Information Seeking Spoken Dialogue Systems—Part II: Multimodal Dialogue

Alexandros Potamianos, *Member, IEEE*, Eric Fosler-Lussier, *Senior Member, IEEE*, Egbert Ammicht, and Manolis Perakakis, *Member, IEEE*

***Abstract*—In this paper, the task and user interface modules of a multimodal dialogue system development platform are presented. The main goal of this work is to provide a simple, application-independent solution to the problem of multimodal dialogue design for information seeking applications. The proposed system architecture clearly separates the task and interface components of the system. A task manager is designed and implemented that consists of two main submodules: the electronic form module that handles the list of attributes that have to be instantiated by the user, and the agenda module that contains the sequence of user and system tasks. Both the electronic forms and the agenda can be dynamically updated by the user. Next a spoken dialogue module is designed that implements the speech interface for the task manager. The dialogue manager can handle complex error correction and clarification user input, building on the semantics and pragmatic modules presented in Part I of this paper. The spoken dialogue system is evaluated for a travel reservation task of the DARPA Communicator research program and shown to yield over 90% task completion and good performance for both objective and subjective evaluation metrics. Finally, a multimodal dialogue system which combines graphical and speech interfaces, is designed, implemented and evaluated. Minor modifications to the unimodal semantic and pragmatic modules were required to build the multimodal system. It is shown that the multimodal system significantly outperforms the unimodal speech-only system both in terms of efficiency (task success and time to completion) and user satisfaction for a travel reservation task.**

***Index Terms*—Multimedia communication, natural language interfaces, speech communication.**

I. Introduction

DESPITE recent progress in the areas of speech recognition and spoken language processing, building state-of-the art multimodal dialogue systems requires large amounts of development time and human expertise. Dialogue and multimodal management algorithms often have little generalization power and are not portable across application domains. Our main goal in this work is to reduce prototyping time and effort by creating application-independent tools and algorithms that automate the design process and can be used by non-expert application developers.

The majority of spoken dialogue systems in use today implement information seeking dialogue, i.e., the user is interacting with the system trying to obtain information from the system and perform a transaction, e.g., travel information, stocks information/banking, movie information and voice portal systems. Information seeking dialogues implement three main tasks: 1) elicit the information goal and related attribute-value pairs from the user, 2) perform database queries with the information supplied by the user, and 3) present query results and allow the user to navigate through the query results. In addition to these three main tasks, the user can also constrain or expand the queries interactively, correct errors, pose clarification questions, and attempt to explicitly modify system beliefs and task definitions. The majority of the system and user communication goals for information seeking systems are both application and interaction-modality independent. The implementation of these communication goals, however, does depend on modality (and sometimes also on the application). Based on this observation we propose a system architecture for multimodal dialogue systems that clearly separates the task and the interface. The task manager is defined as a sequence of broad communication goals that can be updated by the user; we call the sequence of goals along with the interaction context an *agenda* (motivated by work in [33]). Further, a modality-independent dynamic electronic form (e-form) representation is introduced in the task manager, extending the work of [8] and [15].

Recent work on rapid prototyping of dialogue systems and application-independent dialogue management strategies can also be found in [11]. An alternative approach to spoken dialogue system design is the use of an information state that contains semantic, pragmatic and dialogue information as well as update rules and dialogue strategies as discussed in [18], [20], and [24]. The information state approach is powerful and breaks free of the traditional graph-based dialogue management. In this paper, for the purposes of designing information seeking systems, we have found the use of generalized e-forms and a graph-based agenda to be general enough, yet, simpler and more modular.

Another goal of this work is to create an intelligent, efficient and adaptive user interface for information seeking applications. The system should allow the user to explore database information, modify task definitions and system beliefs,

Manuscript received November 16, 2005; revised August 10, 2006. This work was performed in part while A. Potamianos and E. Fosler-Lussier were with Bell Labs, Lucent Technologies, Murray Hill, NJ 07974, USA. This work was supported in part by DARPA under the auspices of the Communicator project and by the FP6 IST Network of Excellence MUSCLE under Contract FP6-5077-52. The associate editor coordinating the review of this manuscript and approving Dr. Jie Yang.

A. Potamianos and M. Perakakis are with the Department of Electronics and Computer Engineering, Technical University of Crete, Kounoupidiana University Campus, Chania 73100, Greece (e-mail: potam@telecom.tuc.gr; perak@telecom.tuc.gr).

E. Fosler-Lussier is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: fosler@cse.ohio-state.edu).

E. Ammicht is with Bell Labs, Lucent Technologies, Whippany, NJ 07981 USA (e-mail: eammicht@lucent.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2006.887999

and use alternate (user-initiated) routes for completing a task. Versatility, customizability, cooperation, and supervision (on a need-only basis) are some of the features of a good interface design [30], [2]. Our work on spoken dialogue interface design for information-seeking systems builds on previous work [1], [8], [9], [45], [33], [22], [35]. In [1], a hierarchical semantic representation is used for spoken dialogue systems design. In [33], [22], dialogue managers are designed using a task definition or a dynamic agenda (list of tasks). In [9], [8], adaptive initiative tracking is introduced for spoken dialogue systems.

Multimodal dialogue interfaces [10], [27], [16], [26], [30], [37] bring the best of two worlds to the user: the navigational aspect of graphical user interfaces and the declarative aspect of speech interfaces. Multimodal dialogue system design is not always done in a principled and efficient manner; often two separate unimodal systems are built and combined at the semantic, pragmatic, task and interface level. In addition to inherent inefficiencies, building separate systems for each modality can also lead to inconsistent user interface design across modalities. In this work, a common semantic and pragmatic representation across modalities is proposed and implemented. The general algorithms and tools of the semantic and pragmatic modules in [4] are used for this purpose. In particular, the input semantics from various modalities are merged by the pragmatic scoring algorithm introduced in [4]. Taken together, this approach provides a general process for designing multimodal dialogue systems for information seeking applications.

The main contributions of this paper are the introduction of 1) a task manager that is clearly separated from the dialogue manager; the task manager uses dynamic electronic forms and a dynamic task definition (agenda), 2) a simple and efficient multimodal system design process that uses common semantic, pragmatic and task representations across modalities, and 3) an adaptive multimodal interface that focuses on the synergies between modalities. These claims are verified by implementing and evaluating the unimodal and multimodal dialogue module for a travel reservation application. The spoken dialogue evaluation was built and formally evaluated under the auspices of the DARPA Communicator project. Specifically, the unimodal (speech-only) system described in this paper is the one used by Lucent Bell Labs in the 2001 DARPA Communicator evaluation (travel reservation application) [42].

The organization of this paper is as follows: Section II gives an overview of the system architecture and a review of the semantic and pragmatic modules. Section III presents the task manager, which consists of two main submodules, respectively, described in Section III-A for the electronic forms and Section III-B for the agenda. Section IV is dedicated to the spoken dialogue interface which implements the task manager agenda states. Section V details the multimodal dialogue system along with the user interface design guidelines, as well as a prototype implementation. Section VI presents the evaluations of the algorithms and systems described: Sections VI-A and VI-B are dedicated to the multimodal system and the spoken dialogue travel reservation system, respectively. Finally, in Section VII porting to other application domains and future enhancements are discussed.

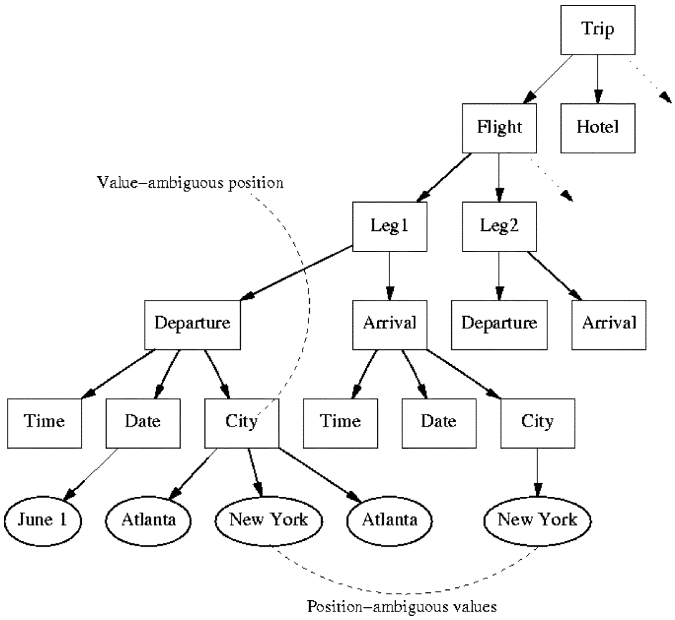

Fig. 1. Raw Data Tree illustrating value ambiguity (departure city Atlanta or New York) and position ambiguity (New York).

II. Architecture and System Overview

In order to fully illuminate the effects of the multimodal dialogue management system, we will first describe the system context in which the dialogue manager operates. Here we summarize the semantic and pragmatic representations that are described more fully elsewhere [4].

The dialogue component of the system is comprised of three modules (Fig. 2): the semantic module, pragmatic module and multimodal dialogue module.[1] The responsibility of understanding the user input is divided between the semantic and pragmatic modules. In a multimodal dialogue setting, the semantic module handles any modality-specific input (converting either a spoken "September eighth" or typed "9/8" to a common object format); the pragmatic module works with modality-independent data structures.

We utilize a series of tree-based ontological structures to hold information collected from the user, system beliefs and database replies that match user constraints [3]. A sample tree for the travel domain can be found in Fig. 1. The four classes of trees utilized include:

1) **Prototype tree** (domain ontology): a tree that encodes all of the is-a and has-a relationships for a domain (e.g., flights have legs). The other three types of trees are all derived from this tree.
2) **Raw data tree:** this tree contains all of the data collected from user input over multiple dialogue turns. Each datum is annotated with its collection turn and the modality-dependent confidence score assigned by the semantic module. It is the semantic module's responsibility to populate this tree; the pragmatic module marks items if they are position-ambiguous or value-ambiguous.

[1]A fourth module, the controller module, handles telephony, speech recognition, text-to-speech, database server and message passing.

