In Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2004, pp. 134-140. Los Alamitos, CA: IEEE Computer Society.

Intelligent Pedagogical Agents with Multiparty Interaction Support

Yi Liu, Yam San Chee
School of Computing, National University of Singapore
Email: liuyi@comp.nus.edu.sg; cheeys@comp.nus.edu.sg

Abstract

Most current virtual world systems focus on the interaction between a single agent and the user. This simplification does not reflect the richness of a real social environment. The quantitative increment from the simple two-party interaction to a multi-party interaction does not merely increase the difficulty linearly. In fact, it leads to a much more complex situation involving multimodal communication, utterance understanding, and interaction style. Here we introduce a four-layer agent architecture with multiparty interaction support. A Newtonian law learning environment based on this agent architecture is presented and how multiple agents cooperate to improve user learning is illustrated.

The agent's interaction ability within a multiparty environment can be realized in three sections: planning and task execution, communication and understanding, as well as learning and coaching. Our entire system can be regarded as a step toward addressing and solving issues related to effective teaching in a multi-user environment within a sophisticated domain.

1. Introduction

Immersive virtual worlds are increasingly favored as a computer-mediated channel for human interaction and communication. Users can act on objects in the world as well as interact with one another. They can also benefit when virtual 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 [1] [2]. Most of these agents operate within a one-to-one tutoring scenario, and their effectiveness has been demonstrated. User learning gains in such dedicated tutoring settings are usually superior to what is achieved using traditional one-to-many teaching in the real world. Technology creates opportunities for innovation in pursuit of supporting computer-mediated forms of collaborative learning. It is possible to create multi-agent single user as well as multi-agent multi-user learning environments, thus fostering student learning in a more social setting. The inclusion of multiple agents allows the designer of a learning environment to engender multiple approaches to solving a problem and for student to appreciate multiple, often diverse, perspectives on an issue. However, several challenges arise when we seek to enlarge the interaction space to one that includes multiple users and multiple agents. First, the functional role of each agent needs to be carefully designed so as to achieve complementarity with just the right amount of overlap and redundancy. Second, interaction between all participants, both real and virtual, in the learning environment must be intelligently handled so that learning and coaching processes unfold in a natural and effective manner. Third, the modeling of student learning needs to be characterized and managed at both the level of the individual as well as that of the group. A flexible agent architecture is essential to create a virtual learning environment that responds dynamically to the situation faced "on the ground."

In the multi-agent system that we are developing, we use Newtonian physics as the learning domain and natural language (spoken and typed) as the form of human-computer interaction. Prior research has revealed that fundamental misconceptions relating to Newtonian physics are deeply-entrenched and widespread. It has proven to be difficult to shift such misunderstandings because of the strong interplay between knowledge, experience, and beliefs. The use of natural language as the basis of interaction between users and machines has the advantages of naturalness and enhanced ease of communication. However, making sense of the goals, intentions, and beliefs of students is hard.

In this paper, we present the design of a multi-agent, multi-user learning environment for studying Newtonian physics. Section 2 reviews the relevant background to the research. Section 3 describes the agent architecture. Sections 4 to 6 of the paper address specific aspects of the architecture, namely, utterance understanding, multiparty interaction, and the pedagogical competence of agents. Section 7 outlines the system in terms of the high-level multi-agent, multi-user interaction. Section 8 presents an example scenario of using the learning environment, and Section 9 concludes the paper.

2. Research Background

To achieve efficiency and effectiveness in an interactive tutoring scenario, an agent needs excellent understanding ability. It facilitates the flow of communication and enhances the ease of use. Elva [3], an embodied virtual museum tour guide, understands users by classifying speech acts of input text. For other input formats such as gestures, Rea [4], a virtual estate seller, applies a Hidden Markov Model for further interpretation. For effectiveness in tutoring, an agent needs a robust task

planning system to maintain a well-organized tutoring process. Steve [1], a virtual instructor for training operators, adopts a hierarchical structure for task representation. He is able to guide, explain, or demonstrate tasks step by step according to the order constraints between two steps. For pedagogical aspects, a misconception detector is indispensable for enhancing user learning. Whizlow [2], a virtual tutor in a 3D CPU City, uses a misconception detector, classifier and corrector to help users. Agents also need to possess sophisticated social interaction knowledge when facing multiple users and other agents in a virtual environment. Traum and Rickel's virtual mission rehearsal exercise project [5] implements a six layer architecture for supporting multiparty dialog considering participants, turn, initiative, and grounding.

