Collaborative Dialogue for Controlling Autonomous Systems

Oliver Lemon, Lawrence Cavedon, Alexander Gruenstein and Stanley Peters
Center for the Study of Language and Information
Stanford University, CA 94305
{lemon,lcavedon,alexgru,peters}@csli.stanford.edu

Abstract

We claim that a natural dialogue interface to a semi-autonomous intelligent agent has important advantages, especially when operating in real-time complex dynamic environments involving multiple concurrent tasks and activities. We discuss some of the requirements of such a dialogue interface, and describe some of the features of a working system built at CSLI, focusing on the data-structures and techniques used to manage multiple interleaved threads of conversation about concurrent activities and their execution status.[1]

Dialogue Interfaces for Autonomous Systems

We believe there is a strong case for natural dialogue interfaces for autonomous systems performing complex tasks in unpredictable dynamic environments. Some authors (e.g. (Schneiderman 2000)) have argued against speech or natural language interfaces for such systems, as opposed to a graphical interface. However, we agree with many of the points made by Allen *et al* (2001): in particular, they argue that for increasingly many of the more complex tasks, GUIs become infeasible due to the complexity of the device and the activities it is required to perform.

It is important to clarify what we mean by "dialogue": a dialogue interface is not simply a matter of adding a speech recognition and generation component to a device (although Allen *et al* make the point that even this can enhance an existing GUI). Dialogue is a truly collaborative process between two (or more) participants (in the case we're interested in, a human operator and a robot or other agent) whereby references and tasks are negotiated and agreed on, often in an incremental manner.

Following Allen *et al*, we claim that collaborative natural dialogue offers a powerful medium for interaction between humans and intelligent devices. This is particularly the case for complex systems operating in dynamic, real-time environments, containing multiple concurrent activities and events which may succeed, fail, become cancelled or revised, or otherwise warrant discussion. Human dialogue in such scenarios are highly collaborative (Clark 1996), with much negotiation between the instructor and the task-performer: dialogue is used to specify and clarify goals and tasks, to monitor progress, and to negotiate the joint solution of any problems. Further, an interface to a device operating in such conditions must be interruptible, context-dependent, and otherwise extremely amenable to multiple threads of conversation.

Moreover, we argue that natural dialogue interfaces have important human-centered advantages over purely-GUI interfaces, including:

- The amount of required specialized training is reduced, allowing task experts (e.g. pilots, for UAVs) to be used as operators of the autonomous systems. Even though some adaption to the interface will generally be needed, much of the interface's power comes from the natural mode of interaction;
- Natural language is an efficient medium for communication and interaction; human communicators are used to making use of previous dialogue context and referring to previously introduced items in a conversational context;
- The collaborative nature of dialogue allows the operator to give timely succinct instructions, further embellishing them if so requested by the agent;
- A speech interface allows hands-free operation; this is critical for controlling intelligent devices while, say, operating a vehicle, or provides an important separate modality of interaction for a robot or autonomous device being (semi-)controlled using some other mode (such as a joystick or remote control);
- The cognitive load on the human operator is lessened since there is less need to focus on the interface itself and its usage; this is critical when the operator-agent team is involved in complex tasks requiring high reactivity, and especially so as we move to situations where a single operator may control multiple agents.

This is not to claim that a GUI interface is not extremely useful; indeed, we believe (as do Allen *et al*), that GUI and dialogue interfaces can be natural supplements to each other—in particular, a GUI can serve as a shared representation of the environment under discussion, allowing simple

[1]This work was supported in part by grant number N00014-02-1-0417 from the Department of the Navy for "High-Level Control of Human-UAV Teams".

gestures to be incorporated into the dialogue process (e.g. pointing by the human, highlighting objects by the agent).[2]

Robust Natural Dialogue Interfaces

Allen *et al* (2001) discuss some of the dialogue phenomena that must be handled as task and domain complexity increases. Of greatest interest and relevance to our focus are the following:

- **Mixed-initiative interaction**: the autonomous agent must be able to both respond to the human operator as well as initiate its own thread of conversation: agent-initiated conversation may arise from its perceptions of a changing environment, or execution-status of its tasks or activities;
- **Collaborative negotiation sub-dialogues**, to refine a command, request, or proffered information: e.g. a human-given command may be ambiguous or not fully specified, or the agent may note that a particular request may be impossible or sub-optimal to perform;
- **Different epistemic/temporal modalities**, e.g., for distinguishing between the current state of the world and a planned one: this allows the user and agent to discuss viability of future tasks or plans to achieve a goal;
- **Context-dependent interpretation** of all interactions: i.e. the preceding dialogue provides an important context for the most current utterance; in particular, noun-phrases are often resolved or disambiguated by referring to previous dialogue, an important aspect in the "efficiency" of natural language as a medium of interaction.