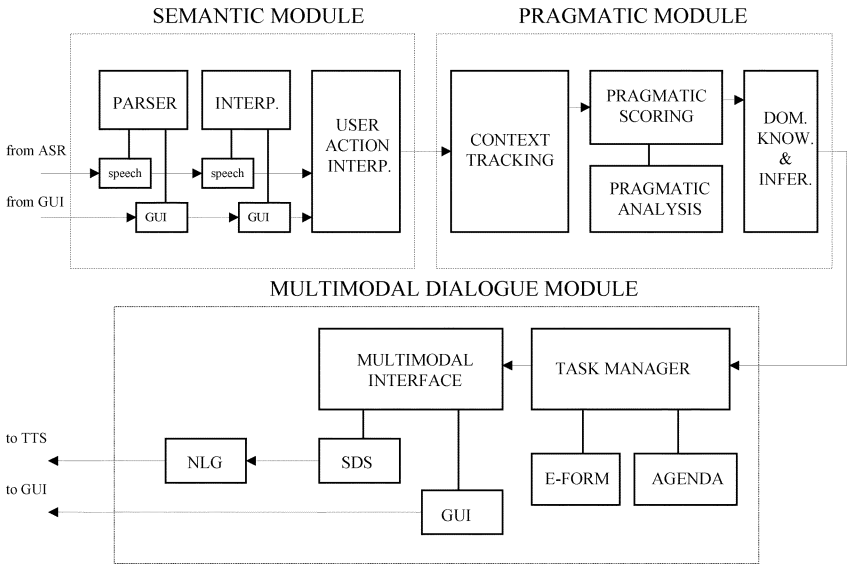

Fig. 2. Architecture of the multimodal dialogue system. Arrows denote the sequence of modules activated in a typical interaction turn.

3) **Candidate data tree:** this tree encodes the current beliefs of the system; it is computed from the information in the raw data tree as well as pragmatic actions, such as implicit confirmation, which increases confidence in an item without explicit input from the user.
4) **Database tree(s):** this is a set of (partially instantiated) trees that hold potential answers to database queries generated from the candidate tree; in the travel domain, we might have one tree for each flight so that we can present different options to the user.

The raw data, candidate data and database trees are similar data structures to the so called *frame representations* or *feature structures*.

By keeping all of these data structures parallel throughout the process, our tree-based algorithms can be reused in different parts of the interaction (form filling versus database navigation). An important feature of the semantic/pragmatic design is the ability to hold multiple conflicting constraints within the knowledge structure so that the ambiguities can be resolved. **Value ambiguities** arise when conflicting information is gathered in multiple turns (e.g., when a user corrects an erroneous system belief) or when the particular input is ambiguous itself (e.g., Rochester – Minnesota or New York?). **Position ambiguities** arise when the context tracker lacks the ability to place a datum exactly in one place in the tree (e.g., if "from New York" is misrecognized as "for New York"). This ability has consequences for both the spoken and graphical interfaces described below.

A. Pragmatic Analysis and Scoring

The pragmatic module performs three main tasks: context tracking, domain-specific inference, and pragmatic analysis and scoring. The context tracking algorithm is an application-independent algorithm for adding dialogue context (pragmatic information in general) to the raw attributes and values extracted by the semantic module. The domain inference uses domain specific rules to infer values about attributes (e.g., for a "round-trip" the first leg departure and the second leg arrival city should be the same) and to update the task structure (e.g., a "round-trip" has two legs). The pragmatic analysis and scoring algorithm combines all the information supplied by the user at the acoustic, linguistic, semantic and pragmatic levels, and produces a rank-ordered list of candidate values for each attribute along with confidence scores. Evidence for or against candidate values come from two main sources: evidence derived from the raw data, and evidence derived from pragmatic considerations by analyzing the interaction. MYCIN style [36] score update formulas are used to combine evidence from multiple sources. Pragmatic confidence scores are normalized to the range $[-1, 1]$, where 1 denotes certainty for the value correctness, 0 denotes ignorance and -1 denotes certainty for the value incorrectness. The pragmatic scoring algorithm is used for combining evidence both in the unimodal and in the multimodal input case as discussed in Section V. Based on the confidence score values (and differences between scores) the system can also detect and resolve semantic ambiguity as discussed in Section IV-E.

Overall, the semantic and pragmatic representation and algorithms are domain-independent and only use domain specific data structures (e.g., tree domain ontology) to represent and argue with task semantics/pragmatics. The representations are also portable across modalities. For more details on the semantic and pragmatic modules see [4]. Extensions of the semantic and pragmatic modules to handle multimodal input are discussed in Section V.

III. Task Manager

Information seeking applications share a great deal of functionality independent of the input/output modalities, the application domain and the system interface design. The main communication goals of an information seeking system are achieved by the following tasks.

T1 elicit the information goal and the related attribute-value pairs from the user.

T2 perform database queries using the information supplied by the user.

T3 present query results and allow the user to navigate through the query results.

In addition to these main tasks the task manager performs the following additional subtasks.[2]

S1 resolve value and position ambiguity interactively.

S2 allow the user to update system beliefs via correction/clarification interactions.

S3 inform the user about updates of system beliefs.

S4 allow the user to constrain/expand the query (similar to S2 but more explicit).

S5 allow the user to navigate among different applications (change information goals) and application states.

S6 allow the user to dynamically update the task definition (agenda).

S7 provide online help and inform the user about system functionality.

The complexity and necessity of each of those tasks and subtasks varies significantly from application to application and from system to system. For example, in web search applications, there is more emphasis on T3, while for a flight information application, T1 is more important. Even for the same application, the dialogue manager emphasis may vary depending on the designer philosophy and resources, e.g., for the travel reservation application in [44] there is more emphasis on S4, while in this work we mostly focus on S1-S3. In many spoken dialogue systems, S5 and S6 functionality is limited or non-existent. For graphical user interface systems, S1 and S2 functionality is limited because traditional graphical input modalities (keyboard, mouse) rarely introduce ambiguities. The task and subtasks presented above are not only application-independent but also modality-independent; however, the communication strategy used to achieve these goals is very much modality-dependent.

The simplest form of a task management algorithm is a graph-based finite state machine [25]. The finite state machine formalism is powerful and is often used for application design independent of the modality (interface). However, the traditional finite state machine formalism has no explicit task model and thus has little generalization power. Motivated by work in graphical user interface design, form-based dialogue management was proposed [15], [8]; the task is described as a series of frames or forms. The form formalism is powerful and portable across application domains and modalities. However, forms and frames are often too static to describe complex applications. A dynamic form-based dialogue manager is presented in [33]; the dynamically created and updated form sequence description is referred to as an agenda.

In this paper, we extend the work on form-based and agenda-based spoken dialogue management. A clear separation of the task and interface management modules is proposed; the interface implements agenda tasks. In addition, forms are augmented with task relevance scores that are dynamically updated by the user and the system; task relevance scores are used to rank-order the importance of the attributes in the forms. The electronic form machinery and agenda task definitions can be dynamically updated by the user and are application and modality-independent.[3] The two main components of the task manager the electronic form and the agenda modules are described next.

A. Electronic Forms

Electronic forms (e-forms) are collections of semantically consistent attributes; the values of some or all of the e-form attributes are needed by the system to construct or to constrain database queries. Forms are commonly used in information seeking systems with a graphical user interface front-end. E-forms have been used in spoken dialogue system design in [15], [9], [8]. In [9], [8], e-forms consist of two pieces of information: 1) a list of attributes and 2) the "relevance" of each attribute, i.e., an attribute can be "irrelevant", "necessary" or "optional" for a specific task.

In this work, e-forms are extended to include a continuous-valued (interface-independent) score that is normalized between 0 and 1 and describes the importance of an attribute. We refer to this score as the *task-relevance* score.[4] An example of an e-form from the travel reservation domain is shown next.

Context	Attribute	Score
trip.flight.leg1	departure.city	1.0
	departure.date	0.9
	departure.time	0.7
	arrival.city	1.0
	arrival.date	0.8
	arrival.time	0.5
	airline	0.6

The e-form is a subtree of the prototype tree, e.g., the e-form above corresponds to the "trip.flight.leg1" branch shown in Fig. 1 and contains as attributes all leaves under the "leg1" attribute. Task-relevance scores are high for "necessary" attributes (e.g., cities) and low for "optional" attributes (e.g., time).[5] Task-relevance scores are normalized from 0 ("unnecessary") to 1 ("necessary"). Continuously valued relevance scores are the first step towards introducing dynamic e-forms described next.

Dynamic e-forms are e-forms with dynamically varying task relevance scores. The scores are updated based on a set of system rules and user feedback. The main mechanism for updating the e-form scores is by attribute-tying, i.e., when the user specifies the value for one attribute in the e-form

[2]Note that these subtasks can be also automatically performed by the system without a user request.

[3]The implementation of the agenda tasks at the interface level is however modality-dependent.

[4]Task relevance scores measure how important it is to instantiate an attribute in order to perform a "successful" query. Since the success of a query is subjective (did the user find the information he/she was looking for?) task relevance scores could be updated/adapted by a user preferences model.

[5]The notion of "irrelevant" attributes used in [8] refers to attributes that are the subject of the user query, i.e., what the user is inquiring about, resulting in the definition of multiple e-forms, one for each possible query subject (e.g., "Where is X playing?" and "When is X playing" have separate e-form definitions). In this paper, we avoid this problem: a single e-form is used for each branch of the ontology tree; the subject of user queries is automatically identified and the task-relevance of the corresponding attribute is dynamically updated to 0 ("unnecessary"); the original score is restored once the user query is serviced by the system.

the relevance of the tied attribute is reduced. For example, departure date and arrival date are tied; when the user specifies the value of one attribute, the other is no longer required (and thus has a score of 0.0). The user can also reduce or increase the relevance of attributes in the e-form via the user profile. The spoken dialogue interface can also update the relevance of attributes based on user feedback and dialogue progress.[6]

When the task manager attempts to specify the context and attribute of the next system-user interaction, the e-form submodule is consulted. This submodule rank-orders the relevance of the attributes based on their (dynamically-updated) scores, checks to see which is the next most important attribute with an unspecified or underspecified value and passes it on to the task manager. When the values for all attributes with task relevance score greater or equal to a fixed threshold have been specified (our system uses the value 0.5), the e-form submodule informs the task-manager that the e-form has been filled.

B. Agenda

The agenda contains the task description and the sequence of tasks in the system. As in [33], [32], the sequence of tasks in the agenda can be updated by the task manager. By designing the agenda tasks to correspond to the main communication goals of our system we can create a simple agenda state graph that is application- and modality-independent. As discussed in Section III, the three main tasks that information seeking systems perform are elicitation of attribute-value pairs and e-form filling (T1), database querying (T2), and presentation, navigation and selection of database query results (T3). These tasks correspond to three main agenda states labeled "fill" (T1), "dbQuery" (T2), and "navigate" (T3), respectively, in our system. In addition, the agenda contains the dialogue context for each of these agenda "states". An example of the agenda for a flight reservation system is shown next.