3. Agent Architecture

An agent is intelligent by virtue of its ability to acquire and apply knowledge. We have designed a four-layer agent architecture for this purpose (see Figure 1). From top to bottom, these layers achieve the agent intelligence in terms of task fulfilling, social communication, pedagogical intelligence and adaptive ability.

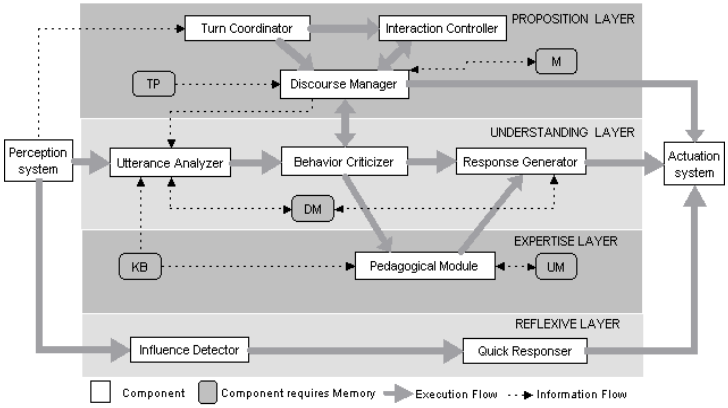

Figure 1. A Four Layer Intelligent Agent Architecture **TP**: Task Planner, **M**: Memory, **DM**: Dialog Model, **KB**: Knowledge Base, **UM**: User Model

The *proposition layer* determines the way the agent carries out his task. A *task planner* first assigns the agent a task then passes control to the *discourse manager*. The *discourse manager* decides the agent's role for the current task by referring to the agent's *memory* module. This role information helps the *discourse manager* determine an interaction pattern for the *interaction controller*. Different agent *interaction controllers* negotiate and synchronize a common interaction pattern. An interaction pattern is defined as a set of primitive interactive behaviors among agents and users in a dialog. The *discourse manager* serves as a bridge whenever the *interaction controller* needs to inform the *actuation system* for the multimodal behavior output. When the *discourse manager* detects any user behaviors that conflict with the current interaction pattern, the *interaction controller* pauses. The *turn coordinator* is invoked to help agent decide on the turn taking in the dialog.

The *understanding layer* helps the agent determine the user's intention. The *utterance analyzer* tracks a user's intention via four modules: (1) a *speech act classifier* categorizes the user's speech; (2) an *ambiguity resolver* tries to achieve grounding in a dialog by cooperating with a *dialog model* which memorizes and manages all the dialog states; (3) an *intention capturer* differentiates between listeners' roles and identifies the implicit intention in a speech act; (4) a *behavior analyzer* infers the user's intention by referring to a series of previous actions. The *discourse manager* also passes the current task information to the *utterance analyzer* for further interpretation. The *utterance analyzer* transfers the determined utterance to the *behavior criticizer* to identify user misconceptions or errors. Finally, the *response generator* engenders a response and control passes to the *actuation system*.

The *expertise layer* endows the agent with pedagogical intelligence. The *behavior criticizer* classifies user problems into errors, misconceptions, or thinking difficulties and passes the result to the *pedagogical module.* After this, different agents with their respective pedagogical abilities solve the user problems with the help of a *user model*. The *user model*, used as a reference, maintains each individual's learning status. The *pedagogical module* passes control to the *response generator* whenever feedback is required.

The *reflexive layer* provides the agent with the capacity for quick, adaptive behavior. The *influence detector* helps the agent to make decision related to joining or leaving a nearby dialog group. The *quick responser* enables the agent to gaze at or walk toward moving users to achieve high social believability.

4. Understanding and Responding

Natural language permits rich communication to take place between machines and users. This section describes how the agent interprets a user's utterance by analyzing both verbal and non-verbal user behaviors and agent understanding in the context of multiple users.

4.1 Utterance Analyzing

Utterance analyzing is divided into four modules: (1) speech act classifier, (2) ambiguity resolver, (3) intention capturer, and (4) behavior analyzer.

