Enhancing Intelligent Tutoring Systems with the Agent Paradigm

Xin Bai, John B. Black
Teachers College, Columbia University Box 8, 525 W 120th Street New York, NY 10027 U.S.A.

Abstract

A cognitive framework called REflective Agent Learning environment (REAL) is developed in this study. REAL is a reusable framework that allows researchers to develop a computer-based learning environment where users can learn through passing their thoughts to some computer-based agents and observe how the agents embodying their knowledge behave as the result of their instruction. Our study benefits from the research in Intelligent Tutoring Systems and agent technologies, stressing reflection as part of the thinking processes. It focuses on the design of the framework and the testing of its usability. The external evaluation of specific implementations serves as the guidance for the future design of the REAL applications. We hope, by grounding themselves in the needs of local practice, the REAL applications can give us opportunities to understand how theoretical claims about teaching and learning can be effectively transformed into meaningful learning.

Keywords

Intelligent tutoring systems; e-learning; artificial intelligence; cognition; intelligent agents; instructional design; knowledge representation; educational games; simulation; expert systems

Introduction

Using computers to provide personalized instruction as an alternative to human tutors has drawn the attention of researchers in the fields of education, psychology, computer science, and cognitive science. Thanks to their cooperative efforts, we have Intelligent Tutoring Systems (ITSs), or Intelligent Computer-Assisted Instruction (ICAI), embodying the computer-as-tutor paradigm. In this sense, ITSs/ICAI can be considered as Pedagogical Agents, which have a set of normative teaching goals and plans for achieving these goals (e.g., teaching strategies), and associated resources in the learning environment (Thalmann, 1997). The agent paradigm now allows researchers to collaborate effectively in an effort to develop other efficient user-centered learning environments. Examples include the computer as a collaborator (Blandford, 1994; Dillenbourg & Self, 1992), the computer as a learning companion (Chan & Baskin, 1990), and the computer as a teachable agent (Biswas, 2005), to name a few.

However, the design and development of computer-based instruction systems are costly (Murray, 1999; Anderson, 1993). Adaptive and broadly applicable cognitive tools are needed to reduce the development cycle time and the level of the required expertise. This will allow for computer-based intelligent tutoring systems to become affordable learning environments in traditional classroom settings.

The primary goal of this research is to design and prototype an intelligent reflective agent situated in an educational gaming environment based upon a cognitive framework, called REAL (REflective Agent Learning Environment). Attempts were made to encourage reflective thinking through having users explicitly externalize their internal knowledge representations by instructing the agent, which, ideally, generates a sequence of behaviors analogous to those

generated by the users' imaginary worlds. It is our hope that when users begin to recognize relationships between their prior knowledge and the newly presented meanings, learning occurs, thus making the new information accessible as part of the learners' active reservoirs of knowledge. The tangible results of this research are some prototypes of the REAL applications.

Background

Our research is inspired by our observations of teenagers highly motivated to control the computer-generated characters. It would be ideal if we could create a reusable framework that allows researchers to develop such a motivational computer-based learning environment where users could learn through passing their thoughts to some computer-based agents and observe how the agents embodying their knowledge behave as the result of their instruction. These agents can use the knowledge acquired from the students to achieve intelligent behaviors. For instance, an agent can predict how events will unfold and plan accordingly. When prediction fails or conflicts with reality, students will need to revise the knowledge to help the agent get out of the problematic situation. Learning will occur when they succeed. Some of the procedures they go through will be compiled and available in their long-term memory to aid their understanding. Those steps are missing in traditional learning practices where, most of the time, students are only dealing with the initial problem state and the final outcome state based upon certain solutions. Students do not have opportunities to search through the problem space and apply operators to change the environment and observe the change during which a deeper understanding of "how thing happens" and "why it has happened" is generated. As Black and Bower (1980) stated (pages 247-248),

> In a story world,… the subject can up-date and "see" dynamic changes of characters, objects, and locations in his storyworld: he has available for inspection not only the starting and ending state of a character's motion but also intermediate points along the dynamic path. The reader can "see" that objects afford or suggest certain actions and prevent others. Furthermore, the reader's storyworld model allows him to experiment with hypothetical changes in his imagination. So he can imagine what would have happened had circumstances been different in the storyworld. Thus, readers can answer questions like, "How could this story have been different if Superman had been unable to fly?" or "If St. George hadn't had a lance, how might he have overpowered the dragon? Could he have pacified and domesticated it?"

To have an agent represent knowledge that a user possesses, we need to look at the ways humans and computer-based agents represent knowledge. Next we will give a theoretical literature review that examines the key areas of research that influence the design of the REAL cognitive framework.

Human knowledge representation

Acquiring knowledge from users and representing it is crucial in determining what they know before passing it to a computer-based agent. In cognitive science, researchers engage in studying how people obtain, store, and process knowledge. In artificial intelligence, researchers aim to design knowledge bases that programs can process to achieve human-like intelligence. We intend to build a cognitive framework that can benefit from both disciplines.

Several knowledge representation paradigms have been proposed in attempts to understand how people think. Theories like propositional representations (Kintsch, 1988), scripts (Schank & Abelson 1977), and semantic networks (Collins & Loftus, 1975) are examples of using symbols to represent cognitive knowledge in computing systems. They have one feature in common – they all represent declarative facts with a certain (true or false) value. Reasoning is achieved through propositional calculus (Boole, 1854). This enables the computer programs to represent factual objects as well as abstract concepts. The notations by Kintsch (1988) and Anderson (1983) are two of the influential ones used as psychological theories. Anderson's ACT architecture is a well developed propositional network model. In ACT, propositions are represented as entities linked to relations labeled by the roles they play. More recently, the technique of concept mapping was developed by Novak (1998) as a teaching tool for representing the knowledge of students. It has subsequently been used as a visualization tool to increase meaningful learning in the sciences and other subjects. However, reasoning with uncertain facts is not possible using these networks. Bayesian networks (Pearl, 1988) have been designed in computer programs that serve as a natural mirror of these kinds of knowledge structures in the human mind. They have formal probabilistic forms that can model human reasoning and optimal decision-making based upon probabilistic facts or uncertain knowledge.

As semantic networks, propositional networks, concept maps, and Bayesian networks can all be represented as networks with nodes chained together through propositional, functional/causal, and system relations, we have decided to categorize them as *network diagrams* in our efforts to adopt a broadly applicable knowledge format in the REAL framework.

Network diagrams, such as semantic networks, propositional networks, and concept maps, are good at representing declarative knowledge. Newell and Simon (1972) suggested that human problem solving behavior could be viewed in terms of goals, plans, and other complex mental structures as well. This knowledge of knowing how to achieve goals is procedural in nature. To accomplish a goal, certain operations need to be carried out based upon certain conditions. A goal is achieved through using *if-then* rules called production systems. A complex goal can be broken down into several smaller pieces of sub-goals. The SOAR architecture (Newell, 1990, 1972; Laird, Newell, & Rosenbloom, 1987) is one of the production systems using a goal-orientated approach. Anderson (1983) states that there is evidence to suggest that knowledge is also stored in the brain in the form of *if-then* production rules that act on declarative propositions. Therefore, the REAL framework needs to allow for the construction of production rules as another type of knowledge representation.

But thinking is more than the manipulation of symbols by rules. Black (1992, 2007) states that mental images are a type of knowledge that exists in human cognition. In his SaW cognitive model, he proposed "knowledge can be represented in symbolic level and deeper abstract level; imaginary world, in abstract level, makes it complete for representing human cognition, together with propositional networks, production systems and mental images." This model becomes the basis for the knowledge representations in the REAL cognitive framework.

