Enabling Effective and Emergent Agent Conversations

Fuhua Lin
Institute for Information Technology
National Research Council
Building M-50, Montreal Rd.
Ottawa, K1A 0R6
CANADA
Tel: (1-613) 993-2535
Fuhua.Lin@iit.nrc.ca

Douglas H. Norrie
Dept. of Mechanical &
Manufacturing Engineering,
University of Calgary, 2500
University Dr. NW, Calgary, AB,
T2N 1N4, CANADA
Tel: (1-403) 220-5787
norrie@enme.ucalgary.ca

Rob Kremer, R.A. Flores-Mendez
Dept. of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta T2N 1N4
CANADA
Tel: (1-403) 220-5112
kremer|robertof@cpsc.ucalgary.ca

ABSTRACT

To enable effective and emergent conversations among software agents in open and distributed environment, this paper proposes a schema-based conversation modeling and manager-based conversation management approach. Conversation policies and interaction patterns among a group of agents are formulated and organized into class hierarchies of goal-directed conversation schemata, which are "sender-initialized" group interaction patterns and "receiver-responded" coordination constraints. Colored Petri Nets formalism is used for verifying conversation schemata. Using conversation schemata, conversation managers are constructed to mediate and administer conversations.

Keywords

Agent technology, schema, conversation, Colored Petri nets

1. INTRODUCTION

People can rely on patterns and incorporate constraints in the world to make their communication more efficient and reliable. In software agent systems, when communicating with others, agents can also rely on communication patterns and group behavioral constraints. Based on the notion of patterns and constraints, a conversation schema (schema, for short) -based method for modeling conversation is presented [1]. We define a conversation schema as a pattern of conversational interactions centered on one or more conversation topics. A goal-directed schema is a schema in which the pattern of interaction is directed towards achieving a specified goal of participating agent(s). When instantiated, a schema governs the conversation between a group of agents by imposing a role on each of the group members. A schema is detailed by communicative acts and implemented as a thread for execution.

2. BACKGROUD

From methodology point of view, there are two extremes in the existing high-level agent conversation approaches. The first is "act-based" approach that leaves low-level tasks to be accomplished by fixed speech acts that are themselves highly conditional. The second is "plan"-based approach.

A conversation plan is a description of how *an agent* acts locally and interacts with other agents by means of communicative acts. This approach is not flexible and practical enough when used in a dynamic environment. The conversation policies (CP) approach tries to find a sweet spot between these two extremes [2].
From the implementation point of view, current techniques for developing agent applications require developers to implement CPs and individual agent behaviors (functionality) together. There are three shortcomings: complicated code, lacking of modularity, and brittle code [3].

3. CONVERSATION MANAGERS

An agent is a more or less complex software module that is able to communicate with others via conversations. For better organization and management, in our approach, an agent system is divided into three layers: the TOP layer consisting of agents like interface agents, task agents, the C&C (communication and cooperation) layer, and the NETWORK layer. In an agent system, the high-level communication is supported by a group of agents called Conversation Managers (CMs) inhabiting the C&C layer. The agents do not communicate with one another in an ad-hoc, point-to-point manner. Instead, agents that are working together form *cooperation domains*. Each agent in a cooperation domain routes all its outgoing messages through the CM, which can direct it to a specific agent (imitating point-to-point communication), to several agents (imitating multicast communication), or to all agents in the cooperation domain (imitating broadcast communication). All incoming messages are received from the CM as well (the original sender's identity is contained in the message header). The benefit of all messages going through the CM is that the CM can then provide several services, such as coordination, conflict resolving, and security control.

A CM sits among conversing agents. After selecting a suitable schema, it keeps track of conversation states, forwards a sender's request to the receiver, and thereafter, relays the receiver's reply to the original sender. This is realized by executing the schema. A CM has the ability to associate messages with a conversation policy for the purpose of (1) determining how to respond to a received message; (2) understanding what a response is from a send message; (3) determining when to stop and resume a conversation; and (4) exception handling, or fault recovering.

A CM has five main components: Schema class library; CPN engine; Active schemata; "Goal-task-schema" inference module; and Schema execution module, an agent naming mechanism, and an I/O module. A local CM in a domain is responsible for managing a conversation of a set of agents to ensure its *local* consistency and coherency. The *global* consistency and coherency are the task of a higher-level CM that manages communication among agents related to different CMs. The organization of CMs forms a hierarchy (see Figure 1).

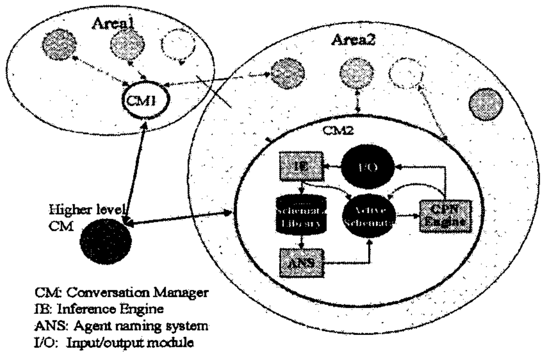

Figure1: Conversation managers

Under the control and coordination of CMs, the agents participating in a conversation located on TOP layer cooperate, negotiate, interact, but they are discharged from resolving communication problems.

