Modelling Cooperative Multi-Agent Systems

Lijun Shan

Department of Computer Science
National Univ. of Defence Technology
Changsha, 410073, China
Email: lijunshancn@yahoo.com

Hong Zhu

Department of Computing
Oxford Brookes University
Oxford OX33 1HX, UK
Email: hzhu@brookes.ac.uk

Abstract. Cooperative computing is becoming inevitable with the emerging of service-oriented computing and GRID becoming a ubiquitous computing resource. It is widely recognized that agent technology can be employed to construct cooperative systems due to agents' autonomous and collaborative characteristics. We devise an agent-oriented modelling language called CAMLE for the analysis and design of MAS (Multi-Agent Systems). This paper presents the collaboration model that captures communication between agents. The structure of the collaboration model and the notation of collaboration diagrams are presented. Uses of the modelling language are illustrated by examples.

1. Introduction

Cooperation between software systems shows its importance as GRID is becoming a ubiquitous computing resource. The recent years has also witnessed the emergence of service-oriented computing such as web services, where services can be dynamically discovered, negotiated, requested and provided. Agent technology has been widely recognized to be a viable approach due to agents' autonomous and collaborative characteristics. Although cooperation is one of the key concepts in MAS, researchers have offered various definitions and typologies [1]. We consider cooperation as the embodiment of agents' social ability. Agents can determine, to certain extent, when, how and with whom to interact at run-time. However, they must obey certain cooperation protocols to achieve their designed objectives. Design and analysis of such protocols is one of the central problems in the research on cooperative computing. This paper addresses this problem from an agent-oriented modelling approach.

Researchers have investigated general problems associated with cooperation. Based on the speech act theory, a number of ACL (agent communication language) have been proposed, including KQML [2], FIPA ACL [3], etc. Recently, graphic notations are employed to model communication in MAS. For example, AUML describes agent communication protocols in a graphic notation that extends UML sequence diagrams [4]. However, few modelling language has been formally defined and reported in the literature.

In [5, 6, 7, 8], we developed SLABS (Specification Language for Agent-Based System) and CAMLE (Caste-centric Agent-oriented Modelling Language and Environment) for engineering MAS. One of the central issues in MAS development is the modelling of agents' cooperative behaviour. We address the problem at three levels. At the top level, a caste model defines the architecture of the system by grouping agents into various castes, which can be roughly considered as agent class; see [6] for more details and formal definition of the concept. At the middle level, communications between agents are specified in a collaboration model. At the lower level, a behaviour model defines the internal behaviour of various agents so that their cooperation with each other is realized by taking certain actions in certain scenarios. This paper focuses on the collaboration model. A collaboration model consists of a number of collaboration diagrams. Horizontally, the diagrams are organized as one general and some scenario-specific collaboration diagrams. Vertically, a hierarchy of collaboration models supports collaboration modelling on different granularity.

The remainder of paper is organized as follows. Section 2 gives the background by briefly reviewing the conceptual model underlying our agent-oriented methodology. Section 3 presents the structure, notation and uses of collaboration model. Section 4 concludes the paper with a brief summary and outline of our related work.

2. Overview of the conceptual model

This section briefly reviews the underlying conceptual model for MAS defined in SLABS and used in CAMLE. The conceptual model is from a software engineering perspective. The basic concepts can be characterized by a set of pseudo-equations. In particular, equation (1) states that agents are defined as real-time active computational entities that encapsulate data, operations and behaviour and situate in their designated environments. Here, data represent an agent's state. Operations are the actions that an agent can take. Behaviour is a collection of sequences of state changes and operations performed by the agent in the context of its environment. By encapsulation, we mean that an agent's state can only be changed by itself and it has its own rules that govern its behaviour in the designated environment to decide 'when to go' and 'whether to say no'.

