

# A General Purpose Architecture for Intelligent Tutoring Systems

Brady Clark, Elizabeth Owen Bratt, Oliver Lemon, Stanley Peters, Heather Pon-Barry,  
Zack Thomsen-Gray, & Pucktada Treeratpituk

Stanford University

Center for the Study of Language Information

Stanford CA 94305-4115 USA

{bzack,ebratt,lemon,peters,ponbarry,ztgray,pucktada}@csli.stanford.edu

## Abstract

The goal of the Conversational Interfaces project at CSLI is to develop a general purpose architecture which supports multi-modal dialogues with devices. Our systems use a common software base consisting of the Open Agent Architecture, Nuance speech recogniser, Gemini (SRI's parser and generator), Festival speech synthesis, and CSLI's "Conversational Intelligence Architecture" (CIA). This paper focuses on one application of this architecture — an automated tutor for shipboard damage control. We discuss the benefits of adopting this architecture for intelligent tutoring.

## Keywords

dialogue system architecture, tutorial dialogue, intelligent tutoring system, multi-modal

## 1 Introduction

Multi-modal, activity-oriented dialogues with devices present a challenge for dialogue system developers. Conversational interaction in these contexts is mixed-initiative and open-ended. Consider dialogue with an intelligent tutoring system (ITS). Dialogue can be unpredictable in tutorial interactions. The user may need to query the system; e.g., ask a definitional question. Further, the tutor must have a way of reacting to various types of user input; e.g., by adjusting the tutorial agenda

when the student asks for clarification about past topics of discussion.

In this paper we discuss a new general purpose architecture for intelligent dialogue systems which addresses these issues: the Conversational Intelligence Architecture (CIA) developed at CSLI.

The CIA has previously been used in a dialogue system for multi-modal conversations with a robot helicopter (the WITAS system; Lemon et al. 2001, 2002). We focus on a parallel deployment of this architecture in the domain of automated tutoring. We will first discuss the ITS we are developing for shipboard damage control. Next, we discuss the CIA for dialogue systems and what benefits it has for intelligent tutoring.

## 2 An Intelligent Tutoring System for Damage Control

Shipboard damage control refers to the task of containing the effects of fire, explosions, and other critical events that can occur aboard Naval vessels. The high-stakes, high-stress nature of this task, together with limited opportunities for real-life training, make damage control an ideal target for AI-enabled educational technologies like intelligent tutoring systems.

We are developing an intelligent tutoring system for automated critiquing of student performance on a damage control simulator (Clark et al. 2001). The simulator is DC-TRAIN (Bulitko and Wilkins 1999), an immersive, multimedia training environment for damage control. DC-TRAIN's training scenarios simulate a mixture of physical phenomena (e.g., fire) and personnel issues (e.g., ca-

sualties). Figure 1 provides a sample of the type of tutorial interaction our system aims to support.

Conversation with automated tutors places the following requirements on dialogue management (see Lemon et al. 2001, Clark 1996):

1. *Mixed-initiative*: in general, both the user and the system should be able to introduce topics
2. *Open-ended*: there are not rigid pre-determined goals for the dialogue

In the next section, we discuss a general purpose architecture for dialogue systems which meets these two demands.

### 3 An Architecture for Multi-modal Dialogue Systems

To facilitate the implementation of multi-modal, mixed-initiative interactions we use the Open Agent Architecture (OAA) (Martin et al. 1999). OAA is a framework for coordinating multiple asynchronous communicating processes. The core of OAA is a ‘facilitator’ which manages message passing between a number of encapsulated software agents that specialize in certain tasks (e.g., speech recognition).

Our system uses OAA to coordinate the following agents:

1. The **Gemini** NLP system (Dowding et al. 1993). Gemini uses a single unification grammar both for *parsing* strings of words into logical forms (LFs) and for *generating* sentences from LF inputs. This agent enables us to give precise and reliable meaning representations which allow us to identify dialogue moves (e.g., *question*) given a linguistic input; e.g., the question “What happened next?” has the LF: (`ask(wh([past,happen])`)).
2. A **Nuance** speech recognition server, which converts spoken utterances to strings of words. The Nuance server relies on a language model, which is compiled directly from the Gemini grammar,

ensuring that every recognized utterance is assigned a LF.

3. The **Festival** text-to-speech system, which ‘speaks’ word strings generated by Gemini.
4. The **Conversational Intelligence Architecture**, which coordinates inputs from the user, interprets the user’s dialogue moves, updates the dialogue context, and delivers speech and graphical outputs to the user. This agent is discussed in Section 4.

Agents 1-3 are ‘off-the-shelf’ dialogue system components (apart from the Gemini grammar, which must be modified for each application). The CIA agent (4) was written in Java for dialogue management applications in general. This OAA/Gemini/Nuance/Festival/CIA architecture has also been deployed successfully in Lemon et al. 2001, 2002.

### 4 Dialogue Management

Dialogue Management with the CIA makes use of several recent ideas in dialogue modeling, described in detail in Lemon et al. (2001, 2002). It creates and updates an *Information State*, corresponding to a notion of dialogue context. Dialogue moves (e.g., *assertion*, *wh-question*) update information states. A user’s dialogue move might send a response to the system, elicit an assertion by the system, or prompt a follow-up question. The system itself generates dialogue moves which are treated just like the user’s conversational contributions.

