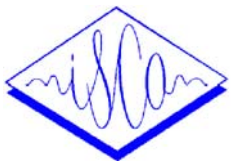

6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

AUTOMATIC DIALOGUE GENERATOR CREATES USER DEFINED APPLICATIONS

Andrew Pargellis, Jeff Kuo, Chin-Hui Lee

Dialogue Systems Research Department
Bell Labs, Lucent Technologies
600 Mountain Ave.
Murray Hill, NJ 07974, USA
({anp, kuo, chl}@research.bell-labs.com)

ABSTRACT

We report on the development of an Automatic Dialogue Generator (ADG), a software engine with associated library files, that simplifies the generation of new applications requiring a speech interface. A key feature of the ADG is that, given any task description specified in tables, the ADG can automatically generate a finite-state dialogue for that task, in a uniform and consistent fashion. The ADG generates a connected graph structure that provides the general dialogue flow between dialogue states, as well as all the dialogue and action components within each state.

Our design philosophy is to build an Application Generator (AG) that is highly portable. The AG contains four modular pieces, of which one is the ADG described here. The ADG consists of domain independent modules used to create a unique dialogue for each individual. The ADG accomplishes this by combining information from a user's profile with information from external sources to generate a dynamic and user-specific dialogue flow. The user specifies types of services, such as Sports and Email, and keywords are defined to prune and organize the information for each service.

1. INTRODUCTION

Our work is motivated by a desire to learn how the various elements of a dialogue session can be automatically generated. Previous dialogue system development has relied on collecting many application-specific examples [1-6]. This design strategy is usually not extendable to other systems built for applications in other domains. A limitation of systems using predefined dialogues is that a user has no opportunity to define, or even modify, a given application. Therefore, a user's expectations are not matched well with the dialogue system's capabilities, resulting in a poor user experience.

Our design strategy is to approximate any dialogue by a finite state specification. This strategy defines each state in a uniform and consistent manner that is portable to multiple domains. A finite state modeling of dialogues is a key feature of the ADG as it enables us to characterize each dialogue state by a fixed set of variables and parameters. This design allows us to automatically generate a unique dialogue for each individual, as well as dynamically adapt the dialogue to changing user preferences and available system resources.

The primary goal of the AG platform [7] is to enable a user to define their own application with a speech interface. The user may be a naïve nonprogrammer, or an experienced application developer who wishes to modify all aspects of a dialogue session including the low-level features of a particular speech engine. The AG is composed of four main components (see Fig.1): *Profile Manager* (PM), *Dialogue Manager* (DM), *Information Services Manager* (ISM, [8]), and the *Automatic Dialogue Generator* (ADG). The system's user interface is a speech platform developed at Bell Labs [9], the Speech Technology Integration Platform (STIP). Applications can access the STIP interface by using the Voice Interface Language (VIL, [10]). The VIL was developed in order to minimize the effort required to generate a unique dialogue, while at the same time retaining the capability to alter any detailed piece of the speech interface and dialogue performance. This language consists of domain independent commands that are used by the DM to run applications with a speech interface. Domain specific components are passed as arguments to each VIL command.

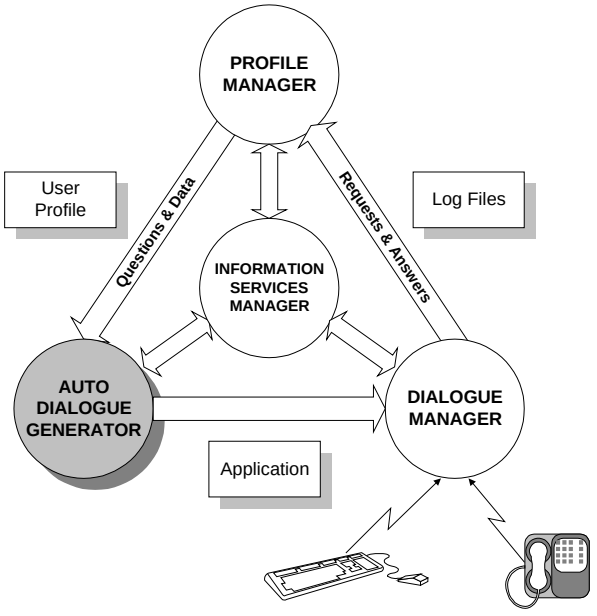

Figure 1. The Automatic Dialogue Generator component of the Application Generator platform. The ADG specifies a user-defined dialogue flow.

