ManyMinds: Supporting Inquiry Based Science with Multiple Intelligent Agents

Eric Eslinger

May 13, 2002

Contents

List of Figures

List of Tables

1 Introduction

Metacognitive skills are a crucial component of a successful learning career. We define metacognition as the ability to plan, monitor progress toward a goal, reflect on the quality of work, and revise the work or plan accordingly. By explicitly addressing certain metacognitive practices in classrooms, researchers have observed improved learning outcomes in both science [14] and mathematical problem solving [10]. Although these efforts were successful, they were also limited in the range of skills that could be addressed at one time and the methods used to address them due to the static nature inherent in traditional pencil-and-paper format. We wished to extend this research to include a larger range of metacognition, and address them in a more dynamic, continuous representation such as that afforded by a computerized learning environment. This paper outlines such an environment and describes pedagogical activities afforded by the system.

2 Background

2.1 Intelligent Agents

Russell and Norvig [9] define an agent as something that perceives and acts upon its environment. They subdivide the space of possible agents by complexity into three categories, reflex agents, goal-based agents, and utility-based agents. Reflex agents

do not maintain any kind of state information in their knowledgebases, they simply map perception to action. Goal-based agents reason using state information about the world and the agent's current goals. Utility-based agents include information about the relative desirability of goal states. Furthermore, agents can be categorized by the type of reasoning system they use, broadly logical or probabilistic. A logical agent uses some sort of logical reasoning system, such as propositional logic, backwards or forwards chaining problem solving, first-order logic, or planning. A probabilistic reasoning agent encodes information about itself and the world using a statistical framework such as a Bayesian network representation, and uses statistical queries to determine the likelihood of goal states given perceptions and possible actions.

ManyMinds agents use a hybrid logical reasoning system that most resembles a propositional logic system. The reasoning system is comprised of condition-action pairs that reason over perceptions, beliefs, and functions acting on perceptions and beliefs. Another useful way to describe a set of agents is using the PAGE description; percepts, actions, goals and environment [9]. ManyMinds agents perceive the work the user has entered into the artifact, judgments the user has reported about that work, and metadata such the time of last time product modification. The fundamental agent action in ManyMinds is giving advice, although they also play roles in constructing the user experience. Agents have the goal of teaching the user how to create a good product (for example a hypothesis). The ManyMinds environment is a Java based software environment.

2.2 Educational Research

A long term goal of the ThinkerTools research group has been teaching general inquiry skills to middle school science students [13, 14, 15]. This has usually been in the context of teaching the students about force and motion using the ThinkerTools microworld, but has also included other content areas such as genetics using the GenScope program [2], or inquiry into the students' own inquiry skills [8]. The ThinkerTools group has found that it is possible to directly support the learning of inquiry skills by providing a scaffolded model of inquiry process, called the Inquiry Cycle. The Inquiry Cycle is a general model of how one does inquiry, starting with Question, then moving to Hypothesize, Investigate, Analyze, Model and Evaluate. By making the process model explicit and also explicitly supporting other inquiry processes such as using representations and reasoning carefully, the ThinkerTools Inquiry curriculum has made significant improvements in students' inquiry skills.

Metacognition includes planning, reflecting on, and revising one's own cognitive activities. Planning can either be the creation of a long term "laundry list" of ideas a learner has about exactly what they are going to do or it can be an on-the-fly activity where a learner picks the best possible next step based on their current needs and abilities (such as a means-ends analysis). Reflection requires the learner to assess themselves in some way. Often this means assessing the quality of the learner's current activity against some criteria in order to inform either the planning or revising metacognitive acts. Revising can occur either on the product level, for example

attempting to improve one's hypothesis, or on the process level, attempting to improve one's hypothesizing plan, or at an even higher level, attempting to improve one's hypothesizing skills. There is another flavor of metacognition, monitoring, which is often discussed in the literature. Called executive control by Brown [1] and self-regulation by Schoenfeld [10], this kind of metacognition is the constant on-line diagnosing of cognitive activities. Where planning, reflecting and revising are discrete actions undertaken by the learner, monitoring includes all the above actions as well as a heuristic for when such actions are appropriate. Successful monitoring requires that the learner be effective at planning their work, reflecting on the quality of their work and the efficacy of their plan, and revising both work and plan when necessary.

