Contextualizing Reflective Dialogue in a Spoken Conversational Tutor

Heather Pon-Barry, Brady Clark, Karl Schultz, Elizabeth Owen Bratt, Stanley Peters and David Haley
Center for the Study of Language and Information
Stanford University, 210 Panama Street
Stanford, CA 94305, USA
Tel: +1 650 725 2317
ponbarry@csli.stanford.edu
bzack@csli.stanford.edu
schultzk@csli.stanford.edu
ebratt@csli.stanford.edu
peters@csli.stanford.edu
dhaley@csli.stanford.edu

ABSTRACT

In this paper we describe the ways that SCoT, a Spoken Conversational Tutor, uses flexible and adaptive planning as well as multimodal task modeling to support the contextualization of learning in reflective dialogues. Past research on human tutoring has shown reflective discussions (discussions occurring after problem-solving) to be effective in helping students reason about their own actions (Katz, Allbritton & Connelly, 2003). However, presenting information in an understandable manner while leading a reflective discussion is difficult and without contextualization it is easy to confuse and frustrate students. This raises the question: how should intelligent tutoring systems effectively contextualize learning in a reflective discussion? We believe that multimodal task modeling, carried out by a flexible and adaptive planning agent, can facilitate this process of contextualization and lead to a more successful dialogue.

Keywords

Intelligent tutoring systems, Spoken dialogue, Reflective tutoring, Multimodality, Speech technology

Introduction

One-on-one human tutoring has been well established as a highly effective mode of instruction: students interacting with expert human tutors have been found to perform 2.0 standard deviations above students in an ordinary classroom setting (Bloom, 1984). In attempts to emulate the effectiveness of human tutors, many developers of intelligent tutoring systems have begun incorporating natural language dialogue into their tutorial systems. While much of this work has focused on using natural language dialogue during the problem-solving session (Graesser et al., 2000; VanLehn et al., 2002; Michael et al., 2003; Litman & Silliman, 2004), very little work has focused on using reflective dialogue after problem-solving (Katz, Allbritton & Connelly, 2003). Various challenges arise while leading discussions after problem-solving that do not come up while leading discussions during problem-solving. For example, students often have difficulty remembering details from their problem-solving session, especially if the session was complicated. We believe that with sufficient contextualization, an intelligent tutoring system can overcome these challenges and also achieve the benefits of reflective dialogue.

In order to successfully converse about past events and to lead a discussion that addresses a student's "trouble spots," a reflective, conversational tutor must be able to contextualize information from the problem-solving session. For example, consider the dialogue excerpt shown below (from the transcript of an actual dialogue with SCoT). In this dialogue, the tutor's goal is to discuss three particular actions the student failed to do (turn 3), but in order to ask the student about them the tutor first describes the events surrounding the time frame in question (turns 1 and 2) to facilitate student understanding.

1	Tutor:	Ok, the next crisis was a flood here [highlights compartment] in Engine room No. 2 (compartment 4-254-0-E).
2	Tutor:	After the flood alarm sounded, the first thing you did was order repair team 5 to dewater the compartment.
3	Tutor:	However, there are 3 other things you should have done before ordering dewatering. What is one of them?

Furthermore, effective human tutors often use student behavior during the dialogue to guide the manner in which new information is presented (Merrill et al., 1992). For example, if a tutor employs multiple hinting strategies and a student seems to do well using strategy A but is confused by strategy B, then the tutor ought to take this into account when deciding between strategies A and B in subsequent dialogue. Issues such as these have motivated the design choices made in our development of SCoT. In this paper, we describe the architecture and functionality of SCoT's tutorial component and explain how it allows SCoT to facilitate contextualized and reflective tutorial dialogue.

Our Spoken Conversational Tutor has recently been used in three separate evaluation studies. The first study measured the effectiveness of SCoT on damage control novices (Pon-Barry et al., 2004); the results showed tutorial dialogues with SCoT to be more effective than practice alone, and also suggested that the accuracy of the speech recognition did not affect learning. The second study compared the relative effectiveness of two tutoring strategies, again on damage control novices (Pon-Barry, 2004); the results showed that subtle variation in tutorial language can effect overall learning gains. The third study compared the effectiveness of four combinations of multimodal input and output in SCoT on students at the US Naval Academy; the results are currently being analyzed.

