Multi-Level Architectures for Natural Activity-Oriented Dialogue

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

Lawrence Cavedon
Center for the Study of Language and Information
Stanford University
210 Panama Street
Stanford, CA 94306, USA
lcavedon@csli.stanford.edu

Abstract

We argue for the development of distinct high and low levels of processing in dialogue management. An architecture is described, which cleanly separates high-level communicative processes (e.g. semantic and pragmatic interpretation, content planning), from low-level interaction processes which maintain the communication channel with the user (e.g. feedback, grounding, turn-management). These levels operate asychronously and semi-independently. We present four examples of processes at the interaction-level in an implemented dialogue system: "Helper" feedback, turn-management, incremental aggregation, and generation of NPs.

1 Introduction

Real dialogues between humans involve utterances and gestures whose main purpose is to manipulate and maintain the communication channel, e.g. grounding, turn-taking, signalling misunderstading, requesting repetition, requesting pause, direction, space-filling, feedback. Seeing dialogue as a joint activity (Clark, 1996), particularly when participants are engaged in collaborative tasks, highlights the importance of these adaptive "low-level" processes of channel management. Dialogue participants often also "align" with each others' vocabulary, syntax, and communication styles. On the other hand, dialogue also involves simultaneous high-level processes such as utterance planning and interpretation, which interact with the low-level processes, and vice versa.

For such reasons, we argue that it is appropriate to view dialogue management similarly to the multi-level architectures for goal-oriented autonomous agents, e.g. (Firby, 1994; Müller, 1996):

- a planning and activity-focused layer manages the bulk of the communication arising from the negotiation and performance of tasks or activities (e.g. interpreting new commands, requesting clarification, reporting on task status, dialogue structure, utterance content planning);
- an "interaction" management layer uses typically "shallow" NL processing techniques to maintain the communication channel and respond to any issues that may arise (e.g. managing turn-taking, acknowledging communications quickly, using alternative processing for handling unrecognized utterances, surface realization of utterance content).

As with the agent-architectures, the two levels of processing cannot be entirely independent: data-structures described below mediate communication between the components. In particular, the interaction level may need to notify the activity-planning level if it notices the user behaving in a manner that indicates a problem (e.g. a previously undetected misunderstanding or lack of a required grounding response).

We argue that for a dialogue system to be efficient, to feel "natural" to a human user, and to be satisfactorily robust, it must address the issue of maintaining and mangaging the communication channel with the user. Recently, several systems have been built to manage activity-oriented dialogue between a human and intelligent devices, e.g. (Allen et al., 1996; Lemon et al., 2002b) with some degree of success. These systems have paid limited attention to the need to maintain the communication channel.

While a dialogue system clearly should not overload the user with feedback, it must nonetheless keep the user engaged, and monitor their attention, which will also help to focus on the activity being discussed. Unfortunately, very little work has been done in dialogue systems research regarding low-level interactions in conversation, and still less has been achieved with regard to dialogue management architectures which respect this distinction. Our aim is to argue that dialogue management architectures should embody asychronous and encapsulated levels of processing respecting these high and low levels of conversational interaction, and we present such an architecture.

The following low-level processing techniques are currently implemented in our system:

- 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 prior dialogue move. If the system wants 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.

The resulting system is able to participate in conversations such as the one illustrated in Figure 1.

Thus we have gone some way towards a multi-level architecture for activity-oriented dialogue systems, analogous to the multi-level architectures for goal-oriented autonomous agents, e.g. (Firby, 1994; Müller, 1996). Such agent architectures generally involve a higher, more abstract level at which goals are represented and plans are defined or constructed. A lower-level manages more direct interaction with the environment, or monitors the performance of long-running tasks. For example, the goal-level may plan the route to move a robot to a destination while the lower-level handles obstacle avoidance and corrects for wheel slippage.

Below, we describe a number of techniques we have used at such a "low" interaction level to enhance the user's dialogue experience, and the ways that they interact with high-level dialogue processing. We also suggest ways in which this strategy could be enhanced to address further issues, for instance using reactive planning and learning techniques, e.g. (Shapiro et al., 2001).

2 Motivation for an Interaction Layer

Studies in linguistics and psychology provide evidence for the claim that dialogue consists of two parallel "channels" of interaction, and that participants attempt to co-ordinate and align these channel through conversation. For example:

> "There is co-ordination of both *content*, what the participants intend to do, and *process*, the physical and mental systems they recruit in carrying out those intentions." (Clark, 1996), p. 59.

Human-human dialogue involves immediate acknowledgement and grounding of utterances ("yep" or "okay"), often in the middle of a

Figure 1: 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

speaker's sentence (if that sentence contains multiple information items). Moreover, human dialogue participants often *expect* such behaviour from their partner—the absence of such acknowledgement often makes us pause to wait for it.[1] A dialogue system should exploit such natural patterns of behaviour both when speaking – to assure the human speaker that their utterances are being understood, and when listening – to use such feedback, or lack of it, as early evidence of problems requiring repair.

Other dialogue phenomena that must be accounted for include managing turn-taking, as well as handling participants speaking out of turn. The dialogue system must react quickly to users interrupting it and then decide whether to continue (e.g. if the user seems to be simply acknowledging a portion of the utterance), or whether to stop what it is saying and revise its planned output.

