In Proceedings of the Twelfth International Conference on Computers in Education (ICCE 2004), Melbourne, Australia, pp. 293–302.

Designing Interaction Models in a Multiparty 3D Learning Environment

Liu Yi, Yam San Chee
National University of Singapore
liuyi@comp.nus.edu.sg, cheeys@comp.nus.edu.sg

Abstract: *The technology of virtual 3D learning environment creates the opportunity for virtual human interactions. These interactions should enrich learning experience and improve learning quality. However, inappropriate interactions often confuse users about the real underlying learning goal, instead of achieving effective and efficient learning. This situation becomes more serious in a learning environment which contains multiple agents and users. Thus, there is a strong motivation to design a robust interaction model to guide collaboration activities effectively. In this paper, a group interaction pattern approach is introduced to improve the quality of collaboration activities, hence achieves effective learning. A planning system integrated with interaction controller is purposed to guide the learning process. Our discussion focuses on how the design of interaction models flexibly supports task-oriented scenarios consisting of multiple agents and users, and how the interaction patterns enhance learning. In the later part, a Newtonian physics 3D learning environment is presented to illustrate the interaction among agents and users in a virtual pedagogical scenario based on our design.*

Keywords: Pedagogical Agents, Multiparty Interactions, Multi-agent Communication

1 Introduction

Immersive virtual worlds are increasingly favored as a computer-mediated channel for human interaction and communication. These worlds present a rich and interactive environment for users to engage in. Users can act on objects in the world as well as interact and converse with one another. Realistic three-dimensional representations of other users in the world create an enhanced sense of social co-presence. Users can benefit when such environments are augmented with believable virtual agents. For instance, they can be aided in task performance in a very natural, social way. In the domain of education, several well known pedagogical agents have been developed. (e.g. Rickel & Johnson, 1997)

Technology creates opportunities for innovation in pursuit of supporting computer-mediated forms of collaborative learning. It is possible to create multi-agent multi-user learning environments, thus fostering student learning in a more social setting. When users and agents work on a learning problem cooperatively, shared social interaction can serve as an instructional function. However, when the number of users and agents increases, the overall interaction in virtual environments becomes complex and hard to manage. Systems allowing free form interactions help users feel the believability of virtual learning experience, but also make them puzzled about the learning goal and process due to the lack of guide. On the other hand, if systems restrict or pre-determine most of interactions, it loses the flexibility of system design and user learning information which can be revealed from rich interactions. These considerations make it crucial to implement effective interactions in a virtual learning world. Here we choose to design a task structure based on group interaction pattern to model and control collaboration activities in a virtual learning environment. This pattern approach enables the separation of interaction design from task content design as well as the whole system design, thus system designer can have more flexibility to achieve effective multiparty interactions. Another advantage of adopting a pattern approach is the reusability. System designer can easily reuse or modify these interaction patterns when they need to change the scenario. To get effective interaction patterns, we can abstract them from real life successful tutoring scenarios. We tend to use them as implicit interaction guidelines for users and agents to follow during task execution, hence a lot of learning time and effort can be saved.

However, there are a few challenges when integrating interaction patterns into a task oriented multi-agent multi-user system. First, the use of interaction patterns should be flexible. Interaction patterns only serve as guidelines and should not prevent users taking the initiative on their own during interactions. Sometimes user behaviors may reflect misconceptions or errors, so agents also need to help users to overcome problems before continuing the interaction pattern. Second, since most interactions may involve more than one computer agents and users, the design of interaction should be generic enough to support any number of agents and users. The agents' roles in an interaction pattern should also be decided at the runtime depending on the agents' availability, expertise and preference. Lastly, the task structure based on the design of interaction patterns

should work consistently with other components in the agent architecture such as communication or planning module.

In this paper, we present a design to achieve the flexibility and effectiveness of multiparty interaction in a virtual learning environment. Section 2 gives a brief research background. Section 3 introduces the task structure based on interaction patterns. Section 4 explains how the agent architecture integrates the interaction model. Section 5 discusses how our design solves the potential problems. Section 6 illustrates our interaction model in the context of multi-agent communication. Section 7 presents a Newtonian physics virtual learning system which is built based on our design of interaction model. Section 8 concludes the paper.