To these important properties we add the following, which provide much of the focus for our discussion below:

- **Ability to handle dialogues about multiple concurrent tasks** in a coherent and natural manner: many conversations between humans have this property, and dialogues between humans and (semi-)autonomous agents will have this feature in as much as such agents are able to carry out activities concurrently;
- **Generation of natural speech**: i.e. *what* to say and *when* to say it: it is important not to overload the human operator with irrelevant information, and to present all information as succinctly as possible, particularly in highly dynamic environments.

These two issues are intertwined: the task of generating natural responses that are understood by the human with minimal cognitive effort is complicated when multiple threads of discussion are available. In this scenario, the problem of establishing and maintaining appropriate context in a natural way becomes difficult.

Generation is also complicated by information becoming available in real-time, from different sources, involving the risk of overloading the operator with irrelevant or highly repetitious utterances. In general, the system should appear as 'natural' as possible from the user's point of view, including using the same language as the user if possible ("echoing"), using anaphoric referring expressions where possible, and aggregating utterances where appropriate. Further, another desirable feature is that the system's generated utterances should be in the coverage of the dialogue system's speech recognizer, so that system-generated utterances effectively prime the user to speak in-grammar.

The CSLI dialogue system addresses these issues using techniques that we describe below, including:

- dialogue context representation to support collaborative activities and concurrent tasking;
- modeling activities to support dialogues for task monitoring and collaborative planning;
- dialogue and conversation management to support free and natural communication over multiple conversation topics;
- natural generation of messages in multi-tasking collaborative dialogues.

Sample Application and Dialogue

The CSLI dialogue system has been applied to multiple applications. For the purposes of illustration here, we describe the WITAS[3] UAV ('unmanned aerial vehicle') application—a small robotic helicopter with on-board planning and deliberative systems, and vision capabilities (for details see e.g. (Doherty *et al.* 2000)), although the current implementation interacts with a simulated UAV.

In this application, mission goals are provided by a human operator, and an on-board planning system then responds. While the helicopter is airborne, an on-board active vision system interprets the scene or focus below to interpret ongoing events, which may be reported (via NL generation) to the operator. The robot can carry out various "activities" such as flying to a location, or following a vehicle, or landing. These activities are specified by the user during dialogue, or can be initiated by the UAV's on-board AI. In any case, a major component of the dialogue, and a way of maintaining its coherence, is tracking the state of current or planned activities of the device.

A more interesting and problematic notion is that of "joint-activities" between the UAV and a human operator. These are activities which the autonomous system cannot complete alone, but which require some human intervention. In our current scenarios, the UAV's vision system is not good enough to determine whether a particular vehicle is the one sought-after, and only the human operator has the authority to determine this, so that human and robot must collaborate in order to find and track a vehicle.

The dialogue in Figure 1 shows how a typical interaction works (other capabilities, such as clarification sub-dialogues, are covered in (Lemon *et al.* 2001)). Note here that the user is able to make explicit queries about the robot's activities (both current and future), that there are concurrent activities, and that conversational initiative centers around the joint activities currently being specified and executed.

[2] See (Suwa & Tversky 2002) for a discussion of the importance of the role of shared representations.

[3] See `http://www.ida.liu.se/ext/witas`

Figure 1: A demonstration of the CSLI dialogue system

Multi-modal Utterances	**Dialogue Move**
Operator (O): *Our job is to look for a red car*	Command (Joint Activity)
UAV (U): *Ok. I am looking for one.*	Report (Confirm Activity)
O: *Fly here please.* [+click on map]	Command (Deictic)
U: *Okay. I will fly to waypoint one.*	Report (Confirm Activity)
U: *Now taking off and flying there.*	Report (Current Activity)
O: *Stop that. Go to the tower instead.*	Command, Revision
U: *I have cancelled flying to waypoint one. I will fly to the tower.*	Report (Activity status)
O: *What are you doing?*	Wh-question (Current Activity)
U: *I am searching for a red car and flying to the tower.*	Answer (Current Activity)
O: *What will you do next?*	Wh-question (Planned Activity)
U: *I have nothing planned.*	Answer(Planned Activity)
U: *I see a red car on main street.* [display on map, show video images] *Is this the right car?*	Report, Yn-question (Activity)
O: *Yes, that's the right car.*	Yn-answer (Positive)
U: *Okay. I am following it.*	Report (Current activity)

Note also the simple use of multi-modal interaction: in the first portion of the dialogue the user indicates the destination by clicking on a map in the accompanying GUI, and later the UAV uses the map and video to illustrate what it sees.

