Proc. of ESCA Workshop on Interactive Dialogue in Multi-Modal Systems, Kloster Irsee, Germany, June 1999.

# DESIGN PRINCIPLES AND TOOLS FOR MULTIMODAL DIALOG SYSTEMS

*Alexandros Potamianos, Hong-Kwang Kuo, Chin-Hui Lee, Andrew Pargellis, Antoine Saad and Qiru Zhou*

Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974-0636, U.S.A.
email: {potam,kuo,chl,anp,saad,qzhou}@research.bell-labs.com

## ABSTRACT

**We review efforts in defining design principles and creating tools for building multimodal dialog systems with emphasis on the speech modality. General design principles and challenges are outlined. The focus is on system architecture, application and speech interface design, data collection and evaluation tools. We conclude that modularity, flexibility, customizability, domain-independence and automatic dialog generation are some important features of successful dialog systems and design tools.**

## 1. INTRODUCTION

Building a successful dialog system requires interdisciplinary expertise that goes beyond having state-of-the-art speech and natural language processing technology. The modules of a dialog system include: *automatic speech recognition* (ASR), *text-to-speech* (TTS) *synthesis*, *natural language understanding* (NLU), *application manager*, *dialog manager*, *database*, *controller/event handler*, and, for multimodal systems, *graphical user interface* (GUI), *gesture/sign recognition*, *visual speech recognition*. Due to the lack of space we will not describe in detail the building blocks of a dialog system. Instead we will focus on the system design process, provide guidelines and outline challenges. Dialog system design is typically comprised of four steps: (i) architectural design, (ii) application design and data collection, (iii) speech and natural language interface design and (iv) user feedback and evaluation. Although these steps can be carried out more all less independently of each other, some degree of coordination is needed to guarantee consistency and smooth integration.

The process of dialog system and interface design is an iterative one, as is the case for applications with traditional input and output modalities. The main reason for the need of iterative enhancement is the lack of accurate user models that could provide objective measures of system performance. In addition, application-specific data needs to be collected and labeled. Given the iterative nature of system design it is important to provide guidelines and tools that can reduce the concept-to-prototype development time and the number of iterations required. In addition, this set of tools has to be *flexible* and *general* enough to be able to build successful, *modular* and *upgradable* dialog systems, with *reusable* components.

Natural language dialog system design has been an active research field for over four decades. Early theoretical contributions to the field were made by scientists from the areas of artificial intelligence and linguistics, e.g., [16]. The advent of robust speech recognition technology and increased computing power towards the end of the 80's made applications in the area of spoken dialog systems possible. As a result, speech engineers started applying machine learning techniques that have been shown to be successful for speech recognition, to natural language understanding and dialog management (see [3] for historical notes). The travel reservation domain and specifically the DARPA Airline Travel and Information System (ATIS) task was the driving force in the beginning of the 90's and provided the seed for most prototype dialog systems in industry and academia [12, 15, 7]. In terms of system architecture, dialog systems have evolved from monolithic, hard-coded system design to a *modular* architecture of re-usable and programmable building blocks [12, 15, 21, 8]. *Stateless* servers were introduced in [12, 15], where dialog and application state information is kept in a template that is passed between and updated by the various modules. The communication between application and interface modules was standardized in [11] using the *Voice Interface Language* (VIL). Concepts such as *abstraction*, *inheritance*, *encapsulation* and *polymorphism* were borrowed from computer science for semantic and dialog state representation [21]. Novel ideas for application and interface design such as *customizability* and *automatic dialog generation* were introduced in [9, 10, 6]. Finally multimodal input/multimodal output system design principles and an interface with personality was demonstrated in [8]. Spoken dialog technology has been successfully used for building telephony applications (there speech provides clear advantages over the traditional touch-tone user interface). A platform architecture that supports multiple telephony channels and applications was proposed in [22]. Systems continue to evolve and new design principles are introduced that improve functionality, modularity and versatility.

The organization of this paper is as follows: We describe the steps involved in dialog system design and provide guidelines for building such systems, in terms of architecture, data collection, application and dialog design, speech and natural language interface, and evaluation. We then review the set of tools available for building dialog systems and conclude the paper.

