Team Tutoring Systems: Reifying Roles in Problem Solving

Mark K. Singley, Peter G. Fairweather, Steven Swerling

Applied Learning Sciences Department, IBM T.J. Watson Research Center

Abstract: We are pursuing the notion of a Team Tutoring System, an intelligent tutoring system that monitors and manages teams of students as they collaborate synchronously and remotely to solve extended, distributed, multi-step problems. Our system, named Algebra Jam, provides opportunities for each team member to take on a variety of roles in order to come to know the domain from a variety of perspectives. To do this, we are defining a typology of collaborative problem solving roles, reifying these roles at the problem solving interface, and accumulating evidence about individual behavior and group problem solving performance in a Bayesian inference network.

Keywords: team tutoring systems, synchronous collaboration, goal reification

Team Tutoring Systems: An Opportunity

We are pursuing the vision of an intelligent tutoring system that monitors and manages teams of students as they work together to solve extended, distributed, multi-step problems. It seems a natural, evolutionary step for intelligent tutoring systems (ITS) to support this kind of multi-user collaborative operation. Traditional ITS implementations, whatever their power, invite criticism for their emphasis on the individual's solitary acquisition of cognitive skills at the expense of other potentially significant social learning interactions.

For example, Brown and Palinscar's work on reciprocal teaching (Brown & Palinscar, 1989; Palinscar & Brown, 1984) powerfully illustrated the benefits of having learners work together and serially adopt the roles of learning leader, learning listener, and supportive critic. The technique provides an external observable model of the comprehension process that learners can examine and internalize by practicing the activities of questioning, clarifying, summarizing, and predicting. The situated learning movement as a whole, inspired by the work of Vygotsky and others, takes as a first principle the importance of social interaction and mutual scaffolding in any effort to build a community of learners.

With the World Wide Web now providing the infrastructure for collaboration, the opportunity is ripe to create virtual communities of problem solvers. We have been working to create an environment in which students can take on a variety of roles and collaborate synchronously and remotely to solve what might be thought of as extended, distributed, highly-situated algebra word problems. In a traditional ITS, the role of the tutor is to monitor and manage an individual's trajectory through a curriculum and provide feedback on individual problem solving actions. However, the role of the tutor in a Team Tutoring System like the one we are building is to monitor and manage an entire team's trajectory through the curriculum and to provide opportunities for each team member to take on a variety of collaborative roles in order to come to know the domain from a variety of perspectives. To do this we are defining a typology of collaborative problem solving roles, reifying these roles at the problem solving interface, and accumulating evidence about individual behavior and group problem solving performance in a

Bayesian inference network.

Collaborative Learning Roles and Schemas

Drawing on Vygotsky and other more contemporary social learning theorists, we are proposing a developmental progression of collaborative roles that students take on as they acquire cognitive skills in group settings. In any particular problem solving episode, a participant may fill out each of these roles to a greater or lesser extent, but our position is that there is pedagogical value in having each participant assume these different points of view as learning progresses.

The roles are:

1) **Observer.** The observer is initiated into the domain of practice by watching other more skillful problem solvers set goals and perform actions. The observer should be given the opportunity to ask those modeling the target skills to explain and justify their actions.

2) **Apprentice.** The apprentice assumes responsibility for managing and performing fairly routine problem solving subgoals, such as filling in a table or computing an average.

3) **Specialist.** The specialist has mastered and is responsible for handling certain problem schemata which appear as major subgoals in the problem, such as determining the rate at which two people work together given their individual work rates. As students work through the curriculum, they have the opportunity to become specialists for a variety of schemata.

4) **Leader.** The leader establishes the overall strategy for solving the problem, determines what resources are available and what additional resources are required, assigns tasks to individual participants, coordinates actions, and evaluates progress.

5) **Coach.** The coach observes and critiques the actions of other participants, responds to help requests, advises the leader, and models problem solving behaviors for other participants.

Of course, this is not necessarily a strict progression, and depending upon the subdomain in question, any particular student may be operating at multiple levels simultaneously. In fact, one possibility is to have the tutor enforce different learning trajectories for different students so that, when combined in teams, each student has some particular strength to contribute to the collaboration. Thus, the tutor might coordinate an entire team's trajectory through the curriculum to enable opportunities for complementary and productive role assignments.