While all four processes of metacognition are critical to successful learning, I have chosen to focus my research mainly on reflection. The work previously done by White and Frederiksen on reflective self assessment has shown a strong link between reflective skill and overall learning outcomes, and the current Inquiry Island software and activities have powerful affordances for reflective activities. Additionally, reflection engages a subset of a learner's personal epistemological theories, either implicitly or explicitly; the interaction between personal epistemology and reflective self assessment is an interesting theoretical space.

When a learner reflects on their work or knowledge, they must engage part of their personal epistemology. During reflection, a learner makes judgments about the quality of what they know and what they have created. But, as discussed by

diSessa [4], knowledge is not necessarily a uniform construct. People can easily switch among different, mutually contradictory belief systems based on their context. Hammer and Elby [6] apply this theory of knowledge in pieces to personal epistemology. Where prior research [3] has characterized epistemological beliefs as a more unified and robust theory-like construct, Hammer and Elby have shown that epistemological beliefs can vary based on the context in which they are assessed as well as the kind of knowledge that they are used on.

Because I am choosing to look at epistemology in a more situated fashion, I want to study it in places where a learners epistemology is critical to their actions. By approaching certain metacognitive activity, such as reflection and revision, as indicative of epistemological beliefs, I will explore the manner in which students' actions can be used to assess their epistemological beliefs. Because the ManyMinds software environment creates a place where students can easily describe their reflective judgments, it is a powerful tool for such a mode of assessment.

3 ManyMinds Design

The ThinkerTools group embarked on a project to provide students with a dynamic scaffold for science inquiry and metacognitive processes. To this end, an advisor-supported general inquiry environment was prototyped and tested on a group of middle-school students in an after-school setting [15, 11, 12]. The results were

very encouraging, and a Java-based implementation of the software was created as a second phase of this project [5]. This software is called ManyMinds; it includes a space for student work, tools to promote self assessment and a set of customizable advisors that help users accomplish their goals.

ManyMinds was designed to support users' metacognitive, sociocognitive and epistemological development. The kind of metacognitive development supported by the system includes reflecting and revising (through the use of reflective self assessment) as well as planning and monitoring as embodied in the advisory agents. The epistemological development of the users was seen to encompass understanding both of the nature of expertise and of oneself as a learner with goals and strategies. ManyMinds is also is an authoring tool for the creation of interactive learning environments. It can be used to create environments that support diverse pedagogical activities, from inquiry science to mathematical problem solving to critical reading of a paper. ManyMinds includes a palette of general purpose interface tools for scaffolding a product, reflective tools for supporting metacognition, and flexible, user programmable agents for providing assistance and modeling expert behavior.

The ThinkerTools group has used ManyMinds to create the Inquiry Island environment for doing inquiry science projects. Inquiry Island includes a research notebook, self assessment prompts specific to doing inquiry projects and agents that support the social, cognitive and metacognitive tasks associated with inquiry science. Sociocognitively, Inquiry Island portrays the scientific enterprise as a social

process, involving debate as well as the collaborative development, communication, and representation of scientific ideas. Finally, the support of scientific inquiry skills and reflective self assessment was seen as a direct outgrowth of the work done with the ThinkerTools Inquiry Curriculum [14]. That curriculum, using scaffolds to teach both inquiry skills and metacognitive reflection and to engage learners in scientific inquiry, was found to effectively develop the inquiry skills of high and low achieving students in urban middle school classrooms.

4 ManyMinds Software Components

The ManyMinds software has two fundamental parts, a student artifact and advisory agents. Both are typically used at the same time, but tend to serve fundamentally different teaching and learning goals. The artifact is a product scaffold that provides an interface for structuring students' work, giving them different text areas to discuss parts of their product as well as self-assessment tools to reflect on the quality of their work. The contents of the artifact and self-assessment tools are modifiable, to allow use with any imaginable subject area context. The advisory agent system runs in parallel with the artifact, giving the student feedback on their work as well as serving as a resource for assistance during any part of the process.

I built the ManyMinds software using Java, primarily for cross-platform compatibility. I knew the ManyMinds software would need support for threads, in-

terprocess communication, and a GUI; I wanted to build software that would run on both Windows and Macintosh platforms. At the time, Java seemed ideal, and I have grown into the Java platform as it expanded, utilizing new features as they became available, such as XML parsing, the collections library, and drag-n-drop gestures in the GUI. The current software is built on Java 1.3.1, and is available for download at `http://thinkertools.soe.berkeley.edu/inquiry_island/`.

