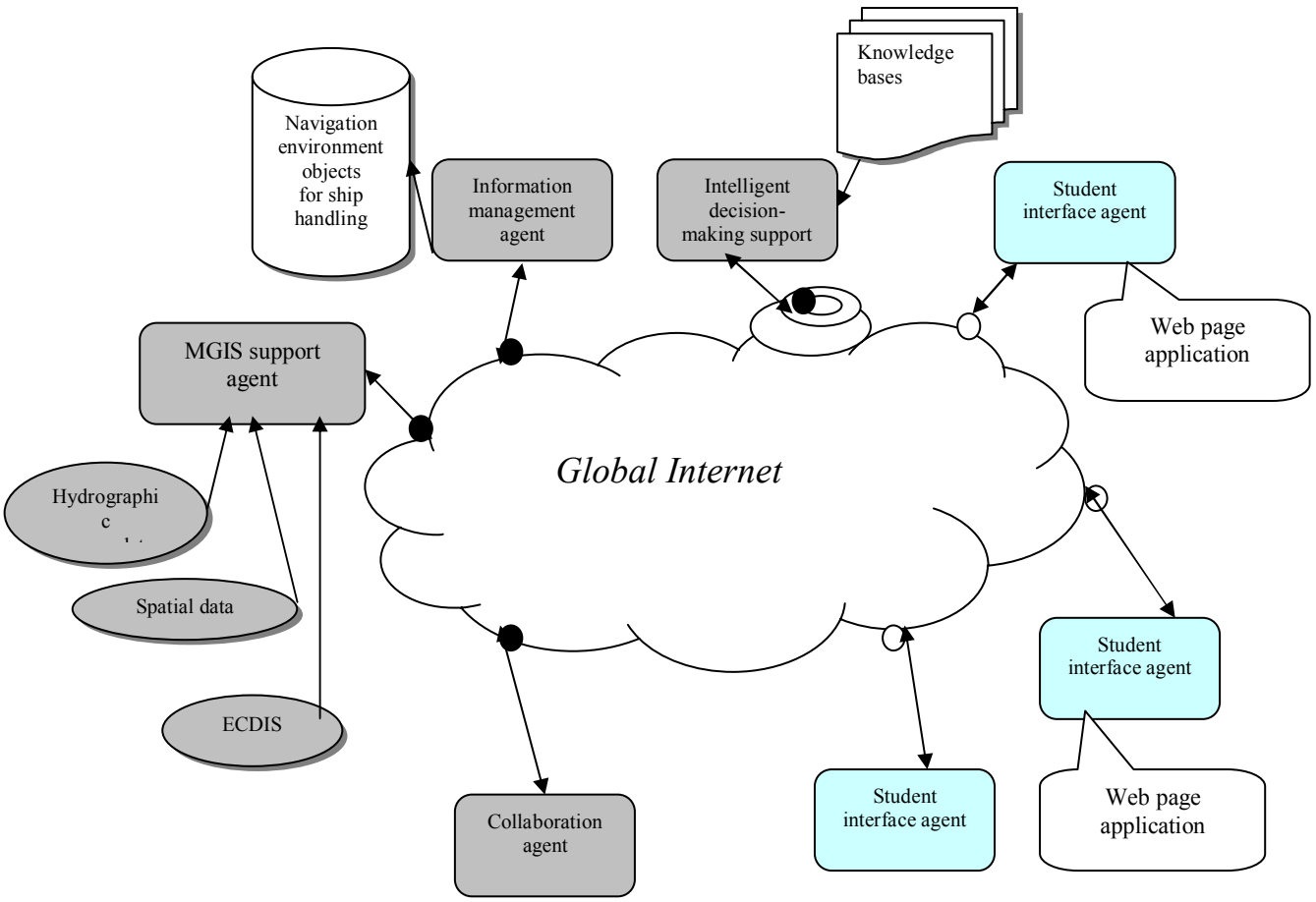

Figure 1, Infrastructure for multiagent-based education systems

```
P = Pre + Post where
    Pre = r:Req, d.Dat →
          f:Fac, d:Dat ←
          f = Pred(r, d)
    Post = f:Fac, d:Dat →
           s:Sol, d:Dat ←
           s = d(f)
Q = r:Req, e:Env →
    s:Sol, e:Env ←
    s = e(r)
R = Pre + Post where
   Pre = r:Req, k:Knl →
         f:Fac, k:Knl ←
         f = Pred(r, k)
   Post = f:Fac, k:Knl →
          s:Sol, k:Knl ←
          s = k(f)
T = s:Stu → s:Stu, r:Req ←
    Send(s, r)
```