Given these roles, what might be called collaborative learning schemas (CLS) organize themselves. For example, if the role of leader has not been assigned among the participants, then usually one emerges as the group organizes itself to solve a problem. But the presence of a leader implies the presence of other participants filling complementary roles if the collaboration is to be productive. The particular constellation of complementary roles defines the CLS. For example, here are some typical role constellations:

Modeler-Observer: Participants watch the coach publicly solve the problem with little involvement other than eliciting explanations.

Leader-Apprentice: The learning leader assigns sub-tasks as part of the problem solution to apprentices.

Critic-Performer: Participants perform sub-tasks, receiving feedback from the coach.

Great Escape: Several specialists combine forces to solve a multi-part problem.

CLS reorganize themselves, prompted by changes in problem type, experience and confidence of the participants, and the group make-up. Moreover, they present themselves in hybrid mixtures, sometimes making it difficult to determine their identity and, consequently, making it more difficult to determine the role a learner has assumed.

Reifying Roles and Gathering Evidence

When designing a team tutoring system, a fundamental concern is how to reify each of the collaborative roles at the interface, and also how to detect which CLS is extant at any particular time. In short, how do we know who is doing what to whom? This is a difficult interface design and inferential challenge, and we can offer only a preliminary and incomplete solution at this time. Our basic strategy is to provide tools at the interface for different kinds of collaborative behaviors and associate the use of particular tools in particular ways with particular roles. An important constraint on our design is our rejection of any reliance on natural language processing to interpret participant actions.

Once this potentially many-to-many mapping between tool and role is established, the system may adopt either a prescriptive or non-prescriptive mode of operation. In the prescriptive mode, roles are assigned explicitly, with only a certain subset of interface actions enabled for each participant depending upon their role assignment (Smith, Hixon, & Horan, 1998). Although it may be argued that this is somewhat constraining and artificial, it simplifies the adoption of a model of collaboration for all participants, reduces the amount of time spent during problem solving on role negotiation, and may provide the necessary structure to allow relative strangers to work together. It also greatly simplifies the inference problem of determining the extant CLS.

In the more liberal, non-prescriptive mode, roles are not assigned, and the interface is fully enabled for all participants. Students freely adopt one or more roles to a greater or lesser degree in the ebb and flow of the session by virtue of the frequency of use (or non-use) of certain role-defining tools. But under this regime, although the tutor's inferential machinery is the same, much less certain inferences can be drawn about who is doing what to whom.

An Example Team Tutoring System: Algebra Jam

As mentioned above, we are building a team tutoring system called Algebra Jam that guides groups of students through a problem-centered curriculum of distributed, extended, highly-situated algebra word problems. At the heart of each problem are one or more instances of problem-solving *schemas*, which are sets of entities and quantitative relationships that tend to cohere in the world and can be used to describe a broad range of problem-solving situations.

We will illustrate the use of schemas to analyze a standard word problem, although as alluded to above, Algebra Jam problems differ from standard word problems in a number of important ways. In algebra, there is a large class of problems that can be construed as involving linear

systems of equations. For example, here is a standard algebra word problem that recently appeared on the SAT:

> Excluding rest stops, it took Juanita a total of 10 hours to hike from the base of a mountain to the top and back down again by the same path. If while hiking she averaged 2 kilometers per hour going up and 3 kilometers per hour coming down, how many kilometers was it from the base to the top of the mountain?

The key to our use of schema theory is the assertion that word problems such as this can be characterized (and categorized) in terms of the underlying set of equations that relate the entities of the problem to one another. According to this analysis, problems that superficially appear quite distinct may in fact be instances of the same underlying problem structure (Marshall, 1995). The Juanita problem is an instance of the *round-trip* schema. The round-trip schema

(1) $d_t = d_u + d_d$
(2) $t_t = t_u + t_d$
(3) $d_u = r_u * t_u$
(4) $d_d = r_d * t_d$
(5) $d_t = r_t * t_t$
(6) $d_u = d_d$

involves the following equations:

This set of variables and primitive equations defines an entire class of problems. Given a set of variables and equations such as this, a particular problem within the class is represented as a set of variable assignments and a goal. Table 1 defines the schema variables and gives their values in the context of the Juanita problem. Table 2 organizes the schema equations into a table. One of the relationships in the table, $r_u + r_d = r_t$, does not hold. However, the other relationships are correct.