The *speech act classifier* uses a pattern matching method to identify a user's intention. Preparation steps including word stemming, reference resolution, stop word removal, synonyms replacement and keyword extraction are applied to facilitate pattern matching on the user's input sentence. The *speech act classifier* uses a finite state

In Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2004, pp. 134-140. Los Alamitos, CA: IEEE Computer Society.

machine to identify the pattern of an input sentence and map the pattern to one of the speech acts defined for our tutoring environment (see Table 1). Different sentence patterns may lead to the same speech act. For example, the patterns of "why", "what causes", and "what is the reason" correspond to the same speech act "question_why". At the end of the speech act categorization procedure, the user's utterance can be represented as a combination of a speech act and certain keywords.

Categories	Speech Acts
Commission	Think, Guess, Compare …
Question	Why, When, Who, Where, YesNo …
Expression	Greet, ISee, interest, Sad, Afraid, …
Request	Explain, Description, Repeat Demo, Suggest…
Declaration	Understand, comment, Summary, Conclude…

Table 1. Speech act Classification (partial)

The *ambiguity resolver* improves interpretation when the reference in a dialog cannot be figured out by the agent with high confidence during the preparation steps of the speech act classification. Names and locations are potential candidates for creating an ambiguity. The *ambiguity resolver* informs the problem to the *dialog model* and the latter notifies the *response generator* to issue a verbal request for the speaker to rephrase his utterance. Once the ambiguity is resolved, the speech act classification procedure can be carried out as usual.

The *intention capturer* recognizes inconspicuous information in a user's utterance such as implicit requests for action and the information related to listeners' roles. Some speakers may require both a verbal reply and some consequent actions. For example, the question "can you do a demo for me?" requests not only a verbal agreement but also a "demo" action. There are two ways to identify these implicit action requests in our system. First, the agent can use predefined templates to match the user's utterance to an implicit action. Second, the agent can analyze the user's previous behaviors through the *behavior analyzer* (discussed in the next paragraph). To determine the listeners' role from an utterance is complex in a multiparty environment. In a two party interaction, a listener is always also the person who will perform an action requested by the speaker. Nevertheless, in a multiparty environment, an intention like "A requests B to inform C to ask D to do something" leads to several chained consequences, and all involved agents need to perform some actions in a timely fashion. A recursive adoption is applied here to separate the header ("A request" in the example) and encapsulate the remaining requests as a whole for the next participator ("B" in the example) to proceed.

The *behavior analyzer* classifies the user's intention relying on the user's previous behaviors. It memorizes the recent behaviors for each user and applies a Hidden Markov Model to classify the user's intention based on an offline form of supervised training. This result helps the *intention capturer* detect the implicit requests for actions.

4.2 Multi-party Dialog Management

The *dialog model* manages the responses from different users in a multiparty environment. For each participator in the current conversational group, the *dialog model* maintains an individual dialog state which records the last few user utterances for future referencing. At the group level, the *dialog model* maintains a *response pool* to store all pending responses. This situation happens when multiple users express their utterances continuously one over another before the agent has the chance to become a speaker. A pruning step is applied to remove any redundancies or conflicts among all the responses in the *response pool* before the agent speaks.

The *dialog model* also recognizes the utterance or intention of a group. Models such as "discussion" and "debate" have been defined to categorize group behaviors. The agent's *discourse manager* receives this group interaction information to analyze the interaction pattern among multiple users.

5. Task-Oriented Multiparty Interaction

Our design of a task-oriented and mixed-initiative multiparty interaction is based on a sophisticated structure. This structure allows agents and users to flexibly execute tasks efficiently. It also deals with the situations when unexpected user behaviors occur.

5.1 Task Execution

Task execution is made flexibility through a graph structure implementation (see Figure 2). Each rounded rectangle denotes a group of several tasks. The arrows indicate the ordering constraints among the tasks and the groups of tasks. The task planner sequentially picks a group when executing tasks. A single task can be compulsory or optional depending on the ordering constraints. For example, at *B*, task 2 and task 3 are both compulsory but the execution ordering between them is flexible. At *C*, finishing either task 4 or task 5 is sufficient to proceed to the next group of tasks. At *D*, task 7 contains a superset knowledge over task 6, hence, finishing task 7 is adequate to advance without task 6 but not vice versa.