Effectiveness of Reflective Tutorial Dialogue

Integrating new information with existing knowledge is a fundamental characteristic of learning (Akhras & Self, 2000). Past research has shown that human tutors can ease this integration process by eliciting self-explanations from the student (*self-explanation* is the process of describing problem-solving steps in one's own words) (Chi et al., 1994). Some dialogue-based tutoring systems have taken the approach of eliciting natural language explanations during problem-solving (e.g., Aleven, Koedinger & Popescu, 2003), but very few have attempted to elicit these explanations after problem-solving. Recent studies provide evidence suggesting that dialogues occurring after problem-solving may be better at eliciting student explanations. For example, one analysis comparing dialogues during problem-solving to reflective dialogues showed that students are more likely to ask questions and to discuss their reasoning processes in the reflective dialogues (Katz, O'Donnell & Kay, 2000). Other analyses have shown that reflective dialogues more frequently involve multi-step interchanges between the tutor and the student (Moore, 1996; Rose, 1997). In addition, a recent study on the instructional role of reflective dialogue found that students who were asked reflective questions by the tutor learned more (as measured by pre-test and post-test scores) than those receiving no reflective questions (Katz, Allbritton & Connelly, 2003).

This evidence suggests that intelligent tutoring systems that can support reflective dialogue have the potential to be more effective than those that do not. However, in order to allow students to integrate new information with existing knowledge during a post-session discussion, a reflective tutor must first have the capability to contextualize the information it presents. We believe that multimodal dialogue-based interaction, carried out by a flexible and adaptive planning agent, can facilitate this process of contextualization.

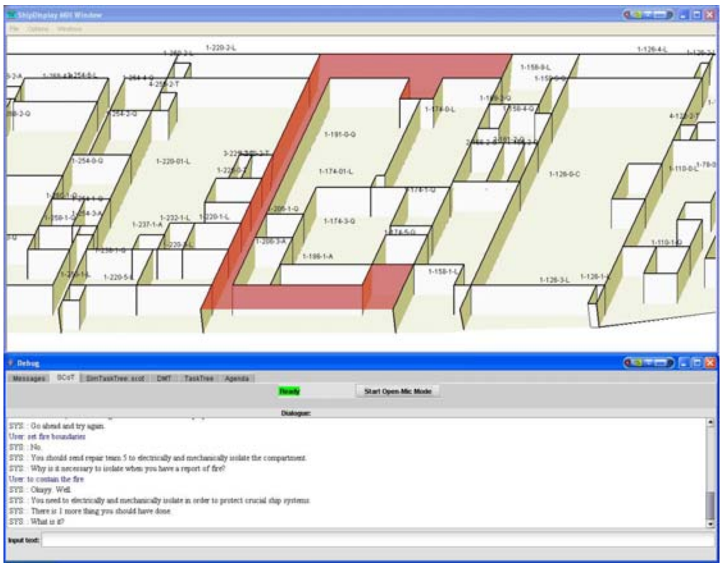
Overview of SCoT

Our approach to tutorial dialogue is based on the assumption that it is a *joint activity*—an activity in which participants have to coordinate with each other in order to succeed (Clark, 1996). Moving a desk, playing a duet, and shaking hands are all examples of joint activities. Joint activities can be subdivided into two types of activities—*basic* and *coordinating*. In most tutorial dialogue the basic activity is solving problems; this is supported by coordinating activities such as identifying gaps in knowledge, verifying understanding, hinting, and the like. Further, the communicative acts by students and tutors can only be properly understood, analyzed, and simulated by viewing them in relation to the current state of their problem solving—as they see it. In other words, the structure of the dialogue is a consequence of the basic joint activity that the dialogue works in service of.

Following this hypothesis, SCoT's architecture separates conversational intelligence (e.g. turn management, use of discourse markers such as 'so' and 'OK') from the activity that the dialogue accomplishes (in this case, reflective tutoring). SCoT is developed within the Architecture for Conversational Intelligence (Lemon, Gruenstein & Peters, 2002), a general purpose architecture which supports multimodal, mixed-initiative dialogue.