Agenda Context	Agenda State
trip.flight	start
trip.flight.leg1	fill
trip.flight.leg1	verify
trip.flight.leg2	create
trip.flight.leg2	fill
trip.flight.leg2	verify
trip.flight	dbQuery
trip.flight	navigate
trip.flight	summary

In addition to the tree main states, the agenda also contains the generic "start" and "summary" states, along with the "verify" and "create" states for e-form values verification and new form creation, respectively. In the travel reservation example above, form creation refers to the optional (user-initiated) creation of a "trip.flight.leg2" return flight leg e-form. The agenda entries describe the sequence of tasks that have to be performed by the system to service an information query. The tasks are encoded in the agenda in time-of-service order; top states get serviced first by the system. However, the user has the option to alter the sequence of states or transition between states. For example, the task manager allows for transitions from the "navigate" to the "fill" state, i.e., the user is allowed to change the e-form attribute values and pose a new query. The implementation of each of the agenda states is interface dependent. For example, the "verify" state is not implemented for GUI interfaces; "verify" is implemented as an explicit confirmation prompt in the spoken dialogue interface. Although the implementation of the agenda states is modality-dependent, the list of agenda states is both application- and modality-independent.

The agenda can be dynamically updated by the user when the "create" agenda state is reached (system-initiated) or at any point in the interaction (user-initiated). For example, the user might say "one-way trip", in which case the agenda items ("trip.flight.leg2","fill") and ("trip.flight.leg2","verify") will be deleted (i.e., "round-trip" restores the agenda to its previous state). These application-specific user actions on the agenda are "hidden" in the application-dependent "domain knowledge and inference" submodule, which is part of the pragmatic module [4].

An example of how the sequence of agenda states is used to define the interaction flow, along with the conditions for transitions between agenda states is shown in Fig. 3. From the generic "start" state the application passes to the "fill" state where the values of the attributes in the e-form are "filled" by the user. Once all the required e-form values have been unambiguously filled, the system passes on to the e-form "verify" state. When the e-form values are confirmed by the user, the next e-form is created (at the "create" state) if applicable. Once all e-forms are filled and confirmed, the system transitions to the "dbQuery" state where a database query is constructed and serviced. The results from the database query are presented to the user at the "navigate" state. The user is allowed to view and select results, or modify the query. A "summary" of the selected results is also given to the user. The explanation for each of the conditions in the state transitions is given in the table shown next.

Condition	Explanation
isFormFull()	all required e-form values instantiated
isFormCorrect()	e-form values explicitly confirmed
noFormCreation()	form creation not allowed
formCreated()	form auto. created (system-initiated)
getResults()	database query returned results
userSelection()	user selected one of database results
newQuery()	system or user initiates new query
modifyQuery()	user modifies query attribute-values
UserInput = '...'	user provided input of type '...'

The implementation of the agenda states described above is modality dependent. For example, as discussed in Sections IV,

[6]For example the spoken dialogue system will reduce the relevance of an attribute by a fixed amount if the user replies "I don't care" or does not provide a value to an explicit prompt eliciting the value of this attribute. The motivation for this strategy is to avoid the user getting "stuck" trying to provide values for optional attributes. The score update rules are empirically defined and no formal evaluation has been performed.

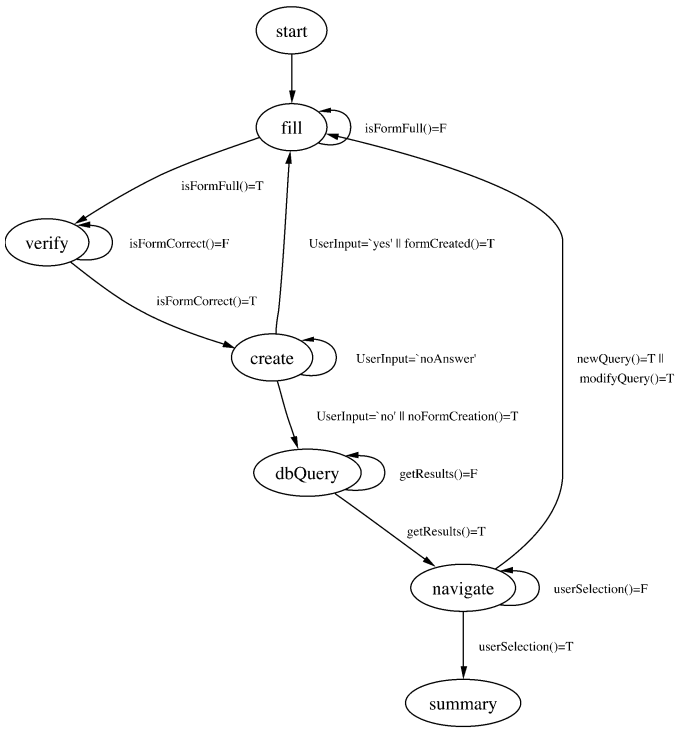

Fig. 3. Dialogue flow using Agenda States. Conditions for transitions between states are also shown, e.g., "$\mathrm{isFormFull}() = \mathrm{T}$" when the e-form values for the specific agenda context are unambiguously specified by the user ($\mathrm{T} = \mathrm{true}$, $\mathrm{F} = \mathrm{false}$).

V the "verify" state is implemented with an explicit confirmation prompt in the unimodal speech interface, while the state is skipped in the multimodal "click-to-talk" interface implementation (i.e., "isFormCorrect()" is always true). Note that in addition to the agenda states there is a list of system actions and user requests that are interface-dependent, e.g., information requests and change requests. These modality-dependent actions along with the speech interface implementation of the agenda states are discussed in the next section. The multimodal interface implementation is discussed in Section V.

IV. Spoken Dialogue System

The spoken dialogue manager promotes mixed-initiative system-user interaction. All types of user requests and user input are allowed at any point in the dialogue, i.e., the full application grammar is active throughout the interaction. The system prompts are focused and try to elicit specific information from the user, e.g., the value of an attribute. Explicit confirmation is used only to confirm the elicited attribute-values at the e-form confirmation state ("verify" state in Fig. 3); implicit confirmation is used in all other cases throughout the interaction. We have found this combination of system-initiated and user-initiated dialogue to provide a good balance for both naive and experienced users.

In the next subsections, we present the spoken dialogue manager submodules in detail. The following submodules (labeled with the corresponding task/subtask identifiers from Section III) are described: e-form filling (T1), database result navigation (T3, S4), information/change requests (S2, S3), focus change requests (S5), ambiguity resolution strategies (S1), and natural language generation.

A. Form Filling Implementation

In this section, we discuss the spoken dialogue implementation of three agenda states namely "fill", "verify" and "create" that correspond to the filling (adding values), the verification and the creation of e-forms, respectively. The implementation of the "fill" agenda state in the spoken dialogue system uses the machinery of the electronic form submodule introduced in Section III-A. Specifically, the spoken dialogue system elicits the values of the attributes in each e-form one by one, in sequence of "task-importance" by asking explicit questions for the value of an attribute, e.g., "What is the arrival city?". At any point in the interaction, the user is allowed to specify attribute-values that are not explicitly requested by the system or to correct/modify the values of attributes (i.e., update system beliefs). At each turn the spoken dialogue system also informs the user of updated system beliefs (implicit confirmation). Ambiguity is resolved via explicit disambiguation questions posed by the dialogue manager during the "fill" agenda state, e.g., "Was the arrival city Boston or New York?" (see also Section IV-E).

Examples of "fill" agenda state interactions are shown in the three examples of Table I from the June 2000 DARPA Communicator evaluation. In the first example, an experienced user enters all information in one turn; there are no speech recognition errors. The second and third examples are more typical examples where there is ambiguity (due to user input or recognition errors). The explicit and implicit ambiguity resolution mechanisms are shown. In addition, the task manager strategy for acquiring the values of attributes in order of importance is shown: first the system elicits information about cities, then about dates and times (as specified in the e-form example of Section III-A) using explicit requests. At any point in the interaction, the user can specify values for additional attributes, e.g., "I wanna depart from La Guardia [uh] next Saturday in the early morning" or modify attribute-values, e.g., "no, not Portland Oregon, Frankfurt Germany".

Once all the required attributes in the e-form have been unambiguously instantiated, the task manager moves on to the "verify" agenda state, as shown in Fig. 3. The "verify" state is implemented in the spoken dialogue system as an explicit yes/no confirmation question on all the elicited attribute-value pairs. An example is shown in the first dialogue fragment of Table II from the 2001 DARPA Communicator evaluation. In the example, the user replies "no" to the explicit confirmation and the system enters into an error correction subdialogue where the user is requested to change the value of one or more attributes. When the attribute-value is successfully updated via a change request (see Section IV-C) the system asks the explicit confirmation question again with the updated attribute-values.

Once the values of an e-form are verified by the user the system moves on to the "create" agenda state. The "create" agenda state is implemented in the spoken dialogue interface as an explicit e-form creation "yes"/"no" question, e.g., "Would you like a hotel in Orlando" or as an explicit question eliciting an attribute-value from the new e-form, e.g., "Where would

TABLE I
THREE DIALOGUE FRAGMENTS FROM THE JUNE 2000 DARPA COMMUNICATOR EVALUATION. S = System Utterance, U = Recognized User Utterance. THE AGENDA STATE, AGENDA CONTEXT AND SYSTEM COMMUNICATION GOALS ARE SHOWN FOR EACH UTTERANCE. THE FIRST EXAMPLE SHOWS AN EXPERIENCED USER INTERACT WITH THE SYSTEM. THE SECOND EXAMPLE DEMONSTRATES VALUE AMBIGUITY RESOLUTION. THE THIRD EXAMPLE DEMONSTRATES ERROR CORRECTION USING A CHANGE REQUEST

Dialogue Fragment I		Agenda State	Agenda Context	Comm. Goals
S:	Hello, welcome to the Bell Labs travel reservation system. How can I help you?	start fill	trip trip.flight.leg1	T1
U:	I want a one-way trip from La Guardia to Boston this Saturday in the morning.			
S:	I currently have you leaving New York-La Guardia on September 23rd in the morning arriving in Boston. Is this correct?	verify	trip.flight.leg1	S3 S2, T1
U:	Yes.			
S:	We're done with this segment.	verify	trip.flight.leg1	S3
S:	Accessing the database; this might take a few seconds.	dbQuery	trip.flight	T2
S:	The following fare costs 91 dollars. From New York-La Guardia U.S. Air has a direct flight leaving on September 23rd at 8:00 am and arriving in Boston at 9:11 am. Do you want to book this flight?	navigate	trip.flight	T3
U:	Yes.			
S:	I am making the reservation. You are leaving New York-La Guardia on September 23rd at 8:00 am on U.S. Air flight number 6804 arriving in Boston at 9:11 am. The total fare price is 91 U.S. dollars. Thank you for using our system. Goodbye.	summary end	trip.flight trip	T3

Dialogue Fragment II		Agenda State	Agenda Context	Comm. Goals
S:	How can I help you?	fill	trip.flight.leg1	T1
U:	I wanna go to New York [um] to Boston.			
S:	Was the arrival city Boston or New York City?	fill	trip.flight.leg1	**S1**, T1
U:	Boston Logan airport.			
S:	Arriving in Boston. Leaving from what city?	fill	trip.flight.leg1	S3, T1
U:	I wanna depart from La Guardia [uh] next Saturday in the early morning.			
S:	I currently have you leaving New York-La Guardia on September 23rd in the morning arriving in Boston. Is this correct?	verify	trip.flight.leg1	S3 S2, T1
U:	Yes, that's correct.			

