Managing Dialogue Interaction: A Multi-Layered Approach

Oliver Lemon
School of Informatics
University of Edinburgh
2 Buccleugh Place
Edinburgh EH8 9LW, UK
olemon@inf.ed.ac.uk

Lawrence Cavedon
CSLI
Stanford University
220 Panama St
Stanford, CA 94306, USA
lcavedon@csli.stanford.edu

Barbara Kelly
Department of Linguistics
UCSB
Santa Barbara
CA 93106-3100, USA
bfk0@umail.ucsb.edu

Keywords: dialogue management architecture, interaction, communication channel management

Abstract

We present evidence for the importance of low-level phenomena in dialogue interaction and use this to motivate a multi-layered approach to dialogue processing. We describe an architecture that separates content-level communicative processes from interaction-level phenomena (such as feedback, grounding, turn-management), and provide details of specific implementations of a number of such phenomena.

1 Introduction

Real dialogue between human participants involves phenomena that do not so much contribute to the content of communication as relate directly to the interactive process between the participants. This includes turn management, providing feedback, utterance fillers, error and false-start management, and utterance timing.

Recent work on dialogue and natural language processing in general has acknowledged the presence of such phenomena in natural speech, and in some cases the importance of its role in dialogue interaction. However, treatment of such phenomena has generally been part of the standard processing model; for example, some parsers are able to handle fillers such as "um", while recent versions of the TRIPS system (Allen et al., 2001) uses incremental parsing and other techniques to handle a range of related phenomena.

We believe that greater focus on "interaction level" phenomena is appropriate and will lead to benefits in building dialogue systems for more robust natural interaction. In this paper, we outline a two-layer architecture for dialogue systems, where one layer uses a range of "shallow" processing techniques to maintain a smooth interaction between the dialogue participants.

1.1 Managing interaction

The inspiration for a clean separation into a two-layer architecture comes from two sources. Clark (1996) distinguishes between two separate communication tracks, which he calls *communicative* and *meta-communicative*. These are simultaneously occurring communications, the first dealing with the information at hand, and the other relating to the performance itself. Dialogue participants use what Clark refers to as *signals* to refer to the performance itself: e.g. timing, delays, re-phrasing, mistakes, repairs, etc.[1]

A second motivation is work on architectures for robots and autonomous agents embedded in complex, dynamic, unpredictable environments. Several researchers in this area have argued for multi-layered architectures for agents that plan action sequences to achieve some goal or task, but need to react quickly to change in the environment (e.g.

[1] Clark's distinction does not necessarily carry over directly to the design of a dialogue system architecture, but it motivates focus on the low-level communication channel.

(Firby, 1994; Müller, 1996)). In such architectures, the role of the bottom layer is to monitor the environment and initiate appropriate actions within the broader context of the goal-directed plan, which is provided by the higher layer of the architecture. The layers operate independently and asynchronously, but communicate as necessary: e.g. goals and plans are passed down to the execution layer, while observations or problems (which may trigger replanning) are passed up to the planning layer.

We view the process of natural interaction with a dialogue participant as analogous to the interaction with a dynamic environment: dialogue phenomena arise which need to be negotiated (as a new obstacle must be avoided by a robot). In the case of a human user involved in activity-oriented dialogue, timeliness is particularly important in order to keep the user engaged and focussed—otherwise, performance of the joint activity may be adversely affected. In particular, dialogic interaction is a continuous process which cannot be broken without the risk of some breakdown: signal-level phenomena must be handled as smoothly as possible, without necessarily resorting to content-level processes, in order to maintain a tight interaction between the participants.

1.2 A multi-layered architecture

Motivated partially by some of the same issues we discuss here, Allen *et al.* (2001) describe a new architecture for their TRIPS system that breaks dialogue management into multiple asynchronous components. We concur with their concerns but focus on a different architectural shift.

We outline below an architecture that separates interaction-focussed techniques from context-management and conversation planning. An initial version of the architecture has been implemented at the Center for the Study of Language and Information (CSLI) at Stanford University.