Table 1: Variables and their values

Variable	Meaning	Value
d_u	distance up	x (goal)
d_d	distance down	unknown
d_t	distance total	unknown
t_u	time up	unknown
t_d	time down	unknown
t_t	time total	10 h
r_u	rate up	2 km/h
r_d	rate down	3 km/h
r_t	rate total	unknown

Table 2: Round-trip schema equation table

	part	+	part	=	whole
distance =	d_u		d_d		d_t
rate *	r_u		r_d		r_t
time	t_u		t_d		t_t

An interesting property of our schemas is that, for any particular problem instantiation, not all of the underlying primitive equations may be required for solution. This is what gives the schemas psychological content and differentiates them from purely structural descriptions of problems. In a purely structural description, only the equations required for solution are included. However, in a schematic description, the entire set of relationships that model the situation are included. In essence, the schema position states that certain sets of relationships tend to co-occur in the world and therefore tend to cohere in a person's head. When someone is confronted with such a situation (e.g., when someone is asked to solve an algebra word problem), the entire set of relationships is activated and brought to bear to try to understand the situation. One large part of the problem is determining which subset of the equations contained in the schema is actually relevant for solving the current problem (Singley, 1995).

Elsewhere we have described the Schema Compiler (Singley, Fairweather, & Alpert, 1998), an analytical engine that takes schematic descriptions and generates detailed and exhaustive problem graphs which can be used for model tracing and remediation (Anderson, 1993). The Schema Compiler is used to provide model tracing capabilities in Algebra Jam.

Figure 1 shows a screen image from Algebra Jam. Participants are placed in a problem-solving scenario rich with algebraic modeling opportunities (e.g. running a business) and are presented with somewhat vague and ill-defined tasks. For example, in the example problem shown, participants find themselves cast as proprietors of a landscaping service and receive a request to schedule a job. Unlike traditional word problems, all the information required for solution is not presented explicitly in the statement of Algebra Jam problems. Instead, information that may be necessary and/or sufficient to solve the problem is buried in a rich set of problem resources (work logs, ledgers, records of correspondence, etc.) that metaphorically either sit on a bookshelf or are placed elsewhere in the environment. Thus, students engage in what has been termed *problem finding* in addition to more traditional problem solving activities.

As students work through problems, they search through the resources provided on their desktop. Resources that are "opened" reside in a tabbed notebook, as shown in the lower right corner of Figure 1. The various kinds of problem resources (e.g. equation scratchpad, table, graph, map; a table is shown in the figure) require students to derive information from a number of widely-used quantitative representations, thereby providing opportunities to improve quantitative literacy as well as algebra problem solving abilities. Each resource is governed by its own schema. In Figure 1, a table is shown that is governed by the *selective average* schema: the problem solver is required to take an average of a selected subset of the row data to find John's mowing rate. This information is passed to the root schema of the problem, a *work* schema that relates John's work rate and the amount of work to be done to the expected work time. By embedding one schema in another, the Algebra Jam architecture permits a hierarchical schematic organization for problems. Thus, Algebra Jam problems can be made arbitrarily complex and afford many

opportunities for decomposition and distribution of tasks among participants.

Contained within the Algebra Jam interface are not only the representations of the problem and the resources used to solve it but the communication tools necessary for collaboration. We cannot be satisfied with generic collaborative tools such as whiteboards because they require more natural language understanding capability than we can provide. Instead, we are developing a set of problem-oriented communication tools that not only help reify collaborative roles but also narrow the potentially unmanageably wide variety of interpretative tasks facing the tutor. Algebra Jam uses four main mechanisms to support collaboration and role reification:

Synchronous Collaboration with Public and Private Resources. Using the flexible event reflection capabilities provided by the Shared Data Objects framework (Banavar, Doddapaneni, Miller, & Mukherjee, 1998), the system supports both public and private workspaces. Students can either work synchronously with a shared resource or work "offline" with a private copy of a resource.