$$Agent = \langle Data, Operations, Behaviour\rangle_{Environment} \tag{1}$$

As an extension to the notion of class in object-orientation, a caste has a set of agents as its members. As stated in equation (2), these members share a set of structural and behavioural characteristics defined by the caste. An agent can dynamically change its membership to castes during its existence by joining in a caste or retreating from its current caste at run-time. A caste may inherit from a number of other castes. Fig. 1 shows the structure of the description of a caste.

$$Caste_t = \{agents \mid structure\ characteristics\ \&\ behaviour\ characteristics\} \tag{2}$$

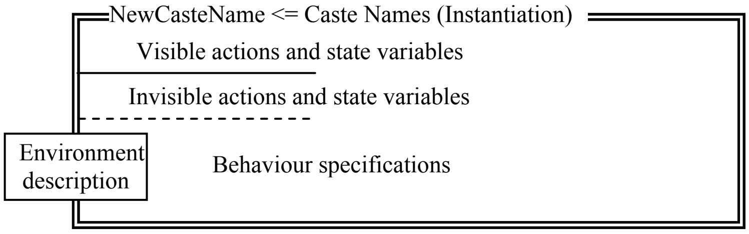

NewCasteName <= Caste Names (Instantiation)
Visible actions and state variables
Invisible actions and state variables
Environment description
Behaviour specifications

Fig. 1. Structure of Caste Description in SLABS

Equation (3) states that a MAS consists of a set of agents. The environment of an agent is a subset of all agents in the system, as stated in equation (4). The environment description of an agent defines which agents are visible.

$$MAS = \{Agent_n\}_{n \in I} \quad (3)$$

$$Environment_t\,(Agent, MAS) \subseteq MAS - \{Agent\} \quad (4)$$

The mechanism of communication is that an agent's actions and states are divided into two parts, the visible and invisible ones. Agents communicate with each other by taking visible actions and changing visible state variables, and by observing other agent's visible actions and state variables, as expressed in equation (5).

$$Communication\ from\ agent\ A\ to\ B = A.\ Action + B.\ Observation \quad (5)$$

3. The Collaboration Model

A collaboration model captures cooperation in a MAS by a collection of diagrams. Communication defined in section 2 is represented in collaboration diagrams by a notation shown in Fig. 2. An agent node denotes a specific agent. Agents are the basic components of a system and are considered as black boxes with only their names inscribed in the nodes. A caste node denotes any agent in the caste. Interaction between agents is modeled by communication links that connect agent/caste nodes. A communication link labeled with a list of actions from node N_1 to N_2 represents that agent N_1 influences N_2 by N_1 taking and N_2 observing the actions. Actions can be numbered to denote the temporal order of their occurrence.

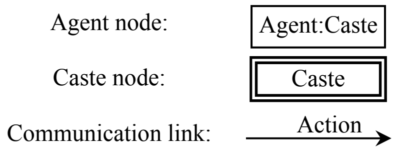


Fig. 2. Notation of Collaboration Diagrams

Fig. 3 shows an example of collaboration diagram that represents the interactions between the members of a university. For instance, an undergraduate student listens to his personal tutor for academic advice on selection of modules, attends lectures given by faculty members and practical classes given by PhD students. When he graduates, he may want to apply for graduate course.