System Architecture

The CSLI dialogue system architecture is component-based, making use of existing systems for speech recognition (Nuance (Nuance 2000)), Natural Language parsing and generation (Gemini (Dowding *et al.* 1993)), and speech synthesis (Festival (Taylor, Black, & Caley 1998)). These and other components (e.g. interactive map GUI) are integrated using the Open Agent Architecture (OAA (Cheyer & Martin 2001)), a technology for flexibly integrating and coordinating multiple asynchronous processes. The overall high-level architecture is shown in Figure 2.

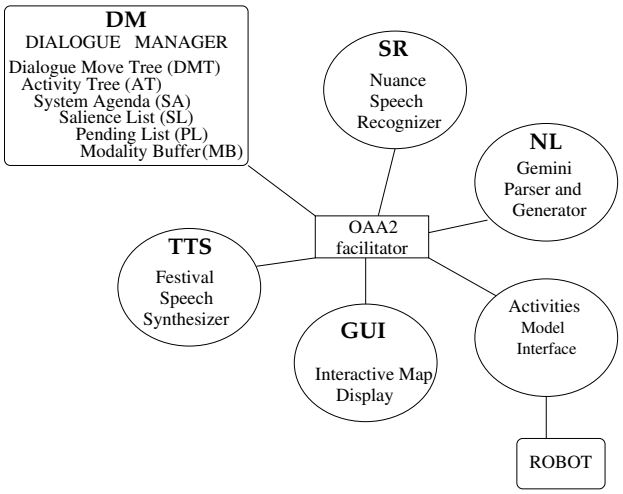

Figure 2: The CSLI dialogue system architecture

An advantage of this architectural approach is that we are not committed to any specific component—e.g. a different parser can be substituted for Gemini simply by writing an appropriate OAA "wrapper".[4]

Each component in the architecture is effectively decoupled from the others and operates autonomously and asynchronously, advertising its "capabilities" to the system via the OAA Facilitator (see (Cheyer & Martin 2001) for details). However, there is a general control flow:

1. incoming speech utterances are handled by the Speech Recognition component and converted into text strings; these strings are passed to the NL Parser, which converts them into a Logical Form (LF), a domain-independent logic-based representation of the utterance's content;
2. the LF is passed to the Dialogue Manager, and used to update the dialogue's Information State, which represents the context of all current conversational threads. This typically results in a *dialogue move* (e.g. generating a response), activating an *activity*, or querying the agent;
3. the Dialogue Manager constructs responses from the agent: these may be generated as a response to a user request, as a report from an activity, or as unsolicited information from the environment (e.g. a new observation). The DM constructs an LF appropriate to the current conversational context and thread, to be communicated to the human operator;
4. the LF is passed to the NL (reverse) parser, generating an appropriate textual NL string; this is in turn passed to the Speech Generation component, which renders the string as a spoken utterance.

Note that the mixed-initiative requirement of the dialogue system means that an utterance may begin at Step 3: a communique from the agent may not be in response to a request or command from the human. Further, the human may provide new inputs before an earlier input is processed end-to-end. The requirements of natural dialogue generation we

[4]This is currently being done for a separate application of the dialogue system.

described earlier means that Step 3 may require multiple responses to be aggregated into a single utterance, or not, depending on what is deemed "natural" for human digestion.

In the following section, we provide further details on the following components:

- the Activity Model Interface: activity models interface to the task-capabilities of the agent, and are used to interpret and generate dialogue utterances salient to the tasks it can perform; and
- the Dialogue Manager: this manages dialogue context, using it to resolve under-specified references, generate more natural responses, and otherwise manage the interaction between human and agent.

Dialogue Management

This section outlines the main properties of the Dialogue Manager; see (Lemon, Gruenstein, & Peters 2002; Lemon *et al.* 2002) for more detail.