Agent knowledge representation in video games

In video games like Sims (http://thesims.ea.com), Second Life (http://secondlife.com), or World of Warcraft (www.worldofwarcraft.com), the characters are controlled by the users who can dress them, feed them, rest them, entertain them, clean them, and engage them in social activities. For instance, in Warcraft, people control a character avatar to explore the landscape,

fight enemies, and interact with other players, all achieved through users' immediate commands using either the classic "point and click" control or the analog stick. Those characters themselves do not have the abilities or knowledge to plan or solve problems. In other words, they are more like passive puppets waiting for users to tell them what they need to do next. Thus, users are not able to plan ahead strategies that a character can use for dealing with specific situations or solving problems. For instance, a Sim, a computer-generated character in the Sims video game, may not have such knowledge as *a tiger is a kind of carnivore* or *energy flows from a producer to an herbivore*. Therefore, it is not able to make decisions such as *assigning enough plants to a planet to maintain a balanced energy flow*. Or, to operate a machine, a Sim may need to know a rule like this: *if a machine malfunctions and the light turns red, turn off the power immediately; otherwise, turn on the fan to cool it down.* It is not easy to develop an interface that allows a Sim to understand and use such rules.

Also, in Sims, the emphasis is more on relations between some Sims and other Sims or the Sims and their environment. The knowledge about how the systems work is not exposed for users to reflect upon. This is because, like any other games, once all the tricks are exposed, the Sims game will lose its motivational factor. Users will soon get bored in a transparent environment where there is nothing mysterious to explore. In other words, the mechanism of how a system works is purposely hidden from the user to sustain their engagement.

We would like our learning environment to be transparent as well as motivational. To make it transparent, we need to develop tools for users to construct knowledge to be passed to the agents. We also need to have tools to represent knowledge of how a system works. We believe users will benefit from the opportunities to observe the consequences of an agent's behaviors in a simulation and reflect upon the knowledge represented that drives the events.

There are other attributes of agents that implement concepts that apply more to humans: for example, notions like knowledge, belief, desire, motivation, emotion, intentions, and obligation. In REAL, the reflective agent is designed to have the potential of reflecting users' mental models and personalities, exposing their knowledge and reasoning processes, and displaying their emotions.

Integrating ITSs with simulation games

Simulation games are the exact learning environments we are looking for to expose mental models. According to Hmelo-Silver (2004), making sense of complex systems requires that a person construct a network of concepts and principles about some domain that represents key phenomena and the interrelationships among different levels of the system, whether it is macro to micro or structure to function. Thus, computer-based science simulation games offer the potential to improve students' understanding of scientific concepts (Jacobson & Kozma, 2000). Gee (2003) claims that games are "Not only pushing the creative boundaries of interactive digital media but also suggesting powerful models of next-generation interactive learning environments." Aguilera and M'endiz (2003) also reported that "in addition to stimulating motivation, video games are considered very useful in acquiring practical skills, as well as increasing perception and stimulation and developing skills in problem-solving, strategy assessment, media and tools organization and obtaining intelligent answers. Of all the games available, simulations stand out for their enormous educational potential."

However, empirical studies have also shown that, although educational games are usually highly engaging, they often do not trigger the constructive reasoning necessary for learning (Conati & Zhao, 2004; Klawe, 1998). It is often possible to learn how to play a game effectively without necessarily reasoning about the underlying domain knowledge.

In this study, the potential of educational games will be further explored by integrating ITSs with simulation games, thus helping students learn effectively with the help of the intelligent tutors while maintaining a high level of engagement and motivation. With the rapid progress of computer technology, researchers have attempted to adopt artificial intelligence (AI) to develop computer-aided instruction systems. There is a strong argument in AI in education which advocates that computer-based learning systems need to adapt to the needs of learners if they are to provide for effective personalized instruction (Self, 1999). For a computer-based system to provide such an adaptive instruction, it needs to reason about the subject domain and the learner. This leads to the development of ITSs, which are characterized by the philosophy of high tutor control − the system plays the active role of domain expert, controlling the selection of task for problems, while the students are supposed to solve the problem. The knowledge base is organized in such a way that it is not only composed of the traditional components in such a format as practice-and-drill, but also contains a student model that depicts the students' cognitive capacities, which is a crucial issue in building adaptive systems. Furthermore, it has defined goals for learning, which may be the mastery of factual knowledge and procedural skills. Such a system achieves its intelligence through its ability to make pedagogical decisions about how and when to provide pedagogical assistance based on individual students needs and goals. ITSs have been proved to be highly effective at improving students' performances and increasing motivation (Anderson, Boyle, Corbett, & Lewis,1990; Koedinger, Anderson, Hadley, & Mark, 1997; Mark & Greer, 1991; Lajoie & Lesgold, 1989; Shute, Glaser, & Raghaven, 1989).

We hope, with the goal of winning a game, students will actively review instructional materials, revise game rules and appreciate just-in-time feedback, in the process of which they master the target cognitive skills. Next, we will discuss other approaches applied in REAL in an effort to facilitate learning.

Betty's Brain is a generic agent that is domain-independent. The benefit of this is that it can be applied to any knowledge domain without the cost of re-inventing the wheel. Such an approach emphasizes the learner's beliefs about the domain, but lacks the correct world view embodied in the traditional expert model of an ITS. Therefore, the system can not reason beyond the students' teaching scope. Hence students lose the opportunity of getting the "deeper" feedback that domain experts can provide. It would be helpful if we could add a correct mental model, i.e., an expert model, which could provide insightful domain-specific guidance that is beyond students' current understanding capacities.

A domain expert may not necessarily be a good teacher, who can provide "intellectual scaffolding" to help students learn and progress through the different stages of knowledge development. We need to have a pedagogical model that contains teaching strategies and essential instructions. These strategies should be tailored to students' needs. Hence, based on students' performances, instructional feedback can be given to students in different ways including immediate, delayed, informative, motivational, positive, or negative. The main purpose of this model is to reduce the differences between the expert and the novice to a minimum or none using adaptive instructional strategies.

The REAL cognitive framework and its applications

We designed the REAL cognitive framework in Figure 1, which contains the components for constructing intelligent agents in the REAL learning environment. As in a traditional ITS, the REAL framework consists of expert knowledge, instructional strategies, the student model, and human-computer interaction components. These components are modularized and represented by such agents as the reflective agent, the expert agent, the pedagogical agent, and the communication agent in REAL. They are the electronic human alternatives of the student, the domain expert, the teacher, and the collaborator.