4.1 Project Artifact

4.1.1 Artifact Interface

The ManyMinds artifact is where the user works on their product. It is structured around a tabbed pane interface, to which the agents can add sections and pages. The pages mostly contain text areas for the user to type into, but they can also contain tables and lists of text as well. In Figure 1, the user is in the "Hypothesize" section of the notebook and is working on a hypothesis about how bodies would behave without friction. The pages and sections can be color-coded, as shown here, with the color linked to the agent that created them; this helps tie together the task and the advisors for that task in the user's mind. Pages that contain lists can collate parts of a user's work. There can be multiple pages of hypotheses, and a single overview page that lists all the user's hypotheses. The artifact also includes a specialized tool for entering tables of data (such as during a science project) and generating charts from that data. Future development of the artifact will include tools for sketching

pictures, displaying graphics downloaded from the Internet, formatting displayed text fonts, and possibly a display mode for mathematical formulae.

4.1.2 Reflective Self Assessment Tools

Previous research showed that reflective self assessment plays an important role in the development of students' inquiry skills while they work with the ThinkerTools inquiry curriculum. While engaged in ThinkerTools activities, students were given reflective prompts that encouraged them to assess their progress regarding high-level social and cognitive goals, such as communicating effectively and reasoning carefully. In order to support this kind of reflection, the ManyMinds artifact can contain reflective self assessment "sliders". Each slider represents one criterion that can be applied to either the product or the process of inquiry. Each slider has several settings, corresponding to the possible values of a qualitative variable. For example, the "question answerability" slider ranges from "Not Rated" to "I already know the answer" to "This question would take years to research". The agents have advice for helping users understand and apply these criteria. These sliders help the user make continuous on-line judgments about their product and the process that is generating it. Each slider also has a location for the user to enter a short comment about why they set the slider the way they did.

4.1.3 Artifact Technologies

The ManyMinds artifact tracks a large amount of student data and must distribute that data to other ManyMinds applications, such as the agent software. To track that information, I built a client-server database using Java RMI. The data server stores string data about the state of ManyMinds documents and will notify clients of change in that data. Data arrays and tables are built on top of the string data, using groups of data objects. Reflective Self Assessment tools also store their information in the data model. Using remote method invocation and the standard Java event processing framework, I have created a data system that updates all clients when data changes. The data server also logs any changes to the system data, providing a convenient way to track and reconstruct user actions using the software. Currently, the data system is run locally, but it is also possible to run the server remotely, on a different machine or process from the artifact and data clients. It is also possible to connect multiple data clients to the same data server, either local or remote. This feature would allow multiple students to work concurrently on the same project using different computers, or enable future teacher support software to connect to all running ManyMinds programs and track student work remotely.

The pages in the ManyMinds artifact are dynamically generated at run-time. They are added to the artifact by ManyMinds agents, who have the page definition stored as XML files. That XML file is parsed and compiled into a page. A page can contain text areas, tables, lists and self-assessment tools as well as color and label

information. Most pages are also cloneable; for example, the number of hypothesis pages in the notebook is variable, and the page markup shows which documents should be shared across all instantiations of a page and which should be unique to that page. All instances of the hypothesis progress slider, for example, should use the same model, but each hypothesis should get its own plausibility slider.

The ManyMinds artifact communicates with the agent subsystem using an implementation of the KQML [7] agent communication language. This implementation is built around a "post office and mailbox" metaphor using Java RMI. While my implementation conforms to standards for processing and handling of KQML messages, there is no standard for the transport layer. For example, my implementation uses RMI to transport string representations of the messages, but could just as easily use server sockets and streams, or even http-style communication. Should a standard arise, switching to one of these transport protocols would be trivial. At the time of writing, I did not envision any other agent-based software implementations that ManyMinds would need to communicate with, but future developments in this area will be easily addressed.

4.2 Advisory System

ManyMinds agents contain a web of HTML pages that form their advice space. Each agent also contains a reasoning system made up of condition/action rules. Agents that are appropriate to the user's current task are shown on the notebook

screen. Agents have expertise in task domains (such as hypothesizing), sociocognitive domains (such as collaborating), metacognitive domains (such as planning), or system development domains (such as improving or adapting).