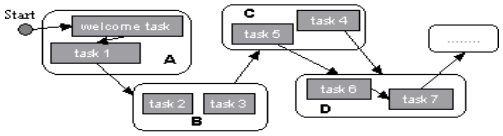

Figure 2. Task Planner

Each task is designed in terms of a three layer topology, comprising: (1) *topic layer*, (2) *interacting*

function layer, and (3) *interaction pattern layer*. The *topic layer* consists of the task description, the conditions for achieving the different stages of the task, the ordering constraints with other tasks, the procedure information such as what tools are used during the task and some common misconceptions about Newtonian laws. The *interaction function* denotes the high level pedagogical techniques such as "explanation" or "demo" which is usually some complex task in a tutoring domain. The *interaction pattern* describes basic turn taking information for multiparty scenarios. Fifteen interaction patterns have been defined for our tutoring scenario (see Table 2).

Categories	Interaction Patterns
Social	Initiate, Join, Leave, Terminate, Greet
Explanation	Provide info, Knowledge linking
Collaboration	Greet, ISee, interest, Sad, Afraid
Miscellaneous	Integration, Agreement, Suggestion

Table 2. Interaction patterns

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* (see figure 3). The content of the lower layer node is partially determined by its upper layer node. For example, 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". "demo" then references its own parent node for retrieving further detailed interaction information. In this example, the *interaction pattern* designs the way to "provide information". It informs agents what the desired turn taking behaviors are. It enables agents to evaluate users, as well as other agents' behaviors. The *interaction function* "demo" restricts the type of the information to provide so that the interaction pattern only provides information relating to a demo such as the steps needed to execute in the demo. The *topic layer* determines the detailed content of the information such as "which demo should be illustrated".

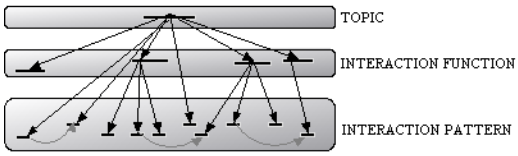

Figure 3. Hierarchical task topology

5.2 Interaction and Turn Takings

An interaction pattern regulates turning takings during a multi-party interaction. Figure 4 shows 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 feedback given by the agent. The benefit of having such an interaction pattern is to achieve an efficient interaction style among multiple agents and users during learning. Nevertheless, there are three challenges that need to be addressed when designing an interaction pattern: (1) how to identify user interaction type (2) what if users do not follow the interaction pattern, and (3) which agent to carry out an interaction pattern.

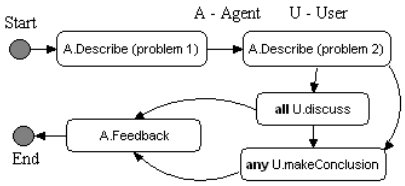

Figure 4. Interaction pattern definition for "knowledge linking"

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 draws a conclusion. For user verbal behaviors, the *discourse manager* checks the group intention through the *dialog model* which 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 a valid behavior defined in an interaction pattern. In this case, the *discourse manager* informs the *dialog model* to question the user for a confirmation or to request a more detailed explanation. Second, the user behavior is obviously contradictory to or unrelated with the required behavior. In this situation, the agent does not immediately "force" the user to behave according to the interaction pattern. Instead, the *turn coordinator* is employed to allow the agent to follow the user's turn setting in a dialog until the time for the user initiated dialog exceeds a threshold or user behaviors become coherent with the interaction pattern again. The execution of the *interaction pattern* then resumes. Third, users have encountered difficulties in problem solving or displayed certain knowledge misconceptions. In this case, the *behavior criticizer* invokes the agent's relevant teaching modules. Once the user's difficulties have been solved, the execution of the interaction pattern resumes.

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 determines

the potential competitors who are nearby and free at that point in time through the *perception system*. Among all the competitors, the priority information defined in the task is referenced, and the agent with the highest priority becomes the winner to initiate the interaction pattern. The winner agent assigns roles to the other agents if the interaction pattern requires the involvement of more than one agent. Agents also need to inform each other when they complete the current turn behavior so as to achieve synchronization of their behaviors.