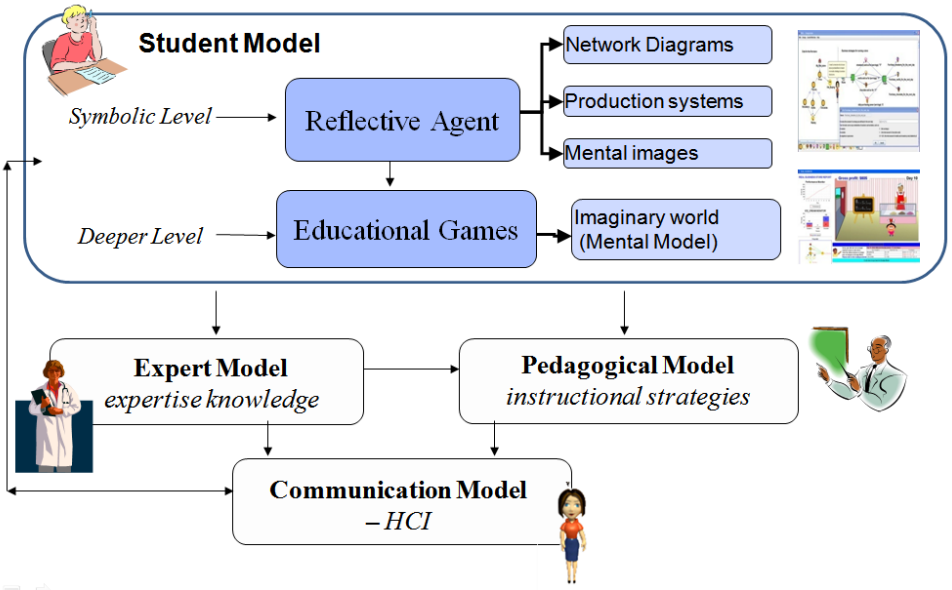

Figure1. The REAL System Architecture

The knowledge bases of these agents are acquired differently. Users pass their thoughts about how a system works to the reflective agent in the form of network diagrams, production systems, and images in Design Mode. These symbolic knowledge representations become the knowledge base of the reflective agent in the simulation, which is another form of users' knowledge representation, their imaginary world. Together, they give the reflective agent the ability to reason and make decisions. The use of the term *reflective* gives the agent two layers of meaning: it reflects the users' domain knowledge back to them (similar to looking into a glass box and seeing their own reasoning); the users reflect on that knowledge (similar to looking at a mirror, criticizing their own thinking processes).

The expert agent's knowledge base is pre-defined and domain-specific. It contains what a user should know ideally. The pedagogical agent relies upon the teaching strategies designed by the researchers to figure out when to give feedback, critique, instruction, or more learning materials. The communication agent acts as a collaborator for facilitating human-computer interactions. It is an animated character that communicates with the user via synthesized speech, facial expressions, and simple hand gestures.

The result of the compiling of these knowledge representations is the unfolding of a simulation where the reflective agent makes decisions based upon what is taught. The

pedagogical agents provide just-in-time feedback which is generated through comparing the knowledge processed by the expert agent and the reflective agent. The communication agent guides students throughout the running of the REAL application to ensure they know how to use different sets of tools.

REAL Planet as a proof of concept

The first REAL application, REAL Planet, was developed in the domain of ecology. We chose ecosystems as our initial prototype because they are complex emergent systems. Emergent phenomena result from the local interactions among many individual entities with different characteristics. The interactions result in unanticipated patterns that emerge in a macro level. Therefore, designing such dynamic emergent systems involves consideration of individual agent behaviors as well as system functions. Simulation is a powerful tool for illustrating the dynamics of such complex systems. In agent-based simulations, where each entity is modeled as an autonomous agent following its own goals, the interaction among the agents produces the emergent phenomena. As Holland (1998) stated, one of the characteristics of emergence is it occurs in systems composed of small components that obey simple laws. It helps students gain insights into situations where simple local rules can lead to emergent macro-level structures. For instance, in an ecosystem simulation, each animal's goal can be as simple as hunting for food in order to survive. Observing that the tigers do not have enough food in a simulation, students may add more rabbits to feed the tigers. The tigers will eat the rabbits following the simple *a predator goes after a prey* law. But more rabbits demand more grass as food. This may result in less grass available, thus fewer rabbits left due to the lack of food. In the end, the tigers still do not have enough food. Hence, adding more rabbits to a system where tigers are starving has not solved the problem. The emergent phenomenon generated in this process can be a teaching moment that gives students opportunities to understand how the food-chain system works in the macro-level, thus determining how problems can be better solved through changing the law (e.g. maintaining a certain amount of bio-energy in each category in the food chain) in the micro-level.

In REAL Planet, students are supposed to embody knowledge about how to design an ecological system on an alien planet, which has environmental conditions similar to that of the Earth. To do so, users pass their thoughts to the reflective agent (the alien) through building propositional networks in Design Mode. The nodes in the propositional networks represent entities in the physical world such as tigers, fungi, and grass. These entities have virtual representations as images at the bottom of the screen (Figure 2). Users construct a network through dragging these images to the main screen and connecting them with relations through arrows. Users can click on an entity to modify its properties (e.g., age, size, energy level, food preferences) or click on a relation (e.g., energy flow) to update it. When the teaching is done, users can click on Game Mode (Figure 3) at the top of the screen and launch the simulation game, where the quality of their teachings will be evaluated. The simulation is controlled by the rules of Mother Nature embedded in the expert agent. The pedagogical agent decides which event to trigger, with the goal of exposing users' missed concepts and misconceptions. For instance, students can observe how many organisms are created at the beginning of the game. As time progresses, some organisms may be consumed by others based upon the energy-flow relations defined by students. If there is a conflict between the expert agent and the reflective agent, say, on whether tigers should eat grass, the pedagogical agent may intervene, illustrating tigers have no desire to eat grass or they find it yucky. Later, users or teachers can save their propositional network in Design Mode and load them again when they would like to improve or

make changes. Anytime users switch from Game to Design Mode, their propositional networks are saved as XML files to the local computer. This allows them to roll back to an earlier session through loading in that file in Design Mode and continue from there.

We benefited from creating this prototype when applying REAL to other domains like probability. The design of an agent with complex characteristics makes it possible to represent an agent in almost any possible domain.

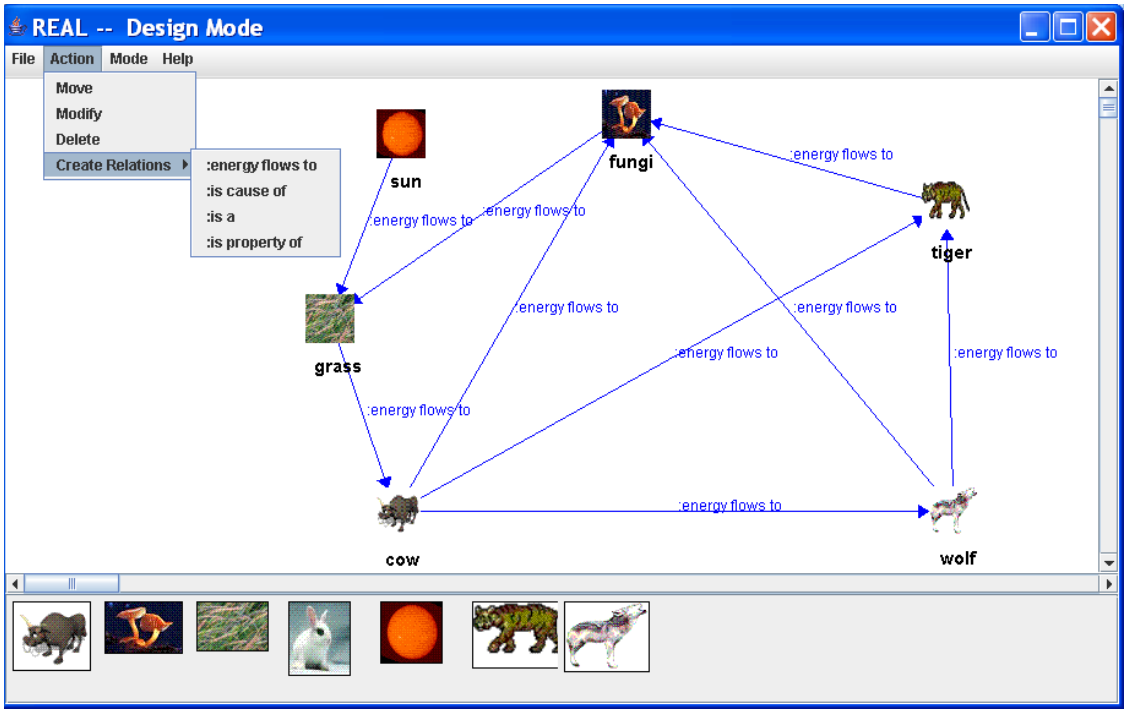

Figure 2. REAL Planet in Design Mode

(This is REAL Planet in Design Mode. Each entity is represented as an image icon and a title. Users can drag and drop entities to the canvas and choose relations from a drop-down list to connect two nodes. This screen shot displays an example network a student designed illustrating how energy flows in the food chain.)