SCoT-DC (Spoken Conversational Tutor for Damage Control), the current instantiation of our tutoring system, is applied to the domain of shipboard damage control. Shipboard damage control refers to the task of controlling fires, floods, and other critical events that occur aboard Navy vessels. Students carry out a reflective discussion with SCoT-DC after completing a problem-solving session with DC-Train (Bulitko & Wilkins, 1999), a fast-paced, real-time, multimedia training environment for damage control. Because problem-solving in DC-Train occurs in real-time, reflective tutorial dialogue is more appropriate than tutorial dialogue during the simulation. Because crises co-occur and demand immediate attention (e.g., in an average scenario there may be 4 fires and 2 floods going on simultaneously), accurate contextualization is a challenging task.

Figure 1 below shows the graphical user interface of SCoT-DC. The bottom window contains a history of the tutorial dialogue; the top window is the *common workspace*—a space where both student and tutor can zoom in or out and select (i.e., point to or circle) compartments, regions, or bulkheads (lateral walls) in the ship. In Figure 1, the tutor is "pointing" to the location of a crisis by zooming in to the deck it is on and highlighting the compartment.



SYS : Go ahead and try again.
User: set fire boundaries
SYS : No.
SYS : You should send repair team 5 to electrically and mechanically isolate the compartment.
SYS : Why is it necessary to isolate when you have a report of fire?
User: to contain the fire
SYS : Okayy. Well.
SYS : You need to electrically and mechanically isolate in order to protect crucial ship systems.
SYS : There is 1 more thing you should have done.
SYS : What is it?

Figure 1. SCoT-DC Graphical User Interface

SCoT is composed of three primary components: a dialogue manager, a knowledge representation, and a tutor. These three components, as well as the natural language tools that support spoken interaction, are described in the next four sections.

Dialogue Manager

The dialogue manager handles aspects of conversational intelligence, helping the tutor interpret student utterances in context. It contains the following dynamically updated components:

- The *Dialogue Move Tree*: a structured history of dialogue moves and dialogue threads
- The A*ctivity Tree*: a temporal and hierarchical representation of the past, current, and planned activities initiated by either the tutor or the student (see Figure 3)
- The *System Agenda*: issues to be raised by the system
- The *Salience List*: objects referenced in the dialogue thus far, used for anaphora resolution
- The *Pending List*: questions asked by the system but not yet answered
- The *Modality Buffer*: a place to store gestures for later resolution

The activity tree serves as the communication interface between the tutor component and the rest of the dialogue manager. Each activity initiated by the tutor corresponds to a tutorial goal such as discussing actions the student forgot to perform, or drilling the student on a particular knowledge area. The decompositions of these goals are specified by activity recipes contained in the *recipe library* (see section below). An in-depth description of SCoT's dialogue management module can be found in Clark et al. (2005).

Knowledge Representation and Reasoner

The knowledge representation provides a domain-general interface to domain-specific information. In accordance with production-system theories of cognition (Anderson, 1993), knowledge specifying causal relationships between problem states (events and crises on the ship) and student actions is encoded in a set of production rules. A knowledge reasoner operates over this production system to provide the tutor with procedural explanations of domain-specific actions as well as information about the student's problem-solving session.

Tutor

The tutor consists of two components: one for planning and executing tutorial activities, and one that contains recipes specifying how to decompose these activities into other tutorial activities or into low-level actions. These components are described in detail below.

Natural Language Components

The natural language components that make the spoken dialogue possible include a bidirectional unification grammar and off-the-shelf tools for automatic speech recognition and text-to-speech synthesis. Incoming student utterances are recognized using Nuance's Automatic Speech Recognizer (www.nuance.com) with a language model compiled from a Gemini natural language understanding grammar. Gemini (Dowding et al., 1993) translates word strings from Nuance into logical forms, which the dialogue manager interprets in context and routes to the tutor. The system responds to the student via a FestVox (festvox.org) limited domain synthesized voice.

SCoT's Tutorial Architecture

SCoT's tutor component contains the tutorial knowledge necessary to plan and carry out a flexible and coherent tutorial dialogue. One aspect of leading a reflective discussion is determining how to contextualize information in the most effective manner. Students will likely provide evidence during the dialogue that alters the tutor's original assessment as well as their plan for how to contextualize information. This emphasizes the need for a planning architecture that allows for revisions to the original dialogue plan. We have chosen an approach that separates tutorial knowledge (i.e. how to lead a tutorial dialogue) from all other sources of information (e.g. domain knowledge, student model). The tutorial knowledge is divided between a *planning and execution system* and a *recipe library*. Figure 2 depicts how the planning and execution system and the recipe library fit into the overall SCoT architecture. In Figure 2 below, ASR (Automatic Speech Recognition) refers to the speech recognizer, and TTS (Text-to-Speech) refers to the speech synthesizer.