The general theoretical framework underpinning our approach is a variant of "conversational games", also known as "dialogue games" (Power 1979; Carlson 1983), "interactions" (Houghton 1986), and, in the context of task-oriented dialogues, "discourse segments" (Grosz & Sidner 1986). All these accounts rely on the observation that answers generally follow questions, commands are generally acknowledged, and so on, so that dialogues can be partially described as consisting of "adjacency pairs" of such dialogue moves. Our notion of "attachment" of dialogue moves embodies this idea (see below).

Many dialogue moves are generic to the domain, but specific to the type of dialogue—we are specifically concerned here with activity- or task-oriented dialogue, and the specific operations we have designed reflect that. However, the design of the DM is such that new moves can easily be added to extend the dialogue capabilities, or for a new application.[5]

Activity Models

To be able to converse about its plans, goals and activities, an autonomous device needs a representation of such available to the dialogue system. The Activity Modeling Interface provides a layer between the Dialogue Management and the device or agent which actually performs the activities.

Integrating Dialogue and Action The use of Activity Models ensures a tight integration between the dialogue interface and the autonomous agent: any tasks or actions about which the agent is to be conversant must be explicitly represented as an Activity. Conversely, we assume the "dialogue-enabled" device is able to perform some class of specified actions (e.g. switch lights on, record on channel n, send email E to X, search for vehicle v), and the Activity Model must be able to translate linguistically-specified commands into activities that the device can perform.

Note that we make no strong assumptions regarding the actions that can be performed by the device. A relatively simple device, with little or no planning capabilities, may require a more sophisticated Activity Model, mapping higher-level linguistically-specified commands into a sequence of the simple actions that the device understands (e.g. see Figure 3). As such, Activity Models can be seen as playing a role analogous to a declarative plan language. For a more complex autonomous device, i.e. one with planning capabilities, the Activity Models will more closely correspond to the abstract higher-level goals and tasks, leaving the agent to manage how those goals and tasks are achieved or performed.

We are developing one representation and reasoning scheme to cover this spectrum of cases from devices with no planning capabilities to some more impressive on-board AI. Both dialogue manager and robot/device have access to a single "Activity Tree" which is a shared representation of current and planned activities and their execution status, involving temporal and hierarchical ordering. This tree is built top-down by processing verbal input from the user, and its nodes are then expanded by the device's planner (if it has one). In cases where no planner exists, the dialogue manager itself expands the whole tree (via the Activity Model for the device) until only leaves with atomic actions are left for the device to execute in sequence. The device reports completion of activities that it is performing and any errors that occur for an activity.

Note that because the device and dialogue system share the same representation of the device's activities, they are always properly coordinated. This follows (Rich, Sidner, & Lesh 2001)'s idea that declarative descriptions of goal decomposition form a vital layer of representation between a dialogue system and the device with which it interacts. Note also that some activities can themselves be speech acts, and that this allows us to build collaborative dialogue into the system.

An example Activity Model An example LOCATE activity model for the UAV is shown in Figure 3. It is used when constructing parts of the activity tree involving commands such as "search for", "look for" and so on. For instance, if the user says "We're looking for a truck", that utterance is parsed into a logical form involving the structure **(locate, np[det(a),truck])**. The Dialogue Manager accesses the Activity Model for LOCATE and adds a node to the Activity Tree describing it.

An Activity Model specifies what sub-activities should be invoked, and under what conditions, and what the postconditions of the activity are; they play a role analogous to the "recipes" of (Rich, Sidner, & Lesh 2001). For example, in Figure 3 the Activity Model for LOCATE states that:

- it uses the *camera* resource (so that any other activity using the camera must be suspended, or a dialogue about resource conflict must be initiated);
- the preconditions of the activity are that the UAV must be airborne, with fuel and engine indicators satisfactory;
- the whole activity can be skipped if the UAV is already "locked-on" to the sought object;
- the postcondition of the activity is that the UAV is

[5] E.g. this is the case in an application of the dialogue system to tutoring (Clark *et al.* 2001).

"locked-on" to the sought object;

- the activity breaks into three sequential sub-activities: WATCH-FOR, FOLLOW-OBJ, and ASK-COMPLETE.

Nodes on the Activity Tree can be either *active*, *complete*, *failed*, *suspended*, or *canceled*. Any change in the state of a node is placed onto the System Agenda for potential verbal report to the user, via the message selection and generation module.