Dialogue Fragment III		Agenda State	Agenda Context	Comm. Goals
S:	How can I help you?	fill	trip.flight.leg1	T1
U:	I am leaving Baltimore.			
S:	Leaving Baltimore. What is the arrival city?	fill	trip.flight.leg1	S3, T1
U:	Fine, Portland.			
S:	Arriving in Portland Oregon. On what date do you want to depart Baltimore?	fill	trip.flight.leg1	S3, T1
U:	About no, not Portland Oregon, Frankfurt Germany			**S2** (user-init.)
S:	Arriving in Frankfurt. On what date do you want to depart Baltimore?	fill	trip.flight.leg1	S3 T1
U:	. . .			

you like to go next?" as shown in the two examples of Table II. Note that the "create" agenda state can be skipped based on user input, e.g., in the first dialogue fragment of Table I the user specified "one-way trip" in which case the second and third flight leg entries are deleted from the agenda. This agenda modification actions are domain dependent and are "hidden" in the "Domain Knowledge and Inference" submodule of the pragmatic module.

B. Database Results Navigation Implementation

The implementation of the "navigate" state for the spoken dialogue interface allows the user to browse through a rank-ordered list of database results or to modify the query constraints. In addition, the user is allowed to request information or select an item in the result list. Given the limited interaction bandwidth of spoken dialogue interfaces and the large amount of information that has to be communicated to the user (typically there are many database hits), special care was taken to present results to the user in order of importance, i.e., results that better fit the query constraints and the stereotypical user model preferences are presented first. Database query results are ordered (and re-ordered on the fly based on user input) using the following cost function:

$$C_n = \sum_{k=1}^{K} w_k \, d(v_{nk}, \hat{v}_k) \tag{1}$$

where C_n is the cost of selecting the nth database query result, v_{nk} is the value of the kth attribute of the nth database query result, $\hat{v}_k$ is the user specified value of the kth attribute, $d()$ is the distance between the two values and w_k is the weight (importance) assigned to attribute k by the user. Both w_k and $\hat{v}_k$ can be implicitly or explicitly modified by the user. w_k can be specified by the user at the user profile. Note that $d()$ and w_k are application specific.

Typical user requests for the "navigation" agenda state of the travel reservation application are "next flight" or "get me an earlier flight". The "next flight" request presents the database query result with the lowest cost C that has not yet been presented to the user. The "earlier flight" request updates the query constraints $\hat{v}$ and causes a "re-order database results" operation; the best match (lowest cost) in the result list based on the new constraints is presented to the user. An example interaction is shown

in the second dialogue fragment of Table II where the user requests a "later flight". The user can also request a new flight search or modify the travel cities or travel dates; in both cases the system goes back to the "fill" agenda state as shown in Fig. 3 (see also Section IV-D).

C. Error Correction and Information Requests

In addition to the basic functionality described in the previous sections, the spoken dialogue system user interface was enhanced with the following (interface-specific) request types:

- *Information requests*: requesting the value of a specific attribute, e.g., the user requests "what is the departure city?", to ascertain the value of the departure city attribute. These requests were available both in "fill" and in "navigate" agenda states.
- *Clear requests*: deleting the values for a specific attribute, e.g., "clear the departure city", forces the removal of all candidate values for a given attribute.
- *Freeze requests*: inhibiting the system from further changing the value of a particular attribute, e.g., "freeze the departure city".
- *Change requests*: explicitly requesting to change the value of an attribute to a new value, e.g., "change Atlanta to New York", "change the departure city to New York", or "not Atlanta, New York". The system clears all previous values for the specific attribute and creates a new candidate value. Change requests are mostly used in the "verify" and "fill" agenda states.

In practice, the clear and (especially the) freeze requests were not used much by the users. However, the information and (especially the) change requests were used often and significantly enhanced the spoken dialogue interface. Information requests were especially useful in the "navigate" state. Change requests were a useful tool in the "fill" state for resolving value ambiguity and in the "verify" state to correct the values of attributes. Naive users, however, did not expect these advanced system capabilities; the features had to be advertised in the system tutorial.

Information, clear, freeze and change requests are serviced within a single dialogue turn by the system. Semantic and pragmatic analysis of change requests is especially challenging. The application-independent implementation of these request types in the semantic and pragmatic modules is presented in [4].

Examples of change requests are shown in the third dialogue fragment of Table I and first dialogue fragment of Table II. The first example is an indirect change request: "No, not Portland Oregon, Frankfurt Germany" that happens during the "fill" agenda state. The second example is an explicit change request: "Change the date September 27th" that happens during the "verify" agenda state. In both cases, the user is able to quickly update system beliefs without entering in a long error correction subdialogue.

D. Focus Change Request Implementation

Focus change requests are implicit or explicit user requests that attempt to change the dialogue context, i.e, navigate among different application states. Examples of explicit focus change requests in our spoken dialogue system are: "start over", "modify query", "go back to the first leg", "let's first make a car reservation". Once the system detects a focus change request it first acknowledges the request and then either 1) services the request immediately, e.g., "Going back to the first leg … ", 2) requests explicit confirmation, e.g., "Are you sure you want to start over?", and 3) acknowledges the request and informs the user that it will not be serviced, e.g., "Let us first complete your flight reservation …". Servicing a focus change request consists of a change of agenda context and agenda state (note that for all of the above examples the new agenda state is "fill"). The most common focus change request in the 2000/2001 evaluation data is "start over"; the rest of the focus change requests were rare.

E. Ambiguity Resolution Strategies

As discussed in [4], pragmatic confidence scores are used to detect value ambiguity, i.e., when the value of an attribute is not known with certainty. Pragmatic confidence scores are determined based on user input, dialogue history and pragmatic analysis. The pragmatic and context tracking modules attempt to resolve value and position ambiguity (respectively) using algorithms and rules discussed in [4]; if this fails, ambiguities are passed on to the dialogue manager for resolution. In our system, we consider the value i of an attribute j to be unambiguously known if its pragmatic score $s_{i,j}$ 1) is the highest among all candidate values of attribute j, 2) is above a threshold σ_1, and 3) the difference between $s_{i,j}$ and the next best candidate score is above a threshold σ_2. Our current system uses the settings $\sigma_1 = .2$, $\sigma_2 = .2$.

Once ambiguity is detected, it is up to the dialogue manager to decide when and how to best resolve the ambiguity. Typically the dialogue manager will attempt to resolve it as early as possible, either using implicit confirmation, explicit confirmation or direct disambiguation subdialogues. The selected disambiguation method depends on the "degree" and "importance" of the existing ambiguity, i.e., how close are the pragmatic scores and score differences to the thresholds σ_1 and σ_2 and how important are the attribute-values in question for the successful completion of the task. Specifically, in the "fill" or "verify" agenda state and a value or position ambiguity is detected the system will attempt to resolve the ambiguity in the next dialogue turn by asking an explicit disambiguation question. An example is shown in the second turn of the second dialogue fragment of Table I. However, for the "navigate" agenda state the system does not try to resolve value and position ambiguity; instead the system re-orders the database query results based on all available input from the user and presents the best result to the user (see Section IV-B). Ambiguity resolution can also be initiated by the user, e.g., by a change request (see Section IV-C).

F. Natural Language Generation

The natural language generation submodule is template based. For each type of prompt (e.g., explicit confirmation, implicit confirmation, value disambiguation) a template is built. Most templates are application and attribute dependent (e.g., information request prompts). Others (e.g., value disambiguation, implicit confirmation) are more general. The detailed discussion of the natural language generation module used in the 2000/2001 DARPA evaluation system does not present much research interest and is beyond the scope of this paper.

TABLE II
TWO DIALOGUE FRAGMENTS FROM THE 2001 DARPA COMMUNICATOR EVALUATION. S = System Utterance, U = Recognized User Utterance. THE FIRST FRAGMENT IS AN EXAMPLE OF A CORRECTION SUB-DIALOGUE WHERE CHANGE REQUESTS ARE USED TO QUICKLY UPDATE SYSTEM BELIEFS. THE SECOND DIALOGUE FRAGMENT IS A DATABASE RESULT NAVIGATION EXAMPLE

Dialogue Fragment IV		Agenda State	Agenda Context	Comm. Goals
S:	I currently have you leaving Cleveland on Saturday, September 22nd in the afternoon arriving in Los Angeles. Is this correct?	verify	trip.flight.leg1	S3 S2, T1
U:	No.			
S:	Please change the date and city information; or say done to move on.	verify	trip.flight.leg1	S2, T1
U:	I would like any clear Bangor			
S:	Please change the date and city information; or say done to move on.	verify	trip.flight.leg1	**S2**, T1
U:	Change the date September 27th.			
S:	I currently have you leaving Cleveland on Thursday, September 27th in the afternoon arriving in Los Angeles. Is this correct?	verify	trip.flight.leg1	S3 S2, T1
U:	Yes.			
S:	Where would you like to go next?	create	trip.flight.leg2	**S6**, T1
U:	Osaka Japan.			
S:	Arriving in Osaka Japan. Leaving Los Angeles on what date?	fill	trip.flight.leg2	S3, T1
U:	. . .			

Dialogue Fragment V		Agenda State	Agenda Context	Comm. Goals
S:	Would you like to continue on to another destination?	create	trip.flight.leg2	S6, T1
U:	No.			
S:	Accessing the database; this might take a few seconds.	dbQuery	trip.flight	T2
S:	The following fare costs 564 dollars. From Hartford United has a direct flight leaving on Wednesday, October 31st at 6:15 a.m. and arriving in Orlando at 11:15 a.m. Is this OK?	navigate	trip.flight	T3
U:	No, I'd like to leave on a later flight.			
S:	United has a flight for 568 dollars leaving at 9:45 a.m. and arriving at 4:42 p.m. with a stop in Chicago O'Hare airport. Is this OK?	navigate	trip.flight	T3
U:	Yes.			
S:	I am making the reservation. You are leaving Hartford on Wednesday, October 31st at 9:45 a.m. on United flight number 174 arriving in Orlando at 4:42 p.m. If you don't need a car or hotel reservation, hang up and your itinerary will be mailed to you.	summary	trip.flight	T3
S:	Would you like to rent a car in Orlando?	create	trip.car.rental1	S6, T1
U:	No.			
S:	Would you like a hotel room in Orlando?	create	trip.hotel.stay1	S6, T1
U:	. . .			