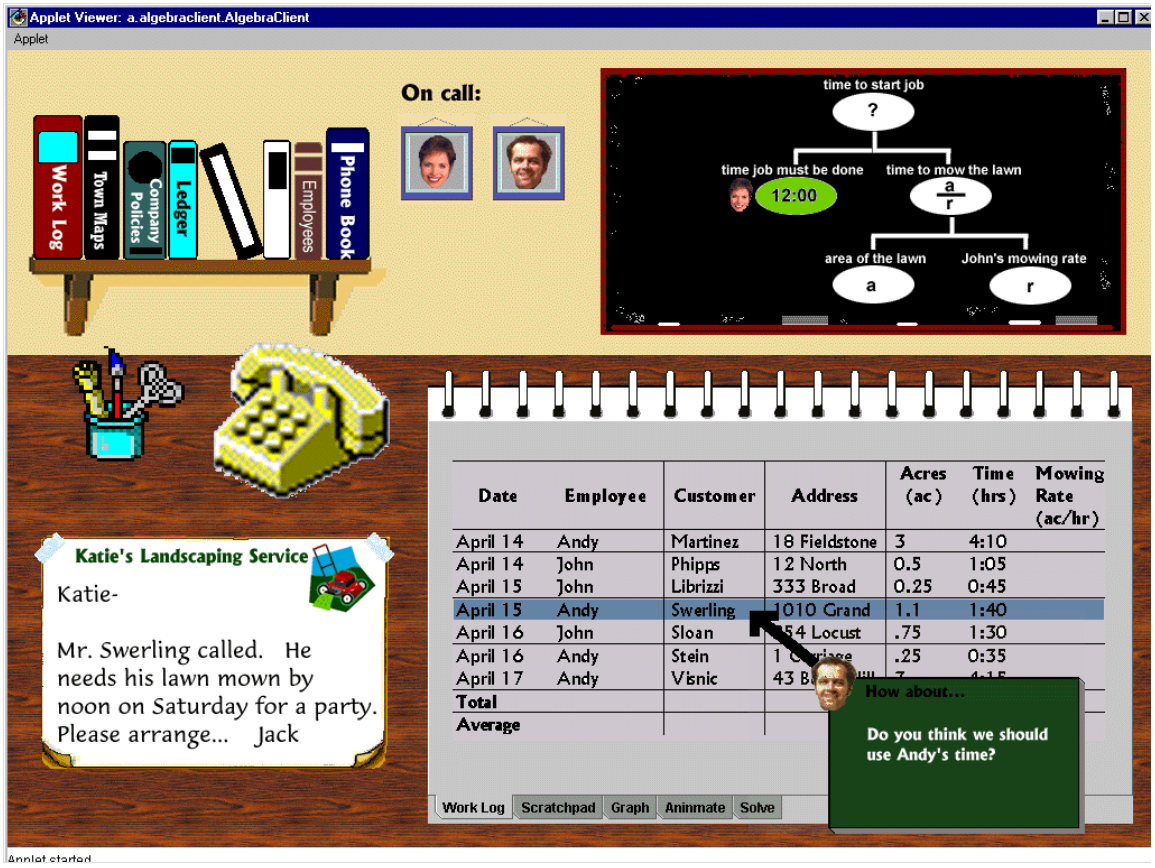

Figure 1. Screen image from Algebra Jam.

Team Blackboard. In most conventional problem solving environments, as students solve

problems, the abstract goals and plans that organize and motivate their low-level problem solving actions are often left implicit. As students work together to solve problems with Algebra Jam, however, these abstract goals and plans are reified as graphical objects in the user interface and become the focus of collaborative and tutorial discourse. Students have an opportunity not only to perform a sequence of actions that will solve the problem, but also to produce the higher level abstractions, the goals and plans, that organize those actions. Of course, it is just these abstract goals and plans that represent the deep structure of the domain and form the basis for transfer across problems, so it is critically important to make them explicit during problem solving (Singley, 1995). The goal structures provide the rationale for taking a particular problem solving action, and thereby support reasoning about the range of application of that action in other problems. Once reified at the user interface, the goal and plan structures provide an abstract annotation of the problem solving trace, and the trace itself becomes an exemplification of the abstraction. This reification and exemplification of abstractions is useful not only when the problem-solving actions are being generated in the first place, but also after the problem has been solved and students are reflecting on their work (Collins & Brown, 1988).