## 2. DESIGN GUIDELINES

A fundamental issue in designing dialog systems is the choice of input and output modes in the user interface. Few guidelines exist for selecting the appropriate mix of modalities [2], however, it is clear that a spoken language interface is not the best choice for all applications. Other important issues include selecting a modular architecture and a *cooperative* interface. Some of these issues are discussed in the following sections. Specifically, the features of a successful system and the challenges that lie ahead are outlined. Note that general principles from computer science such as abstraction, inheritance and encapsulation

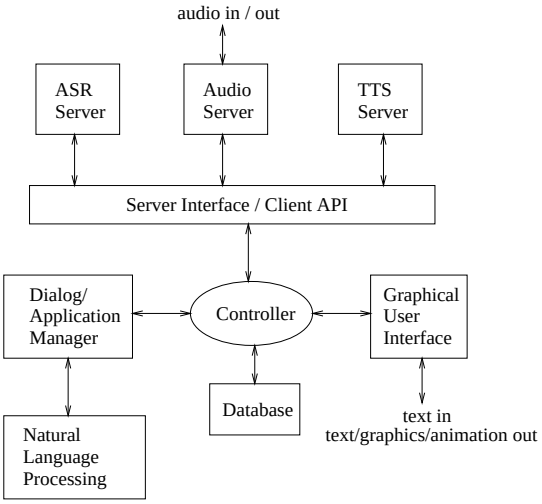


Figure 1: Multimodal system architecture.

should also be applied to software development of spoken dialog systems.

### 2.1. Architectural Guidelines

The architecture of dialog and multimodal systems has steadily evolved from a monolithic to a *modular* architecture with well-defined communication protocols between modules, e.g., [1]. In Fig. 1, a typical architecture is shown for a multimodal system with emphasis on the speech modality (adapted from [11]). The audio, speech recognizer and text-to-speech servers are controlled by a client layer through a well-defined *application programming interface* (API). The client brokers connections between the audio and ASR server (speech input), and the TTS and audio server (speech output). Multiple ASR, TTS or audio servers can be handled by the client API. The controller determines the generic multimodal application control flow, merges the input streams (speech, text, gestures) and synchronizes the output streams (speech, text, graphics, animation) [8]. The typical control flow is as follows: get transcription(s) from ASR and/or GUI, send them to the dialog manager/NLU module for processing, get results from dialog manager and present them to the user. Database information can be optionally requested from the controller by the application manager. A scripting language is often used for the controller, e.g., PERL in [10, 8]; a new scripting language is defined in [15]. Variations on the architecture in Fig. 1 can be found in the literature, e.g., in [15] some of the functionality of the application manager has been transferred to the controller for increased modularity; this architecture was adopted by the DARPA Communicator project.

There is no consensus among researchers on the communication protocol between the various modules. Typically both synchronous and asynchronous (event-driven) communication are allowed, although synchronous communication is preferred when possible. In some systems, information is communicated among and within servers through message-passing. The message contains all state information and thus the servers are *stateless* [12, 15, 8]. However, the format and content of these messages are very different among systems. The APIs for the communication between the dialog manager, controller, NLP and database modules need to be carefully defined. In [11], the

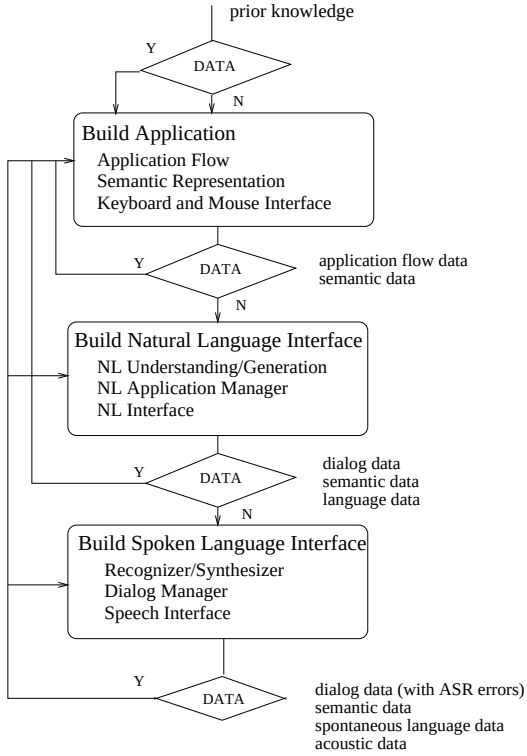


Figure 2: Iterative dialog system design using (optional) multi-stage data collection.

communication between the application manager and the audio, recognition and synthesis server is done through the Voice Interface Language (VIL).