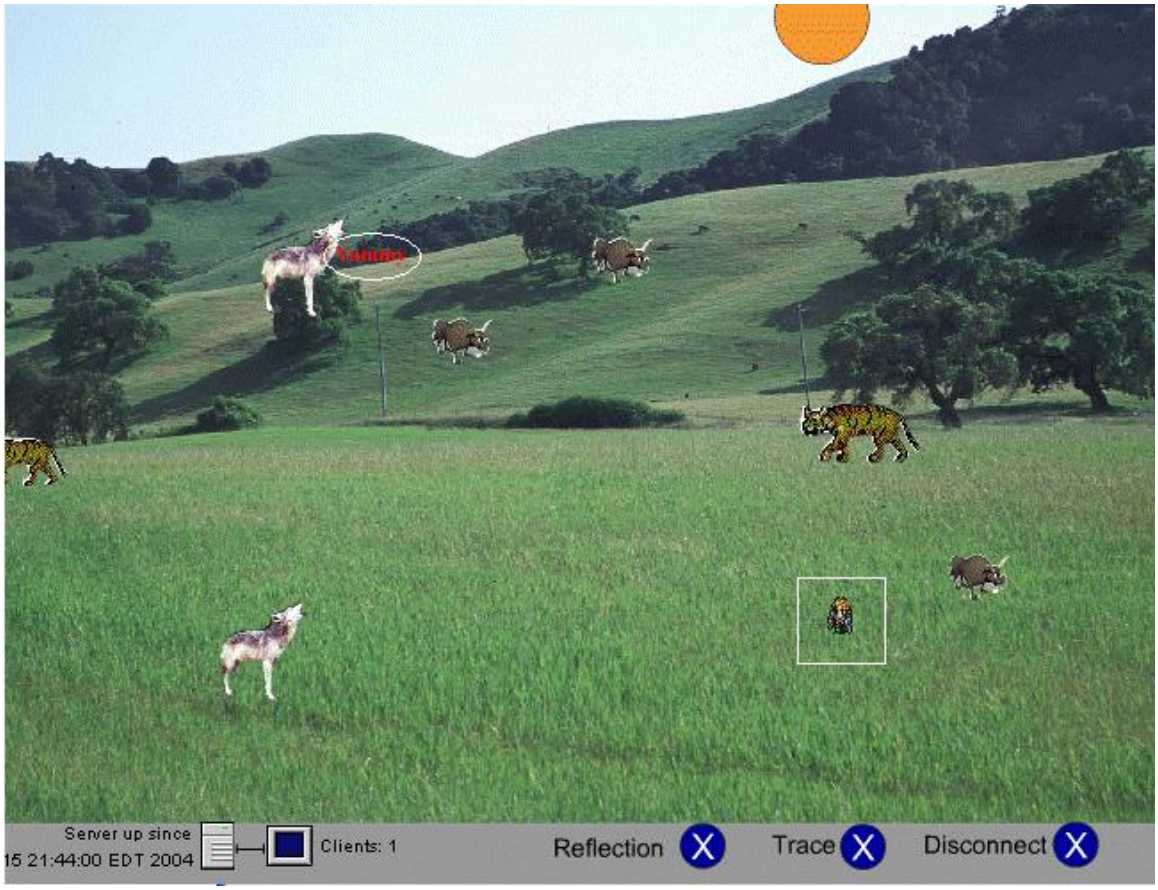

Figure 3. REAL Planet – Game Mode

(In this version of REAL Planet, each animal agent has a "biological clock" that starts to tick once the agent is born. The result of the time decay is the decreasing of energy, the increasing of age and size. The property status in each agent affects the agent's cognitive states. For instance, energy below threshold may cause an agent to pursue a goal of hunting for food. But if the health level is also extremely low, the agent might decide not to take risks to get the food that is difficult to obtain. "Yummy" appears in a wolf's thought bubble (in the white oval) when that wolf eats a cow. The reflective agent (in the white square) can walk around observing what is happening in the imagery world.)

REAL Business

The time and efforts on the design of REAL Planet paid off when we started to design the second REAL variant, REAL Business, in the domain of probability. It took two weeks of programming to build the application. The design team spent the rest of the semester putting together instructional strategies that are mostly relevant in this specific domain (Bai, Black, Vikaros, Vitale, Li, Xia, Kim, & Kang, 2007; Bai, Black, & Vitale, 2007.).

REAL Business simulates the running of an ice-cream store in order to present the probability concepts and problem solving skills in an authentic context. It can be used to facilitate the development of probabilistic reasoning in middle school students, who will have opportunities to predict the chance of an event happening, observe the data generated in the

process, identify the patterns that emerge, compare the results with those they predicted, and draw conclusions. Simulation is a good tool for emerging students in this kind of learning environment.

Unlike the knowledge representation in REAL Planet, REAL Business requires users to apply production rules in the macro-level or abstract level. It asks users to observe interactions between elements of the system in the micro-level over time, derive patterns, and provide production rules that can be used in the macro-level based upon those patterns. For example, in REAL Business, users are asked to help the reflective agent (i.e. an ice-cream store owner) figure out the proper amount of ice cream to purchase for each day. Then they are challenged further – the same rules they provide should also work for running ice cream stores in other communities with a totally different set of probabilistic data. Thus, they need to learn to go beyond the basic rules governing a local context in the micro-level and think about business strategies in the macro-level.

In REAL Business, when the program starts, a communication agent jumps to the center of the screen (Figure 4) and tells users what they are supposed to do to win a game. She explains what problem is to be solved and what users can do to solve it. When she starts to introduce Design Mode, the researchers open Design Mode. The communication agent points to the left side of the screen in Design Mode and asks users to observe some statistical data in a tree diagram. Then she flies to the right side of the screen and tells users that they need to design mathematical formulas that the store owner can use to decide how many servings to buy each day. After users provide formulas, the communication agent tells users to click on Game Mode and observe how the business goes in the simulation. When Game Mode (Figure 5) starts, users will see customers walking in the store and the store owner selling ice creams behind the counter. As customers have different flavor preferences, they may be happy or sad depending on whether they can get the flavor they want. The store owner and customers have thought bubbles illustrating what they are thinking. Users can switch back to Design Mode in the middle of the game to improve the business formulas. Data is collected, reorganized, and displayed in such formats as bar charts, line charts, or dynamic tree diagrams in Reflection Mode (Figure 5) so that users can justify and evaluate their strategies.