Goal reification is important for problem solvers working alone, but it is even more important for problem solvers working together. As the focus of collaborative discourse, reified goal structures support the sharing of information, the distribution of tasks, and the coordination of work. They promote a shared understanding of what's been done and what needs to be done.

In Algebra Jam, the goals that appear in the team blackboard are schema variables, problem entities whose values are needed to solve the problem. Once posted, the goal structures reveal the hierarchical schematic organization of problem and represent the logical dependencies between variables. For example, in the Team Blackboard in Figure 1, traversing the tree from the root we see that, in order to determine the starting time for mowing the lawn, we need to determine the required stopping time and the expected duration of the work. In order to determine the duration of the work, we need to determine the amount of work to be performed and the work rate, and so on. It is important to note that Algebra Jam problems can be overdetermined, i.e. there may be more than one way to derive the value of a variable. It may be possible to simply look up a required value in one resource, or compute the value in another. These alternative methods raise issues of efficiency and optimality, which stimulate discussions among the participants as well as the tutor. Once completed, goals turn green and an image of the participant who found the value is associated with the goal. Learning leaders may assign unsatisfied goals to participants as well.

Object-Oriented Chat. Object-oriented topical chats permit one learner to communicate with another by typing a message while pointing to a particular interface object on the screen. As the user mouses over the screen, interface objects (table cells, written equations, blackboard variables, bookshelf resources, etc.) are highlighted. If the right mouse button is clicked over a highlighted object, a special chat dialog appears that is tethered to the highlighted object by an arrow. The exact position of the dialog window and the length of the arrow can be adjusted by dragging the participant image which acts as a pivot between the arrow and the dialog window. For example, in Figure 1, one participant is using the chat facility to ask his partner whether the data held in a particular table row is relevant to their task.

The arrow functions as a demonstrative and a locative in the statement or question in the dialog, thereby easing the typing load on the sender and the comprehension load on the recipient. The

conversation is grounded as being about something. Additionally, the object-oriented nature of the chat gives the tutor additional leverage on the thorny inference problem of determining who is doing what to whom. Consider the following scenario: The learning leader highlights a particular unopened resource and sends a message about it to an apprentice. The apprentice then opens the resource and uses it productively to derive the value of a relevant variable. In this circumstance, the tutor has some warrant to conclude that the learning leader successfully performed a role-defining behavior by identifying an appropriate resource for the apprentice. This inference is drawn without any reliance on natural language processing.

Collabicons. To further reduce the generation and comprehension loads on participants and provide even more interpretive leverage to the tutor, we are attempting to define a message typology which when completed will constitute a basic ontology of collaborative acts (cf. Jona, Bell, & Birnbaum, 1991). The message types, which we have dubbed *collabicons*, provide the sender with a small but comprehensive range of collaborative intentions with which to frame a message. The message types fall into the four categories of assistance, control, information, and evaluation, as shown in Table 3. When sending a message, a participant is first prompted for the type of message being sent. We are currently experimenting with cartoonish facial expressions and other icons to accompany the textual tags for the collabicons.

When the intentional cues provided by the collabicons are combined and crossed with the content cues provided by the object-oriented pointing facility described above, the resulting chat mechanism supports a fairly rich tag language which we hope will simplify the generative and interpretive tasks of both the participants and the tutor.

Table 3: Collabicons from Algebra Jam.

Message Type	Collabicons
Assistance	Help Me
	Show Me
	Want Help?
	Watch Me
Control	I'll Do It
	You Do It
	Let's Work Together
	Let's Quit
Information	Huh?
	For Your Information…
	Why?
	How About…
Evaluation	Thumbs Up
	Thumbs Down
	Thumbs Sideways

Team Modeling

Team modeling generalizes the notion of student modeling. Instead of simply monitoring a single student's trajectory through a domain, our task is to monitor an entire group's. In addition to modeling each participant's domain knowledge as in a traditional tutoring system, we are attempting to model each participant's collaborative proficiency, as well as overall team performance. Again, our motivation is the belief that there is value in having students progress through a range of collaborative roles as they acquire cognitive skills in group settings. If role behaviors can be recognized and monitored by the tutor, student interactions might be engineered to enhance learning.