Modeling Dialogue Context

As we argued earlier, the key to efficient and natural dialogue is a rich model of the dialogue context. Dialogue context is used to resolve anaphora, interpret under-specified references, and as a central resource in all collaborative planning conversations. The CSLI dialogue manager models context via an Information State; each incoming utterance is interpreted against the current Information State and typically involves updating it.

The CSLI dialogue Information State has the following components (most of these are described below):

- Dialogue Move Tree (DMT);
- Activity Tree (AT);
- System Agenda (SA);
- Pending List (PL);
- Salience List (SL);
- Modality Buffer (MB);

Figure 4 shows how the Dialogue Move Tree relates to other parts of the dialogue manager as a whole. The solid arrows represent possible update functions, and the dashed arrows represent query functions. For example, the Dialogue Move Tree can update the Salience List, System Agenda, Pending List, and Activity Tree, while the Activity Tree can update only the System Agenda and send execution requests to the robot, and it can query the Activity Model (when adding nodes). Similarly, the Message Generation component queries the System Agenda and the Pending List, and updates the Dialogue Move Tree whenever a synthesized utterance is produced.

The Dialogue Move Tree Dialogue management typically involves a set of abstract dialogue move classes which are domain independent (e.g. *command, activity-query, wh-question, revision*, ...). Any ongoing dialogue constructs a particular Dialogue Move Tree (DMT) representing the current state of the conversation, whose nodes are instances of the dialogue move classes, and which are linked to nodes on the Activity Tree where appropriate, via an *activity tag* (see below).

Incoming logical forms (LFs) from the parsing process are always tagged with a dialogue move (see e.g. (Ginzburg, Sag, & Purver 2001)), which precedes more detailed information about an utterance. For example, the logical form:

command([go], [param-list ([pp-loc(to, arg([np(det([def],the), [n(tower,sg)])]))])])

corresponds to the utterance "go to the tower", which is flagged as a *command*.

A slightly more complex example is:

report(inform, agent([np([n(uav,sg)])]), compl-activity([command([take-off])]))

which corresponds to "I have taken off"—a *report* from the UAV about a completed 'taking-off' activity.

The Dialogue Move Tree is used to interpret how an incoming LF relates to the current dialogue context. In particular, since multi-tasking may result in multiple concurrent dialogue threads, an LF must be attached to a specific thread in the complete conversation. In brief, the DMT does the following:

1. acts as a history or "message board" of dialogue contributions, organized by "thread", based on activities;
2. classifies *which* incoming utterances can be interpreted in the current dialogue context, and which cannot. It thus delimits a space of possible Information State update functions;
3. uses an *Active Node List* to control the order in which this function space is searched;
4. classifies *how* incoming utterances are to be interpreted in the current dialogue context.

In general, the DMT can be seen as representing a function space of dialogue Information State update functions. The details of any particular update function are determined by the node type (e.g. *command, question*) and incoming dialogue move type and their contents, as well as the values of *Activity Tag* and *Agent*.

The multi-threaded nature of a DMT is an important way in which it differs from similar concepts (e.g. (Ahrenberg, Jonsson, & Dalhbeck 1990)). In particular, since *all* threads of a dialogue are represented and can be active simultaneously, a new utterance can be flexibly interpreted, even when it is not directly related to the current thread (e.g. a user can ignore a system question and give a new command, or ask their own question). This enhances the power of the dialogue interface for controlling autonomous agents, particularly in unpredictable dynamic environments.

Interpretation and State Update The central algorithm controlling dialogue management has two main steps:

1. *Attachment*: process incoming input conversational move c with respect to the current DMT and Active Node List, and "attach" a new node N interpreting c to the tree, if possible;
2. *Process Node*: process the new node N, if it exists, with respect to the current information state. Perform an Information State update using the dialogue move type and content of N. Any references are also resolved at this stage (using anaphora resolution, GUI input, etc).