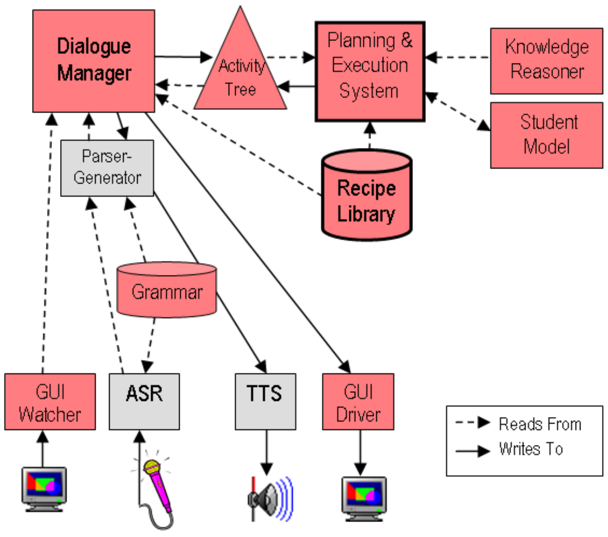


Figure 2. SCoT architecture

Planning and Execution System

The *planning and execution system* is responsible for selecting initial dialogue plans, revising plans during the dialogue, classifying student utterances, and deciding how to respond to the student. All of these tasks rely on external knowledge sources such as the *knowledge reasoner*, the *student model*, and the *activity tree* (collectively referred to as the *Information State*). The Information State provides the tutor with information such as the current context of the dialogue, or a history of the student's interactions with SCoT. By separating tutorial knowledge from other knowledge sources, we not only create a tutor that is reusable in other domains (Schultz et al., 2003), we also provide a clean and consistent way for each component of the tutor to access information pertaining to the current context. This allows SCoT to lead a flexible dialogue and to continually reassess information from the Information State in order to select the most appropriate tutorial tactic. The planning and execution system "executes" tutorial activities by placing them on the activity tree, where they are then interpreted and executed by the dialogue manager.

Recipe Library

The *recipe library* contains activity recipes that specify how to decompose a tutorial activity into other activities and low-level actions. An *activity recipe* can be thought of as a tutorial goal and a plan for how the tutor will achieve the goal. The recipes are written in a scripted language (Gruenstein, 2002) which allows for automatic translation into system activities—the same activities that are stored in the activity tree (each node on the activity tree corresponds to one tutorial goal). Consider the following excerpt from a dialogue that highlights the capabilities of SCoT:

1	Tutor:	OK, in this session there was a fire here [highlights compartment], in the supply department storeroom.
2	Tutor:	You had difficulty controlling this fire, so we should review the various areas of damage control that were covered in this problem.
3	Tutor:	After the fire alarm sounded, you sent repair 3 to set fire boundaries. However, there is one other thing you should have done before setting fire boundaries. Can you tell me what it is?
4	Student:	I'm not sure.
5	Tutor:	Well, the supply department storeroom is an unmanned space. When the alarm sounds, are you certain that there really is a fire?
6	Student:	No, I guess not.
7	Tutor:	Can you tell me now what you forgot to do?
8	Student:	I should have sent investigators to verify the alarm.
9	Tutor:	Yes, that's right.

This dialogue illustrates one tutorial approach to contextualizing the information being presented (namely, stepping through the student's actions). Other approaches to contextualization include discussing an analogous hypothetical situation, or exhaustively recreating the details of a problem-solving session. The dialogue excerpt above corresponds to the tutorial goal *discuss_problem_solving_sequence* (see recipe in Figure 4). After the tutor puts this activity on the activity tree, the system executes the recipe. This causes the activity to be expanded into four more-specific activities (i.e., subgoals). The activity tree in Figure 3 shows this decomposition. Note that in Figure 3 the activity *situate_problem_context* has also been expanded, but the other three sub-goals are not yet expanded—this diagram represents the state of the activity tree after turn (1) and before turn (2) in the dialogue excerpt above.