Examples of prompts can be seen in the dialogue fragments of Tables I and II. However, a novel experimental natural language generation system that is application independent was independently tested with the Communicator system and is presented in [14].

G. Dialogue Flow Examples

The sequence of agenda states and the conditions for transitions between states used in the spoken dialogue system are shown in Fig. 3 and are illustrated in Tables I and II, which list dialogue fragments from the 2000 and 2001 DARPA Communicator evaluation campaigns. The dialogue fragments are labeled with agenda state and context information that identify the agenda states shown in Fig. 3. For a formal evaluation of the spoken dialogue interface for the travel reservation application, see Section VI.

V. MULTIMODAL DIALOGUE SYSTEM

The Communicator task manager and the spoken dialogue system have been designed to be domain and modality-independent: consequently, adding the visual modality required only a few enhancements that proved easy to design and implement. We describe next how the semantic and pragmatic modules were updated and how the graphical user interface (GUI) was built. The visual input modalities are keyboard and mouse input in a desktop environment, or pen and graffiti input in a personal digital assistant (PDA) environment. The visual output modalities are text and graphics.

A. Semantic and Pragmatic Modules

The semantic representation used for both the visual and speech interfaces is identical. The semantics of the application (encoded in the prototype tree) are unchanged since the visual interface has the same functionality as the speech interface. The raw data and candidate data trees now encode the instantiated *joined* semantics of the visual and speech modalities. The GUI parser is the recursive finite-state parser used in the unimodal Communicator with an augmented grammar. In addition to spoken forms the GUI parser understands abbreviations such as "10/3/02" for date or "15:00" for time. For details see [4].

The context tracking, domain act classification and pragmatic analysis algorithms developed for the speech modality are also used for graphical input, although, in practice only a small subset of their functionality is exercised. The pragmatic scoring algorithm introduced in [4] is also unchanged. It has been designed to allow integration of any type of evidence for or against candidate values and thus provides a very useful framework for merging often conflicting information collected from the two input modalities: given multiple candidates for a given attribute, we update the pragmatic confidence score for

Fig. 4. (a)-(b) "GUI-only" interaction mode screen-shot examples for flight/hotel tab panes. (c)-(e) "open-mike" and "click-to-talk" interaction modes examples (user input: "From New York to Chicago on April 25").

each candidate using MYCIN style formulae as in the single modality case [36], [4]. The only difference is that now the initial confidence scores for values entered via the GUI are much higher than corresponding values provided by the speech recognizer. Similarly, the ambiguity detection, representation and resolution algorithms are identical for both the unimodal and the multimodal Communicator.

For example, consider two pieces of information supplied by the user: speech input "July fifth" and GUI input "7/10" (both for interaction context "trip.flight.leg1.departure.date"). The speech and GUI parser/interpreter processes the input and produces two raw attribute-value pair candidates, namely ("date", "July 5, 2005") and ("date","July 10, 2005"). The context tracker and pragmatic scoring modules are then applied to produce the pairs ("trip.flight.leg1.departure.date", "July 5, 2005") and ("trip.flight.leg1.departure.date", "July 10, 2005"), with pragmatic confidence scores 0.4 and 0.85, respectively.[7] Both candidates are placed on the candidate data tree along with their updated scores. Based on the (large) difference of the pragmatic scores the system argues that the attribute "trip.flight.leg1.departure.date" is unambiguously instantiated with value "July 10, 2005". Future speech or GUI user input can alter the values of the attributes and their corresponding pragmatic scores.

[7]The pragmatic scoring algorithm assumes that the original modality-dependent confidence scores for speech and GUI input are 0.5 and 0.9, respectively, where 1 denotes certainty and 0 ignorance of the true value. Due to the conflicting nature of the evidence, the scores are then updated to 0.4 and 0.85, respectively, (assuming that no other evidence exist for the attribute "trip.flight.leg1.departure.date"). The algorithm for updating the pragmatic scoring algorithm is described in detail in [4].

As originally intended, the Communicator pragmatic module design proved to be a natural and efficient way of combining information from different input modalities.

B. Graphical User Interface

The GUI forms are generated automatically from the prototype tree, the e-form definition and the agenda in the Communicator task manager. The e-form that generates the GUI form shown in Fig. 4(a) was given in Section III-A.

Note that the fields in the GUI are ordered automatically according to the e-form score that encodes the "task-relevance" of an attribute.[8] Flight reservation, hotel reservation and car rental forms are accessible as separate *tab panes* (see Fig. 4).[9] Navigation to different flight legs is implemented with buttons at the bottom of the form. Selected attribute fields, e.g., "departure time", "airline" and "car rental company" are implemented as a combo box, i.e., a pull-down menu that contains all possible values. The combo box automatically pops up when it comes into focus, i.e., when the application manager expects that the user is going to provide information about this attribute. Only attribute fields that have less than ten value options were implemented as combo boxes.

[8]It is arguable whether the grouping should be derived from the e-form score or determined from semantic coherence, e.g., all "date" fields grouped together. To give more freedom to the designer the system allows for a manual re-ordering of the fields. This option was used in Fig. 4 to bring the "airline" field to the bottom of the form.

[9]In general, parent nodes of forms in the domain ontology tree are now implemented as tab panes. This is consistent with GUI design principles and with the formalism that parents of forms represent different domains within an application. This information is extracted and dynamically updated from the agenda.

Value ambiguity is shown as a pull-down box with a list of choices and highlighted in red; position ambiguity and error messages as represented in the GUI as pop-up windows. Fields and buttons that become inaccessible in the course of the interaction are "grayed out". Finally, the context (or focus) of the interaction is highlighted in yellow.

In Fig. 4(a)-(b), GUI screen-shots for "GUI-only" interaction mode are shown, highlighting the flight and hotel reservation tab panes, respectively. Note that for consistency with the multimodal user interfaces we have added a speech activation button at the bottom of the GUI; this button is disabled in the "GUI-only" interaction mode. In Fig. 4(a) note the combo box automatically pops up when " *trip.flight.leg1.departure time*" comes into interaction focus. Additional information about the design of the multimodal user interface can be found in the next section.

C. Multimodal User Interface

A fundamental issue when designing multimodal systems is the choice, integration and appropriate mix of input and output modes in the user interface [5], [27], [26], [30]. Few guidelines exist for selecting the appropriate mix of modalities [5], [6]. It is clear, however, that a spoken language interface is not always the best choice. This is especially true for system output, where speech plays a secondary role to visual input (with the exception of hands-free, eyes-busy applications). It is often the case when designing multimodal user interfaces, that the developer is biased either towards the speech or the visual modalities. Our goal is to follow an approach that respects both modalities, creating an interface that is both *natural* and *efficient*. The first step towards the seamless integration of the input and output modalities is a *consistent* user interface: the GUI represents system state and possible actions (including those specific to the speech-modality). A second important step towards a truly multimodal experience is creating a user interface that utilizes the *synergies* between the input and output modalities, e.g., using visual feedback from the speech recognizer (including possible ambiguities).

Consistency is maintained via: 1) a common semantic representation, 2) the use of e-forms as the building block for the application manager for both modalities, 3) a multimodal pragmatic scoring algorithm (as described in the previous section), and 4) common system functionality via the speech or the visual modalities. More importantly, the modalities have been integrated in a way such that the user has the freedom to pick and choose the modality of preference. A short list of synergies between the speech and visual modalities as manifested in the multimodal Communicator follows: 1) the system semantic state is represented visually, 2) speech prompts are thus significantly shorter, mostly used to emphasize visual information, 3) speech recognition errors are easily corrected via the GUI, 4) position and value ambiguities are displayed visually and are easily resolved via the GUI, 5) the focus (or context) of the dialogue is highlighted visually and can easily be changed via the GUI, 6) the GUI takes full advantage of speech-interface "intelligence", and 7) conflicting GUI and speech input are seamlessly integrated via the pragmatic scoring algorithm.

Examples of the seamless integration between the visual and speech modalities are shown in Fig. 4. In Fig. 4(a), the semantic state is displayed visually; attribute-value pairs ("departure city", "New York City"), ("arrival city", "Chicago") and ("departure date", "7/18/05") are derived from the user input. In addition, the "arrival date" field is disabled since the "departure date" is specified by the user (using the intelligence of the speech interface) and the focus of the next dialogue turn, i.e., "departure time", is highlighted.

D. Multimodal Interaction Modes and Systems

Two different multimodal (MM) interaction modes have been implemented for combining the visual and speech modalities, namely: "click-to-talk" and "open-mike". "Click-to-talk" mode assumes that visual input is the default input modality and allows users to switch to the speech modality by clicking on a speech activation GUI button. "Open-mike" mode assumes that speech is the default input modality and allows the user to switch to visual input by clicking on the GUI.

Next we discuss the implementation of the MM modes in detail. The output interface is common for each interaction mode to allow us to better investigate the effectiveness of the "optimum" input modality mix. The visual output is identical to the "GUI-only" mode output. Audio output prompts were significantly shortened compared to the unimodal "speech-only" case. Specifically, implicit confirmation prompts were not used in the MM case where confirmation was efficiently done via the visual modality. In addition, form creation prompts and explicit confirmation prompts were significantly shortened or not used at all depending on the interaction context. Finally, information request prompts were shortened (typically to the name of the attribute requested, e.g., "Departure city?").

Note that in all multimodal modes only one modality is active at a time, i.e., the system does not allow for concurrent multimodal input.[10] In our current MM implementation, visual input is not allowed (GUI is "greyed-out") while speech input is selected and speech activity is detected.

1) Click-to-Talk Mode: By default the system expects input via the GUI; the user can provide spoken input by pressing the "Speech Input" button on the GUI. Upon being pressed, the "Speech Input" button turns red (indicating that the speech recognizer is active), the speech prompt is stopped (barge-in event) and the GUI is disabled ("grayed-out") for the duration of the speech recognition event. At the end of the speech input turn the system returns to the default GUI input state.

2) Open-Mike Mode: By default the system is listening for spoken input ("Speech Input" button is yellow). When voice activity is detected (a simple voice activity detector (VAD) algorithm is used for this purpose) the "Speech Input" button turns red, the GUI is disabled and the speech recognizer is activated. Upon completion of the speech recognition event the system returns to the "waiting for speech input" state. The user can provide visual input by clicking on the GUI area. Once GUI input is completed the system returns to the default "waiting for speech input" state.