2 Background

A lot of one-agent with one-user pedagogical system has been developed successfully. Steve (Rickel & Johnson, 1997), a virtual instructor for machine operation training, explains or demonstrates tasks step by step for learners. Whizlow (Lester et al., 1999), a tutor in a 3D virtual CPU City, helps user improve understanding by correcting users misconception. Although effective, the interactions happened in these systems are relatively simple. One to one interaction can not reflect the richness of interactions in a real social environment. Realizing this limitation, some researches have been working on the multiparty interactions for virtual environments. Virtual mission rehearsal exercise project (Traum & Rickel, 2002) implements a six layer architecture for supporting multiparty dialog including participants, turn, initiative, grounding, topic, and rhetorical. It focuses on the issues that arise with face to face communication during multiparty interactions. To analyze the effective way for multiparty interaction, some experiments have been carried out. Padilha (Padilha & Carletta, 2002) did a simulation of small group discussion and proposes some rule based computational models for multi-party interactions. Suh (Suh, 2001) also presents a taxonomy of group interaction patterns after conducting some experiments on evaluating students' collaborative knowledge construction process.

3 Design of Structure

We use interaction patterns to manage the group interactions for task orientated learning environment systems. A three-level topology structure is constructed to allow flexible task execution under certain efficiency criteria.

Each task consists of *Topic* layer, *Interacting Function* layer, and *Interaction Pattern* layer. *Topic* layer contains the task description, the conditions for achieving different stages of the task, the ordering constraints with other tasks, the procedure information such as what tools are used by agents during this task as well as other domain related information such as common knowledge misconception. *Interaction Function* denotes sophisticated pedagogical expertise which agents process in a tutoring domain such as step by step demo or explanation. Interaction patterns describe basic turn taking information for some primitive interaction scenarios (Tab 1). We create fifteen interaction patterns for the tutoring scenario based on five main categories.

Categories	**Interaction Patterns**
Social Interaction	Initiate topic, Invite User, Leave topic, Terminate topic and Greet
Understanding, Explanation Interaction	Provide information, Q & A, Knowledge linking, Comparing theorem
Collaboration Interaction	Integration, Agreement, Suggestion
Miscellaneous Interaction	Conflict, Giving example
Supervising Interaction	Give feedback

Table1: Interaction Patterns

The task execution follows the terminal nodes of the hierarchical tree with the ordering constraints. A terminal node is either an *interaction function* or an *interaction pattern* (Fig 1). The content of the lower layer node is partially determined by its upper layer node. E.g. To execute an interaction pattern called "provide information", the interaction pattern retrieves the description from its parent node which is an interaction function called "demo". Since "demo" is still not the root, the root node Topic needs to be referenced for further detailed information. In this example, the *interaction pattern* designs the way to "provide information". It tells agents what are the desired turn taking behaviors. This information will help agents decide and evaluate

everyone's behaviors. The *interaction function* "demo" restricts the type of the information to provide. Due to this restriction, the interaction pattern only provides information relating to a demo such as the information of steps in the demo. The *Topic* layer determines the detailed *content* of the information such as which demo should be illustrated.

As we said before, an interaction pattern regulates the turning takings during a multiparty interaction. Fig 2 describes a flow diagram for an interaction pattern called "knowledge linking". The agent initiates the interaction by describing two related problems, followed by either a group of users' discussion or a single user's conclusion. The interaction pattern finally ends with some feedbacks given by the agent. The benefit of having

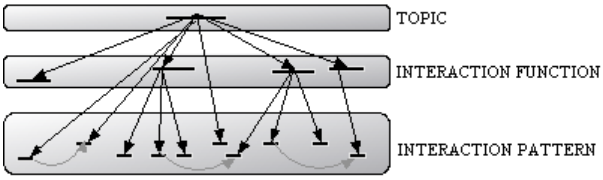

Figure 1: Task Topology

such an interaction pattern is to achieve an efficient interaction style among multiple agents and users during learning.

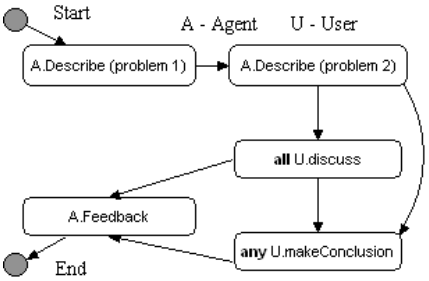

Figure 2: Interaction pattern definition for "knowledge linking"

An interaction pattern is triggered when some pre-conditions are met (Tab 2). When several different interaction patterns are satisfied at the same time, agents choose the interaction pattern with the highest priority. A higher priority interaction pattern also preempts the executing interaction pattern when its precondition is satisfied. The suspended lower priority interaction pattern does not continue until the higher priority one has finished.