4.2.1 Advisor Interface

ManyMinds agents do much more than define the contents of the artifact. They also can become active as the user works, having their icon put into the "agent panel" (see Figure 2). While active, agents can give advice by placing a short sentence into the advice preview panel. If the user wishes to read more about that advice, clicking the "More" button will display an HTML page in the accompanying web browser. The HTML files are pre-written, not generated at run-time. They do contain, however, variables that can be substituted with values from the datamodel at run time, thereby somewhat personalizing the advice; for example, the agents can say "I see your research question is:" and insert the student's question into the advice window.

The advice web pages are broken into several important categories. Agents can give advice about goals, motives, strategies, and plans as well as examples and stories that further personify the agent. Goal advice is about a particular goal or criteria the agent has for the user, for example Quentin Questioner has the goal that the research question be possible to answer and advice about how to know when a question is answerable. Motive advice explains why a particular goal is important,

for example why it is important to have an answerable question. Strategies give help in developing a product that meets a particular goal. An agent's plan advice can be a collection of strategies, but usually addresses the creation of a product in a more holistic fashion. A research question is not usually developed by creating an answerable question, making it interesting and then changing it so it builds on previous research. While those individual goals can be addressed by strategies, the overall questioning plan would follow a different path through the question space followed by assessment of the question against the agent's stated goals for a question. The planning advice is usually rather abstract, requiring users to apply their new knowledge to their particular situation. To assist this process, agents can provide examples or vignettes describing how hypothetical students used their advice. Agents also include definitions of important concepts including "hypothesis" and "experiment".

4.2.2 Advisor Technologies

ManyMinds agents are built using the KQML messaging system discussed above (see Section 4.1.3). Agents will respond to messages, usually about the value of variables in the datamodel, and will generate messages for actions like giving advice or adding pages to the research notebook. The agents each posess a production system for reasoning consisting of a collection of "if/then" rules that match preconditions to actions. Preconditions take the form of beliefs the agent holds (propositions) as well as functions that can be applied to the global data (for example the

length of the hypothesis field or the timestamp of the last modification of the hypothesis completeness slider). See Table 4.2.2 for a detailed description of agent preconditions. Agent actions include modifying their own beliefs, asking questions of each other and the user, giving advice, forming teams of agents to give advice and modifying the research notebook. See Table 4.2.2 for more details. ManyMinds agents, therefore, are largely reflex or goal-based, as described by Russell and Norvig [9]. An agent designer could create a simple reflex mapping from slider state to appropriate advice, or a more complex representation that makes use of agent beliefs to mediate what kind of advice is given and how the advice is phrased and delivered.

This reasoning system was implemented using the standard Java event dispatching model. Preconditions, when becoming true, notify the rules that contain them, which in turn notify their actions. The reasoning system was designed to be powerful enough to model complex behavior, while maintaining complete transparency to the user. The agent reasoning and messaging system is built to make use of multiple threads when available. The current version of ManyMinds uses a single thread for message dispatching and a single thread for knowledge event dispatching, but additional threads could be added if it would improve performance (for example, if the software were run on a multiple processor system).

The agent production system is stored as an XML file and packaged in a jar file, which also contains the XML definitions of pages and raters the agent can add to the notebook and the agent's HTML advice files. At program startup, Many-

Minds loads all the agent definition jar files found in its agent folder. To add a new agent to the system one must simply package the agent's definition and place it in the appropriate location in the file system. At the current time, most agent editing is done with a generic XML editor, rather than a specialized tool for agent editing. Several rounds of design and modification of that tool have occurred (See Figure 3), and to date no tool has proven more useful than a basic XML editor. Future design of ManyMinds will focus on that tool, however, as well as a tool for supporting the editing and creation of the agent's HTML files.