The effects of an update depend on the details of both the incoming input (in particular, to the dialogue move type and the content of the logical form) and the DMT node to which it attaches. Similarly, the attachment allowed on a node depends on the node type: for example, in the WITAS

Figure 3: A "Locate" Activity Model for a UAV, exhibiting collaborative dialogue

```
Locate// locate is "find-by-type", collaborative activity.
      // Breaks into sub-activities: watch_for, follow, ask_complete.
         {ResourcesUsed {camera;}    // will be checked for conflicts.
           PreConditions             //check truth of KIF statements.
            {(Status flight inair) (Status engine ok) (Status fuel ok);}
           SkipConditions // skip this Activity if KIF condition true.
            {(Status locked-on THIS.np);}
           PostConditions// assert these KIF statements when completed.
            {(Status locked-on THIS.np) ;}
           Children SEQ                  //sequential sub-activities.
            {TaskProperties
               {command = "watch_for";   // basic robot action ---
                          np = THIS.np;} // set sensors to search.
             TaskProperties
               {command = "follow_obj";  //triggers complex activity --
                          np = THIS.np;} //following a candidate object.
             TaskProperties              //collaborative speech action:
               {command = "ask_complete";//asks user whether this is
                  np = THIS.np;  }}}     //object we are looking for.
```

application, if the human operator issues the system a command (resulting in a *command* node), then the attachment (which determines the agent's response) must be a *confirmation*, *wh-question*, *yn-question*, or *report* node. The exact set of dialogue moves depends on the application and the type of dialogue appropriate to it: a standard set of moves is built-in to the core system—covering a wide range of multi-tasking activity execution and monitoring dialogues—but a given application may extend this set if appropriate.[6]

It is worth noting that the node type created after attachment may not be the same as the dialogue move type of the incoming conversational move *c*. Depending on the particular node which attaches the new input, and the move type of that input, the created node may be of a different type. For example, if a *wh-question* node attaches an input which is simply a *command*, the *wh-question* node may interpret the input as an answer, and attach a *wh-answer*. These interpretation rules are local to the node to which the input is attached. In this way, the DMT interprets new input in context, and the pragmatics of each new input is contextually determined, rather than completely specified via parsing using conversational move types.

In the current system, if the user produces an utterance which can attach to several nodes on the DMT, only the "most active" node (as defined by the Active Node List) will attach the incoming move. It would be interesting to explore such events as triggers for clarification questions, in future work.

Message generation

As described earlier, the message generation phase component of the dialogue interface raises important challenges for systems in complex dynamic environments. We attempt to address these by use of the following techniques: *relevance filtering; recency filtering; echoing; variability; aggregation; symmetry; real-time generation.*

Inputs to the generation module are "concept" logical forms (LFs) describing the communicative goals of the system. These are structures consisting of *context tags* (e.g. activity identifier, dialogue move tree node, turn tag) and a *content logical form* consisting of a Dialogue Move (e.g. report, wh-question), a priority tag (e.g. *warn* or *inform*), and some additional content tags (e.g. for objects referred to). An example input logical form is,

report(inform, agent(AgentID),
cancel-activity(ActivityID))

which corresponds to the report "I have cancelled flying to the tower" when AgentID refers to the robot and ActivityID refers to a "fly to the tower" task.

The output of this phase is a fully specified logical form which is sent to the Generation component of the Gemini parser (Shieber *et al.* 1990). The bi-directionality of Gemini (i.e. the same grammar is used for both parsing and generation) automatically confers a useful "symmetry" property on the system—that it only produces sentences which it can also understand. This means that the user will not be misled by the system into employing out-of-vocabulary items, or out-of-grammar constructions. Another side effect of this is that the system utterances prime the user to make in-grammar utterances, thus enhancing coordination between user and system in the dialogues.

We briefly outline these techniques and their impact.

Message selection: filtering Relevance and recency filtering techniques are used to ensure that the dialogue stays "on topic": i.e. the agent focuses (as appropriate) on the cur-

[6] See (Lemon *et al.* 2002) for a list of current dialogue moves and corresponding attachments for the WITAS application.