Interaction Pattern	**Pre Condition**	**Priority**
Invite user	During conversation, a new user comes	1
Greet	New user/agent comes, or leaves	1
Provide information	There is some unknown knowledge for use, and it is not introduced before	2
Agreement	There is at least one user have correct answer, but not all users agree with him	2
Suggestion	User does not know what to do	2
…	…	…

Table 2: Preconditions and priorities of Interaction Patterns

4 Integration with Agent Architecture

To integrate the design of interaction model into the agent architecture, let us review some important components an agent requires (Fig 3). First, an agent needs the *utterance analyzer* to track and interpret the user intention via an analysis of the conversation. There should also be a *dialog model* to save all these interpreted

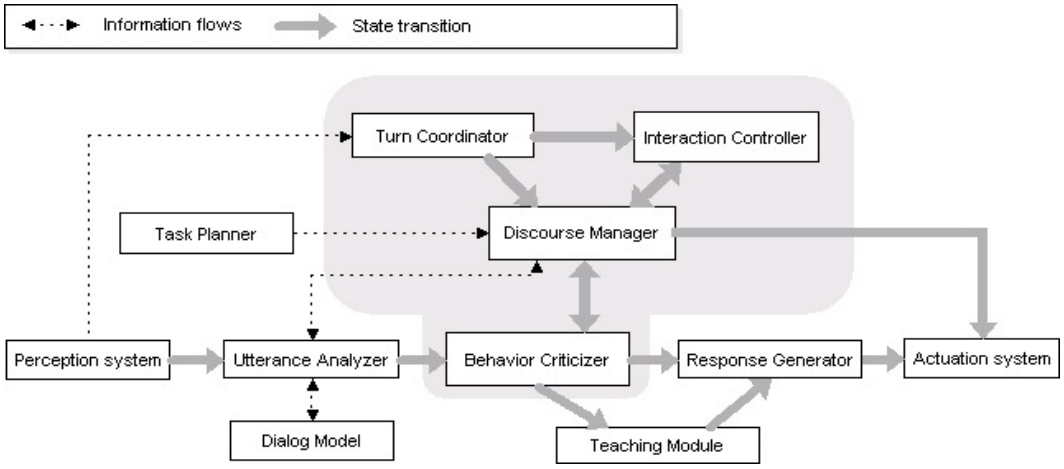

Figure 3: Agent architecture integrated with the design of interaction

information for future reference. To respond, an agent also needs a *response generator*. *Perception system* and *actuation system* help agent to receive information from environment and output animated behaviors or voices to the environment. If an agent need to process some pedagogical expertise, a teaching module is also necessary.

The grey color background area contains the system components which determine the way the agent carries out its task using interaction patterns. The central *task planner* always starts with picking a task, and then assigns the agent a task stage and passes the control to the *discourse manager*. A task stage contains the descriptions and interaction functions. When the condition for an interaction pattern is met, agents' *discourse managers* negotiate the agents' role for it. The execution of interaction pattern is managed by the *interaction controller*. Different agents' *interaction controllers* synchronize the execution progress of the interaction pattern. The *discourse manager* serves as a bridge whenever the *interaction controller* needs to inform the *actuation system* for the multimodal behavior output. *Behavior criticizer* consistently reports the user behaviors to the *discourse manager*. Whenever the *discourse manager* detects any conflict between user behaviors and the interaction pattern, the *interaction controller* pauses. The dialog then turns into a user initiated conversation. The *turn coordinator* helps agent to decide the turn takings in the dialog until time exceeds the limitation and the *interaction controller* resumes.

5 Potential Problems and Solutions

There are some potential problems we need to address when carrying out our task execution based on the design of interaction patterns: (1) how to identify user interaction type (2) what if users do not follow the interaction pattern (3) which agent to carry out an interaction pattern.

Identify the user interaction pattern: The *discourse manager* identifies user behavior through the *behavior criticizer*. The *behavior criticizer* receives both verbal and non-verbal user behavior information from the *utterance analyzer*. To recognize an interaction pattern from user non-verbal behavior, the *discourse manager* evaluates the environment state to analyze the effect of users' behavior and make a conclusion. For user verbal behaviors, the *discourse manager* checks the group intention through the *dialog model* (Since the dialog model saves the history of conversation for every user). If only single user behavior is involved, the *discourse manager* analyzes the information from the *utterance analyzer* directly.