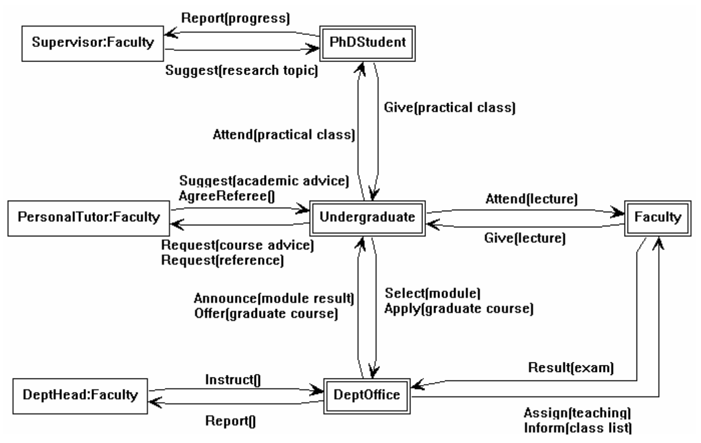


Fig. 3. Example of Collaboration diagram

Although the notation of our collaboration diagrams looks similar to that of collaboration diagrams in object-oriented methodologies such as UML [9], there are significant differences in the semantics. In OO paradigm, when a message is passed from object A to object B, object B must execute the corresponding method. Therefore, actions annotated on the link from A to B in UML diagrams are actually methods of B. In our model, however, the actions annotated on a link from A to B are visible actions of A, and agent B does not necessarily respond to agent A's action. It fits well with the autonomous nature of agents.

A flat diagram representation does not scale well for complex systems, so we extend the basic collaboration diagram to a collaboration model that comprises a set of diagrams to help handle systems' complexity. We consider collaboration modelling from two perspectives: the agent perspective, viz. which agents are to be involved in each scenario of system behaviour, and the communication perspective viz. what communication the agents take to meet a specific global requirement. Therefore, the collaboration model is organized from the two aspects: the hierarchical organization of super-sub diagrams makes explicit the modelling domain, and the horizontal organization of general-specific diagrams characterizes various scenarios the agents participate in. Fig. 4 shows the example of a collaboration model's structure. The system is directly composed of agents of three castes: *A*, *B* and *C*. Each of them can be decomposed into some components, called *component agents*. The process of decomposition terminates when some agents, such as M_1, M_2 and M_n are identified as atomic components. An agent that is consists of a number of agents as component is called a *compound agent*. For each compound agent, such as the System, A, B and C, a collaboration model including one general and a number of specific diagrams is constructed to describe the collaboration between its components.

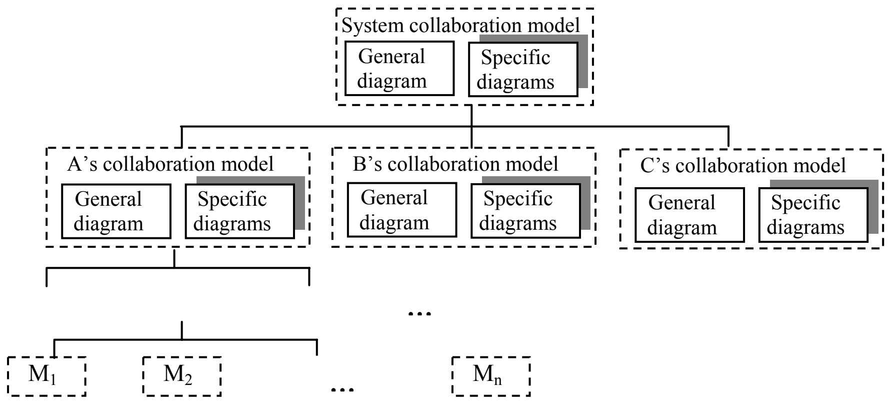
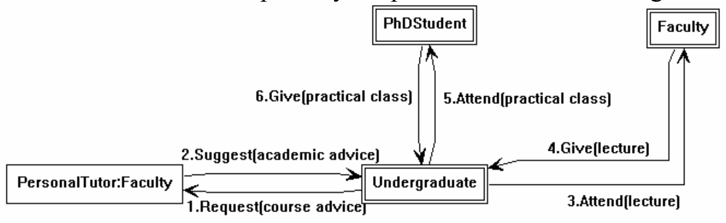


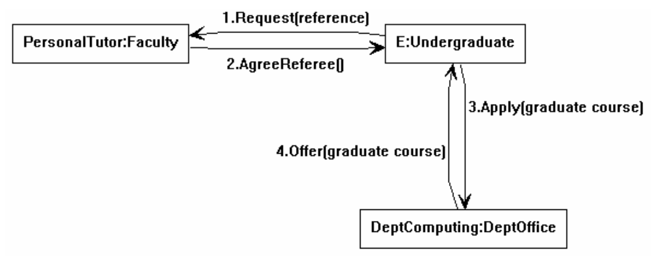
Fig. 4. Example: A Collaboration Model's Structure