As stated previously, the *turn coordinator* activates when unexpected user behaviors occur. With the *turn coordinator*, every agent can express his turn request at any time if a conversation does not follow an interaction pattern. The turn taking bidding scores for agents are calculated individually 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 is computed as $m * t * (a * f + b * d) * r$ where,

- *m*: indicates the agent's name has been mentioned by the speaker. If yes, m = 1, otherwise m = 0.1
- *t*: the amount of time elapsed from the moment of the turn request until the moment when the turn scores are compared
- *f:* the angle between the agent and the speaker's face orientation
- *d:* distance between the speaker and the agent
- *r:* importance level of the utterance the agent is going to articulate

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

6. Pedagogical Capability

The pedagogical capability is customized for agents to fulfill the goal of effective teaching. Three different agents in our system are introduced, and we illustrate how the pedagogical module supports misconception detection and correction.

6.1 Design of Pedagogical Functions

Three agents with different functional roles cohabit our virtual learning environment. All agents utilize the same architecture and they differ only in respect of their distinct *pedagogical modules* or in having different priorities for tasks execution. The modularized design makes it easy to implement characteristic pedagogical agents on the existing agent architecture. In our system, both the instruction and evaluation agent are equipped with a pedagogical module which supports misconception correction. The evaluation agent has a higher priority in helping students overcome misconceptions while the instruction agent has a higher preference for describing tasks and giving instructions. Another agent, called the thinking agent, provides scaffolding when users are not able to engage in critical thinking on their own. Inspired by UC Berkeley's Thinker Tools [6], the thinking agent enhances of user thinking ability through the use of questions, hypotheses formation, investigation, and evaluation activities. As soon as user behavior reveals that user has difficulty continuing a task, the thinking agent's *pedagogical module* is invoked to pose reflection questions for the user. If the user is still puzzled about the task, the thinking agent requests other agents to assist in multi-agent tutoring process..

6.2 Misconception Detection and Correction

The *behavior criticizer* detects whether a user's current action or utterance could lead to an error, a misconception, or a difficulty in task solving. The following situations illustrate some of the sensitive scenarios:

1. Speech act shows the users do not agree with agent
2. Speech act shows that users are making some declarations, comparisons, or conclusions.
3. Users realize a misconception themselves
4. There are missing or wrong steps during the task execution
5. Users make some contradictory statements.
6. After a long time discussion, users still have no conclusion.

Once they are detected, the *behavior criticizer* will report them to the *pedagogical module.* The misconception detection is based on the first order predicate calculus (FOPC) which describes objects, relations, properties, and events for our Newtonian laws learning domain. Some terms are listed in Table 3.

TermsType	Instances
Constant	Force, Gravity, Mass, Object, Vacuum..
Function	SpeedOf, accelerationOf, SumOf …
Predicate	Move, Drop, Rotate, Equal, Environment …

Table 3: Related FOPC terms for Newtonian laws

Thus, an expression such as "Objects with different mass drop at the same speed rate in a vacuum condition" is represented as:

$$\forall x,y \in Object, Enviroment(vacuum) \wedge Drop(x) \wedge Drop(y) \wedge (\neg Equal(massof(x), massOf(y)) \Rightarrow Equal(accelerationOf(x), accelerationOf(y)))$$

Initially, there are a few correct FOPC expressions defined for each task. They are the basic guidelines to validate the user's utterance. A misconception is identified if the user's utterance conflicts with the existing facts.

A pattern matching algorithm is used to transform a user's utterance (if the user expresses a full meaning) or both the agent and the user's expression (if the user answers a question from the agent) to a FOPC expression. If information from the user's utterance is insufficient to convert to a FOPC expression, two strategies can be utilized. The agent may choose to ask the user for a detailed explanation, or the agent may leave the current

user's utterance for future processing. If there is no contradiction between the user's utterance with correct FOPC expression in the existing facts base, this utterance is regarded as the user's correct conception towards the current topic, and it is saved. Otherwise, the user's misconception needs to be corrected.

The agent needs to ensure that the user behavior is not simply a careless mistake before attempting to correct the misconception. The agent allows the user to re-evaluate his last utterance by asking him to confirm his declaration. If the user reasserts his wrong answer, the agent then regards it as a misconception. To correct the misconception, the agent has three strategies available: *Recall*, *Relate*, and *Reflect*. *Recall* triggers the agent to search for some previous strategies when solving the same misconception for other users; *relate* refers to the related discourse plan, example, or experiment which can be used to help the user refine the understanding. *Reflect* refers to the agent's request for the user to re-evaluate his own misconceptions. During a user's reflection, if the misconception is realized and corrected by the user himself, the entire misconception correction process is completed. The corrected conception and the procedures are saved into the *user model*.