This breakdown into separate architectural levels is analogous to the multi-level agent/robot architectures. However, many of the same motivations pertain, especially those related to design considerations (e.g. separating different types of phenomena into different layers) and performance (e.g. high-level planning from low-level execution and monitoring running in parallel[2]). Further, the manner in which Müller and Firby's systems handle reactive tasks (e.g. obstacle avoidance, object tracking, etc.) completely at the low-level whenever possible reflects our view of how certain dialogue interaction phenomena are best handled. Much like these systems, dialogue communicative goals are produced at the higher level and imposed as constraints on the lower-level. Environment-level processes fill in the detail of these goals and handle contingencies which may otherwise prevent the achievement of these goals.

A number of interaction-management techniques are present in the current implementation, including:

- A back-up recognition pass, using statistical processing to extend grammar-based coverage and provide immediate user "help" feedback for unrecognized utterances (Hockey et al., 2003);
- Turn management—timing of system output is governed by monitoring the speech channel and the (prioritized) agenda of speech outputs. If the system need to take the turn, it grabs it using only low-level processing;
- Handling user barge-in—user speech interrupts system output and automatically grabs the turn;
- Immediate Grounding of recognized commands (e.g. system says "OK" immediately after recognizing the user: "fly to the tower");
- NP selection — choosing anaphoric or salient noun-phrases at the point of generation;
- Incremental aggregation of system-generated utterances — appropriately condensing and forming elliptical system output at the point of generation.

While this accounts for only a small number of signals that arise during natural dialogue, the architecture provides a framework for incorporating further techniques—in particular, using shallow

[2]Note: we are talking about very different parallel threads here than those which occur in multi-modal fusion, such as occurs in the SmartKom (Wahlster, 2002) system.

processing—for making use of such signals to provide more natural and robust interactions between dialogue systems and human participants.

In the next section, we describe work from the linguistic and psychology literature that demonstrates the importance of asynchronous interaction-level processing. In Section 3, we propose a specific architecture that provides a framework for integrating various processes for channel-management. In Sections 4 and 5, we describe specifics of the CSLI implementation, outlining first the more abstract dialogue management layer, followed by techniques employed at the interaction layer. In Section 6, we discuss further possibilities and in Section 7 we conclude.

2 The Importance of Channel Phenomena

The standard processing model for dialogue systems involves a sequence of modules from speech recognition to speech synthesis, as illustrated in Figure 1, which essentially illustrates (a simplification of) the original TRIPS architecture, as described in (Ferguson and Allen, 1998). Typically, each module is self-contained and relatively independent of other modules.

Recent findings in the psycholinguistic literature have suggested various shortcomings of this modular approach. For example, work on *alignment* indicates that conversation participants' processing interacts on multiple levels, contravening the strict modular model (Pickering and Garrod, 2003). This is one of the considerations we address below, but we are primarily concerned with other interaction-level phenomena.

One of our prime motivations for an interaction level processing layer is to ensure timely response to interaction. Parsing and processing takes time—this can be alleviated by incremental parsing techniques, but meta-communication signals typically do not need to be interpreted and processed to the same extent as communicative utterances, and instead require immediate attention that precludes full processing.

For example, researchers have looked at the use of *um* and *uh* in conversation and found that these are often used as place-holders for a speaker who wants to maintain their speaking turn (Clark and Fox Tree, 2002). The detection of fillers such as these generally acts to inhibit (to some extent) the listener from interrupting or taking the turn from the current speaker. Hence, not only should such discontinuities not be ignored but they must also be processed immediately in order to maintain the ongoing interaction.

Conversely, listeners also use what is known as *back-channel feedback* to indicate to the speaker that they are listening and paying attention. For English, back-channels include *uh-huh*, *mhm* and *yeah*. Back-channels differ from other devices used to keep a conversation flowing, such as repetitions and collaborative finishes, in that they tend to be non-specific to the current utterance. Moreover, back-channel feedback is often produced without thinking, in response to simple prosodic clues such as a speaker pause, a lowering of speaker pitch, or a rise in speaker intonation (Ward and Tsukahara, 1999).