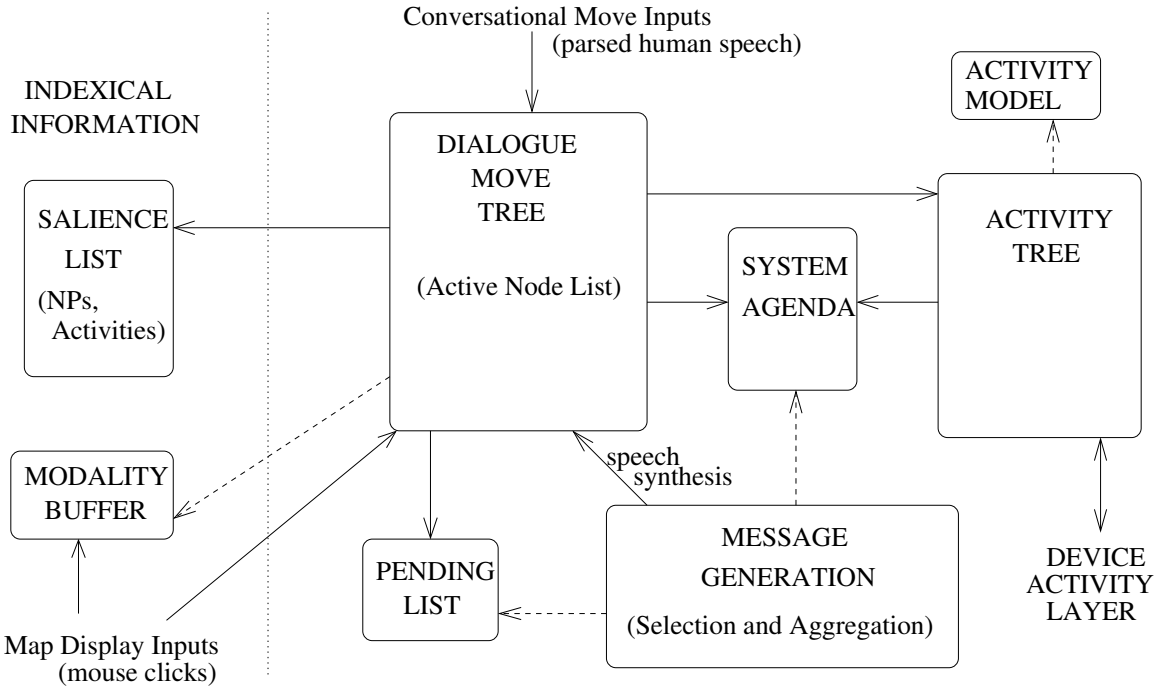

Figure 4: Dialogue Manager Architecture (solid arrows denote possible updates, dashed arrows represent possible queries)

rent topic(s) of conversation. LFs for message-generation are placed on a "System Agenda" (SA), which is the part of the dialogue Information State which stores communicative goals of the system. Communicative goals may also exist on a "Pending List" (PL) which is the part of the information state which stores questions that the system has asked, but which the user has not answered, so that they may be re-raised by the system. At any time, there may a number of "Current Activities" which the user and system are performing (e.g. fly to the tower, search for a red car): these activities represent current topics of conversation, and system reports can be generated by them, or be relevant to an activity by virtue of being about an object involved in an activity.

Some system reports are more urgent than others (e.g. "I am running out of fuel") and these carry the label *warning*. Warnings are always relevant, no matter what activities are current—they always pass the recency and relevance filters.

Echoing (for noun-phrases) 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. If the object is top of the salience list, the generator will select an anaphoric expression.

Incremental aggregation Aggregation combines and compresses utterances to make them more concise, avoid repetitious language structure, and make the system's speech more natural and understandable. When constructing an utterance, there is usually 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. Only occasionally does the System Agenda contain enough unsaid utterances to perform reasonable "pre-aggregation". Each dialogue move type (e.g. *report*, *wh-question*) has its own aggregation rules, specifying which other dialogue move types can aggregate with it, and how this is done.

As an example, the LF that represents the phrase "I will fly to the tower and I will land at the parking lot", is 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". The "retro-aggregation" rules result in sequences of system utterances such as, "I have cancelled flying to the school. And the tower. And landing at the base."

This phase of processing results in a fully specified Logical Form, representing an utterance exactly as it is to be generated into speech. At this point, the LF is passed to Gemini to be converted into natural language string, which is in turn passed to Festival and rendered into speech output.

System Status and Future Work

The system described here has been implemented and runs under both Unix or Windows 2000.[7] The core dialogue system contains a number of built-in dialogue operators, specifically suited to collaborative task-oriented conversation; however, new dialogue operators (for other types of dialogues or different application domains) are easily added.

[7]Video footage of the system can be found at `http://www-csli.stanford.edu/semlab/witas/`.

As well as the WITAS application, the CSLI dialogue system has also been applied to an intelligent tutoring system (Clark *et al.* 2001) and an in-car system for controlling devices. Testing of the WITAS system indicates that naive users can complete 80% of simple missions with a simulated UAV (e.g. search for vehicle, find it, follow it, land) after only four training tasks, lending some support to the claim that dialogue provides a natural interface for task specification.