It is possible that the agent may be unable to correct the user's misconception within the time available. In this case, the misconception is saved into the *user model* and referenced when a future similar misconception is encountered. Similarity between two statements is calculated by comparing the physics keywords.

The information saved in the *user model* is stored individually for each user. But the procedures for correcting misconception can be retrieved when the agent interacts with other users as well. Overall statistics show the frequent of the misconceptions that occur. Questions relating to those "hot" misconceptions are raised by the agent more frequently as a strategy to test a user's knowledge and understanding.

7. System Overview

Focusing on the multiparty interaction, the entire system can be visualized as a combination of different interaction levels (see Figure 5).

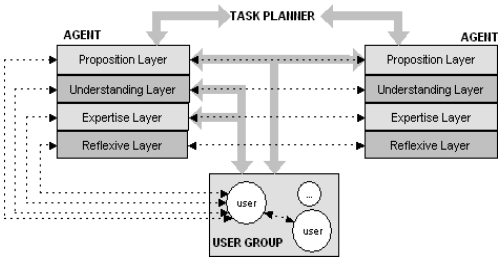

Figure 5. System level view of Multi-party Interaction

Reflexive behavior always works in terms of a two party interaction, either between agent and agent or between agent and user. Understanding behavior occurs at both the single user and multiple user level. Single agent single user understanding is the basic activity for communication in our learning environment. Single agent multiple user understanding takes place when the agent needs to identify the goal of a user group. The agent's *pedagogical module* also functions in both single user and multiple users mode. The agent corrects misconceptions for each user and the correction procedures are saved to the agent's memory for subsequent use. For task execution, the *task planner* serves as a coordinator for action taken by the multiple agents. The discourse *manager* and *interaction controller* always treat multi-agent multi-user interaction as group behavior. Similarly, turn taking is negotiated among the agents and influenced by the users. Thus, it also supports multi-agent multi-user interaction.

8. Example Scenarios

Our system has been designed using the design framework of C-VISions [7], a socialized, collaborative, virtual interactive simulation learning envrionment. The C-VISions browser allows the user to interact with the virtual worlds. Our visual environment is located a virtual spaceship (see Figure 6) and users are challenged with Newtonian laws related tasks.

Figure 6. Multi-agent Virtual Spaceship

The following simulation protocol illustrates how the interaction between three agents and two users may proceed as part of the users' learning experience.

Evaluation agent – Eddy, Thinking agent – Tera, Instruction agent – Ivan.

User Liu keeps silent and does nothing.

1 Tera: What can I do to help you?
2 Liu I really have no idea on how to start.
3 Tera: Brainstorm some related questions.

Liu still encountered difficulties. Tera calls Ivan for help.

8 Tera: Ivan, can you give Liu some hints?
9 Ivan: OK.

Ivan faces Liu.

10 Ivan: Liu, here is one suggestion. . . .
. . .
14 Ivan: Do you think this method works?
15 Liu: I think it works.
16 Ivan: Which force do you think is larger; the force from spaceship to acceleration machine, or vice versa?
17 Liu: I think the first one is larger.

18 Eddy: You think the force from spaceship to acceleration machine is bigger?
19 Liu: Yes.
20 Eddy: Can you tell me why you think so?
21 Liu: I think it is bigger because the mass of the spaceship is larger.
22 Eddy: Ivan, please help Liu do the experiment
23 Ivan: OK. Follow me, Liu.
Another user Jay enters the virtual environment.
30 Tera: Welcome to our spaceship. …
Jay walks towards Liu while Liu is talking with Ivan. Ivan notices this, greets Jay.
36 Ivan: Jay, would you mind joining us?
37 Jay: Sure.
38 Ivan: We are talking about the relation between force and mass.
39 Ivan: Can you share some idea on this?
40 Jay: Hi, Liu, nice to meet you.
41 Liu: Me too.
42 Jay: Ivan, how many tasks are there left?
43 Ivan: 6
44 Liu: That is a lot. Let's hurry up.
45 Jay: OK.
46 Liu: I think there is no relation between force and mass.
47 Jay: I dun agree. . .
. . .
53 Ivan: Can any one of you draw a conclusion?