Precondition	*Description*
Must Believe X	The agent must believe X to be true
Must Not Believe X	The agent must believe X to be false
Global Comparator	Compares Metaglobals, can use greater than, less than, equal, not equal, semantics of string or numerical representation depends on type of global being compared.
Value X Metaglobal	The value of X in the data model
Length X Metaglobal	The length of X in the data model
Timestamp X Metaglobal	The last modification time of X in the data model
Count X Metaglobal	The number of X in the data model

Table 1: Preconditions in ManyMinds agent reasoning system

Action	*Description*
Give Advice	Set the text in the advice preview window, and give URL to display if the user hits "more".
Form Team	Request team members to enter the agent panel, also kicks out current team.
Add Section	Adds a section to the top level of the artifact, also gives color and ordering preference.
Add Page	Adds a page to a section in the notebook.
Add Rater	Adds a reflective self assessment slider to the notebook

Table 2: Actions in ManyMinds agent reasoning system

5 A ManyMinds Case Study: Inquiry Island

Inquiry Island is a particular instantiation of the ManyMinds system based around supporting inquiry science projects in a middle school classroom. This project grew out of the ThinkerTools Inquiry Curriculum [14]. The current Inquiry Island artifact is a research notebook for doing inquiry projects such as science fair projects. This artifact builds on prior research regarding effective ways to scaffold the inquiry process. Each of the six stages of the Inquiry Cycle is given a separate, color-coded section of the research notebook. A typical page in the research notebook, in this case the Hypothesis page, (Figure 1) includes a text area for working on a hypothesis, another for working on the rationale for that hypothesis, and the self-assessment tools (see Section 4.1.2) necessary for judging the quality of that hypothesis. The agents developed for Inquiry Island correspond to the steps of the Inquiry Cycle (Quentin Questioner, Ivy Investigator) as well as the other supported cognitive (Ingrid Inventor), metacognitive (Molly Monitor) and sociocognitive (Keiko Collaborator) skills specifically targeted by the ThinkerTools Inquiry Curriculum.

5.1 Science Fair Club Pilot Study and Analysis

The ThinkerTools group started an after-school club at an urban middle school with the goal of testing ManyMinds Inquiry Island by providing it to support students' work on their science fair projects. This club met two times a week for ninety minutes and had seven members. All seven students used the Inquiry Island software

to develop their projects, and also engaged in many different activities designed to teach inquiry and reflective skills. Those activities included using the science fair judge worksheet to judge examples of prior science fair projects, using the Many-Minds reflective self assessment system to assess and give feedback about each others' projects, and modifying the agents' advice. The students were shown how to use the software and encouraged to use the reflective self assessment tools, but they worked independently for the most part. Teacher intervention in the students' work flow was primarily at the students' request, so the advisory agents were the main source of reminders to reflect and revise.

Overall, we found that students of varying ability levels were able to use the software to do a science fair project. While not all the findings discussed below are positive, the general result is encouraging. Of the seven full-time participants in the science fair club, there was a first place project, two second place projects and a third place project. This performance is not solely attributable to the software, since these students also spent longer working on their science fair projects than the average student.

The results discussed here are preliminary and observational, based on trials with only seven students. Our findings show what a student can accomplish using Inquiry Island without much teacher intervention. These results will be used to inform the development of advice, software, and activities for future deployment of Inquiry Island in classrooms. The three key results are that students mostly ignore advice from agents, that students are able to assess their work using the reflective

self assessment sliders, and that students do not readily revise their work.

During the course of the club intervention, different mechanisms for presenting advice were tested. Agents would sometimes simply show a “light bulb” to indicate they had advice. Other times, agents would interrupt the student by popping up a web browser window with their advice in front of the notebook . A middle ground, where agents showed a small text area describing the advice they could give is shown in Figure 2. The light bulb mechanism for displaying advice went largely ignored, and the much more aggressive pop up display annoyed the students to the point where they turned off the advice entirely. The third technique was a comfortable balance, where students did notice that the advisors had advice without getting annoyed by it, but students for the most part still did not read offered advice. Our observations of the students’ use of advice led to changes that have been made in the agents. First of all, the advice synopses (Figure 2) have been reworded in a less generic fashion to help students differentiate between advice they have seen before and advice they have not seen. Second, the advice itself was further refined to eliminate “blind spots” that became obvious only through use. Since the students discovered those blind spots, they soon learned that the agents were not as helpful as they thought they would be, and they did not view the agents as a very useful source of information. Thirdly, we have increased the sophistication of the rules that govern agent behavior. During this pilot we aimed for a minimal implementation of agent rules, but future implementations will have several important additions. Agents now wait for a break in the user’s work to give advice, rather

than interrupting with advice when it first becomes appropriate. Agents also have the capability to become more aggressive in giving advice if they sense they are being ignored. Finally, advice will be more tailored to the user's situation.