Unexpected user behavior for the interaction pattern: There are three types of unexpected behaviors during the execution of an interaction pattern. First, the user behavior does not reveal sufficient information to be recognized as any behaviors defined in an interaction pattern. In this case, the *discourse manager* informs the *dialog model* to question the user for a confirmation or detailed explanation. Second, the user behavior is

obviously contradictory to or unrelated with the required behavior. Under this situation, the agent does not immediately force the user to behave according to the interaction pattern. *Turn coordinator* is employed at this moment to allow the agent to follow the turn setting in a user initiated dialog until the time exceeds a threshold or user behaviors become coherent with the interaction pattern again. The execution of interaction pattern then resumes. Third, users have met troubles in problem solving or displayed certain knowledge misconception. In this case, the *behavior criticizer* invokes agent's relevant teaching modules. Once the user problems have been solved, the execution of the interaction pattern resumes again.

Which agent to carry out the interaction pattern: Since the description of an interaction pattern does not specify which agent to initiate the interaction pattern, it is possible for two agents to compete for the same role. In this case, the agent's *discourse manager* first finds out the potential competitors who are nearby and free at the current moment through the *perception system*. Among all the competitors, the priority information defined in the task stage is checked and the agent with the highest priority becomes the winner to initiate the interaction pattern. The winner agent assign role for other agents if this interaction pattern requires more than one agent. Agents also need to inform each other when finishing the current turn behavior for the purpose of synchronization.

As we said before, the *turn coordinator* functions when there are some unexpected user behaviors. With the *turn coordinator*, every agent can express his turn request at any time during a conversation which does not follow an interaction pattern. The turn taking bidding scores for each agent are calculated and compared whenever the silence or the content in the dialog indicate a speaker shift may occur. The agent with the highest turn bidding score is the next speaker.

The turn bidding score = m * t * (a * f + b * d) * r

m: whether agent's name has been mentioned by the speaker. If yes, m = 1, otherwise m = 0.1. These values are subjected to be adjusted in order to achieve better results.
t: the amount of time elapsed from the moment the turn request is issued to the moment when the turn score is compared.
f: this value reflects the relation between the agent and speaker's face orientation
d: distance between the speaker and the agent
r: this value denotes how important is the utterance the agent is going to say

a, b are the coefficient values for adjusting the importance of physical position during turn coordination.

6 Agent Communication

Our design of interaction model helps virtual learning environments to model effective multiparty interactions. Multiple agents need to negotiate and inform the interaction progress to each other. In Fig 4, the dotted lines denote the inter-agent communication, and the normal lines refer to the message or control passing within one agent. Different agents' *discourse managers* communicate with each other and also with the central task planner. The discourse manager also sends and receives intra-agent communication messages to the turn coordinator and the interaction controller to manage the entire interaction process.