Most importantly, however, back-channel feedback is important to the speaker.[3] Bavelas *et al.* (2000) investigated how a speaker in a conversation (in this case someone narrating a story) is affected when listener responses are inhibited. They found that speakers with distracted and unresponsive listeners did not finish their stories effectively, measurably faltering at what should have been the dramatic ending. Speakers needed an interlocutor's feedback to be able to maintain fluency and continue the dialogue effectively. Bavelas *et al* also found that response latency in one-on-one conversations is extremely short and may be simultaneous: listeners can provide back-channels without fully listening to the conversation partner and without being responsible for taking up a speaking turn.

These results indicate that the nature of interaction between participants is crucial to the collaborative act of dialogue—signals and feedback that carry effectively no communicative content are still important for keeping the interaction smooth and to ensure that the participants stay attentive and focussed on the task at hand. When the dialogue task involves, say, a human user being guided through a safety-critical activity by an automated system, then such issues are of particular importance.

[3] Allwood (1995) refers to such feedback morphemes as the most important cohesion device in spoken language.

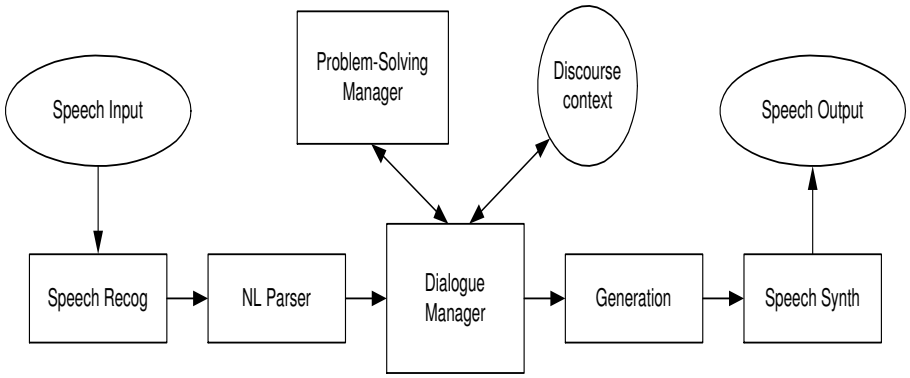

Figure 1: Traditional Dialogue System Architecture

Conversely, communicative behavior contains signals regarding a participant's attention, and in particular may indicate a loss of focus. In tutorial settings—one of the dialogue applications we are specifically concerned with—this can be used to determine students' confidence in their responses. For example, phenomena such as timing between responses, hesitance markers, and intonation can all be implicit clues that a student is having a problem (Graesser et al., 1995).

3 A Two-Level Architecture

The traditional architecture for dialogue systems basically involves a linear approach to processing, as illustrated in Figure 1. In this standard architecture, modules tend to be self-contained and only loosely dependent. Evidence outlined above, particularly that related to alignment, suggests that this tightly encapsulated approach will deal poorly with the interactive nature of real dialogue. Allen *et al*'s (2001) revised TRIPS architecture introduces a more non-linear approach to dialogue processing, with asynchronous processes managing interpretation, generation, and interface to behavioral aspects.

We augment the TRIPS approach by combining multiple processes for interpreting utterances (e.g. structured parsing versus statistical techniques) and for generating responses (e.g. generation from semantic representation versus template-based). More fundamental to the architectural distinction we propose, the processing of an utterance and generating an appropriate response may proceed without full processing by the Dialogue Management component: information gleaned from an utterance will always be passed up to the Dialogue Manager, but to ensure timely response, an appropriate response may be produced directly from a low-level component. Other processes included at the interaction layer detect non-communicative information, such as gaps or delays in the user's speech.

Figure 2 illustrates various aspects of the specific two-level architecture we are developing. The lower level interfaces directly with the user and, importantly, is driven by this interaction. For example the low level includes a Turn Manager which manipulates the speech channel to ensure that:

- user inputs are respected without interruption (except when necessary);
- turn passes to the appropriate participant, based on the highest priority Agenda item and the dialogue move that generated it;
- generated outputs are natural and timely;
- recognized user inputs are acknowledged quickly using simple feedback utterances.