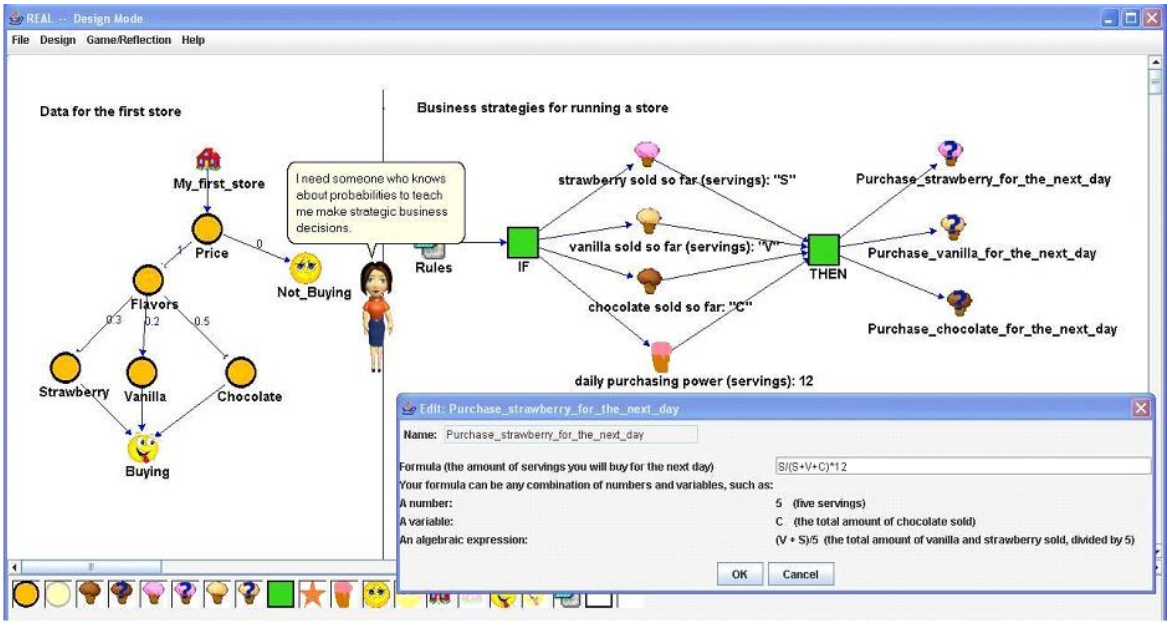

Figure 4. Design Mode in REAL Business

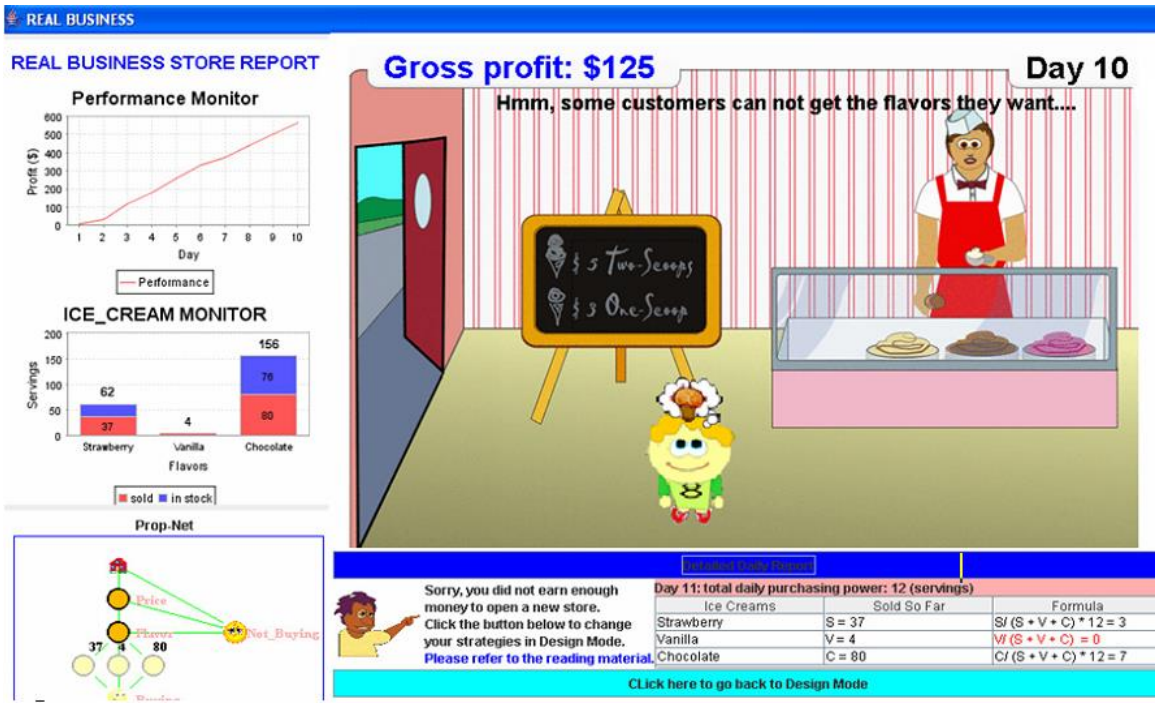

Figure 5. Game Mode (right) and Reflection Mode (left) in REAL Business

Agent design

The agent technologies allow us to embody objects with Belief-Desire-Intention (BDI) (Bratman, 1999), hence giving those otherwise abstract symbolic objects human or animal-like

decision-making abilities. The intelligent agents in REAL are entities empowered with their own thread of control. They are able to initiate actions in order to achieve goals, thus being able to behave autonomously in an environment.

Some of these agents are embodied with visual representations. They can interact with the simulation environment with a body that is represented graphically. For instance, organisms, such as animals, are displayed as cartoon-like characters wandering around hunting for food in REAL Planet. Customers and store owners are represented as animated characters that can walk and have facial expressions in REAL Business.

There are two types of agents in REAL: (1) entity agents in simulation in a subject domain, whose beliefs, goals, and intentions users must incorporate into their knowledge representations within Design Mode. Most of these agents are embodied agent; (2) cognitive agents, which represent human alternatives in a classroom. They include a reflective agent (modeling learners), expert agent, pedagogical agent and communication agent. Only some of these agents have graphical embodiment. For instance, a reflective agent (e.g., the store owner in REAL Business) can be seen serving customers in an ice cream store in REAL Business. Most of the current cognitive agents function as invisible automated agents that manage the learning processes and control the simulation games.

Agents as transparent entities in Design Mode

Upon launching a REAL application, users are presented with an overall goal, such as creating a zoo in REAL Planet or developing a successful ice cream shop in REAL Business. These goals require the user to correctly predict or direct entity behaviors in the simulated world and provide effective instruction to their reflective agents. Constructing effective knowledge representations of entity behaviors, relations, and properties in Design Mode allows the reflective agent to deal with the emergent events successfully within the game world.

The basic knowledge units in Design Mode are nodes and relations. Nodes may represent either a single biological agent or a specific state of an agent. In the case of REAL Planet, each node is a separate organism, such as fungus, grass, and tiger, which resides in the simulated world. In REAL Business, nodes represent the possible actions that the customer agent may take within the simulation.