To construct our team model, we must confront two difficult questions: How do we gather evidence about collaborative activities (who is doing what to whom?), and how do we use that evidence to update beliefs about knowledge states (who knows what?).

Table 4: Gathering evidence on collaboration

Role	Evidence
Observer	• observes problem solving actions of others (shares resource but performs no actions) • asks clarification questions (sends "Huh?" collabicon) • initiates help requests (sends "Help Me, Show Me" collabicons) • accepts help (receives "Want Help, Watch Me" collabicons)
Apprentice	• accomplishes routine subgoals (leaf)
Specialist	• accomplishes non-routine subgoals (non-leaf)
Leader	• posts subgoals on the Team Blackboard • makes subgoal assignments on the Team Blackboard • identifies appropriate resources (sends "How About" collabicon while pointing to relevant resource) • evaluates work products (sends "Thumbs Up, Down, Sideways" collabicons)
Coach	• responds to help requests from other participants (replies to "Help Me, Show Me" collabicons) • offers help (sends "Want Help, Watch Me" collabicons) • evaluates work products ("Thumbs Up, Down, Sideways" collabicons)

To answer the first question, we must define a set of events at the interface that, when observed, provide some warrant to update one's beliefs about student domain proficiency and collaborative skill. Many of these observable events will be no different from the kinds of events observed in traditional tutoring systems, i.e. they will reflect a student doing productive work with the tools provided by the interface to solve the problem. However, some of the observable events will reflect collaborative activities that do not directly represent a move through the problem space but rather are meant primarily to either elicit or convey knowledge to other participants. Nevertheless, just like the more productive events, these events provide important clues about what students do and do not know…if they can be deciphered. Table 4 provides a list of events that constitute evidence for a participant taking on a role.

As mentioned above, certain roles imply complementary roles, which is captured in our notion of CLS. Accordingly, we use the CLS to squeeze more information out of our observations. Thus, evidence of role behavior may be "active" (we know the participant is assuming the role because of something s/he has done) or "passive" (we know the participant is assuming the role because of something he hasn't done, or because of some action of another participant).

Bayesian Interpretation of CLS-Based Evidence

Given the high level of uncertainty, the multiple sources of evidence, and the underlying multivariate model, we have turned to Bayesian inference networks to help us manage our inferences about participant knowledge states (Mislevy, 1995).

Figure 2 presents a portion of the team model for Algrebra Jam; portions of the network for a team consisting of two participants is shown. The team model consists of a model for each team member as well as a construct representing the team as a whole. Each member's model consists of a domain model (which after Gitomer, Steinberg, & Mislevy (1995) consists of separate declarative, procedural, and strategic components) and a model of collaborative competence. Broadly speaking, collaborative competence consists of knowing 1) what roles exist, 2) how to perform each role, 3) when it is appropriate to assume each role, and 4) how to assert yourself in a role. We attempt to model a participant's proficiency with each of the five roles.

Traditional intelligent tutors focus their interpretive capabilities on inferring a student's mastery of skills or strategies from his or her actions at the application interface. The identification of a CLS and the roles taken up by its participants offer an opportunity to clarify interpretations by capitalizing on the different conditional probabilities with which certain events occur within a CLS. To illustrate, consider how silence is almost always interpreted as contrary evidence for the presence of a skill, as is often done in a classroom. However, if we have identified the CLS appropriately and the role of the learner, we can establish different interpretations of silence (or "doing nothing at the interface") by conditionally relating the evidence for a skill (silence) to particular roles (e.g., observer, specialist, coach).

In our team model, we use *collaborative event structures* to operationalize the CLS. Event structures capture the notion that the directly observed event, known as the primary event, may be fruitfully interpreted as an entire set of events. This set reflects the different facets revealed when the primary event is viewed through the prism of the CLS. When a primary event is received, it is first decomposed into its signature set of events. Then, the primary event and each of its derivatives is submitted separately to the network, with each event triggering its own independent updating of estimates of proficiency. Derived events are either secondary events, which provide additional evidence to update the subnetwork of the participant who produced the primary event, or collaborator events, which update the subnetworks of other participants.