The upper level is responsible for modeling other aspects of the conversational context, as well as communicative goals and intentions. The content (i.e. logical forms) of user utterances are processed using the dialogue model (e.g. updates and adding nodes to the Dialogue Move Tree (Lemon et al., 2002b)), and system utterances are constructed which are in line with the system's communicative

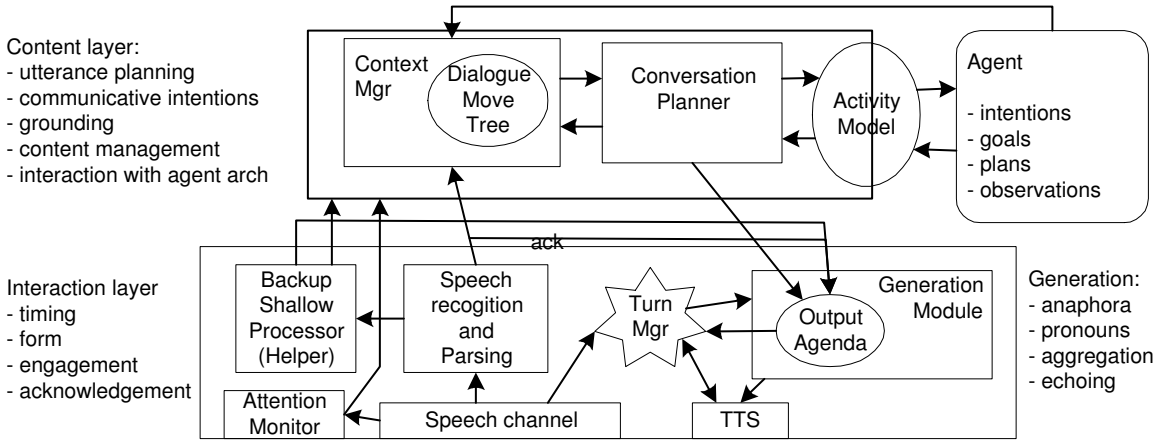

Figure 2: System Architecture

goals and intentions, whether they be imparting information to the user or requesting clarification or further information.

The higher level also interacts with the rest of the agent architecture, mediated by an *Activity Model* (i.e. a representation of the agent activities about which dialogue may occur (Gruenstein, 2002)). The agent may wish to communicate its own goals, the progress of its activities, or report on any observations it makes regarding its environment.

As with multi-layered agent architectures, the two levels operate semi-autonomously and asynchronously: the lower level is driven by tight interaction with the user, while the upper level is driven by longer-range communicative goals from its activities and responses to user utterances. However, various types of information exchange connect the two levels. For instance, user utterances recognized at the lower level must clearly be passed to the content-management level to be parsed and then incorporated into the dialogue context, while high-level communication goals must be passed to the lower level's *Output Agenda* for generation and speech-synthesis.

The Output Agenda plays a crucial role in mediating utterances to be communicated, whether they be system-initiated or responses, and generated from the planner or a low-level component. The Output Agenda is a prioritized list, where an utterance's priority is influenced by a number of factors, such as: whether it is in response to an error or misunderstanding (i.e. "Pardon"); the importance of the communicative content (i.e. an urgent observation); and the dialogue move that generated it (e.g. answering a question). The Agenda runs asynchronously, aggregating multiple utterances when appropriate as well as influencing speaker turn (see below).

Of perhaps greater interest, the interaction level can be used to monitor user engagement and attention in other ways — e.g. time between utterances, speaking rate, use of speech fillers — to detect potential problems as soon as possible, and to provide early warning to the content layer that the user may have, for example, misunderstood some instruction. This can be used to generate a clarification or grounding sub-dialogue, in order to establish mutual understanding before proceeding (thus improving robustness of the system as a whole).

Conversely, expectations at the upper-layer can influence processing at the interaction layer: for example, open points of attachment on the Dialogue Move Tree represent types of utterances the system expects from the user, and these are used to prime the recognition of incoming utterances for faster processing, as well as influencing the turn.