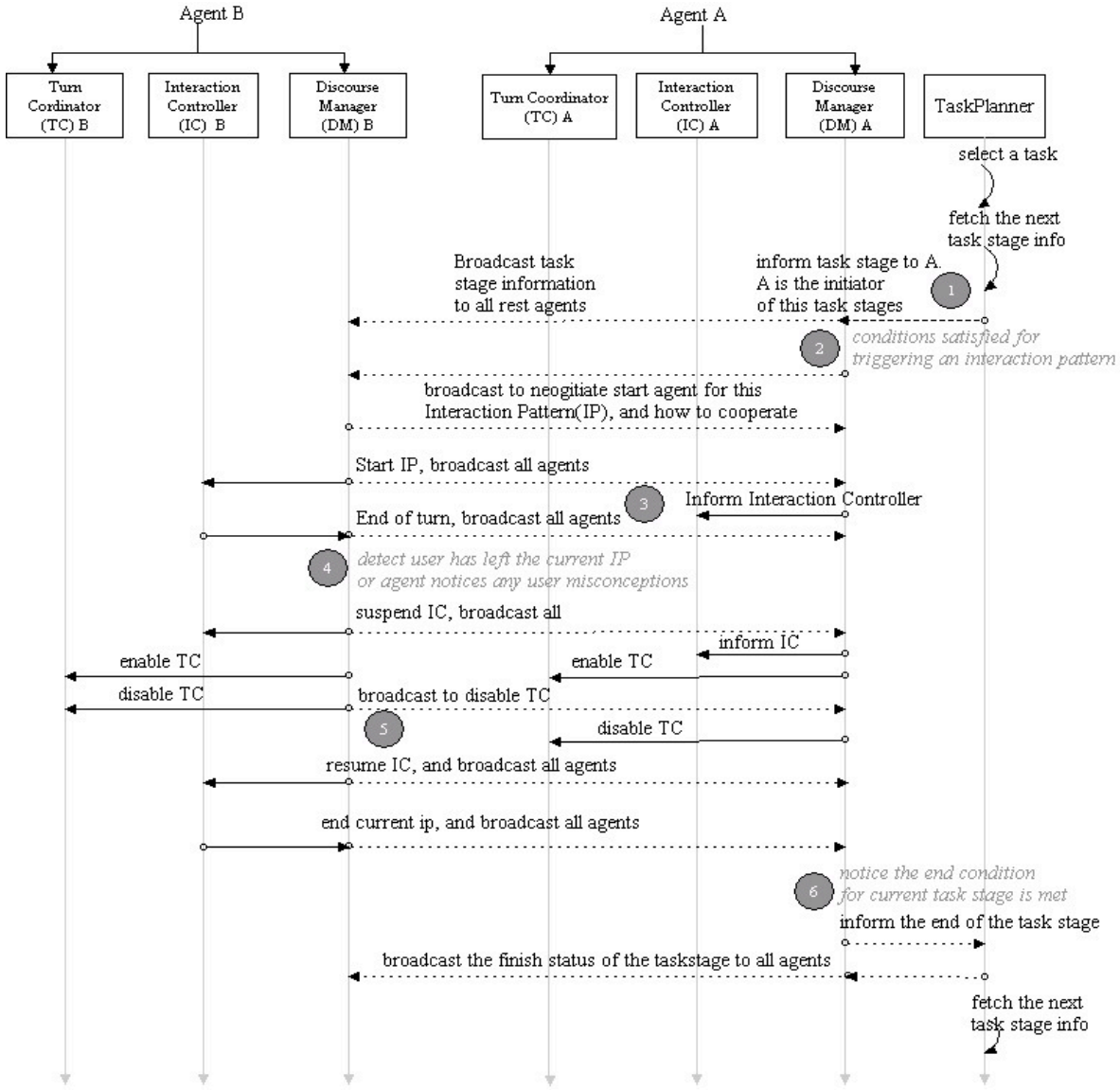

Figure 4: Agent communications for managing the entire group interaction

At Time 1, the *task planner* has just selected a task stage. It informs the details of the task stage to all agents' *discourse managers*. Agent A is selected as the initiator of this task stage according to the description defined in the task stage. At time 2, Agent A notices certain condition has been met to trigger an interaction pattern. It sends this information to Agent B's *discourse manager* to negotiate the right agent to initialize this interaction pattern. At time 3, Agent B finishes its current turn defined in the interaction pattern. It informs this to Agent A's *discourse manager* so that Agent A knows the status of the execution of the interaction pattern. At time 4, Agent B notices some unexpected user behaviors. Its discourse manager suspends the interaction controller, and enables the turn coordinator. Agent A takes the same procedure after receiving the notification from Agent B. At this moment, all agents have stopped the execution of the interaction pattern. At time 5, because the time exceeds a threshold, agent B politely asks users to perform some learning activities according to the interaction pattern. Its *discourse manager* disables the turn coordinator and resumes the interaction controller. It also informs agent A's *discourse manager* to do the same. At time 6, Agent A realizes the end condition for the task stage has been met, so it informs it to the *task planner*. The *task planner* selects the next task stage after announcing the termination status of the last task stage to all the agents.

7 Example Scenario

Based on the design of the interaction control, we have designed a multi-agent virtual physics learning environment. Our system has been designed using the design framework of C-VISions (Chee & Hooi, 2002), a socialized learning collaboration, virtual, interactive simulation. The C-VISions browser allows the user to interface with the virtual worlds.