Several club activities specifically scaffolded the use of the reflective self-assessment tools. First, students were given a paper-and-pencil handout of the slider criteria and asked to assess science fair projects from a previous year. Next, students exchanged Inquiry Island notebooks and were asked to set the sliders and add comments to each others' work. Successful participation in these activities showed that students were capable of engaging in reflective assessment; three of the students also used the sliders during their work without being asked. One of the students even revised his slider settings as he worked, changing the assessment as his product improved. While this was not a widespread activity, we are very encouraged to observe it at all. This behavior will be better supported in future deployments of the system, including improved support for peer assesment activities.

The ManyMinds data logs were searched for times when students engaged in active revision of their work. Those revisions were almost never found. Students in the science fair club tended to work by adding text to whatever they were working on at the time and then move on to their next task. This result indicates that more classroom support for revision of work is necessary in future implementations of Inquiry Island. We hypothesize that the assesment-advice-revision cycle of work is metacognitively sophisticated and not necessarily supported in traditional science classrooms; we hope to coach this behavior during future Inquiry Island

implementations.

6 Future Work

In addition to the design changes suggested by the after school club pilot, future ManyMinds development will focus on extending the capabilities of the artifact, further developing the Inquiry Island system, and designing agent editor and teacher support tools. In May 2002, Inquiry Island will be used in a sixth grade classroom during a four-week unit. Students will use Inquiry Island to structure and carry out a class inquiry project. In this study, my primary research goal is to develop a scheme for parsing and coding the ManyMinds data logs to better understand student use of the system and how that use reflects student metacognitive expertise. We will also begin examining specific classroom techniques that will be used in future Inquiry Island studies, such as peer assessment and role playing. Students will receive feedback on their projects within the Inquiry Island system from teachers, graduate students and researchers. Finally, in the Fall of 2002, Inquiry Island will again be used in middle school classrooms to study its impact on inquiry and metacognitive skills. A longer intervention will be necessary in order to learn what kinds of activities best support growth in metacognitive and inquiry skills. This study will take place in the course of at least two curricular units, an inquiry-based content unit such as ThinkerTools force and motion and a science fair project unit.

7 Bibliography

References

[1] BROWN, A. L. Metacognition, self regulation and other more mysterious processes. In *Constructivism in the computer age* (Hillsdale, NJ, US, 1988), G. Forman and P. B. Pufall, Eds., Erlbaum, pp. 49–70.

[2] BUSH, K. Genscope inquiry curriculum. Unpublished.

[3] CAREY, S., AND SMITH, C. On understanding the nature of scientific knowledge. *Educational Psychologist 28*, 3 (1993).

[4] DISESSA, A. A. Knowledge in pieces. In *Constructivism in the computer age* (Hillsdale, NJ, US, 1988), G. Forman and P. B. Pufall, Eds., Erlbaum, pp. 49–70.

[5] ESLINGER, E. M., WHITE, B. Y., AND FREDERIKSEN, J. R. A modifiable multi-agent system for supporting inquiry learning. In *Artificial Intelligence in Education* (San Antonio, TX, 2001), J. D. Moore, C. L. Redfield, and W. L. Johnson, Eds., IOS Press, pp. 545–547.

[6] HAMMER, D., AND ELBY, A. On the form of a personal epistemology. In *Personal Epistemology: The Psychology of Beliefs about Knowledge and Knowing* (Mahwah, NJ, 2002), B. K. Hofer and P. R. Pintrich, Eds., Erlbaum, pp. 169–190.

[7] LABROU, Y., AND FININ, T. A proposal for a new kqml specification. Tech. Rep. TR CS-97-03, University of Maryland Baltimore County, Baltimore, MD 21250, February 1997.

[8] PANG, P. Initiating and inquiry science curriculum. Master's thesis, University of California at Berkeley, 1999.

[9] RUSSELL, S. J., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1995.

[10] SCHOENFELD, A. H. What's all the fuss about metacognition? In *Cognitive Science and Mathematics Education* (Hillsdale, NJ, 1987), A. H. Schoenfeld, Ed., Lawrence Erlbaum Associates, pp. 189–215.

[11] SHIMODA, T. A. *Student goal orientation in learning inquiry skills with modifiable software advisors*. PhD thesis, University of California at Berkeley, Berkeley, CA 94720, May 1999.