At 1, after observing the idle behavior of the user, Tera's *behavior criticizer* identifies it as an indication of Liu's difficulty. Tera's *pedagogical model* is invoked, and the suggestions on improving the thinking process are provided. At 9, the *intention capturer* in Ivan's understanding module recognizes that utterance 8 from Tera implies both a verbal reply as well as a request for an action-to provide hints to Liu. At 16, Ivan asks a question regarding a common misconception encountered by previous users even though he knows Liu's answer is correct. At 17, a wrong answer is detected. At 18, Eddy issues a question to ascertain that the user did not respond spuriously. At 21, a misconception is identified. Eddy suggests that Liu performs a related experiment in order to correct the misconception. At 30, Tera notices Jay's arrival. She knows that Eddy has a higher priority to welcome new users. However, Tera's *perception system* detects that Eddy is at present busy with another user; so Tera's *interaction controller* assumes the role to welcome Jay. Just before 36, Ivan's *interaction controller* is carrying out a "provide information" interaction pattern. At 36, Jay's arrival interrupts the interaction pattern so that Ivan's *interaction controller* pauses the "provide information" pattern and starts a "welcome" interaction pattern. When Jay has joined the conversation at 37, Ivan's *interaction controller* resumes the previous "provide information" and considers Jay as a participator in the interaction pattern. He asks Jay to share idea with Liu. The discussion content (the relation between force and mass) is restricted by the Topic in the current task. From 40 to 42, Ivan's *utterance analyzer* notices that the users do not follow the instruction to have engaged a discussion as requested. However, Ivan does not immediately force the users back to the task related discussion. Ivan's *turn coordinator* follows the turn assignment by Jay in 43. From 46 to 52, Ivan's *dialog model* realizes that the users are back to the discussion mode. At 53, Ivan follows the instruction from his *interaction controller* to ask the users to conclude.

9. CONCLUSION

By considering multiparty interaction in the context of understanding, planning and teaching, our agents are designed to possess a high level of social and pedagogical intelligence in a multi-agents multi-users environment.

The agent's understanding ability is based on speech act classification, consideration of conversational roles, and the context of the entire dialog. Task planning and discourse management of the agent are achieved through a multi-level topology. Interaction patterns and interaction functions provide guidance for all parties to engage in an efficient and effective tutoring interaction. The pedagogical intelligence is enhanced by designing complementary roles for the multiple agents. Pedagogical ability for correcting misconceptions related to Newton's law is based on an error tolerant mechanism.

Future directions involve challenging with interactions in a larger learning environment involve more than one group. The management for inter and intra group interaction raises further difficulties that must be addressed in order to achieve effective group tutoring process. This will be explored in the future research.

[1] J. Rickel and W. L. Johnson, "Integrating Pedagogical Capabilities in a Virtual Environment Agent," in *ICAA*.1997, pp. 30-88.
[2] J. Grégoire, L. Zettlemoyer, and J. Lester, "Detecting and Correcting Misconceptions with Lifelike Avatars in 3D Learning Environments," in *AIED*. 1999, pp. 586-593.
[3] Yuan X. and Chee Y. S., "Embodied tour guide in an interactive virtual art gallery," in CyberWorld*2003*. pp. 432-439.
[4] J. Cassell, T.Bickmore, L.Campbell, H.Vilhjalmsson, and H.Yan, "More than just a pretty face: conversational protocols and the affordances of embodiment," *Knowledge-Based System*, vol. 14, pp. 55-64, 2001.
[5] D. Traum and J. Rickel, "Embodied Agents for Multi-party Dialogue in Immersive Virtual Worlds," in *AAMAS 2002*, vol. 2. Melbourne,, pp. 766-773.
[6] B. White, J. Frederiksen, T. Frederiksen, E. Eslinger, S. Loper, and A. Collins, "Inquiry Island: Affordances of a Multi-Agent Environment for Scientific Inquiry and Reflective Learning," in *Proceedings of the Fifth ICLS*, 2002.
[7] Chee Y. S. and Hooi C. M., "C-VISions: Socialized learning through collaborative, virtual, interactive simultations," in *CSCL 2002*. Boulder, CO, USA, pp. 687-696.