In engineering terms, this division of labour is attractive in that the clarity and modularity of dialogue management is enhanced. Rather than conflating, for example, turn-management with utterance planning in a single generation component of a dialogue system, the separation into multiple levels of processing allows different turn-taking and utterance

planning strategies to be developed independently, and various combinations to be experimented with.

In the rest of the paper, we discuss our dialogue management architecture and, in particular, the techniques employed so far at each of the two levels described here to enhance user experience and improve overall system performance. The current implementation based on the above architecture is still being refined; we focus on the features that have already been implemented.

4 Top-Level Context Management

The approach to dialogue modeling we have implemented is based on the theory of *dialogue games* (Carlson, 1983; Power, 1979), and, for task-oriented dialogues, *discourse segments* (Grosz and Sidner, 1986). These accounts rely on the observation that answers generally follow questions, commands are usually acknowledged, and so on, so that dialogues can be partially described as consisting of *adjacency pairs* of such dialogue moves. The notion of "attachment" of dialogue moves on a Dialogue Move Tree (DMT) (Lemon et al., 2002b) embodies this idea.

An *Activity Tree* represents hierarchical and temporal information about the task-state of the dialogue. Activities are the joint tasks managed by the dialogue: e.g. booking a flight or moving a robot—again, see (Lemon et al., 2002b) for details. Nodes on the Activity Tree can be in various states (*active*, *complete*, *failed*, ...), and any change in the state of a node (typically because of an action by the agent) is placed onto the system's Output Agenda for potential verbal report to the user, via the low-level message selection and generation module.

This level of the architecture is where conversation planning and generation of system-initiated topics occur. Any planned communication (whether it be system-initiated or in response to a user utterance) is put on to the Output Agenda, where it is scheduled for generation.[4] Conversely, true grounding — i.e. acknowledging that an utterance is understood within the context of the rest of the dialogue — only occurs after the utterance has been interpreted with respect to the DMT. Since a simple acknowledgment may already have been generated after recognition, output after interpretation is only needed if a response is required (e.g. the user asked a question), or if a problem is detected (e.g. an ambiguity must be resolved).

Since system communication is planned here, this layer is also the one that interacts with the rest of the agent architecture: any goals, state-changes, or observations that the agent may wish to communicate are added as communicative goals, typically via the Activity Model. For command-and-control applications (e.g. guiding a robot or UAV), system-initiated utterances tend to be fairly short and simple and conversation-planning is minimal; however, for our dialogue-enabled tutorial application (Clark et al., 2001), conversation-planning is quite complex and the system may generate multiple, relatively long utterances on its own initiative.

5 Low-level Conversation Management: Maintaining the Communication Channel

We currently employ a range of shallow processing techniques to maintain a smooth interaction with the human dialogue participant. By "shallow processing" we mean processing that does not necessarily result in or concern itself with the semantic representation or pragmatic interpretation of the utterance in the context of the dialogue. In particular, information at this level is not processed in the context of the Dialogue Move Tree or the Activity Tree.

In the following, we describe a number of the low-level processing techniques currently implemented in our system. Future work will address more of the interaction phenomena described earlier.

5.1 Case study 1: Helper Feedback

In cases where a user utterance is not recognized, the input is passed to a statistical recognizer of wider coverage. This recognizer is often able to detect lexical items and grammatical structures in the input that are not covered by the first (grammar-based) recognizer. In these cases, the results of the second recognition pass are used to inform the user of the system's shortcomings, for example: "The system heard you say 'Look around for a red car', but the system does not know the word 'around'. You could say 'Look for a red car' ".

[4] The order in which outputs are generated, or even whether they end up generated at all, depends on the priority of the corresponding information as well other interactions with the user.

None of these utterances is planned or represented at the top level of dialogue management. They are produced simply to inform the user of a communication breakdown and to try to keep the communication flowing. If the user were to indulge in meta-dialogue about the help message, then that message would need to be represented in the high-level context. However, we present the help message as being generated by a different "helper" agent, which disappears (from the GUI) as soon as the help message is produced, thus discouraging the user from engaging it in dialogue.