The interaction between these various entities or states is guided by relations. In Design Mode, a relation connects two entities. Each relation is associated with some relational property, such as "is a," "energy flows to," or "35% (chance of causing)." In REAL Planet, the reflective agent needs to design the energy flow to maintain energy balance, e.g., each agent should have enough food to survive although they may be food of other agents. The energy flows from the agent as food to the agent as the food hunter. In REAL Business, the relations establish the set of behaviors that a customer will engage in while shopping. Thus a behavior can only lead to a finite set of other behaviors and is determined probabilistically.

Furthermore, each entity may maintain properties which represent either static characteristics of the entity or possible temporary states. For instance, an entity "tiger" may have such properties as "birth rate," "category," or "hunting for food." An entity "customer" may have properties like "price preference" or "flavor preference." The user typically decides how the characteristics of the entity should affect its actions. For example, a user may decide that a

customer will decide to buy ice cream if his or her preferred flavor is in stock. Thus, in Design Mode the user is able to create production rules to adjust daily purchase amount in order to make each customer agent achieve their goal of getting the ice cream.

Autonomous agents in Game Mode

When Game Mode is launched, the rule engine grabs the entities and their associated properties and relations from the XML file, converts them to shadow facts and loads them to the REAL system in the form of Java Beans. The thread in each Java Bean will immediately start a heart beat − an agent is born at this point.

The user-constructed information is stored in Java Beans. The Genuts (http://www.genuts.com) game engine picks up these Java Beans and provides them with visual representations in the form of Sprite objects. A sprite in Genuts is an animation of pictures in one image. Its goal is to provide an object with animated picture along the time with a sequence of pictures. These sprites are contained in a playfield with a collision manger handling the collision detection. The agents represented visually by these sprites have different kinds of mobility. Some agents, particularly those representing humans or other animals, are active. They move around in the simulation, while other entities, such as grass or a cashier, remain in a relatively fixed location. Sprites can be static images or animations (with a sequence of images that loop over). They can be wrapped with other sprites (which layer to unwrap depends on which behavior to display in the real time). For instance, a "customer" sprite" can be wrapped with a sprite representing a thought bubble.

Agents' reasoning abilities

Such agents embedded in a rule-based system reason through pattern matching. A rule-based system consists of facts, rules, and a reasoning engine that acts on them. Facts represent an agent's goals, beliefs, desires, and physical conditions. Rules represent procedural knowledge, instructing how an agent should act at certain conditions (patterns).

These shadow facts are represented by observable java objects in the REAL game (Figure 6). The change of the status of these java objects triggers a notification that is sent to the rule engine. The rule engine checks on the working memory and decides which patterns are matched, thus triggering the corresponding rules.

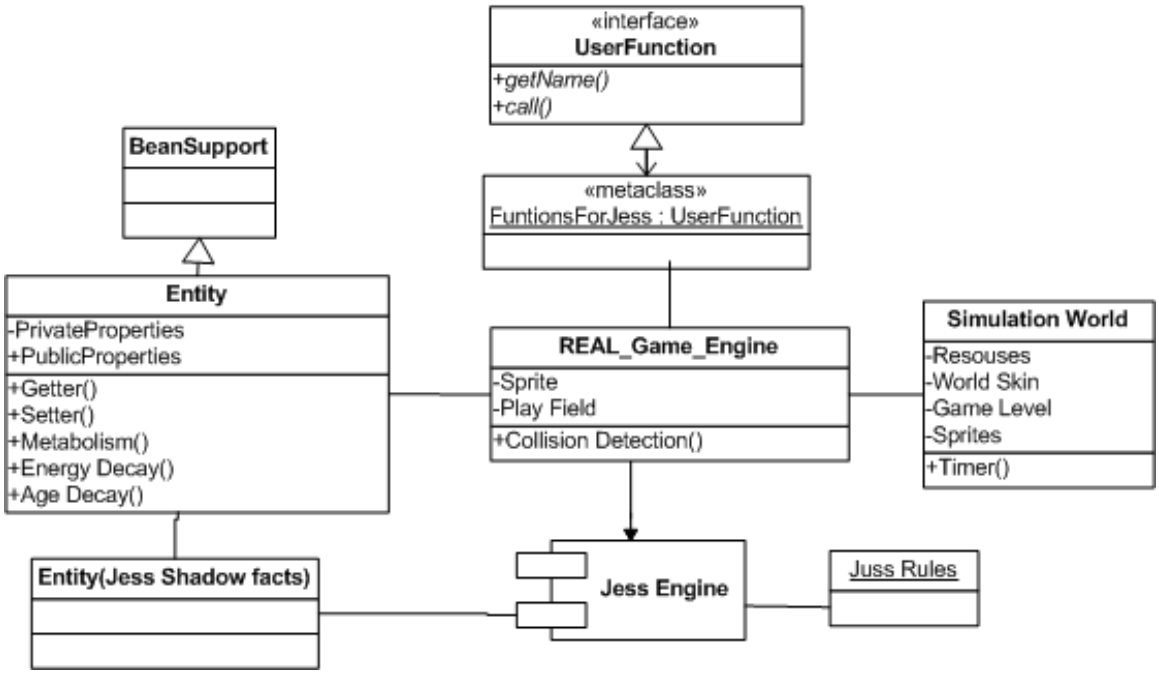

Figure 6. REAL Agent Architecture

The Jess reasoning engine monitors agents' statuses as declarative facts in a virtually parallel fashion. It will trigger events whenever it finds pattern matches in the pre-defined rules residing in the expert module or the student module, thus allowing an agent to sense, reason and act. These agents can decide whether or not to perform an action in a conflict situation according to their goals and beliefs. They are also capable of flexible (reactive, pro-active, social) behaviors (Wooldridge, 1995). For instance, if a customer has the desire to buy ice cream of a chocolate flavor, but that flavor is out of stock, this customer leaves the store without achieving his or her goal. The store owner identifies this situation and decides to order more ice cream of this flavor for the next day.

Cognitive agents

Cognitive agents in REAL are software entities that carry out some set of operations on behalf of a user in a learning environment with some degree of autonomy, and in so doing, employ some knowledge or representation of the users it represents. In REAL, cognitive agents are employed to represent the goals or desires of the user, an expert, and a teacher. Not all the cognitive agents are necessarily embodied as visual characters in a specific simulation environment. Rather, some may be designed as software agents that serve as a paradigm for developers to engage in intelligent agent design and for researchers to explore ways of using computers to embed teaching strategies, model learners, or design knowledge base in a subject domain.

Reflective agent

This agent represents the mental states of users. Built in Design Mode by the user, this agent combines the knowledge representations for the entity agents with specific rules to guide its actions. Like the entity agents, reflective agent behaviors are constructed in the form of propositional networks, production rules and mental images. In the case of REAL Business, the reflective agent is a store owner who is given the task of establishing an effective business. This requires the reflective agent to not only understand the behavior of his customers, but to act appropriately in response to these behaviors. The user is given the task of constructing production rules for the reflective agent that prompts it to buy more ice cream. To engage in this task effectively the user must incorporate information from the entity (i.e. costumer) agent's knowledge representation, including such concepts as sample size, decision trees, and probability formulas.

In Game Mode the reflective agent, along with the entity agents, will act according to the knowledge representations given by the user. Furthermore, the Game Mode itself should simulate the user's mental model for the specific scenario. Thus, the reflective agent can be viewed as the user's own embodiment within the simulation.