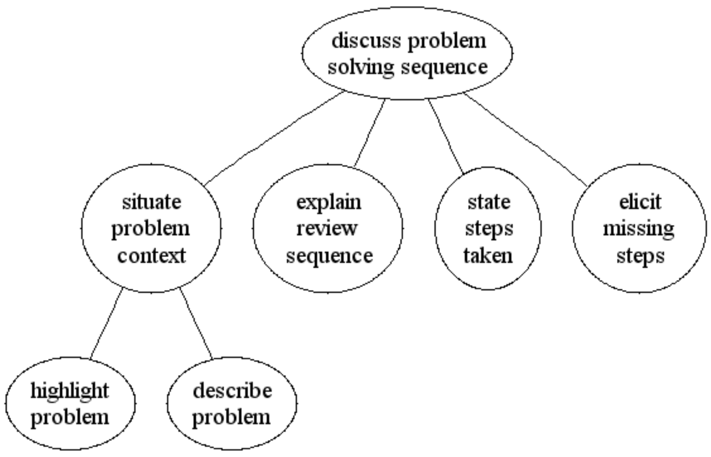


Figure 3. Sample Activity Tree

The tutorial goals (activities) in Figure 3 give rise to a contextualized dialogue in the following ways:

- In turn (1) of the dialogue, it is the tutor's first mention of this problem, so the *situate_problem_context* activity is added to the activity tree, and the tutor describes the type of problem while highlighting its location in the ship display (regions are colored according to the type of compartment).
- In turn (2) of the dialogue, the tutor tells the student why it chose to review this sequence so that the student will understand the tutor's subsequent turns. This corresponds to the activity *explain_review_sequence*.
- In turn (3) of the dialogue, the tutor contextualizes the problem by reminding the student what they did (they sent repair 3 to set fire boundaries). This corresponds to the activity *state_steps_taken*.
- Also in turn (3), the tutor asks the student what step of the sequence they omitted. Since the student does not provide the information the tutor is looking for (in turn (4)), the tutor provides further information about the context (turn (5)), and reasks the question (turn (7)). This interaction is specified in the decomposition of the *elicit_missing_steps* activity (decomposition not shown in Figure 3).

Figure 4 below shows the recipe corresponding to the tutorial goal *discuss_problem_solving_sequence*. An activity recipe contains three sections: the *DefinableSlots*, the *MonitorSlots*, and a *Body*. The *DefinableSlots* specify what information is passed into the recipe, the *MonitorSlots* specify which parts of the information state are used in determining how to execute the recipe, and *Body* specifies how to decompose the activity into other activities and low-level actions. The recipe below decomposes the activity of discussing a problem solving sequence into either three or four other activities (depending on whether the problem has already been discussed). When this recipe is executed, these activities (i.e., subgoals) are added to the activity tree, and the tutor begins to process their respective recipes.

```
Activity
<discuss_problem_solving_sequence> {

  DefinableSlots {
    currentProblem;
  }

  MonitorSlots {
    currentProblem.alreadyDiscussed;
  }

  Body {
    if (!currentProblem.alreadyDiscussed)
{
      situate_problem_context;
    }
    state_review_purpose;
    state_steps_taken;
    elicit_missing_steps;
  }
}
```

Figure 4. Sample activity recipe (I)

The activity recipe scripting language provides a framework for expressing these tutorial tactics and contextualization strategies. Furthermore, the modular nature of the recipes makes it easy to test the effect of different pedagogical and conversational approaches to contextualization. In the second evaluation study mentioned above, one such manipulation of activity recipes was used to alternate between tutorial strategies having subtle variations in language. The results from this experiment suggested that tutors who explicitly refer back to prior dialogue and paraphrase student utterances produce larger learning gains than tutors who used alternative linguistic devices (Pon-Barry, 2004).