In engineering terms, such a division of labour is also attractive in that the clarity and modularity of dialogue managment 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.

Allen *et al.* have also noted the importance of accounting for such phenomena and have used it as one of the motivations for the revised architecture of their TRIPS system (Allen et al., 2001). They separate dialogue management into a number of different asynchronous modules, and use incremental interpretation and generation to ensure high reactivity to user activity. We believe

[1]For example, think of giving someone a phone number and waiting for them to echo the first half of it before continuing.

that identifying the two streams of dialogue (i.e. activity-oriented versus channel-maintenance) is an important one, and that clean separation into the separate layers allows use of shallow techniques for fast response without need to access the content planning layer.

3 A Two-Level Architecture

Figure 2 illustrates various aspects of the particular two-level architecture which 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),
- the turn passes to the appropriate participant, based on the preceding dialogue move (passed in from the top level),
- 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 goals and intentions, whether they be imparting information to the user or requesting clarification or further information. This 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 about its 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.

Perhaps of more 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 level 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 bias the recognition of incoming utterances for faster processing, as well as influencing the turn.

In the rest of the paper, we discuss our dialogue management architecture and, in particular, the techniques employed at each of the two levels described here to enhance user experience and improve overall system performance.

4 Top-level context management

The approach to dialogue modelling 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.

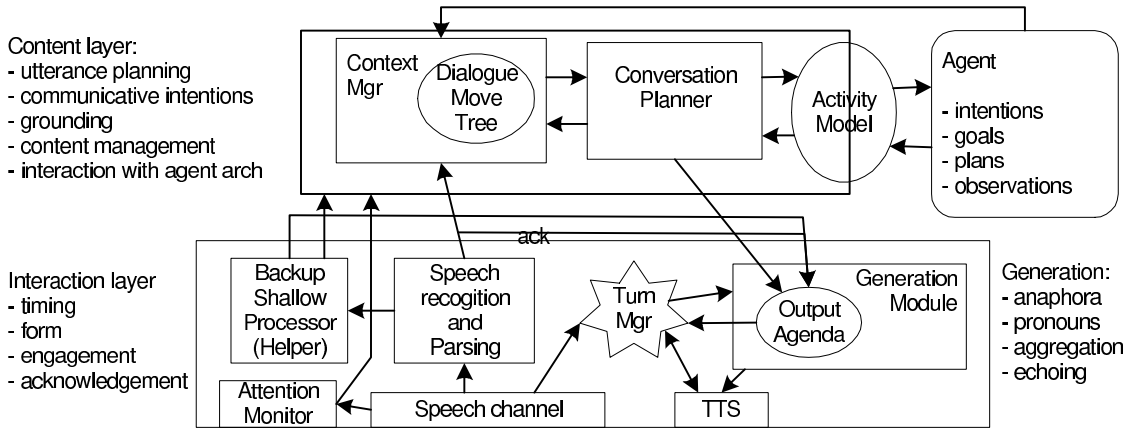

Figure 2: System Architecture

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 possible verbal report to the user, via the low-level message selection and generation module.

This level of the architecture is also where conversation planning and generation of system-initiated topics occurs. Any planned communication (whether it be in response to a user utterance or system-initiated) is put on to the Output Agenda, where it is scheduled for generation.[2] 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 acknowledgement 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).

[2]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.

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.

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.

5.1 Case study 1: "Helper Feedback"

In cases where the 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' ".

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, but it 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 watch 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,

```
System:  I have cancelled flying
to the base
System:  and the tower
System:  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.

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 Further Possibilities

There are many other dialogue phenomena that could be treated along these lines. 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 important in the genre of tutorial dialogue (Zinn et al., 2002), 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.

We are currently investigating using Icarus (Shapiro et al., 2001), a reactive planning system that learns and adapts to user behavior, for implementing and integrating different low-level modules. We hope 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 argue for the development of distinct high and low levels of processing in dialogue management. There are many aspects of dialogue management which have to do with manipulating and maintaining the communication channel, but do not have any content that is relevant to the tasks being co-ordinated by the conversation. We advocate separating these processes into modules in a "low-level" of dialogue management, independent of "high-level" semantic and pragmatic interpretation and utterance content planning, analogously to the multi-level architectures used in robotics.

We gave four examples of such asynchronous and semi-autonomous low-level processes, currently implemented in a full working dialogue system: Helper feedback; Turn-taking; Aggregation; NP selection.

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.

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

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

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

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

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

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. Master's thesis, Stanford.

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 of European Association for Computational Linguistics (EACL 03)*, page (in press).

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.

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

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

Dan Shapiro, Pat Langley, and Ross Shachter. 2001. Using background knowledge to speed reinforcement learning. In *Fifth International Conference on Autonomous Agents*, pages 254 – 261. ACM Press.

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

Claus Zinn, Johanna D. Moore, and Mark G. Core. 2002. A 3-tier planning architecture for managing tutorial dialogue,. In *Intelligent Tutoring Systems, Sixth International Conference (ITS 2002)*.