Let's follow what happens when an Algebra Jam participant enters a new equation in the scratchpad, as shown schematically in Figure 2. The entering of the equation is considered the primary event by the event structure (a "produces" event). The correctness of this particular equation (Work = Rate * Time) has a direct bearing on the model's estimates not only of Jack's procedural proficiency at writing equations but also his declarative knowledge of the Work schema, from which this equation is drawn. Also, the fact that Jack is producing the equation (as opposed to say, observing or evaluating it) means that the model will be updating its estimate of

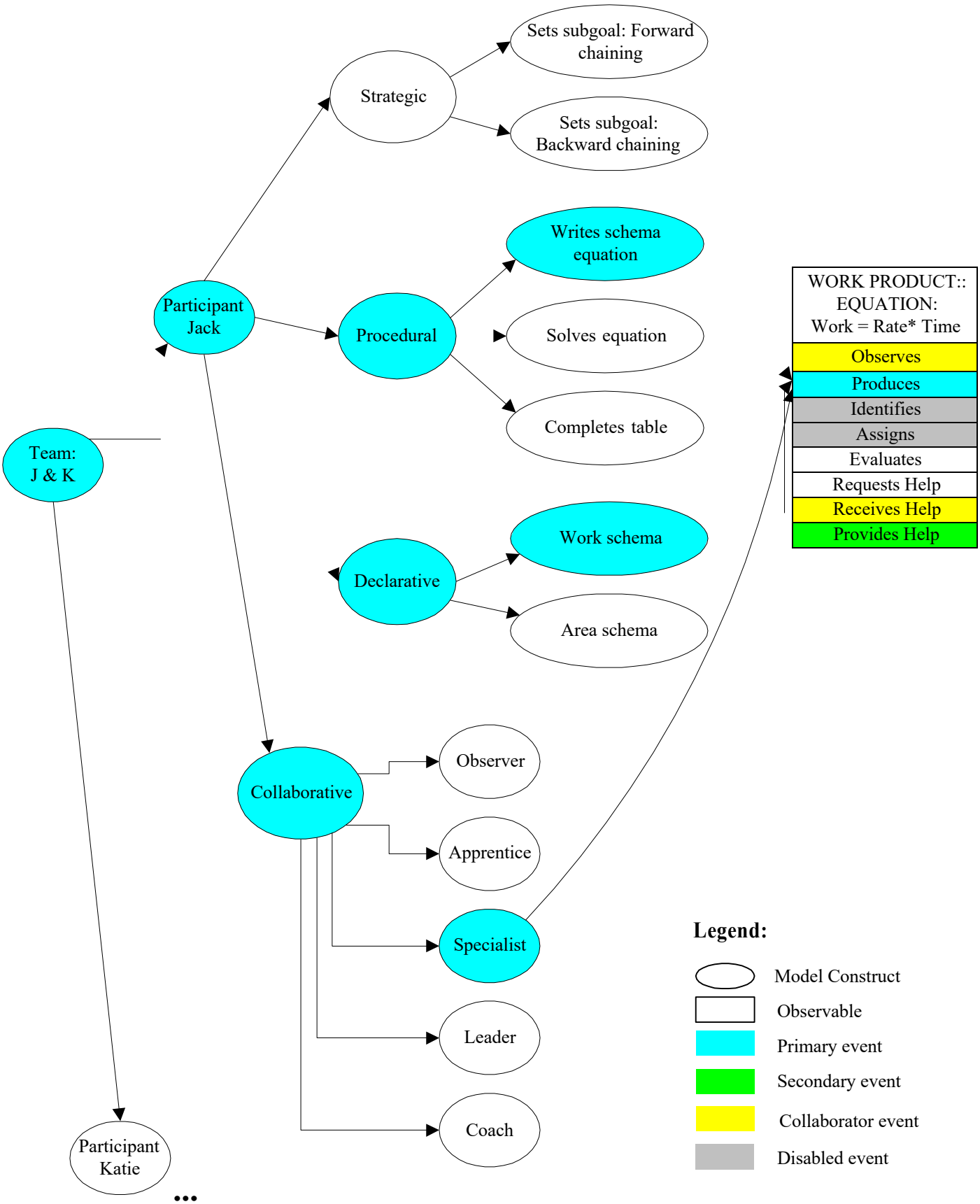

Figure 2. A small portion of the team model for two participants, Jack and Katie. Updates are shown in response to Jack writing a new equation in the scratchpad.