Multimodal Interaction

Multimodal interaction is another important way that SCoT contextualizes information while leading reflective dialogues. By coordinating spoken and visual input and output in the *common workspace*, the tutor has increased flexibility in how it chooses to present information. One way in which the tutor contextualizes problems being discussed is by highlighting compartments in certain colors while speaking to indicate the location of the crisis and the compartment type. An example of this coordination can be seen in the way the activity *situate_problem_context* (shown below Figure 5) is decomposed into both visual and spoken actions.

```
Activity <situate_problem_context> {

  DefinableSlots {}

  MonitorSlots {
    currentProblem.type;
    currentProblem.location;
  }

  Body {
    highlight_problem(currentProblem.location);
    describe_problem(currentProblem.type,
      currentProblem.location);
  }
}
```

Figure 5. Sample Activity Recipe (II)

Because the dialogue in SCoT is spoken rather than typed, students are free to use their hands to make gestures in the common workspace while they are speaking. This allows them to "point to" compartments, regions, or bulkheads (for setting boundaries) in the ship display while explaining an action they took in the session or asking a hypothetical question. This coordination of speech and gesture allows SCoT to support interchanges such as:

1	**Tutor:**	If there is a fire here [highlights compartment], in compartment 1-126-0-A, which bulkheads should you set fire boundaries on?
2	**Student:**	I should set primary boundaries here [selects bulkhead], and here [selects other bulkhead].

This hypothetical dialogue shows a student selecting bulkheads on the ship in conjunction with their verbal input ("I should set primary boundaries *here* and *here*"). Another way for the student to incorporate gesture with speech is to drag boundaries forward or backward while describing the action, e.g., "I set this boundary too far forward. It should go here instead." or "I should have placed the aft boundary closer to the compartment, like this." Figure 6 illustrates what the common workspace looks like when a student is selecting particular bulkheads on the ship.

Studies investigating how people combine speech with gestures and diagrams have suggested that participants construct shared models of what they are discussing in order to facilitate communication of difficult content (Engle, 1998). Allowing the student to explain their reasoning while pointing to objects in the workspace creates a common mode of communication between the student and the tutor, and makes it easier for the tutor to know if the student is contextualizing the problem appropriately. This is one way in which multimodal interaction is extremely helpful in contextualizing reflective tutorial discussions.

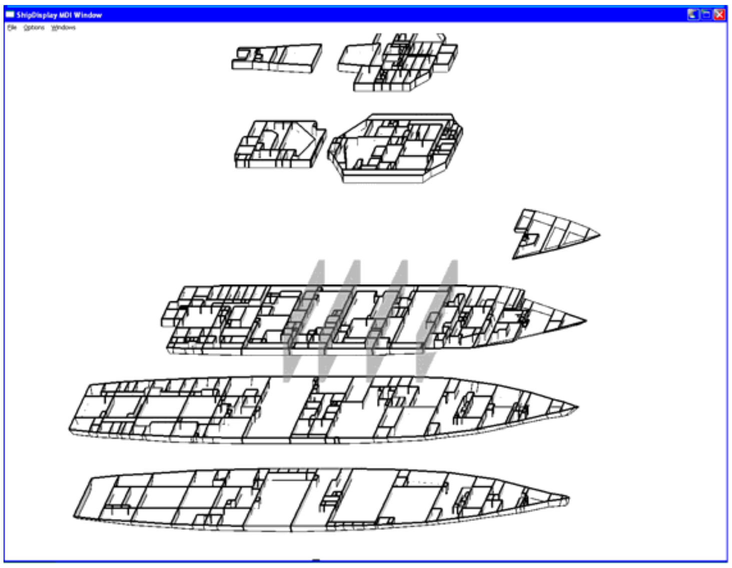


Figure 6. Common Workspace in boundary selection mode

The common workspace also incorporates a symbolic representation of the changing states of various crises on the ship. Figure 7 below shows an example of this symbology, which was incorporated into SCoT-DC for the third (most recent) evaluation study. Standard damage control symbology starts with a line drawn from the affected compartment in a ship diagram out to the side, where lines incrementally build triangles with each action (e.g., the report of smoke, the start of clearing smoke, and the completion of clearing smoke). SCoT-DC's use of this symbology is in line with standard Navy practice and training.