### 2.2. Data Collection and WoZ Experiments

Data collection and analysis form a crucial part of system development. In Fig. 2, we show the various stages of application development, as the interface evolves from keyboard and mouse to spoken language. Data can be collected at various stages either automatically or assisted by supervisors (wizards). The types of data collected are acoustic, lexical, semantic and application/dialog flow data as shown in the figure. Word level transcriptions and application-specific semantic labels are assigned to the data either manually or semi-automatically (using the output of the recognition and understanding module). Collected data are typically very much dependent on the system configuration and the application scenario. As a result, dialog system development is an iterative process: collected data are used to improve the system, and more data are collected using the updated system configuration.

*Wizard of Oz* (WoZ) scenarios have been used extensively for interface design [4]. WoZ systems can be categorized into: (i) *fully wizarded* systems, (ii) *wizard-supervised* systems. Fully wizarded dialog systems use two wizards: one transcribing the user input and another typing the system output. As a result, the response time is large and the data obtained can be unrealistic. In practice, WoZ scenarios use a fully automatic prototype system and the wizard acts as a supervisor for the system, correcting errors and intervening when things break down, e.g., [5]. WoZ scenarios are especially useful for tactile input systems to emulate the voice modality ("voicing-over" of an application) [13].

### 2.3. Application and Dialog Manager Design

Designing a good application is the first step towards building a successful dialog system. The importance of application design is often overlooked by the speech community, where the emphasis is on a unimodal speech interface. As a result, applications often focus on the interface, and have limited functionality. In Fig. 2, an *application-centric* approach to dialog system design is outlined, where the application evolves as the interface is augmented with natural language and spoken language capabilities. The main advantages of this approach are: increased modularity (separation of interface and application), flexible multi-stage data collection as the interface and the application evolves, and an interface that can easily be tailored for "universal access", i.e., create the appropriate mix of modalities based on the way the application is accessed (telephone, personal digital assistant or PC).

One important application area of dialog systems is the *personal assistant/agent*, which encompasses a variety of sub-applications such as name-dialing, messaging (voice, e-mail, fax), information retrieval (news, weather), finance (banking, stock trading), travel reservations (air, car, hotel) etc. In [9, 10], a *hierarchical top-down* approach to application design is proposed. The services that the agent supports are organized in a hierarchical tree-structure which users navigate using voice. A similar hierarchical organization is used for navigating the world wide web using a mouse interface in Internet portals such as Yahoo and Excite.

For large dialog applications, the manual annotation of all possible state transitions can be a gruesome task. In [9, 10], a paradigm for *automatic application and dialog flow generation* is proposed. Specifically, given a hierarchical tree-structure of dialog states the dialog flow is generated automatically as follows: the system can transition to states that are children nodes, sibling nodes or the parent node of the current application state. Automatic dialog generation can also be based on user profile information and database constraints. Automatic and *dynamic* dialog flow generation is a challenging new direction in dialog management.

An important feature of any good application is *customizability*. This can be achieved by building a user profile either by explicitly querying the user or implicitly based on past interaction between the user and the system. In [6], the user profile determines the services to be included in a particular application as well as the format of the data presented to the user. For example, in an information retrieval application the user can specify the genre of news or sports reports he is interesting in and the format the news headlines are presented in. The importance of tailoring application content and form to the user's needs is clearly demonstrated by web-based news and information retrieval gateways such as Pointcast.