[10]For many information-seeking/form-filling MM applications this is not a major limitation, but see Section VII for some thoughts on how concurrency may be achieved with modest extensions.

In Fig. 4(c)-(e), examples of the MM modes are shown. Initially the interaction focus is on "departure city", the speech modality is selected and the system goes to the "waiting for speech input state" state. The speech user input "from New York to Chicago on Monday" activates the speech recognizer (VAD event) and the GUI becomes disabled. The ASR event completes and the GUI is updated with the recognized information. For the next turn, visual input is selected by the user and the system goes to the GUI input state.

VI. Evaluation

In this section, we present the results of the spoken dialogue and multimodal system evaluation. The spoken dialogue system was evaluated for a travel application under the auspices of the DARPA Communicator program in the 2001 evaluation [42]. The multimodal (speech and GUI) systems were implemented and evaluated (independently) also for the travel reservation task. The "click-to-talk" and "open-mike" multimodal systems were evaluated and compared with the unimodal speech-only and GUI-only systems. Both objective and subjective evaluation criteria were used.

A few notes on the audio platform used in the spoken dialogue and multimodal evaluation campaigns. 1) The automatic speech recognizers (ASR) employed both for the spoken dialogue and the multimodal evaluation were identical. The Bell Labs ASR [47] was used with state-of-the-art tied triphone acoustic models and a class-based trigram language model (a single grammar was used for the whole application).[11] Because the ASR engine did not produce word level confidence scores, default confidence values of 0.5 and 0.9 were assigned for speech and graphical input, respectively, by the pragmatic scoring algorithm [4]. 2) The text-to-speech synthesizer (TTS) employed in the spoken dialogue evaluation used concatenated pre-recorded prompts and when such prompts were unavailable the Bell Labs TTS system was used instead [47]. The FreeTTS system [13] was used throughout the multimodal evaluation experiments.

A. Spoken Dialogue System Evaluation

In this section, we present evaluation results for the travel reservation system based on data collected during the DARPA Communicator 2001 evaluation campaign. The evaluation was performed over a period of six months; during this period 215 calls were made by 28 paid subjects. Callers were also asked to judge whether the task was successfully completed and to answer a set of five user satisfaction survey questions on the system usability based on the NIST-derived Likert paradigm [19], [38]. Among the 215 dialogues collected, we present results for 139 dialogues for which user survey data exists. For the assigned task of finding a suitable travel itinerary 115 out of the 139 interactions examined were judged successful by the user; a 83% perceived task completion (PTC) rate. Note that about half of the task failures were due to database back-end failures,[12] excluding database failures the task completion rate was 91%.

TABLE III
Objective and Subjective Dialogue Metrics

	Task Comp.		No Comp.	
Objective Metric	mean	std	mean	std
Dialogue Duration (secs)	547	428	429	423
# of User Turns	33.6	25.8	31.2	24.8
# of User Words	85.6	79.4	94.3	99.7
# of User Words per Turn	2.61	1.36	3.33	1.42
# of Concepts	35.2	27.9	28.7	20.2
# of Concepts per Turn	1.13	0.23	1.18	0.31
Word Error Rate (%)	23.5	12.9	24.9	12.7
Sentence Error Rate (%)	28.8	15.6	34.3	15.9
Concept Err. Rate (CER) (%)	16.5	11.9	20.9	15.9
Sentence CER (%)	15.5	11.2	18.9	15.9

	Task Comp.		No Comp.	
Subjective Metric	mean	std	mean	std
Easy to Get Info	3.61	1.37	1.62	1.05
Easy to Understand	4.02	1.09	3.10	1.42
Know What to Say/Do	3.56	1.27	2.97	1.47
Know What to Expect	3.55	1.29	1.98	1.30
Future Use	3.31	1.50	1.69	1.04
Overall	3.61	1.15	2.23	0.90

Objective dialogue metrics computed for each dialogue included dialogue duration, the number of user turns, the number of user words, the number of user words per turn, the number of user concepts and the number of user concepts per turn. Additionally, accuracy metrics, such as word and concept error rates, were computed both at the word and sentence level. Each metric was computed directly from corresponding log files with the exception of concept accuracy. The concept accuracy error rate (CER) was computed by comparing the output of the semantic parser for the recognized and transcribed user utterances, respectively. Sentence CER was computed as the percentage of sentences with at least one concept error. Mean value and standard deviation of each objective dialogue metric are shown in Table III as a function of (perceived) task completion. As expected, task-completed dialogues have a longer dialogue duration and a larger number of user turns. However, the number of user words and user words per turn is larger for task-incomplete dialogues. In [40], it was shown using the PARADISE evaluation framework across all Communicator systems that user satisfaction correlated best with task completion, task duration, turn duration and word accuracy.

In the second part of Table III, the mean and standard variation of the Likert scores in the user survey are shown as a function of perceived task completion. The user survey consisted of the following five questions which were rated from 1 to 5 (5 being strong agreement):

1) "In this conversation, it was easy to get the information that I wanted" (Easy to Get Info),
2) "I found the system easy to understand in this conversation" (Easy to Understand),
3) "In this conversation, I knew what I could say or do at each point of the dialogue" (Know What to Say/Do),

[11]Note that the much worse ASR performance in the multimodal evaluation compared to the spoken dialogue evaluation (Communicator) was due to the different speaker population used: native speakers for the Communicator vs. non-native speakers for the multimodal evaluation.

[12]The database back-end used live data obtained from the internet; back-end failures were due to internet service outages or unavailable data feed.

TABLE IV
OBJECTIVE AND SUBJECTIVE METRICS FOR UNIMODAL AND MULTIMODAL SYSTEMS

Objective Metric	speech-only	GUI-only	click-to-talk	open-mike
Task Success (%)	62	100	98	96
Speech Turns (%)	100	0	56	65
GUI Turns (%)	0	100	44	35
Avg. session duration (sec)	193.4	86.2	112.5	115.0
Avg. # turns per session	16.3	13.5	13.6	14.8
Avg. turn duration (sec)	11.9	6.4	8.3	7.8

Subjective Metric	speech-only mean (std)	GUI-only mean (std)	click-to-talk mean (std)	open-mike mean (std)
Easy to Get Info	3.57 (1.60)	4.54 (0.73)	3.85 (1.05)	4.06 (1.15)
Easy to Understand	3.5 (1.73)	4.48 (0.84)	3.74 (1.14)	4.06 (1.24)
Know What to Say/Do	3.61 (1.57)	4.62 (0.67)	3.93 (1.06)	3.85 (1.27)
Know What to Expect	3.57 (1.62)	4.54 (0.76)	3.78 (1.25)	3.66 (1.39)
Future Use	3.54 (1.60)	4.24 (1.06)	3.74 (1.16)	3.72 (1.44)
Overall	3.56 (1.60)	4.48 (0.83)	3.81 (1.13)	3.87 (1.30)

4) "The system worked the way I expected it to in this conversation" (Know What to Expect), and
5) "Based on my experience in this conversation using this system to get travel information, I would like to use this system again" (Future Use).

All subjective scores are significantly higher when the task is completed.

Overall, the dialogue system was judged favorably by evaluation users despite the relatively high word and concept error rates (24% and 18%, respectively). The (perceived) task completion rate was satisfactory at 83% (91% when database failures were excluded) but low compared to commercial systems that use system-initiated dialogue for simpler tasks. Our system also performed comparably to the rest of the travel reservation systems in the DARPA Communicator project 2001 evaluation (see [39], [41], [42]). Overall, the evaluation results show that our system based on general application-independent algorithms for task and dialogue management can achieve state-of-the art performance. For more details on system evaluation see [21] and for the DARPA Communicator evaluation in general see [39], [41], [42].

B. Multimodal Dialogue System Evaluation

In this section, we evaluate the multimodal dialogue system performance of both the click-to-talk and open-mike systems, and compare it with the unimodal systems (speech-only, GUI-only). This evaluation is not part of the official DARPA Communicator evaluation and was performed by ten non-native English-speaking users that were asked to complete five travel reservation scenarios of varying complexity: one/two/three-legged flight reservations, round-trip flight with hotel reservation, round-trip flight with car reservation. Evaluation took place in an office environment with all software (spoken dialogue system, speech platform, visual interface) running on a desktop P4 computer. The ten users evaluated all four systems on all five scenarios (a total of 20 runs per user). Users did not have prior experience using spoken dialogue systems. Systems were evaluated in random order.

Each user was given a short introductory document explaining the task, the scenarios, the four systems that were being evaluated and the subjective evaluation questions. The user was then asked to complete a simple demo scenario using all four different systems. The demo scenario included example phrases that could be used to interact with the speech interface and examples of how to use the system barge-in capability. The 15 minute introduction, was followed by the 20 evaluation runs. The user was asked to fill out a questionnaire (used for subjective evaluation) after each run. The user survey used was identical to the one used in the official DARPA Communicator evaluation. Upon completion of all experiments an exit interview was conducted. The results from the evaluation objective and subjective measures for all four systems are shown in Table IV.

Objective evaluation metrics are shown in the top part of Table IV. For each interaction mode we measure percentage of scenarios completed, percentage (of number of turns) of usage of the speech and visual input modalities, task and turn duration statistics. Note that duration statistics are computed only for completed scenarios. In terms of task completion, all modes are equivalent (no statistical significant difference) with the exception of the "speech-only" mode which performs significantly worse. In terms of task and turn duration, the "GUI-only" mode is the fastest, followed by the multimodal modes (no significant difference among them) and the much slower "speech-only" mode. Note that the lack of efficiency of the "speech-only" interface is partially due to the large (over 40% word error rate) speech recognition error rates for the non-native speakers evaluation population. Compare the 62% task completion result for this experiment with 91% task completion for DARPA Communicator evaluation in the previous section (we have excluded database errors to make a direct comparison).

The results from the subjective evaluation of the four systems are shown in the bottom part of Table IV. The mean and standard deviation of the Likert scores are shown for each of the five questions in the evaluation questionnaire. Results are shown only for the successfully completed tasks. Note that the 95% confidence interval of the mean subjective metric value x is $[x-0.27, x+0.27]$ for individual questions and $[x-0.12, x+0.12]$ for the overall score. In terms of subjective scores, the "GUI-only" system significantly outperforms all other systems. The multimodal systems outperform the "speech-only" system

in terms of subjective measures. Note that the "speech-only" interface subjective scores are similar to the scores of the DARPA Communicator evaluation in the previous section.[13]

Overall, we have found that 1) "GUI-only" was the most efficient interaction mode in terms of task completion and task duration, 2) the multimodal systems clearly outperformed the "speech-only" systems both in terms of objective and subjective measures, and 3) among the multimodal modes there was no statistical significance in terms of objective measure performance. Given the low speech recognition accuracy and the availability of the very efficient keyboard input in the GUI interface the above results are not surprising.