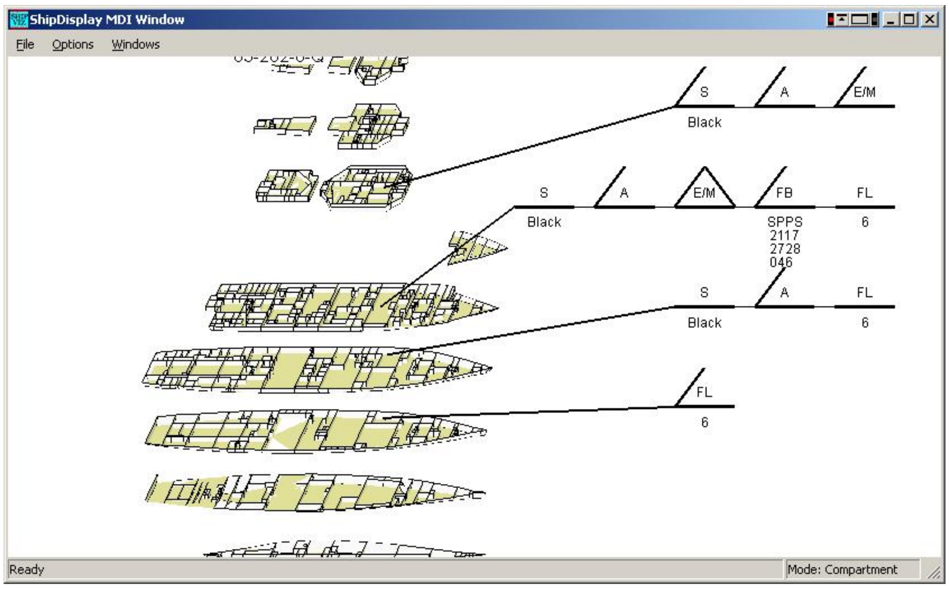


Figure 7. Common Workspace with DCA Symbology

Using this symbology allows the tutor to concisely present a sequence of events in a single depiction. The tutor can rely on the fact that the student is seeing the overall context of the problem state, while the tutorial dialogue concentrates on particular errors. The tutor can also show repeated patterns of mistakes by highlighting portions of symbology, without having to take time to list each one in words. For example, if the student repeatedly forgets to desmoke compartments where smoke has been reported, the system can present the symbology for an entire session, and color lines for desmoking in red, showing the student where the actions were missing. This color-coding, when combined with verbal output, presents combinations of crises in such a way that a student can easily understand what the state of the ship was, and make connections between their mistakes across problems.

Conclusion

In this paper we presented the framework of SCoT, our Spoken Conversational Tutor, and described how it uses multimodal interaction to support the contextualization of dialogue in a reflective tutorial discussion. By separating tutorial knowledge from domain knowledge and by writing activity recipes in a modular way, we have a framework that makes it easy to revise plans as the information state changes and appropriately contextualize the conversation through dialogue and through gesture. This framework is domain independent and has the potential to support reflective dialogue in any number of educational domains.

We are continuing development efforts to expand the recipe library and address several aspects of conducting reflective tutorial dialogues. One focus is to support self-explanation, in which students use free-form language to explain their own reasoning. A second focus is to round out further tactics for contextualization through graphic support of system and user speech, and experimentally evaluate their comparative effects.

Acknowledgments

This work is supported by the Office of Naval Research under research grant N000140010660, a multidisciplinary university research initiative on natural language interaction with intelligent tutoring systems. Further information is available at http://www-csli.stanford.edu/semlab/muri.

References

Akhras, F., & Self, J. (2000). System intelligence in constructivist learning. *International Journal of Artificial Intelligence in Education, 11,* 344-376.

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

Aleven, V., Koedinger, K., & Popescu, O. (2003). A tutorial dialog system to support self-explanation: Evaluation and open questions. In Hoppe, U., Verdejo, F., & Kay, J. (Eds.), *Proceedings of the 11th International Conference on Artificial Intelligence in Education,* Amsterdam: IOS Press, 39-46.

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

Bulitko, V., & Wilkins., D. C. (1999). Automated instructor assistant for ship damage control. *Paper presented at the 11th Conference on Innovative Applications of Artificial Intelligence,* July 18-22, 1999, Orlando, FL, USA.

Chi, M. T. H., de Leeuw, N., Chiu, M., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18,* 439-477.

Clark, H. (1996). *Using Language,* Cambridge, UK: Cambridge University Press.