Continuing and future work is planned in a number of areas. Incremental improvements are continually being made to the Dialogue Manager, such as: adding new dialogue move operators; improving the generation component, both in terms of filtering and aggregation; improving the interaction between dialogue participants (e.g. improved "turn-taking"). Initiatives with longer-term scope include

- **Interfacing to more powerful agent/control languages:** reporting on activities and their status requires the Activity Model to be synchronized with both the dialogue status as well as whatever control language is being used to control and monitor the agent's activities, requiring some integration effort;
- **Enhancing the mixed-mode interaction:** the current multi-modal interaction involves simple clicking and highlighting. More sophisticated gesturing requires more careful consideration of timing between gesture and utterance, as well as use of other technologies (pen-pad, touch-screen, etc);
- **Controlling teams of agents:** this raises the need for new dialogue operators (e.g. for *negotiating* the composition of the team and breakdown of tasks); multi-agent dialogue also complicates the issue of maintaining the conversational "common ground", since inter-agent communication would likely take place using a communication language which may not be transparent to the human.

References

Ahrenberg, L.; Jonsson, A.; and Dalhbeck, N. 1990. Discourse representation and discourse management for natural language interfaces. In *In Proceedings of the Second Nordic Conference on Text Comprehension in Man and machine*.

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2001. Toward conversational human-computer interaction. *AI Magazine* 22(4):27–37.

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

Cheyer, A., and Martin, D. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems* 4(1/2):143–148.

Clark, B.; Fry, J.; Ginzton, M.; Peters, S.; Pon-Barry, H.; and Thomsen-Gray, Z. 2001. Automated tutoring dialogues for training in shipboard damage control. In *Proceedings of SIGdial 2001*.

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

Doherty, P.; Granlund, G.; Kuchcinski, K.; Sandewall, E.; Nordberg, K.; Skarman, E.; and Wiklund, J. 2000. The WITAS unmanned aerial vehicle project. In *European Conference on Artificial Intelligence (ECAI 2000)*.

Dowding, J.; Gawron, J. M.; Appelt, D.; Bear, J.; Cherny, L.; Moore, R.; and Moran, D. 1993. GEMINI: a natural language system for spoken-language understanding. In *Proc. 31st Annual Meeting of the ACL*.

Ginzburg, J.; Sag, I. A.; and Purver, M. 2001. Integrating Conversational Move Types in the Grammar of Conversation. In *Bi-Dialog 2001—Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue*, 45–56.

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

Hockey, B.-A.; Aist, G.; Hieronymous, J.; Lemon, O.; and Dowding, J. 2002. Targeted help: Embedded training and methods for evaluation. In *Proceedings of Intelligent Tutoring Systems (ITS)*. (to appear).

Houghton, G. 1986. *The Production of Language in Dialogue: A Computational Model*. Ph.D. Dissertation, University of Sussex.

Lemon, O.; Bracy, A.; Gruenstein, A.; and Peters, S. 2001. Information states in a multi-modal dialogue system for human-robot conversation. In Kühnlein, P.; Reiser, H.; and Zeevat, H., eds., *5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog 2001)*, 57 – 67.

Lemon, O.; Gruenstein, A.; Battle, A.; and Peters, S. 2002. Multi-tasking and collaborative activities in dialogue systems. In *Proceedings of 3rd SIGdial Workshop on Discourse and Dialogue*, 113 – 124.

Lemon, O.; Gruenstein, A.; and Peters, S. 2002. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL)*. Special Issue on Dialogue (to appear).

Nuance. 2000. `http://www.nuance.com`.

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

Rich, C.; Sidner, C.; and Lesh, N. 2001. Collagen: applying collaborative discourse theory to human-computer interaction. *AI Magazine* 22(4):15–25.

Schneiderman, B. 2000. The limits of speech recognition. *Communications of the ACM* 43(9):63–65.

Shieber, S. M.; van Noord, G.; Pereira, F. C. N.; and Moore, R. C. 1990. Semantic-head-driven generation. *Computational Linguistics* 16(1):30–42.

Suwa, M., and Tversky, B. 2002. External representations contribute to the dynamic construction of ideas. In *Diagrams 2002*, 341–343.

Taylor, P.; Black, A.; and Caley, R. 1998. The architecture of the the festival speech synthesis system. In *Third International Workshop on Speech Synthesis, Sydney, Australia*.