Pedagogical Agent

The strategies in pedagogical agent are designed by instructional designers. They may learn appropriate pedagogical practices for REAL applications in the design team as well as through pilot studies with some students. The procedure of developing pedagogical strategies requires designers to generalize common misconceptions students make and record human teacher interventions that get students back on track. These misconception patterns and the corresponding pedagogical strategies are implemented in the pedagogical module in the rule-based system that allows the pedagogical agent to decide what misconceptions to look for, what feedback to give and when. Although the developers are required to create rules for capturing common misconceptions, the process has been designed in such a way that it allows non-technical researchers to modify the feedback texts and the timing of certain feedback in a notepad with no recompilation needed. Thus the researchers can change the wording and adjust the feedback to fit his or her own style. The feedback is sent to the user in the form of thought bubbles next to the store owner.

Expert Agent

An expert agent exhibits mastery knowledge in a domain and can performance a task in an optimized way. Discrepancies between the behaviors of the reflective and expert agents can be seen as missed concepts or misconceptions in student understanding and may lead to interventions within the game to help the user overcome his or her conceptual difficulties.

In the case of REAL Planet, the expert agent incorporates the rules by which animals behave. If a user creates a behavior which does not match the expert's (e.g. a buffalo taking energy from a lion), the user will receive feedback from the pedagogical and communication agent.

In REAL Business, an expert agent possesses strategies that allow one to run a business successfully. For instance, it tends to ignore sales data on specific days, instead paying more attention to accumulated data in a longer ranger of time. The skill involved is the understanding of the concept of "sample size." If the reflective agent's profits are significantly lower than the expert agent then the user may be prompted to redesign his or her agent. The expert agent is also

capable of providing assistance through statistical charts to facilitate the discovery of meaningful patterns in data.

Communication Agent

A communication agent acts as a collaborator that facilitates user-computer interaction. It is designed using Microsoft Agent, utilizing configurable features such as speech, tonality, gesture, facial expression, gender, and screen navigation that can be combined together to emulate a human face-to-face communication act. It can also stay on top of different REAL modes and is disposable upon a user's request. For instance, in REAL Business, the communication agent introduces the REAL tools and game rules with a female voice. It asks for help from the users to design business strategies for the store owners. It can navigate to different locations in the screen and make gestures. Text in thought bubbles and a machine-generated voice are both available to users.

Evaluation

We evaluated REAL Planet using the internal evaluation method and evaluated REAL Business with both the internal and external evaluation methods. We focused on system design for REAL Planet. The time and efforts on its design paid off when we started to design REAL Business, which took two developer weeks of programming to get built. The external evaluation of REAL aims at testing the usefulness of the REAL system for the user, such as its ability to foster learning, motivate learners, and encourage scientific exploration. The evaluation of the learning outcome is not a focus in this study. Lessons derived from the process are intended to be used for the design of specific REAL applications and studies in the future. The result will influence the direction for the future research and development.

Our main focus is on the internal evaluation of system performance, addressing the question: "what is the relationship between the REAL cognitive framework and the performance of the REAL application?" More specifically, internal evaluation answers the following questions: "What agents do we need? What do these agents need to know? How do these agents do what they do based upon what they know?" This is an iterative design process. We evaluated the advantages and disadvantages of using different sets of technologies. We also tested out a few reasoning engines and game engines in our platform. We then looked at ways to standardize our design so that this framework can be developed and deployed to other local contexts easily.

Internal Evaluation

Originally, we had planned to set up REAL in a 3D virtual reality environment where users could interact with the computer-generated images, observe the happening of events in a 3D environment, and get the look and feel of the simulation in as realistic a way as possible. At the same time, we had intended to have these entities in the simulation to be controlled by a rule-based engine in the back-end, which could make critical real-time decisions regarding the control of the entities. Since the sequences of the events as well as the interactions among entities do not and should not be pre-defined, we needed to find an efficient solution that allowed the graphical game modules and the reasoning modules to collaborate seamlessly.

When we started to develop REAL Planet, we chose Virtual Reality Modeling Language (VRML) as the 3D virtual reality scripting language. VRML is a standard file format for representing 3D interactive vector graphics that can be viewed through an internet browser. In

the back-end, we used the traditional artificial intelligence programming language called LISP for the handling of the rule-based reasoning. But we found this architecture could not readily meet one of our development goals – to timely instruct what the agents should do based upon what they know. The rendering of a complex simulation like this involved tedious coding in VRML, and the resulting VRML file was big in size. This made it difficult to control a 3D environment through sending dynamic data from the rule-based system. This lack of efficient direct communication kept us from getting full control of the virtual agents.

Also, because of the lack of support of LISP from other industrial vendors, we might have had to think about the compatibility issues for every component we would have needed to add to REAL in the future stages. Most of the time, no application programming interface (API) is available to LISP. In the long run, this might have posed serious risks to REAL as it might have ended up as an obsolete application quickly due to its lack of scalability.

We decided to look for other technologies that would allow us to have efficient control of the simulation. JESS is similar to LISP. Both have rule-based engines with similar syntaxes. JESS, written in Java, has the ability to interface with other components through a Java-based adapter. This gives JESS an edge over other rule-based shells, because Java-based systems are platform independent – this allows for a program to be compiled once and run on any other platform. Later, we found we had greatly benefited from this design which makes REAL extensible and scalable.

Flash was adopted as the GUI developmental environment. It has built-in support for socket connection with the XML Socket object. This object allowed us to establish a communication channel with a socket server application. It used XML as the format for data transfer between the client and the server. For instance, when a tiger encountered a rabbit, a collision was detected. The event was sent to other interested parties through a dedicated socket channel. The corresponding feedback from those parties was also sent back to Flash through the same channel. Thus, our concern about the real-time two-way communication was solved.

We developed a game engine written in ActionScript in Flash and used a socket server to push the collision detected in the simulation back to JESS in XML. However, we had a bottleneck later when the number of entities increased beyond a threshold – a single thread looping through entities was not able to manage all the changes and the corresponding updates. The computational demand for processing these entities increased in exponential order. As a result, it slowed down the simulation. This contradicted our original goal – designing a system that is scalable. Later, we tried Director, which has a built-in 3D Havok physics engine. We found that although Flash and Director can provide fancier and more user-friendly interfaces and can communicate with other components, they are not able to handle a complex situation where each component in the virtual environment demands constant attention. Also, in our case, we had wanted the data to be processed through a rule-based engine based upon users' knowledge construction. Therefore, a front-end component like a tiger only needed to be a passive receiver of instructions for what to do next from a rule-based engine.

Finally, we decided to choose a game developmental environment that came with a built-in game engine – Genuts. Another advantage of adopting a java-based game engine is that a Java-based object (e.g., a Sprite in Genuts), once attached to a Java Bean, can be directly manipulated as a native shadow fact in the JESS rule engine. This greatly simplifies the development process. For instance, no XML files are needed for data exchange between the client and the server. Also,

once entities are defined in Design Mode, they are automatically converted to Sprites for visual representation in Game Mode and to shadow facts for knowledge representation in JESS. This approach helps streamline the development process and makes REAL a generic platform where knowledge construction, representation, and manipulation can be achieved with less programming effort.

External Evaluation of REAL Business

The external evaluation of REAL Business examined the usability of the REAL application through surveys, interviews, a pre-test, and a post-test. As the participants have had background knowledge on the basics of probability, pre-test and post-test were not used for evaluating learning outcomes. Rather, they were designed and improved by the research team to have a better estimate of the tests in terms of content and structure.