Note that the evaluation results presented above are only a single measurement point for the given application task, interface implementation and efficiency of the speech and GUI input modalities. The results could be very different (for the same systems) implemented on a personal digital assistant (PDA) rather than the desktop, where the efficiency of the GUI modality is reduced due to more cumbersome keyboard input. Improved recognition accuracy could also improve the efficiency of the speech input, thus changing the objective and subjective measures of the four systems.

VII. Discussion

In this section, our experience from porting the multimodal dialogue system to a movie information application is discussed, limitations of the current system are pointed out and future work opportunities are proposed. In addition to the movie information domain, porting to various other applications was investigated (yellow pages, corporate directory information system) but not fully implemented.

The list of implementation steps for porting the system to a new application domain, e.g., movie information system, were the following: (1) *application semantics*: design the prototype tree and the database semantics, (2) *parser*: design parser rules and map parser concepts to prototype tree attributes, (3) *interpreter*: define mapping from spoken/written form to internal value representation for each attribute in prototype tree; define application-dependent user requests in "User Action Interpreter", (4) *pragmatics*: define application-dependent context ambiguity resolution rules; implement "Domain Knowledge and Inference" rules, (5) *task manager*: define agenda and e-forms, (6) *generation*: define prompts and prompt templates; define mapping from internal value representation to spoken form for each attribute, (7) *database*: design and implement database back-end; define database results navigation information.

Note that only the application-dependent interpreter, domain knowledge/inference, natural language generation and database module code-base had to be modified in order to port the system to the movie information application domain. Most of the time and effort went into designing and implementing the database back-end; designing the parser grammar rules and writing interpreters for new concepts also required significant effort. Note that the pragmatic and multimodal dialogue interface modules required little or no modification. Overall, we were able to port the system to a new application domain with relatively little effort (with the exception of the database back-end implementation) and were thus able to verify the domain-independence claims for our system.

Despite the success of the porting exercise, there were some limitations in the resulting movie information system both in terms of system functionality and user interface. These limitations are a starting point for future research and are detailed next: 1) Spoken dialogue (SD) implementation of "database navigation": presenting the database results one by one was not appropriate for the movie information system; a more flexible algorithm is needed that can group/summarize database results and presents them in a single interaction turn to the user, 2) SD implementation of "focus change": the current implementation of focus change requests in the spoken dialogue system (SDS) is too rigid for the movie application; a flexible version of focus change request that allows the user to "abandon" an e-form and start a new one should also be implemented; the designer should decide on the flavor of the focus change request implementation that should be used, 3) SD implementation of "verify" agenda state: in most cases the explicit confirmation of e-form values was not appropriate for the movie application; to remedy the problem a "light" version of the "verify" agenda state should be implemented in the SDS; the designer can select the appropriate flavor of "verify" state in the agenda definition, (4) semantic persistence: in the current system, it is only possible to reset system beliefs explicitly with a "start over" or a "clear" request; we have found semantic persistence to be a problem in the movie application domain where users often implicitly start a new query; adding an implicit semantics forgetting mechanism to the system is a hard problem that requires further research, (5) complex queries: the system is currently unable to service "follow-up" queries, e.g., "Where is x playing?" followed by "Where is the movie theater located?", and compound queries, i.e., user requests that require a "join" operation between two or more databases in the back-end. In addition to these limitations, the automatic algorithm for the design of the graphical user interface does not always produce the expected results. More research is needed to improve on the interface design and system functionality.

We conclude this section with some thoughts on how to extend the proposed system to handle simultaneous speech and GUI input (concurrent multimodality) as well as to other input modalities. To handle concurrent multimodality the finite-state machine (FSM) speech/GUI parsers and interpreters [4] have to be extended to be able to cope with simultaneous speech and GUI input, e.g., "draw a line from here [mouse click] to there [mouse click]". For this purpose, an extension of the traditional two-tape (input/output) FSM semantic parser to a three-tape (two input/one output) FSM parser can be used as proposed in [17]. In addition to the parser and interpreters, no other modification to the semantic and pragmatic modules is necessary to handle concurrent multimodal input. Similarly, integration of other input modalities, e.g., gestures, in the current system can be achieved by adding appropriate semantic parsers and interpreters. For each new modality the parser/interpreter should

[13]The scores for the "Easy to Understand" question are not directly comparable because different text-to-speech systems were used in the two evaluation campaigns.

map user input to attribute-values as specified in the domain ontology (prototype tree). No other modification in the semantic or pragmatic modules is necessary.

VIII. Conclusions

In this paper, we have developed a new application- and modality-independent framework for task and multimodal dialogue management of information seeking dialogue systems. The proposed framework clearly separates the task module and the user interface module. Three user interface modules were implemented as part of a travel reservation application: speech-only, GUI-only and multimodal (speech and GUI) which all share the same task manager. The user interface modules are an implementation of the task manager methods and agenda states. The spoken dialogue interface was shown to handle complex user input including error correction and references to attributes. The spoken dialogue system was evaluated and it was shown to compete favorably with state-of-the-art systems. Task completion was higher than 90% for the speech-only interface (ignoring database failures). Average subjective user rating was 3.5 out of 5 in a Likert scale (note that during the last two months of the six month evaluation campaign the user ratings were 4 out of 5).

Introducing multimodality into our system required a small development effort; the (unimodal) semantic, pragmatic and task module were shown to be portable to the visual modality. The multimodal interface was designed to emphasize consistency and synergies between modalities. It was shown that the desktop multimodal system compared favorably to the speech-only system both in terms of objective and subjective evaluation criteria. The desktop GUI system outperformed the multimodal systems for a population of non-native speakers. Some of these results might be biased due to low speech recognition accuracy; more experimentation is needed to verify these results for PDA environments (see also [48]).

The unimodal and multimodal systems were also ported to the movie information domain to verify application-independence claims. Overall, we have shown that the proposed multimodal dialogue module is both portable across applications and modalities and can produce state-of-the-art performance. We have also shown that application-independent and modality-independent algorithms can significantly reduce prototyping time with no or little loss in system performance. This work is a first step towards creating general and portable algorithms for multimodal dialogue systems.

Acknowledgment

The spoken dialogue systems data collection was designed and organized by NIST. The user survey and scenarios were created by the DARPA Communicator evaluation committee. The authors would like to express their sincere appreciation to Dr. S. Lee, Dr. J. Kuo, Dr. A. Pargellis, Prof. C.-H. Lee, and Prof. J. Olive for their many contributions to the Bell Labs DARPA Communicator program; to A. Saad and Dr. Q. Zhou for building the Communicator-compliant Bell Labs audio platform; to T. Ren, M. Toutoudakis, F. Garitos, R. Barkan, M. Einbinder for their help with system implementation and data collection; to Dr. J. Chu-Carroll, Dr. M. Tsangaris, and Prof. S. Narayanan for many helpful discussions; and to the Colorado University Communicator team (and especially Dr. B. Pellom) for providing and supporting the travel information database back-end. The authors also wish to thank the anonymous reviewers for many helpful comments and suggestions.

References

[1] A. Abella and A. Gorin, "Construct algebra: Analytical dialog management," in *Proc. Annu. Meeting Association of Computational Linguists (ACL)*, Washington, D.C., Jun. 1999.

[2] A. Abella, M. K. Brown, and B. Buntschuh, "Development principles for dialog-based interfaces," in *Proc. Eur. Conf. Artificial Intelligence*, Budapest, Hungary, 1996.

[3] E. Ammicht, A. Potamianos, and E. Fosler-Lussier, "Ambiguity representation and resolution in spoken dialogue systems," in *Proc. European Conf. on Speech Communication and Technology*, Aalborg, Denmark, Sep. 2001.

[4] E. Ammicht, E. Fosler-Lussier, and A. Potamianos, "Information seeking spoken dialogue systems—Part I: Semantics and Pragmatics," *IEEE Trans. Multimedia*, vol. 9, no. 3, Apr. 2007, to be published.

[5] N. O. Bernsen and L. Dybkjaer, "Is speech the right thing for your application?," in *Proc. Int. Conf. Speech Language Processing*, Oct. 1998.

[6] V. Bilici, E. Krahmer, S. te Riele, and R. Veldhuis, "Preferred modalities in dialogue systems," in *Proc. Int. Conf. Speech Language Processing'2000*, Beijing, China, Oct. 2000.

[7] B. Carpenter, *Type-Logical Semantics*. Cambridge, MA: MIT Press, 1998.

[8] J. Chu-Carroll, "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries," in *Proc. of the 6th ACL Conference on Applied Natural Language Processing*, Seattle, WA, May 2000.

[9] ——, "Form-based reasoning for mixed-initiative dialogue management in information-query systems," in *Proc. Eur. Conf. Speech Communication and Technology*, Budapest, Hungary, Sep. 1999, pp. 1519–1522.

[10] P. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Clow, and J. Smith, "The efficiency of multimodal interaction: A case study," in *Proc. Int. Conf. Speech Language Processing*, Sydney, Australia, 1998.

[11] M. Denecke, "Rapid prototyping for spoken dialogue systems," in *Proc. Int. Conf. Computational Linguistics*, Taipei, Taiwan, R.O.C., Aug. 2002.

[12] M. Denecke and A. Waibel, "Dialogue strategies guiding users to their communicative goals," in *Proc. Eur. Conf. on Speech Communication and Technology*, Rhodes, Greece, Sep. 1999.

[13] *FreeTTS : A Speech Synthesizer*, [Online]. Available: http://freetts.sourceforge.net/docs/

[14] M. Galley, E. Fosler-Lussier, and A. Potamianos, "Hybrid natural language generation for spoken dialogue systems," in *Proc. Eur. Conf. Speech Communication and Technology*, Aalborg, Denmark, Oct. 2001.

[15] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, "A form-based dialogue manager for spoken language applications," in *Proc. Int. Conf. Speech Language Processing*, Philadelphia, PA, Oct. 1996.

[16] X. Huang, A. Acero, C. Chelba, L. Deng, D. Duchene, J. Goodman, H.-W. Hon, D. Jacoby, L. Jiang, R. Loynd, M. Mahajan, P. Mau, S. Meredith, S. Mughal, S. Neto, M. Plumpe, K. Wand, and Y. Wang, "MIPAD: A next generation PDA prototype," in *Proc. Int. Conf. Speech Language Processing*, Beijing, China, Oct. 2000.