User tests have shown that the use of this low level module (which can be installed independently of the high-level dialogue manager) significantly improves task completion (both percentage of tasks completed and time taken). By the fifth task, 100% of users with the helper completed the task as compared with 80% of those without, and those without the helper took on average 53% longer to complete the tasks. For full details of the evaluation see (Hockey et al., 2003).

5.2 Case study 2: Turn Taking

Here we use a turn-marker at the low-level of dialogue processing. The turn can be marked as *user, system* or *none*, and is set in a variety of ways. If the user begins to speak (start-of-speech signal is received from the recognizer) the turn becomes *user* and any system audio output is stopped. If the system needs to take the turn (e.g. if it has urgent information it needs to communicate), but turn is set to *user*, and the user is not speaking, the system will output "Just a moment" and so take the turn before generating its required utterance. Again, note that this turn-grabbing utterance is not planned or represented at the top-level of dialogue moves. It does not need to enter into such high-level plans or representations because it is required only in order to manipulate and maintain the channel, and does not carry any content of its own.

The demonstration system displays a turn marker on the GUI, allowing observers to monitor the changing possession of the turn.

5.3 Case study 3: Incremental aggregation

Aggregation (Appelt, 1985) combines and compresses utterances to make them more concise, avoid repetitious language structure, and make the system's speech more natural and understandable overall. In our system, this process is carried out not at the level of content planning, but at the lower-level of processing, where content logical forms are manipulated (possibly combined) and converted into strings for speech synthesis. Indeed, it is important that aggregation functions at this lower level, because the process needs access to:

- the message to be uttered (A),
- what has just been said (B),
- what is to be said next (C),

and the precise surface form of B is only represented at the low-level. High-level processing only plans the content of the utterance to be generated, and passes it down, and so cannot determine the details of the eventual surface form of the generated utterance.

Aggregation techniques on a prewritten body of text combine and compress sentences that have already been determined and ordered. In a complex dialogue system however, aggregation should produce similarly natural output, but must function incrementally because utterances are generated on the fly. In fact, when constructing an utterance we often have no information about the utterances that will follow it, and thus the best we can do is to compress it or "retro-aggregate" it with utterances that preceded it (see the example below). Only occasionally does the Output Agenda contain enough unsaid utterances to perform reasonable "pre-aggregation".

At the low-level of processing, the generator receives an item (on the Output Agenda) to be converted into synthesized speech. This item consists of a dialogue move type along with some content (e.g. `wh-answer, location(tower)`).

Each dialogue move type (e.g. *report, wh-question, wh-answer*) has its own aggregation rules, stored in the class for that logical form (LF) type. In each type, rules specify which other dialogue move types can aggregate with it, and exactly how aggregation works. The rules note identical portions of LFs and unify them, and then combine the non-identical portions appropriately.

For example, the LF that represents the phrase "I will fly to the tower and I will land at the parking

lot", will be converted to one representing "I will fly to the tower and land at the parking lot" according to the compression rules. Similarly, "I will fly to the tower and fly to the hospital" gets converted to "I will fly to the tower and the hospital".

In contrast, the "retro-aggregation" rules result in sequences of system utterances such as,

```
Sys:  I have cancelled flying to the base
Sys:  and the tower
Sys:  and landing at the school
```

Again, this process happens only at the low-level processing stage of content realization, and needs no access to the high-level representations of dialogue structure, history, and plans. A separate thread running in the Output Agenda component asynchronously performs aggregation as needed and appropriate.

5.4 Case study 4: Choosing NPs

Another low-level process in utterance realization is choosing appropriate NPs – anaphoric expressions such as "it" or "there", or NPs which "echo" those already used by the human operator. Again, this routine does not need access to the high-level dialogue management representations, but only to the list of NPs employed in the dialogue thus far (the *Salience List*).

Echoing is achieved by accessing the Salience List whenever generating referential terms, and using whatever noun-phrase (if any) the user has previously employed to refer to the object in question. Anaphoric phrases are generated whenever the reference object is the same as the one at the top of the Salience List.