Figure 5: Virtual Spaceship Physics Learning World

The visual scene is within a virtual spaceship (Fig 5) and the users are challenged with Newtonian law related tasks. The users can communicate with the agents through typing in a chat window. The following episodes illustrate the interactions between two agents and two users as part of the learning experience. Evaluation agent detecting and correcting the misconception, and thinking agent helps users to overcome their difficulties during task solving.

```
     Evaluation agent – Eddy, Thinking agent – Tera
     Eddy is having conversation with user Jay, another user Liu comes closer to Eddy.
     Eddy stops the conversation with Jay, and faces to Liu
1    Eddy:          Hi, Liu, would you like to join our discussion?
2    Liu:           Hi, what are you doing?
3    Eddy:          We are discussing the relation between force and velocity.
4    Liu            yah, I want to join your discussion
     Eddy continues the discussion on the previous physics problem.
     …
8    Eddy.          Which force do you think is larger, the force from spaceship to
                    acceleration machine, or vice versa?
9    Liu            I think the first one is larger.
10   Eddy           You think the force from spaceship to acceleration machine is bigger?
11   Liu            yes
12   Eddy           Can any of you tell me why do you think so?
     User Liu and Jay discussed for a while…
20   Jay            We think it is bigger because the mass of the spaceship is larger
21   Eddy           Can you do an experiment to prove it?
     Jay encounters a problem during task execution. Tera is coming to help.
30   Tera           Can I help you, Jay?
31   Jay            I have troubles to continue the task.
32   Tera:          Why discuss it with others peers first?
     Jay did not go to discuss with Liu. It seems he does not want the discussion
33   Jay            The spaceship is so big
34   Tera:          Why do you think it is big?
     …
```

At 1, when Eddy is carrying out an interaction pattern with Jay, he realizes a higher priority interaction pattern “greet” has met the condition to start. Eddy stops the conversation with Jay, greets user Liu and invites him to join the discussion. After Liu joins the conversation, Eddy resumes the previous paused interaction

pattern. At 11, Liu gives the correct answer to Eddy's question. Eddy feels Liu's answer although correct, but is too simple. Thus, Eddy starts an interaction pattern called "question and answer" to know more about Liu's understanding. According to the definition of this interaction pattern, at 12, Eddy asks either Liu or Jay to answer the question. Jay answers the question at 20. Eddy regards it as a consistent behavior to the one defined in the interaction pattern, and proceeds to the next. At 20, Eddy also realizes a user misconception since the answer from Jay is not correct. He stops the interaction pattern and asks the users to carry out a related experiment to improve their understanding. At 30, Eddy realizes Jay's difficulty in solving task, so he wants to start a "suggestion" interaction pattern. But after negotiating with Tera, he decides to let Tera to start this interaction pattern because Tera has higher preference on carrying out this interaction pattern. At 33, Jay does not follow Tera's interaction pattern. Tera does not force him back to the interaction pattern immediately; instead, he enables his turn coordinator to decide the turn taking.

8 Conclusions and Future works

This paper presents a design for achieving efficient and effective interactions in a multiparty learning environment. This design proposes an approach to manage the interaction among the agents and users as a whole. It separates the interaction design from the learning task and scenario design and gives the system designer more control on the interactions itself.

The interaction controller is built within the agent architecture to manage the entire group interaction based on some interaction patterns. Multiple agents can negotiate and synchronize the same interaction pattern with each other. An interaction pattern not only works as guidelines to improve quality of interactions among users and agents, but also allows agents to react to some unexpected user behaviors or users' learning problems.

The direction for future researches is intriguing. At the current stage of design, we focus only on the turn setting behaviors in an interaction pattern. In fact, Interaction design also can reflect the influence among agents and users' facial expression, eye contacting, or voices. To model these multimodal communication behaviors in a group pattern level helps agents to easily achieve the believability as well as improving the understanding among agents and users. This will be explored in our future research.

References

Rickel, J., & Johnson, W. L. (1997). Integrating Pedagogical Capabilities in a Virtual Environment Agent. In First International Conference on Autonomous Agents (pp. 30-88). California.

Lester, J. C., Zettlemoyer, L. S., Gregoire, J. P., & Bares:, W. H. (1999). Explanatory Lifelike Avatars: Performing User-Centered Tasks in 3D Learning Environments. In AGENTS '99. Proceedings of the Third Annual Conference on Autonomous Agents (pp. 24-31). Seattle, WA, USA.: ACM.

Traum, D., & Rickel, J. (2002). Embodied Agents for Multi-party Dialogue in Immersive Virtual Worlds. In 2nd International Conference on Autonomous Agents and Multiagent Systems (Vol. 2, pp. 766-773). Melbourne.

Padilha, E. G., & Carletta, J. (2002). A simulation of small group discussion. In 6th workshop on the semantics and pragmatics of dialogue (pp. 117-124). Edinburgh, UK.

Hee Jeon, S. (2001). A Case Study on Identifying Group Interaction Patterns of Collaborative Knowledge Construction Process. Paper presented at the 9th International Conference on Computers in Education.

Chee, Y. S., & Hooi, C. M. (2002). C-VISions: Socialized learning through collaborative, virtual, interactive simulations. In Conference on Computer Support for Collaborative Learning (pp. 687-696). Boulder, CO, USA.