[17] M. Johnston and S. Bangalore, "Finite-state multimodal integration and understanding," *J. Nat. Lang. Eng.*, vol. 11, no. 2, pp. 159–187, 2005.

[18] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An architecture for multimodal dialogue systems," in *Proc. Annu. Meeting of the Association for Computational Linguistics*, 2002.

[19] L. B. Larsen, "Combining objective and subjective data in evaluation of spoken dialogues," in *Proc. ESCA Workshop on Interactive Dialogue in Multi-Modal Systems*, Kloster Irsee, Germany, Jun. 1999.

[20] S. Larsson and D. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Nat. Lang. Eng.*, vol. 6, pp. 323–340, 2000.

[21] S. Lee, E. Ammicht, E. Fosler-Lussier, J. Kuo, and A. Potamianos, "Spoken dialogue evaluation for the bell labs communicator system," in *Proc. Human Language Technology Conf.*, San Diego, CA, Mar. 2002.

[22] E. Levin, R. Pieraccini, W. Eckert, G. D. Fabbrizio, and S. Narayanan, "Spoken language dialogue: From theory to practice," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO, Dec. 1999.
[23] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialogue strategies," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 11–14, 2000.
[24] C. Matheson, M. Poesio, and D. Traum, "Modelling grounding and discourse obligations using update rules," in *Proc. Annu. Meeting North American Association for Computational Linguistics*, May 2000.
[25] M. F. McTear, "Modelling spoken dialogues with state transitions diagrams: Experiences with the CSLU toolkit," in *Proc. Int. Conf. Speech Language Processing*, Sydney, Australia, Dec. 1998.
[26] S. Narayanan and A. Potamianos, "Creating conversational interfaces for children," *IEEE Trans. Speech Audio Process.*, vol. 10, pp. 65–78, Feb. 2002.
[27] S. L. Oviatt, P. R. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro, "Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions," *Hum. Comput. Interaction*, vol. 15, no. 4, pp. 263–322, 2000.
[28] M. Perakakis, M. Toutoudakis, and A. Potamianos, "Modality selection for multimodal dialogue systems," in *Proc. Int. Conf. Multimodal Interfaces*, Trento, Italy, Oct. 2005.
[29] A. Potamianos, E. Ammicht, and H.-K. Kuo, "Dialogue management in the bell labs communicator system," in *Proc. Int. Conf. Speech Language Processing*, Beijing, China, Oct. 2000.
[30] A. Potamianos, H.-K. Kuo, C.-H. Lee, A. Pargellis, A. Saad, and Q. Zhou, "Design principles and tools for multimodal dialog systems," in *Proc. ESCA Workshop Interact. Dialog. Multi-Modal Syst.*, Kloster Irsee, Germany, Jun. 1999.
[31] A. Potamianos and H.-K. Kuo, "Speech understanding using finite state transducers," in *Proc. Int. Conf. Speech Language Processing*, Beijing, China, Oct. 2000.
[32] A. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, and A. Oh, "Creating natural dialogs in the Carnegie Mellon communicator system," in *Proc. Eur. Conf. Speech Communication and Technology*, Budapest, Hungary, Sep. 1999.
[33] A. Rudnicky and W. Xu, "An agenda-based dialog management architecture for spoken language systems," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO, Dec. 1999.
[34] D. Sadek and R. D. Mori, "Dialogue systems," in *Spoken Dialogues with Computers*, R. D. Mori, Ed. London, U.K.: Academic, 1998, pp. 523–561.
[35] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A reference architecture for conversational system development," in *Proc. Int. Conf. Speech Language Processing*, Sydney, Australia, Dec. 1998.
[36] E. Shortliffe*, Computer-Based Medical Consultation: MYCIN*. New York: Elsevier, 1976.
[37] M. Tsangaris and A. Potamianos, "AGORA: A GUI approach to multimodal user interfaces," in *Proc. Human Language Technology Conf.*, San Diego, CA, Mar. 2002.
[38] M. Walker, L. Hirschman, and J. Aberdeen, "Evaluation for DARPA communicator dialog systems," in *Proc. Int. Conf. on Language Resources and Evaluation*, Athens, Greece, Jun. 2000.
[39] M. A. Walker, J. Aberdeen, J. Borland, E. Bratt, J. S. Garofolo, A. N. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. I. Rudnicky, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker, "DARPA communicator dialog travel planning systems: The June 2000 data collection," in *Proc. Eur. Conf. Speech Communication and Technology*, Aalborg, Denmark, Sep. 2001.
[40] M. A. Walker, R. J. Passonneau, and J. E. Boland, "Quantitative and qualitative evaluation of DARPA communicator spoken dialogue systems," in *Proc. Annu. Meeting of the Association for Computational Linguistics*, 2001.
[41] M. A. Walker, A. I. Rudnicky, R. Prasad, J. Aberdeen, E. O. Bratt, J. S. Garofolo, H. Hastie, A. N. Le, B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. A. Sanders, S. Seneff, and D. Stallard, "DARPA communicator evaluation: Progress from 2000 to 2001," in *Proc. Int. Conf. Speech Language Processing*, Sep. 2002.
[42] ——, "DARPA communicator: Cross-system results for the 2001 evaluation," in *Proc. Int. Conf. Speech Language Processing*, Sep. 2002.
[43] M. Walker, C. Kamm, and D. Litman, "Towards developing general models of usability with PARADISE," *Nat. Lang. Eng.: Special Issue on Best Practice in Spoken Dialogue Systems*, 2000.
[44] W. Ward and S. Issar, "The CMU ATIS system," in *Proc. 1995 ARPA Spoken Language Systems Technology Workshop*, Austin, TX, Jan. 1995.
[45] W. Ward and B. Pellom, "The CU communicator system," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO, Dec. 1999.
[46] W. Xu, B. Xu, T. Huang, and H. Xia, "Bridging the gap between dialogue management and dialogue models," in *Proc. 3rd SIGdial Workshop on Discourse and Dialogue*, Philadephia, PA, Jul. 2002.
[47] Q. Zhou, A. Saad, and S. Abdou, "An enhanced BLSTIP dialogue research platform," in *Proc. Int. Conf. Speech Language Processing*, Beijing, China, Oct. 2000.
[48] M. Perakakis, M. Toutoudakis, and A. Potamianos, "Blending speech and visual input in multimodal dialogue systems,," in *IEEE/ACM Workshop on Spoken Language Technology*, Aruba, Dec. 2006.

Alexandros Potamianos (M'92) received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Greece, in 1990. He received the M.S and Ph.D. degrees in engineering sciences from Harvard University, Cambridge, MA, in 1991 and 1995, respectively.

From 1991 to June 1993, he was a Research Assistant at the Harvard Robotics Lab, Harvard University. From 1993 to 1995, he was a Research Assistant at the Digital Signal Processing Lab, Georgia Tech, Atlanta. From 1995 to 1999, he was a Senior Technical Staff Member at the Speech and Image Processing Lab, AT&T Shannon Labs, Florham Park, NJ. From 1999 to 2002, he was a Technical Staff Member and Technical Supervisor at the Multimedia Communications Lab at Bell Labs, Lucent Technologies, Murray Hill, NJ. From 1999 to 2001, he was an adjunct Assistant Professor at the Department of Electrical Engineering, Columbia University, New York. In the spring of 2003, he joined the Department of Electronics and Computer Engineering at the Technical University of Crete, Chania, Greece, as Associate Professor. His current research interests include speech processing, analysis, synthesis and recognition, dialogue, and multimodal systems, nonlinear signal processing, natural language understanding, artificial intelligence, and multimodal child-computer interaction. He has authored or co-authored over 60 papers in professional journals and conferences. He is the coauthor of the paper "Creating conversational interfaces for children" that received a 2005 IEEE Signal Processing Society Best Paper Award. He holds four patents.

Dr. Potamianos has been a member of the IEEE Signal Processing Society since 1992 and he served as a member of the IEEE Speech Technical Committee from 2000 to 2003.

Eric Fosler-Lussier (SM'05) received the B.A.S degree in computer and cognitive studies and the B.A. degree in linguistics from the University of Pennsylvania, Philadelphia, in 1993. He received the Ph.D. degree from the University of California, Berkeley, in 1999; his Ph.D. research was conducted at the International Computer Science Institute, where he was also a Postdoctoral Researcher through August 2000.

From August 2000 to December 2002, he was a Member of Technical Staff in the Multimedia Communications Lab at Bell Labs, Lucent Technologies; from January to July 2003, he was a Visiting Scientist in the Department of Electrical Engineering, Columbia University, New York. Currently, he is an Assistant Professor in the Department of Computer Science and Engineering, The Ohio State University, Columbus, with a courtesy appointment in the Department of Linguistics, where he co-directs the Speech and Language Technologies (SLaTe) Laboratory. His interests include linguistic modeling for automatic speech recognition, spoken dialogue systems, statistical pattern recognition, and natural language processing. He has authored or co-authored over 50 journal and conference papers.

Dr. Fosler-Lussier currently serves on the IEEE Speech and Language Technical Committee.

Egbert Ammicht received the B.S. degrees in mathematics and physics from Bogazisi University, Turkey, in 1974 and the Ph.D. degree in applied mathematics from Northwestern University, Evanston, IL, in 1978.

He held post-doctoral and teaching positions at the Courant Institute, New York; the Karlsruhe Nuclear Research Center, Karlsruhe, Germany; Ames Labs, Ames, IA; and the University of Delaware, Newark, prior to joining AT&T Bell Laboratories in 1985. He currently is a Distinguished Member of Technical Staff at Lucent Bell Laboratories. His research interests include signal processing applications in acoustics and E&M, including adaptive beamforming, echo cancellation, noise suppression, as well as image processing and image compression. His most recent areas of work include speech processing applications, natural language understanding algorithms, and wireless MIMO modems. In addition to having publications and seven patents in these areas, he is the main author of a commercialized high performance acoustic echo canceler.

Manolis Perakakis (S'07) was born in Iraklion, Greece, in 1974. He received the Diploma and M.S degrees from the Department of Electronics and Computer Engineering, Technical University of Crete, Greece, in 2001 and 2003, respectively. Since 2004, he has been a Research Assistant with the Department of ECE, Technical University of Crete, where is he is pursuing the Ph.D. degree.

He worked as a Consultant at Dialogos S.A. from 2000 to 2001. His current research interests include distributed speech recognition, speech processing, and multimodal spoken dialogue interfaces.

Mr. Perakakis received the First Award of Excellence in Telecommunications by Ericsson for his senior year thesis. He is also a Trolltech worldwide programming contest beta application winner. He has been a member of the Technical Chamber of Greece since 2004.