As in the case of aggregation, the top level content generation algorithm does not manage the details of utterance realization – this is better handled at the instant that the content logical form is to be translated into a string for the speech synthesizer. Otherwise the top level would have to replan utterances after every intervening dialogue move. This example shows how respecting the multi-level architecture is desirable from an engineering point of view.

6 Current Implementation and Further Possibilities

An initial version of the CSLI dialogue system based on the described architecture has been implemented, and is able to engage in conversations such as illustrated in Figure 3.

The system has been applied to both command-and-control and tutorial applications; this is of interest since the former generally involves user-initiated conversations while in the latter, conversation tends to be system-initiated. The Output Agenda mediates by handling both standard logical forms or generation-templates.

Only a small number of the interaction-level phenomena that arise in human-human dialogue have been implemented, but we believe a number of them could be treated within our framework. For instance, processes at the lower level could detect miscommunication and channel breakdown, and send a request to the top level to replan the long-range dialogue strategy. This is particularly relevant in the tutorial setting, where low-level processes could detect problems with user attention and responsiveness, and prompt a switch to a different high-level strategy. Particularly important for safety-critical applications, but of general use, would be low-level monitoring of channel noise and other environmental factors such as user gestures and gaze. Again, certain combinations of these inputs would have high-level consequences for interpretation and dialogue planning.

Recent work makes use of Icarus (Shapiro, 2001), a reactive planning system that learns and adapts to user behavior, to cover timing and realization of system turns as well as handling delays in the user input. In future, we anticipate that this will allow, for instance, turn-taking facilities to be more easily adapted as personalities or situations require: for example, after noticing a particular event the system may be more likely to interrupt a speaker, or may adapt to become less prone to interruption when interacting with a speaker who responds poorly to system barge-in.

7 Conclusion

We have presented evidence from the psycholinguistic literature that demonstrates the importance of

Figure 3: Demonstration system, November 2002

Utterance	**Dialogue Move**
Operator (O): *Fly around the school*	Command
Helper (H): *I heard you say "Fly around the school" but the system does not know the word "around". You could say "fly to the school"*	Helper feedback
O: *Fly to the school*	Command
Robot (R): *OK.*	Confirmation
O: *Then pick up the medical supplies at the hospital*	Take turn, Command
R: *OK. Wait a minute*	Confirmation, System grab turn
R: *I am supposed to always fly high*	Report Global Constraint
R: *Shall I fly to the hospital at low altitude anyway?*	Negotiate constraint relaxation, Release Turn
O: *Yes. Do it anyway*	Yn-Answer
R: *OK. Wait a minute*	Confirmation, System grab turn
R: *There is a fire at the tower*	Report observation
R: *I am flying to the hospital*	Report Current Activity
O: *Why?*	Why-question, Release Turn
R: *Because I am picking up the medical sup...*	Why-Answer, User barge-in
O: *Stop that.*	Command (Barge-in)
R: *OK. I have cancelled flying there*	Confirmation, Report Activity State, Anaphoric NP
O: *Fight the fire at the tower*	Command
R: *OK. I am flying to the lake*	Confirmation, Report Activity State
O: *Why are you doing that?*	Why-question, Release Turn
R: *Because I am picking up water there.*	Why-Answer, Anaphoric NP

interaction-level phenomena in dialogue. Phenomena such as fillers, timing, and feedback must be dealt with properly and in a timely manner; otherwise, the interaction and even the collaborative activity being performed may be adversely affected.

Our approach is to cleanly separate the architecture of dialogue systems to provide for an asynchronous layer that is designed to handle interaction signals, analogously to agent/robot architectures that include a layer to manage interaction with a dynamic environment. This allows processing outside the full context of a dialogue history when required for processing speed, while allowing the context to still influence such processes when able.

A system has been implemented based on this architecture, containing a range of low-level processes, which we have described here in some detail: shallow-helper feedback; turn-management; aggregation; NP selection. Current work is directed towards incorporating techniques to manage further phenomena—such as predictors of uncertainty and loss of attention—in both command-and-control and tutoring applications.