[12] SHIMODA, T. A., WHITE, B. Y., AND FREDERIKSEN, J. R. Acquiring and transferring intellectual skills with modifiable software agents in a virtual inquiry support environment. In *Proceedings of the 32nd International Conference on System Sciences* (Los Alamos, CA, 1999), IEEE Computer Society.

[13] WHITE, B. Y. Thinkertools: Causal models, conceptual change, and science education. *Cognition & Instruction 10*, 1 (1993), 1–100.

[14] WHITE, B. Y., AND FREDERIKSEN, J. R. Inquiry, modeling, and metacognition: Making science accessible to all students. *Cognition & Instruction 16*, 1 (1998), 3–118.

[15] WHITE, B. Y., SHIMODA, T. A., AND FREDERIKSEN, J. R. Enabling students to construct theories of collaborative inquiry and reflective learning: Computer support for metacognitive development. *International Journal for Artificial Intelligence in Education 10*, 2 (1999), 151–182.

Figure 1: Inquiry Island Main Windows

Figure 2: Agent Advice Box

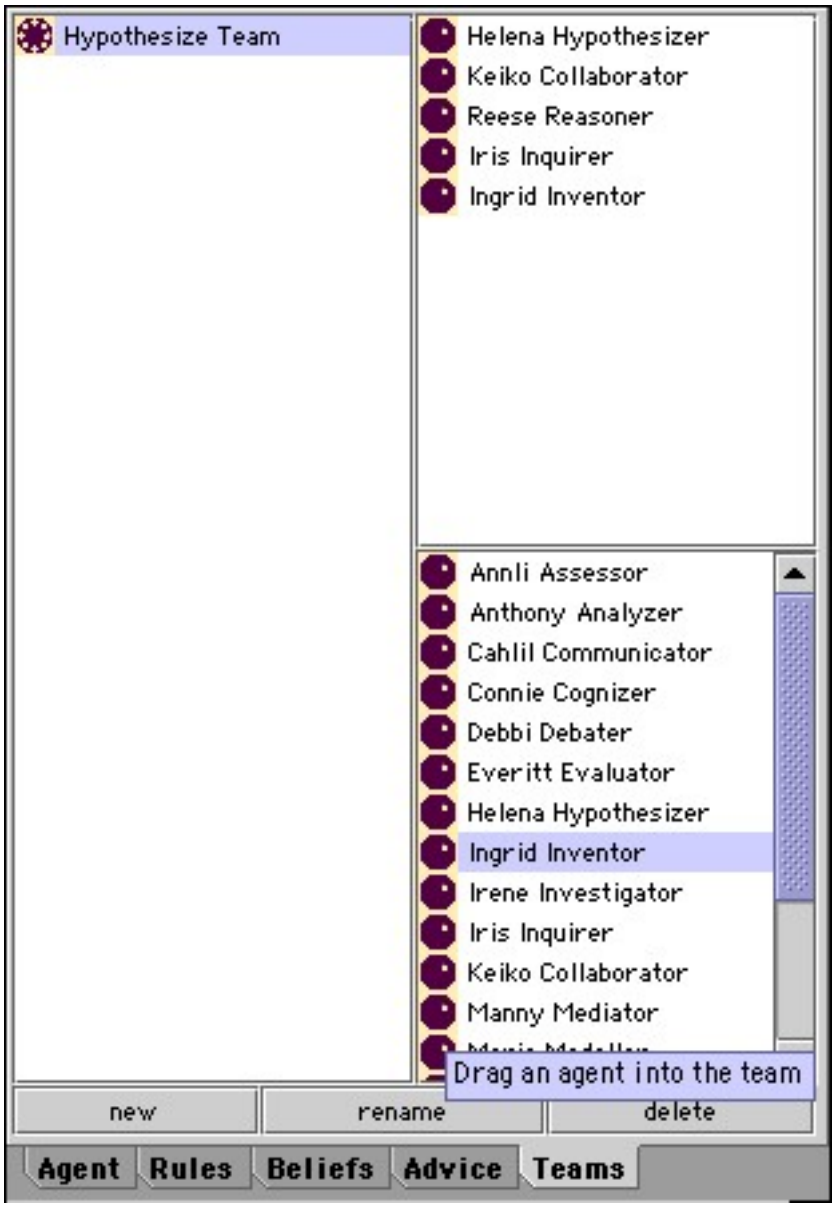

Figure 3: Experimental Agent Editor Tool