3.1 Horizontal structure of collaboration model

One of the complications in collaboration modelling is on account of agents' various behaviour in different scenarios during the system's execution. By scenario, we mean a typical situation in the operation of a system. Various scenarios involving various sets of communications occur in their respective temporal sequences, therefore it is better to describe them separately. The collaboration model supports separation of scenarios by the general-specific diagram organization. A general collaboration diagram gives an overall picture of the communication between all the agents in a system by describing all visible actions an agent may take and all observers of the actions. Specific collaboration diagrams provide the means of grouping communications into separate diagrams in terms of scenarios. Each specific diagram describes a specific scenario by capturing a collection of related communications between some agents. For example, Fig. 5 shows two specific collaboration diagrams for the example of university. Diagrams in (a) and (b) respectively depict the scenarios of undergraduate's study and applying for graduate course. They can be considered as presentation of specific parts described in Fig. 3. In each diagram, the actions are numbered to indicate their temporal orders in the specific scenario. Similarly, other scenarios in the university, such as graduate's study and faculty's work can also be described separately in specific collaboration diagrams.



(a) Scenario of Undergraduate Study



(b) Scenario of Undergraduate Graduate

Fig. 5. Examples of Specific Collaboration Diagram

With the general and specific diagrams as complementary facilities for collaboration modelling, our language supports both decomposition and scenario-driven analysis approach. The decomposition approach means a whole-dividing process that begins by identifying all the agents' actions and communications in a general diagram according to global system requirements. Then various scenarios that may occur during the system's execution are plot out and communications involved in the specific scenarios are elaborated into specific diagrams. This approach may be suitable for the applications with a global requirement. In contrast, the scenario-driven approach means a part-integrating process that starts with specific situations modelling and finishes with a general description. This approach is suitable when a scenario-based representation of the application requirements has been given. It is up to the users to apply either of the two approaches or a hybrid of them in certain applications.

3.2 Vertical structure of collaboration model

The modelling language allows describing systems at a coarse granularity, that is, a system can be viewed as an agent that interacts with users and/or other systems in its external environment. Moreover, a sub-system can also be viewed as an agent that interacts with other sub-systems. As analysis deepens, the agents can be decomposed into components. Analysis of interaction among such component agents is in the same way as the analysis of the whole system. The only difference is that the environment of the components is clearer than the whole system, and such information can be carried over to the analysis of the components. Therefore a lower level collaboration diagram may have *environment nodes*, denoting the agents in the compound agent's environment, drawn on the boundary. The lower level diagram which describes communication among component agents is called a *sub-diagram*. And the higher level diagram is called the sub-diagram's *super-diagram*. Component agents are capable of communicating with the peer component agents as well as with external agents. A communication link from a component to an environment node indicates that the component agents take some particular tasks of its compound agent. In this way, the compound agent has its functionality decomposed through the decomposition of its structure.

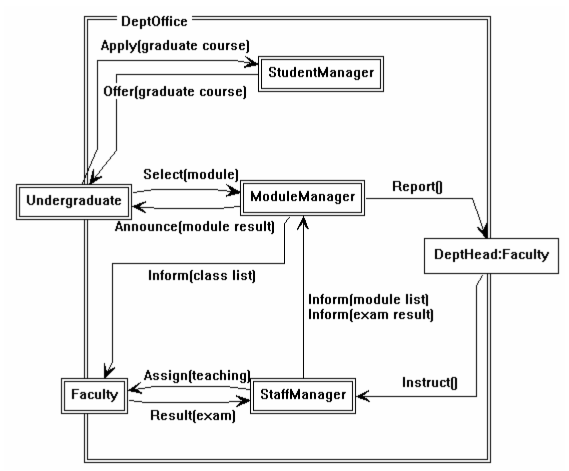


Fig. 6. Collaboration Diagram for Decomposition of DeptOffice

Fig. 6 shows an example of the decomposition of the caste DeptOffice in a lower level collaboration diagram. The caste DeptOffice in Fig. 3 means a department office in the university. The castes Undergraduate and Faculty and agent DeptHead that interact with the caste DeptOffice described in Fig. 3 are carried to Fig. 6 as the environment nodes. The DeptOffice consists of three castes: the StudentManager, ModuleManagers and StaffManagers. This lower level diagram describes the internal structure of the DeptOffice and the interactions between the component agents.