### 2.4. Speech and Natural Language Interface

Spoken and natural language interface design is the most challenging aspect of building a dialog system. A variety of open research problems and practical issues have to be tackled successfully and expertise from different disciplines has to be combined. Clearly good recognition and understanding performance are vital. In addition, the dialog system should allow for user exploration, provide alternate routes to completing a task, and should never let the user get lost or trapped. *Versatility*, *customizability*, *cooperation* and *supervision* (on a need-only basis) are some of the features of a good interface. Active research issues in dialog system design relevant to the goals outlined above include: handling spontaneous speech phenomena and robust speech recognition, dynamic creation of recognition grammars based on dialog history, dynamic generation of system prompts based on updated user beliefs, dynamic control of user/system initiative of the dialog as a function of user progress. Some practical issues that are equally important are echo cancellation of system prompts, timing in turn-taking (system time-outs), user or system interruption of dialog flow (barge-in) etc. Prototypes of dialog systems are starting to incorporate features that improve the flexibility of the spoken language interface, e.g., automatic relaxation of query constraints for queries with no matches, two-levels of system supervision, implicit confirmation of user input, shorter prompts, optional spelling of proper nouns [7]. Even more ambitious goals are to provide the interface with *intelligence* and *personality* [8, 18].

A *user-centric* approach to dialog system design can help address some of the issues raised above. A static or dynamic user model can be used to predict user intent given the current state of dialog. Grammar fragments can be associated with each of the user's intentions and a dialog state dependent grammar can be automatically generated as a weighted combination of these grammar fragments. In addition, a model of user beliefs can be used during generation of system responses; the user belief model is updated based on user input and system responses. User-centric grammar design and dialog management can help customize and automate the dialog system design process.

As mentioned above, dialog system design goes beyond building state of the art components, e.g., speech recognizer, natural language understanding, dialog manager. An important focal point of dialog system design is the *interaction* between the different modules. For example, acoustic confidence scores computed in the recognizer can be used to improve the performance of the understanding module and to determine the appropriate dialog strategy in the dialog manager. Dialog history can be used to improve recognition and understanding performance [14]. How to integrate the various sources of information to improve performance and enhance user experience remains an open research issue.

Finally designing generic modules that are *application independent* to the degree possible (clearly some application-dependent semantic representation is needed for designing the understanding module) is the ultimate goal of dialog system design. Universal algorithms and tools that can be applied with minimal modifications across applications will significantly speed up development time and create new challenging application areas.

### 2.5. User Feedback and System Evaluation

User feedback is a very important source of information for the designer of dialog systems. Despite recent progress, user interface and application design is mostly an empirical field [4]. Focus groups and user-experience forums can be used as a "proof of concept", i.e., to determine if the application and interface meets user's needs and preferences.

Little formal work can be found in the area of dialog system evaluation. Objective evaluation metrics exist for some of the modules of a dialog system, e.g., word and semantic label accuracy for the recognizer and under-

standing module, respectively, provided that transcribed and semantically annotated text corpora are available. End-to-end objective measures such as task completion and total number of dialog turns are often used to measure the functionality and efficiency of dialog systems. Clearly, the ultimate judge of system performance is the user. Subjective measures, such as system satisfaction, perceived efficiency, flexibility and robustness of the system can provide valuable metrics for the evaluation and iterative enhancement of a system. In [5], a mix of subjective and objective criteria was used to evaluate the ATIS systems. In [20], the correlation between objective (word accuracy, task completion, number of dialog turns) and subjective criteria (user satisfaction) was computed; the hope is that the relationship between objective criteria and user-satisfaction is more or less task-independent.

## 3. DISCUSSION

An important effort to provide a set of tools for spoken dialog system design was made at the CSLU at Oregon's Graduate Institute. The toolbox provides the modules needed for system building and a graphical user interface for application design. It has been successfully used by non-experts for building simple applications. The toolbox can be downloaded from the web; for a review see [17].

Most commercially available tools for dialog system design have limited natural language understanding capabilities, e.g., [19]. Dialog flow is typically modeled with a finite state machine and each dialog state corresponds to a single application action. In [19], tools for user-centric grammar design are provided, i.e., dialog state grammars are the union of all grammars that correspond to actions that the user can request. In general, current tools require a significant amount of application-specific work, e.g., manual writing of semantic grammars [15]. Further, most of the tools are designed with a specific set of applications in mind (communication assistant, travel reservation) and tuning of the modules is required for each new application domain.

A set of dialog tools should be able to automatically design an application based on transcribed and semantically annotated (in an action/attribute notation) data. In addition, generic dialog strategies should be available for groups of applications, e.g., database queries, information retrieval and browsing, so that the dialog and application flow are generated automatically based on a user profile. More research is needed to advance the state of the art and achieve these goals.