The CIA includes the following dynamically updated components (see Lemon et al. 2002 for full details):

1. The *Dialogue Move Tree*: a structured history of dialogue moves and ‘threads’, plus a list of ‘active nodes’
2. The *Activity Tree*: a temporal and hierarchical structure of activities initiated by the system or the user, plus their execution status

T: There were two fires, and you put out both of them. You also handled one flood. The first fire was in the officer's baggage room. For a fire, what's the first thing you should do?

S: Send a repair team to set fire boundaries on the compartment.

T: No, you missed a step. What comes before setting fire boundaries?

S: I'm not sure.

T: Okay. First send a team to investigate. For the second fire, you also forgot to send a team to investigate.

T: Okay. After setting fire boundaries, what should you do?

S: Order a repair team to fight the fire.

T: No, you missed a step. What should you do before sending a repair team to fight the fire?

S: Isolate the space.

T: Yes. Always electrically and mechanically isolate a space before fighting a fire. For the second fire, you also forgot to isolate the space.

Figure 1: A Sample dialogue: T=tutor/system, S=student

3. The *System Agenda*: the issues to be raised by the system
4. *Saliency List*: the objects referenced in the dialogue thus far, ordered by recency
5. The *Pending List*: the questions asked but not yet answered
6. The *Modality Buffer*: stores gestures for later resolution

The Dialogue Move Tree (DMT) provides a representation of dialogue context in terms of a structured history of dialogue moves. Further, the DMT determines whether or not user input can be interpreted in the current dialogue context, and how to interpret it. Recall the two requirements placed on automated tutors discussed in Section 2. The DMT structure is able to interpret both user and system input as dialogue moves at any time, thus allowing for *mixed-initiative*. Further, the DMT can handle dialogues with no clear endpoint (*open-ended*). In the next section, we discuss further benefits of the CIA for intelligent tutoring systems, both in the

domain of shipboard damage control and in general.

## 5 Benefits of the Conversational Intelligence Architecture

The CIA has the following useful properties:

1. It embodies Clark's (1996) joint activity theory of dialogue, in which dialogue serves the activity the conversational participants are engaged in. By utilizing the Dialogue Move Tree/Activity Tree distinction, we are able to provide a model of the joint activities (the Activity Tree), and follow the structure of dialogue deployed in service of those activities (the Dialogue Move Tree).
2. While other intelligent tutoring systems employ finite-state automata which constrain the dialogue move option space for any input (e.g., AutoTutor; Graesser et al. 2000), our CIA is not a finite-state machine, and is dynamically updated. The latter property is useful for handling unpredictable input and for shifting the agenda in response to user input.

3. The Dialogue Move Tree provides us with a rich representation of dialogue structure, which allows us to return to past topics of discussion in a principled, orderly way. For example, in the domain of shipboard damage control, the automated tutor might compare the handling of a later crisis to the handling of earlier crises. Further, the student might ask for clarification about the reasons for earlier actions, so we would like to be able to return to the earlier topic, and pick up the context at that point, as well as simply referring to the earlier crisis.
4. Dialogue moves used in the different implementations of the CIA are domain-general, and thus reusable across different domains. We are building a library of dialogue moves for use by any type of dialogue system. For example, tutorial dialogue will share with other systems dialogue moves such as *questions* and *answers*, but not others (e.g., *hints*).
5. The architecture separates dialogue management from “back-end” activities, such as robot control or tutorial strategies. In the tutorial case, it provides a high-level representation of the tutorial strategies (in the form of the Activity Tree) accessible by the Dialogue Move Tree.
6. The architecture supports multimodality by way of the Modality Buffer. For example, we are able to coordinate speech input and output with gestural input and output (e.g., the user can indicate a point on a map with a mouse click or the system can highlight a map region).

## Acknowledgments

This work is supported by the Department of the Navy under research grant N000140010660, a multidisciplinary university research initiative on natural language interaction with intelligent tutoring systems.

## References

- Bulitko, V.V. and D.C. Wilkins. 1999. Automated instructor assistant for ship damage control. *Proceedings of AAAI-99*.
- Clark, B., J. Fry, M. Ginzton, S. Peters, H. Pon-Barry, and Z. Thomsen-Gray. A Multi-Modal Intelligent Tutoring System for Shipboard Damage Control. *Proceedings of IPNMD-2001*: 121-125.
- Clark, H.H. Clark. 1996. *Using Language*. Cambridge University Press.
- Dowding, J., J. Gawron, D. Appelt, J. Bear, L. Cherny, R.C. Moore and D. Moran. 1993. Gemini: A natural language system for spoken-language understanding. *Proceedings of the ARPA Workshop on Human Language Technology*.
- Graesser, A., K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz and the Tutoring Research Group. 2000. AutoTutor: a simulation of a human tutor. *Journal of Cognitive Systems Research*. 1: 35-51.
- Lemon, O., A. Bracy, A. Gruenstein and S. Peters. 2001. Information States in a Multimodal Dialogue System for Human-Robot Conversation. *Proceedings Bi-Dialog, 5th Workshop on Formal Semantics and Pragmatics of Dialogue*, pages 57 - 67, 2001.
- Lemon, O., A. Gruenstein and S. Peters. 2002. Collaborative Activities and Multitasking in Dialogue Systems *Traitement Automatique des Langues (TAL, special issue on dialogue)*, ed. Claire Gardent, (to appear).
- Martin, D., A. Cheyer and D. Moran. 1999. The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence* 13, 1-2.