4. CONVERSATION SCHEMATA

We propose a concept, conversation schema by considering:

First, a schema is designed and used to guide how several messages can be connected and often includes steps by different speakers. Second, a conversation always centers on one or more conversation *topics*. Topics can be described by a set of quantitative and/or qualitative augments. Next, a schema must identify agent classes providing some desired services. Their instances (agents) will participate in the conversation. Moreover, individual "communicative" acts can be then aggregated to create more complex structures. These communicative acts and structures can be used to describe the (goal-directed) collective behaviors of agents. Lastly, a schema can be thought of as a set of *activities* of two or more agents in which information is transmitted. The ultimate result of a conversation is a change of state in the participants. Depending on the components of state of the particular agents, this could be a change in the agent's knowledge, goals, or other attributes. Therefore, it should identify the internal *state* changes, and the external information exchange. Also, it can have optional *steps* that are chosen based on a particular situation and *variables* represented in *topics* that allow them to be adapted.

5. IMPLEMENTATION

By introducing CMs into our agent system architecture, agent-agent conversation problem is converted to agent-CM interaction problem. A schema should contain: data structure to model synchronized variables and so on; facilities to manage (re-direct) messages; mechanism to support negotiation, data exchange and dissemination; mechanism to plan, schedule, and control actions in a task; facilities to monitor the status of a distributed task.

To verify the correctness, completeness, and consistency of schemata, a schema can be organized into a Colored Petri net (CPN) [4], which represents three types of knowledge: (1) Descriptive and fact (goals, tasks, topics, states) knowledge can be represented by knowledge annotations of *color sets* in *places*; (2) Rule knowledge (actions) of conversation can be represented by corresponding knowledge annotations of *transitions*; (3) Control knowledge can be represented by *arc functions* and *flow-controls*, which are meta-rules that select one of several conflicting transitions as the next firing transition in a conflict resolution. If all firing conditions of a transition are satisfied, the transition can fire and the conversation can be conducted.

The objective of CPN graph simulation is to visualize the CPN graph and its executive process. The number, color, and distribution of tokens in places denote the dynamic system states in the executive processes. Any firing sequence which goes from an initial place to a terminal place represents a sequence of messages corresponding to a *complete conversation* among the participating agents. Therefore, the whole CPN of a schema conveys the set of all possible message sequences in which an agent can be involved.

A schema class library of a CM consists of a set of CPN-based schema classes. A CPN-based schema class is a template to construct a schema instance in which the net is converted into a set of "if –then" rules. The instantiation of a schema class implies the instantiation of the net structure and agent classes. For example, a sender mustn't name the addresses of the recipients in a schema class. The addresses of the recipients will be determined by an agent naming mechanism. Once a schema execution reaches an ending state, the schema instance will be destroyed immediately. Schema classes are used to structure a conceptual schema into modules by encapsulation and inheritance. To make schema class reusable and flexible, synchronization constraints are separated from the internal part of each schema class. The behavior of a subclass is obtained from the inherited CPN by adding new transitions and adding new state places, or providing more specific details about them.

Schema execution is a process of "if-then" rule utilization via transition firing, token passing, and state place changing in the corresponding CPN. It keeps track of the current conversation by receiving and analyzing messages from other agents participating in the conversation, and speaks to different agents depending on the current state by sending out messages.

6. CONCLUSION

We have proposed and developed schema-based conversation modeling and conversation manager-based conversation enabling mechanism. The initial experimental results have proved that this approach has the following distinctive advantages: first, it ensures consistency and coherency of agent communication and cooperation. Second, it improves effectiveness of agent collaboration. Third, it reduces complexity of implementation by constructing conversation managers that separate the description of common agents functionality from that of communication and synchronization. As a future research topic, we will add the security and trust into the schema-based conversation management to realize secure and efficient agent communication.

7. REFERENCES

[1]. Lin, F., Norrie, D. H., Shen, W., and Kremer R., "Schema-based Approach to Specifying Conversation Policies" *Agent '99 Workshop on SICP*, Seattle, WA, 1999

[2] Greaves, M. and J. M. Bradshaw (eds.), SICP-99, 2rd Int. Conf. in Autonomous Agents'99, Seattle, WA, May 1-5, 1999

[3] Jamali, N., P. Thati, and Gul A. Agha, An Actor-based Architecture for Customizing and Controlling Agent Ensembles, IEEE Intelligent Systems & their applications, Vol. 14, No. 2, 1999, pp.38-44

[4] Jensen, K. (1992) *Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, vol. 1, Springer-Verlag.