## 4. CONCLUSION

We have reviewed some important principles and trends in dialog system design. Previously dialog system development has relied on collecting many application-specific examples. Dialog systems were usually not extendable to other applications domains and provided little or no customizability to the user. Our work is motivated by the desire to learn how the various building blocks of a dialog system interact and how a dialog session can be automatically and dynamically generated based on user's preferences. The ultimate goal is to provide flexible and upgradable dialog systems, with general and reusable components. Designing an interface that is versatile and customizable is essential for enhancing user experience. More research is needed to meet these challenges. Modular architecture, automatic dialog generation, user modeling and the voice interface language are important contributions towards achieving these goals.

## 5. REFERENCES


[1] G. Antoniol, R. Fiutem, G. Lazzari, and R. De Mori, "System architecture and applications," in *Spoken Dialogues with Computers* R. De Mori, ed., Academic Press, pp. 583–621, 1997.

[2] N. Bernsen and L. Dybkjoer, "Is speech the right thing for your application?," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[3] R. De Mori, "Problems and methods for solution," in *Spoken Dialogues with Computers* R. De Mori, ed., Academic Press, pp. 1–22, 1997.

[4] R. Eberts, *User Interface Design*. Englewood Cliffs, NJ: Prentice–Hall, 1994.

[5] L. Hirschman et al, "Multi-site data collection and evaluation in spoken language understanding," in *Proc. of the Human Language Technology Workshop*, Mar. 1993.

[6] H.-K. Kuo, A. Pargellis, and C.-H. Lee, "Information and services manager customizes dialogue-based applications," in *Proc. ESCA Workshop Interact. Dialog. Multi-Modal Syst.*, Kloster Irsee, Germany, June 1999.

[7] L. Lamel, "Spoken language dialog system development and evaluation at LIMSI," in *Proc Internat. Symposium on Spoken Dialogue*, Sydney, Australia, Nov. 1998.

[8] S. Narayanan, A. Potamianos, and H. Wang, "Multimodal systems for children: Building a prototype," in *Proc. EUROSPEECH*, Budapest, Hungary, Sept. 1999.

[9] A. Pargellis, H.-K. Kuo, and C.-H. Lee, "Automatic application generator matches user expectations to system capabilities," in *Proc. ESCA Workshop Interact. Dialog. Multi-Modal Syst.*, Kloster Irsee, Germany, June 1999.

[10] A. Pargellis, H.-K. Kuo, and C.-H. Lee, "Automatic dialogue generator creates user defined applications," in *Proc. EUROSPEECH*, Budapest, Hungary, Sept. 1999.

[11] A. Pargellis, Q. Zhou, A. Saad, and C.-H. Lee, "A language for creating speech applications," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[12] R. Pieraccini, E. Levin, and W. Eckert, "AMICA: the AT&T mixed initiative conversational architecture," in *Proc. EUROSPEECH*, Rhodos, Greece, Sept. 1997.

[13] A. Potamianos and S. Narayanan, "Spoken dialog systems for children," in *Proc. ICASSP*, Seattle,Washington, pp. 197–201, May 1998.

[14] A. Potamianos, G. Riccardi, and S. Narayanan, "Categorical understanding using statistical n-gram models," in *Proc. EUROSPEECH*, Budapest, Hungary, Sept. 1999.

[15] S. Seneff et al, "Galaxy-II: A reference architecture for conversational system development," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[16] R. W. Smith and D. R. Hipp, *Spoken Natural Language Dialog Systems*. New York, NY: Oxford University Press, 1994.

[17] S. Sutton et al, "Universal speech tools: The CLSU toolkit," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[18] N. Suzuki, K. Ishii, and M. Okada, "Organizing self-motivated dialogue with autonomous creatures," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[19] Unisys Corporation, *Unisys Natural Language Speech Assistant*. 1999.

[20] M. Walker, D. Litman, C. Kamm, and A. Abella, "Evaluating spoken dialogue agents with paradise: Two case studies," *Computer Speech and Language*, pp. 317–347, 1998.

[21] K. Wang, "An event driven model for dialogue systems," in *Proc. ICSLP*, Sydney, Australia, Dec. 1998.

[22] Q. Zhou, C.-H. Lee, W. Chou, and A. Pargellis, "Speech technology integration and research platform: A system study," in *Proc. EUROSPEECH*, Rhodes, Greece, Sept. 1997.