Clark, B., Lemon, O., Gruenstein, A., Bratt, E., Fry, J., Peters, S., Pon-Barry, H., Schultz, K., Thomsen-Gray, Z., & Treeratpituk, P. (2005). A general purpose architecture for intelligent tutoring systems. In Ole Bernsen, N., Dybkjaer, L. &. van Kuppevelt, J. (Eds.), *Advances in Natural Multimodal Dialogue Systems*. Dordrecht: Kluwer, 107-123.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction, 12,* 371-417.

Dowding, J., Gawron, M., Appelt, D., Cherny, L., Moore, R., & Moran, D. (1993). Gemini: A natural language system for spoken language understanding. *Paper presented at the 31st Annual Meeting of the Association for Computational Linguistics*, June 22-26, 1993, Columbus, Ohio, USA.

Engle, R. A. (1998). Not channels but composite signals: Speech, gesture, diagrams and object demonstrations are integrated in multimodal explanations. In Gernsbacher, M. A., & Derry, S. J. (Eds.), *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, Mahwah, NJ, USA. Erlbaum, 321-326.

Graesser, A., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & the Tutoring Research Group. (2000). AutoTutor: a simulation of a human tutor. *Journal of Cognitive Systems Research, 1,* 35-51.

Gruenstein, A. (2002). *Conversational interfaces: A domain-independent architecture for task-oriented dialogues*, Unpublished MS Thesis, Stanford University, USA.

Katz, S., O'Donnell, G., & Kay, H. (2000). An approach to analyzing the the role and structure of reflective dialogue. *International Journal of Artificial Intelligence and Education, 11,* 320-333.

Katz, S., Allbritton, D., & Connelly, J. (2003). Going beyond the problem given: How human tutors use post-solution discussions to support transfer. *International Journal of Artificial Intelligence and Education, 13,* 79-116.

Lemon, O., Gruenstein, A., & Peters, S. (2002). Collaborative activities and multitasking in dialogue systems. In Gardent, C. (Ed.), *Traitement Automatique des Langues, 43* (2), 131-154.

Litman, D., & Silliman, S. (2004). ITSPOKE: An intelligent tutoring spoken dialogue system. *Paper presented at the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics*, Retrieved October 25, 2005, from, http://www.cs.pitt.edu/~litman/demo-final.pdf.

Merrill, D., Reiser, B., Ranney, M., & Trafton, J. G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences, 2* (3), 277-305.

Michael, J., Rovick, A., Zhou, Y., Glass, M., & Evens, M. (2003). Learning from a computer tutor with natural language capabilities. *Interactive Learning Environments, 11* (3), 233-262.

Moore, J. D. (1996). Making computer tutors more like humans. *International Journal of Artificial Intelligence in Education, 7* (2), 181-214.

Pon-Barry, H. (2004). In search of Bloom's missing sigma: Adding the conversational intelligence of human tutors to an intelligent tutoring system, *Unpublished MS Thesis*, Stanford University, USA, Retrieved October 25, 2005, from http://www-csli.stanford.edu/semlab/muri/papers/HeatherPonBarryThesis.pdf.

Pon-Barry, H., Clark, B., Bratt, E., Schultz, K., & Peters, S. (2004). Evaluating the effectiveness of SCoT: a Spoken Conversational Tutor. In Mostow, J. & Tedesco, P. (Eds.), *ITS 2004 Workshop on Dialog-based Intelligent Tutoring Systems,* 23-32, Retrieved October 25, 205, from, http://www-csli.stanford.edu/semlab-hold/muri/papers/ITS_2004_Workshop.pdf.

Rosé, C. P. (1997). The role of natural language interaction in electronics troubleshooting. In *Proceedings of the Eighth Annual International Energy Week Conference and Exhibition*, January 28-30, 1997, Houston, TX, USA.

Schultz, K., Bratt, E., Clark, B., Peters, S., Pon-Barry, H., & Treeratpituk, P. (2003). A scalable, reusable spoken conversational tutor: SCoT. *Paper presented at the AIED 2003 Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom,* Retrieved October 25, 2005, from, http://www.cs.usyd.edu.au/~aied/vol6/vol6_Schultz.pdf.

VanLehn, K., Jordan, P., Rosé, C. P., Bhembe, D., Böttner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., Roque, A., Siler, S., & Srivastava, R. (2002). The architecture of Why2-Atlas: A coach for qualitative physics essay writing. *Lecture Notes in Computer Science, 2363*, 158-167.