Jack's proficiency at assuming the “specialist” role.

Depending upon the collaborative context, however, this equation-writing event could be interpreted in a number of additional ways. For example, if this event is in response to a “Help Me” or “Show Me” request from another participant, then the secondary event of “provides help” is warranted. Also, this would trigger a “receives help” collaborator event for the recipient of the assistance. Finally, if any other participant was sharing the scratchpad at the time the equation was written, this would result in an “observes” collaborator event.

Enhanced Opportunities for Intervention

Conditional interpretation of a learner’s actions, depending on the role that he or she has assumed, not only enhances the tutor’s estimate what of the learner knows, but suggests pedagogical strategies the tutor might adopt. For example, if the tutor needs more evidence that a learner understands the sequences of steps in a problem, it could ask that learner model the solution and ask other learners to function as critics. Might not a learner profit from being assigned the role of having to defend the sequence of steps taken in the solution to a problem?

Tutors typically measure a learner’s progress in terms of proportion of the set of skills that learner uses to solve certain problems (Anderson, 1993). However, would it not be useful to also measure a learner’s progress in terms of the roles he or she assumes within a collaborative learning schema? At least role information could be used as another source of evidence regarding skill acquisition. For example, consider one situation where the tutor occasionally observes actions that implied the use of a cognitive skill. Compare it with another where the tutor repeatedly observes a critic correctly accepting or rejecting the explanations of a solution from other learners. Even though the evidence in the second case is more difficult to detect, in some sense it is more certain than evidence in the first case.

Bibliography

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Banavar,G., Doddapaneni, S., Miller, K., & Mukherjee, B. (1998). Rapidly building synchronous collaborative applications by direct manipulation. In *Proceedings of ACM 1998 Conference on Computer-Supported Cooperative Work* (pp. 139-148). New York: Association for Computing Machinery.

Brown, A. L., & Palincsar, A. S. (1989). Guided, cooperative learning and individual knowledge acquisition. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Collins, A., & Brown, J.S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems* (pp. 1-18). New York: Springer-Verlag.

Jona, M., Bell, B, & Birnbaum, L. (1991). Using button theory to structure student-teacher interactions in computer-based learning environments. In *International Conference on the Learning Sciences, Proceedings of the 1991 Conference*. Charlottesville, VA: Association for the Advancement of Computing in Education.

Marshall, S. (1995). *Schemas in problem solving*. New York: Cambridge University Press.

Mislevy, R. (1995). Probability-based inference in cognitive diagnosis. In P. Nichols, S. Chipman, & R. Brennan (Eds.), Cognitively Diagnostic Assessment (pp. 43-71). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gitomer, D., Steinberg, L, & Mislevy, R. (1995). Diagnostic assessment of troubleshooting skill in an intelligent tutoring system. In P. Nichols, S. Chipman, & R. Brennan (Eds.), *Cognitively Diagnostic Assessment* (pp. 73-101). Hillsdale, NJ: Lawrence Erlbaum Associates.

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, *1*, pp. 117-175.

Singley, M.K. (1995). Promoting Transfer through Model Tracing. In A. McKeough, J. Lupart, & A. Marini (Eds.), *Teaching for Transfer: Fostering Generalization in Learning* (pp.197-206). Hillsdale, NJ: Lawrence Erlbaum Associates.

Singley, M.K., Fairweather, P., & Alpert, S. (1998). The schema compiler. In *International Conference on the Learning Sciences, Proceedings of the 1998 Conference* (pp. 351-353). Charlottesville, VA: Association for the Advancement of Computing in Education.

Smith, R., Hixon, R., & Horan, B. (1998). Supporting flexible roles in a shared space. In *Proceedings of ACM 1998 Conference on Computer-Supported Cooperative Work* (pp. 197-206). New York: Association for Computing Machinery.

Author's Addresses

Mark K. Singley (ksingley@watson.ibm.com), Peter Fairweather (peterf@watson.ibm.com)

Applied Learning Sciences Department, IBM T.J. Watson Research Center; Route 134; Yorktown Heights, NY 10598. Tel. (914) 945-2138.