Component agents can be further decomposed into a set of components if necessary, followed by analysis of their communications in lower level diagrams. Such a refinement can be carried on until the problem is specified adequately in detail. Thus, a collaboration diagram on system level that specifies the boundary of the application can be eventually refined into a collaboration model comprising a hierarchy of collaboration diagrams on various abstract levels. Of course, the hierarchical structure of collaboration diagrams can also be used for bottom-up design and composition of existing components to form a system.

In order to obtain a meaningful collaboration model, consistency between general and specific diagrams and that between models at different levels must be assured. Consistency constraints on collaboration model as well as other constraints for CAMLE model are defined in [8].

4. Conclusion

This paper presents a collaboration model that captures communications in MAS by describing the agents' interconnections through action taking and observing. Thus actions as a part of an agent's internal capability are related to its external behaviour in terms of its cooperation with others. This view of communication leads to the independence of collaboration model to *ad hoc* communication languages or protocols, therefore makes it easy to model cooperation in a rather early stage of system analysis and enable engineers to focus on the conceptual analysis and design of agent communication. Diagrams in a collaboration model are organized into a hierarchy to represent agents on different levels. Separation of concerns in terms of various scenarios of system behaviour helps engineers to manage complexity and to employ decomposition analysis or scenario-driven approach in specific applications.

The work reported in this paper is a part of our research for modelling, formally specifying and developing MAS. An environment supporting multi-view modelling of MAS in CAMLE language has been designed and implemented. Besides the support to model construction, the environment can perform consistency checking for models of an application against the consistency constraints and can transform diagrammatic models in CAMLE to formal specifications in SLABS. Work in this direction will be reported separately.

Acknowledgement

The work reported in this paper is partly supported by China High-Technology Programme (863) under the Grant 2002AA116070.

Reference

[1] J. E. Doran, S. Franklin, N. R. Jennings & T. J. Norman. On Cooperation in Multi-Agent Systems. Panel discussion at the First UK Workshop on Foundations of Multi-Agent Systems (held at the University of Warwick on Oct. 23rd 1996).

[2] Y. Labrou and T. Finin. A Proposal for a New KQML Specification. Tech. Report TR-CS-97-03, Computer Science and Electrical Engineering Dept., Univ. of Maryland, Baltimore County, Baltimore, Md., 1997.

[3] FIPA. FIPA'99 Specification Part 2: Agent Communication Language. Available at http: www.fipa.org

[4] B. Bauer, J. P. Muller and J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. *International Journal of Software Engineering and Knowledge Engineering*. Vol. 11, No. 3, pp.1-24, 2001.

[5] H. Zhu. SLABS: A Formal Specification Language for Agent-Based Systems. *International Journal of Software Engineering and Knowledge Engineering*, Vol. 11. No. 5, pp529~558. 2001

[6] H. Zhu. Representation of roles in caste, Technical report TR-DoC-03-01, Department of Computing, Oxford Brookes University, 2003.

[7] L. Shan and H. Zhu. Modelling and specification of scenarios and agent behaviour, To appear in *IEEE/WIC Conference on Intelligent Agent Technology* (IAT'03), Halifax, Canada, Oct. 2003.

[8] L. Shan and H. Zhu Consistency Constraints on Agent-Oriented Modeling of Multi-Agent Systems, Technical Report TR-DOC-03-03, Department of Computing, Oxford Brookes University, Oxford, UK, Nov. 2003.

[9] G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide. Addison Wesley. 1999