NT is the main configuration of the navigation training system. S, D, E, and K denotes the set of students, the database maintained by the MGIS agent, the environment information base maintained by IMA, and the knowledge and inference rule base maintained by IDMS agent. In NT configuration, each agent is represented by a configuration, viz., MGIS, IMA, IDMS, and SIA. CA is represented as a set of higher level rules that operate on other configurations. The Up1, Up2, and Up3 rules are used to transfer requests from SIA to IDMS, from IDMS to MGIS, and from MGIS to IMA, respectively. Dn1, Dn2, and Dn3 rules transfer solutions from IMA to MGIS, from MGIS to IDMS, and from IDMS to SIA, respectively. The Pre rules in MGIS and IDMS are used to form outgoing requests based on received requests; and the Post rules are used to send solutions to the requesting agent.

Other notes: If c is a configuration and e1 an environment variable in c, c[e1 = e2] denotes c with e1 replaced by e2. If d is a deduction rule and f a fact, s = d(f) means that s is the solution found out by applying d to f. If d is a deduction rule and r a request, f = Pred(r, d) means that f is a factor needed to apply d to r.

4. Conclusions

Modeling MASs in the chemical reaction metaphor is effective in the design of MASs due to its capacity in specifying architectural properties. As a case study, we discussed how to specify multi-agent-based education systems for navigation training in the Gamma language. We would like to point out that this approach is useful not only for designing multi-agent-based education systems but also for other multi-agent-based systems. This approach supports the agent system ontology by effectively separate architectural properties from proprietary platform features, and thus allows step-by-step modular designs of MASs.

Acknowledgements

This research is supported from U.S. Army Research Office Award #W911NF-04-1-0024 through Scholars Academy of University of Houston-Downtown.

References

[1] E. Yu, "Agent-oriented modeling: software versus the world," Agent-Oriented Software Engineering AOSE-2001 Workshop Proceedings, LNCS 2222, Springer Verlag, pp. 206-225.

[2] G. Paquette, "Implementing a virtual learning center in an organization," Proc. of ITHET-2001, Kumamoto, Japan, July 2001.

[3] J.-P. Banâtre, P. Fradet, and Y. Radenac, "Chemical specification of autonomic systems," In Proc. of the 13th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE'04), July 2004.

[4] J.-P. Banatre, and D. Le Metayer, "The Gamma model and its discipline of programming," *Science of Computer Programming*, Vol. 15, 1990, pp. 55-77.

[5] J.-P. Banatre, and D. Le Metayer, "Programming by multiset transformation," *CACM*, vol. 36, No. 1, 1993, pp. 98-111.

[6] J.-P. Banâtre, P. Fradet, and Y. Radenac, "Principles of chemical programming," In S. Abdennadher and C. Ringeissen (eds.): Proc. of the 5th International Workshop on Rule-Based Programming (RULE'04), Vol. 124, ENTCS, (June), 2005, pp. 133-147.

[7] D. Le Metayer, "Higher-order multiset processing," DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 18, 1994, pp. 179-200.

[8] H. Lin, "A language for specifying agent systems in E-Learning environments," in: F.O. Lin (ed.), *Designing Distributed Learning Environments with Intelligent Software Agents*. 2004, pp. 242-272.

[9] H. Lin, and C. Yang, "Specifying Distributed Multi-Agent Systems in Chemical Reaction Metaphor," *The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, Springer-Verlag, Vol. 24, No. 2, April 2006, pp.155-168.

[10] M. Griss, and G. Pour, "Accelerating development with agent components," *Computer,* IEEE, May, 2001, pp. 37-43.

[11] C. Yang, et al. "Applying Collision Avoidance Expert System to Navigation Training Systems as an Intelligent Tutor", *Proceedings of IEA/AIE 14th International Conference*, Budapest, Hungary, June, 2001, pp. 941-947.

[12] F. Bellifemine, A Poggi,. and G. Rimassa, "Developing multi-agent systems with JADE," In the *Seventh International Workshop on Agent Theories, Architectures and Languages (ATAL-2000)*, Boston, MA., 2000 (Available at:http://jade.cselt.it).