Acknowledgements

This research was partially supported by the Wallenberg Foundation's WITAS project, Linköping University, Sweden, and by grant number N00014-02-1-0417 from the Department of the US Navy. The dialogue system was implemented while the first author was employed at CSLI, Stanford University.

References

James F. Allen, Bradford W. Miller, Eric K. Ringger, and Teresa Sikorski. 1996. A robust system for natural spoken dialogue. In *Proceedings of ACL*.

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational sys-

tems. In *Proceedings of Intelligent User Interfaces 2001*, Santa Fe, NM.

Jens Allwood. 1995. An activity based approach to pragmatics. In *Gothenburg Papers in Theoretical Linguistics 76*, Dept. of Linguistics, Uni. of Göteborg.

Douglas E. Appelt. 1985. Planning english referring expressions. *Artificial Intelligence*, 26(1):1 – 33.

J. B. Bavelas, L. Coates, and T. Johnson. 2000. Listeners and co-narrators. *Journal of Personality and Social Psychology*, 79:941–952.

Lauri Carlson. 1983. *Dialogue Games: An Approach to Discourse Analysis*. D. Reidel.

Herbert H. Clark and Jean E. Fox Tree. 2002. Using *uh* and *um* in spontaneous speaking. *Cognition*, 84:73–111.

Brady Clark, John Fry, Matt Ginzton, Stanley Peters, Heather Pon-Barry, and Zachary Thomsen-Gray. 2001. Automated tutoring dialogues for training in shipboard damage control. In *Proceedings of SIGdial 2001*.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

George Ferguson and James Allen. 1998. TRIPS: An intelligent integrated problem-solving assistant. In *Proceedings 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573, Madison, WI.

James Firby. 1994. Task networks for controlling continuous processes. In *Proceedings 2nd Int'l Conf. on AI Planning Systems*, pages 49–54.

A. C. Graesser, N. K. Person, and J. P. Magliano. 1995. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology*, 9:1–28.

Barbara Grosz and Candace Sidner. 1986. Attentions, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Alexander H. Gruenstein. 2002. Conversational interfaces: A domain-independent architecture for task-oriented dialogues. Masters thesis, Computer Science Department, Stanford University.

Beth-Ann Hockey, Oliver Lemon, Ellen Campana, Laura Hiatt, Gregory Aist, Jim Hieronymus, Alexander Gruenstein, and John Dowding. 2003. Targeted help for spoken dialogue systems: intelligent feed back improves naive users' performance. In *Proceedings European Assoc. for Computational Linguistics (EACL 03)*.

Oliver Lemon, Alexander Gruenstein, Alexis Battle, and Stanley Peters. 2002a. Multi-tasking and collaborative activities in dialogue systems. In *Proceedings of 3rd SIGdial Workshop on Discourse and Dialogue*, pages 113 – 124, Philadelphia.

Oliver Lemon, Alexander Gruenstein, and Stanley Peters. 2002b. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL)*, 43(2):131 – 154. Special Issue on Dialogue.

Jorge P. Müller. 1996. *The Design of Intelligent Agents—A Layered Approach*. Springer Verlag, Heidelberg, Germany.

Martin Pickering and Simon Garrod. 2003. Toward a mechanistic psychology of dialogue. *Brain and Behavioral Science*. to appear.

Richard Power. 1979. The organization of purposeful dialogues. *Linguistics*, 17:107–152.

Daniel Shapiro. 2001. *Value-driven agents*. Ph.D. thesis, Department of Management Science and Engineering, Stanford University.

Jan van Kuppevelt, Ulrich Heid, and Hans Kamp. 2000. Best practice in spoken language dialogue system engineering. *Natural Language Engineering*, 6.

Wolfgang Wahlster. 2002. SmartKom: fusion and fission of speech, gestures, and facial expressions. In *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, pages 213–225, Kyoto, Japan.

N. Ward and W. Tsukahara. 1999. A responsive dialog system. In Y. Wilks, editor, *Machine Conversations*, pages 169–174. Kluwer.