Results

The survey results in Figure 7 displays the evaluation result of the usability of REAL on the scale of 1 – 5. A scale with "Yes/No" was used in the first study session. It was converted to "4.5/1.5" and compiled into the rest of the evaluation result. Some students were not able to finish the survey due to the lack of time.

QUESTION	MEAN	SCALE	SIZE
Does playing REAL increase your interest in probability?	4.0	1 = strongly disagree 2 = disagree 3 = neutral 4 = agree 5 = strongly agree 4.5 = Yes* 1.5 = No*	20
Does REAL give you greater confidence that you can understand and use probabilities?	3.7		
Do you want to learn more about probability?	3.9		

Figure 7. The Survey Result on REAL Business

(The scale of "Yes/No" was used in the first session; it was converted to "4.5/1.5" in the other sessions)

Future trend

With our approach of designing ITS with agents embedded in simulations, it is natural to think about extending REAL to multiple-user and m-learning versions. The modularized design of the reflective agent over a common interface makes it possible for a user to develop his/her own agents in Design Mode and share it with other users in Game Mode over the internet, where each reflect agent can carry out actions representing its creator. This will allow users to observe the consequences of their actions interacting with other reflective agents in simulations. The current java-based rule-based engine and game engine allow the applications to be developed once and deployed to different platforms, including mobile platforms. The advantage of developing the REAL applications over mobile devices is that such portable devices can encourage learners to observe the emergent phenomena and collaborate with peers anywhere anytime. For instance, certain situations may trigger a communication request from a reflective agent to another reflective agent. The users can thus talk over the phone and work together to

solve a problem. Or a pedagogical agent may capture a moment when it needs to inform a reflective agent what goes wrong and what needs to be done to correct it. This can be achieved by having the pedagogical agent send text messages to a user's mobile phone with detailed pedagogical feedback. The multi-level communication among the reflective agents and the users will make REAL a rich motivational learning environment where users are willing to collaborate to solve problems with active reflection of their internal thinking processes.

Conclusion

From the design and study on the REAL applications, we found the following value for REAL as a cognitive framework. (1) The learning environment is *motivational*. Through survey and interviews, participants in our studies showed great interest in our application. Most of them indicated that the game was very fun and playing with REAL increased their interest in the target domain. (2) The REAL framework is *reusable* and *extensible*. We had a quick turn-around time in terms of application development, for instance, in the second implementation – REAL Business. The intelligent agents, which are embedded in a rule–based system hooked up to a game engine, can be easily extended for representing humans or organisms. The modules were well defined, thus the developers were able to easily jump in and work on specific areas. (3) The REAL platform encourages *collaboration* among researchers with interest in such areas as artificial intelligence, intelligent computer-aided instruction, human computer interaction, human cognition, and educational games, to name a few. It can serve as a generic platform for those researchers to embed their ideas and test out their hypotheses.

This study builds a cognitive framework for developing intelligent agents in simulation games to encourage reflective thinking and engaged learning. We believe that our cognitive framework could contribute to the much-needed ongoing effort to develop intelligent agents for simulation in a constructive learning environment.

References

Aguilera, M. and M'endiz A. (2003). Video games in education. *Computers in Entertainment*: 1(1), 1-14.

Anderson, J. R. (1993). *Rules of the Mind*. Mahwah, NJ: Lawrence Erlbaum.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge: MA, Harvard University Press.

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42, 7-49.

Bai, X., Black, J. B. & Vitale, J. (2007). REAL: Learn with the Assistance of a Reflective Agent. *Agent-Based Systems for Human Learning Conference*, Hawaii.

Bai, X., Black, J.B., Vikaros L., Vitale, J, Li, D., Xia, Q., Kim, S., Kang, S. (2007). Learning in One's Own Imaginary World. *American Educational Research Association*, Chicago

Biswas, G., Schwartz, D. L., Leelawong, K., Vye, N., & TAG-V (2005). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence* 19: 363-392.

Blandford, A. E. (1994). Teaching through collaborative problem solving. *Journal of Artificial Intelligence in Education* 5(1): 51-84.

Black, J.B. (2007) Imaginary Worlds In Gluck, M. A., Anderson, J. R , & Kosslyn, S. M. (Eds.). *Memory and Mind: A Festschrift for Gordon H. Bower.* New Jersey: Lawrence Erlbaum Associates

Black, J. B. (1992). Types of Knowledge Representation *CCT Report* New York: Teachers College, Columbia University.

Black, J.B. and Bower, G.H. (1980) Story understanding as problem-solving. *Poetics*, 9, 223-250.

Boole, George (1854). *An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities*, Macmillan, 1854/1958. New York, NY: Dover Publications,

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* 13, 4-16.

Bratman, M. E. (1999). *Intention, Plans, and Practical Reason*. CSLI Publications.

Chan, T. W. and Baskin, A. B. Eds. (1990). Learning companion systems. *Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*. Norwood, NJ, Ablex.

Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, 82, 407–428.

Conati C. and Zhao X. (2004). Building and evaluating an intelligent pedagogical agent to improve the effectiveness of an educational game. *Proceedings of IUI '04, International Conference on Intelligent User Interfaces*, Island of Madeira, Portugal, p. 6-13.

Dillenbourg, P., & Self, J.A. (1992). A computational approach to socially distributed cognition. *European Journal of Psychology of Education*, 3 (4), 353-372

Friedman-Hill, E. (2003). *Jess in Action: Rule-Based Systems in Java*, Manning Publications.

Gee, J. P. (2003). *What video games have to teach us about learning*. New York: Pal grave.

Hmelo-Silver, Merav, G. (2004). Comparing expert and novice understanding of a compex system from the perspective of structures, behaviors and functions. *Cognitive Science*, vol. 28, pages 127-138.

Holland, J. (1998). Emergence: from chaos to order. *Reading* MA: Addison-Wesley.

Jacobson, M. J., & Kozma, R. B. (2000). *Innovations in science and mathematics education: Advanced designs for technologies of learning*. Mahwah, NJ: Erlbaum.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction–integration model. *Psychological Review*, 95, 163–182.

Klawe, M. (1998.). When Does the Use of Computer Games and Other Interactive Multimedia Software Help Students Learn Mathematics? In *NCTM Standards 2000 Technology* Conference, Arlington, VA.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, 30-43.

Laird, J., Newell, Allen and Rosenbloom, Paul (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33: 1-64.

Lajoie, S.P. and Lesgold, A. (1989). Apprenticeship training in the workplace: Computer coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning*, 3, 7-28

Mark, M.A. and Greer, J.E. (1991). The VCR tutor: Evaluating instructional effectiveness. InHammond, K.J. and Gentner, D.Q. (Eds.): Proceedings of *13thAnnual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Hillsdale, NJ, 564-569.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, pp. 98-129.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA, Harvard University Press.

Newell, A. & Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Novak, J. D. (1998). *Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations*. Mahwah, NJ: Lawrence Erlbaum Associates.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann

Self, J. (1999). The defining characteristics of intelligent tutoring systems research: ITSs care, precisely. *International Journal of Artificial Intelligence in Education* (10): 350-364.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.* Hillsdale, NJ: Erlbaum.

Thalmann, D., Noser, H, Huang, Z. (1997). Autonomous virtual actors based on virtual sensors, in: Creating Personalities (Ed. R.Trappl, P. Petta), *Lecture Notes in Computer Science*, Springer Verlag, pp.25-42.

Wooldridge, M. J., N. (1995), Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review* 10 (2), 115-152.