The PM enables a user to generate and manage a profile that defines their personalized application. The DM is a speech interface between the user and the STIP platform. For information retrieval services, the ISM manages information from local library sources and external sources such as websites

on the Internet. The ADG, discussed in this report and shaded in Figure 1, interfaces with the other components (ISM and DM) to generate the dialogue states and flow for the user-specified application. Each dialogue state contains specification tables that the DM uses to define the system's actions during a dialogue session. We used the Sports domain for the initial testing of the overall AG platform.

2. AUTO DIALOGUE GENERATOR

A dialogue session is modeled by a set of dialogue states and allowed transitions between them. Based on a finite state dialogue modeling strategy, each dialogue state is characterized by a uniform set of speech and application specific parameters. Such parameters may include Automatic Speech Recognizer (ASR) parameters (acoustic and language models, rejection and Barge-In thresholds), Text to Speech (TTS) parameters, prompts, state transitions and library functions defining implied semantic actions.

The ADG creates a dialogue session that is unique to each user by combining three sources of information: the user profile, information retrieved by the ISM, and a library that contains predefined library specification tables such as task specific subgrammars and URL addresses. These three sources are used to generate the user's personalized dialogue flow.

2.1. Organization of the Dialogue States

Figure 2 is a schematic overview of the ADG, shown for a particular user's application. The ADG requires data from three sources: the user profile, information from external databases such as those residing on the Internet, and task specification tables from a library. The ADG uses these data to create an application consisting of a dialogue flow that, at the highest level, is encoded in a series of dialogue states. Each state is a node in a connected graph and is represented by a subdirectory. These subdirectories form a directory tree that encodes the dialogue for a particular user's application. The dialogue specification files for each state are stored in the subdirectory for that state. In addition, each user has a profile directory and a data directory (mailbox) into which documents retrieved by the ISM are downloaded.

The subdirectories that describe the dialogue structure are shown in the lower left of Figure 2. In this example, the user chose the following services: Email, News, Sports, and Weather. From each dialogue state, certain allowed transitions are defined by the ADG based on the hierarchical relationships in the task specification tables. The local flow provides transitions to subtopics associated with that state and the global flow, accessible from all states, allows for shortcut transitions to menu states providing access to main groups of services.

2.2. Data Sources Used by the ADG

The user's intent is conveyed to the ADG through their user profile. The profile contains information such as the User Name, Password, list of desired services, and keywords for certain information services. For the Sports service shown in Figure 2, the user requested a set of subtopics: MLB (Major League Baseball) and Extreme Sports. The user also specified a list of keywords for each subtopic.

Many task specification tables are in an external library. These files predefine various aspects of dialogue flows, including: organization of different services, subgrammars to be used in concatenating state specific grammars, and domain specific variables such as a URL for information retrieval. The hierarchical relationships between services are listed in a file, for example, "main/info: help, news, sports, stocks, weather". This defines the flow between different dialogue states. A subgrammar library contains phrases that the ADG's grammar generator uses to create grammars for each dialogue state. The grammar generator also uses a synonym table that defines various ways a user may request something, such as voice-mail or voice-messages. Another library file contains domain specific variables such as a URL or sound file.

The ISM daily accesses web sites associated with the user's particular sports interests and downloads a set of documents that may be of interest to the user. At present, we use various www.espn.com sites. The documents are downloaded to the user's cache and preprocessed, thus avoiding network access delays.

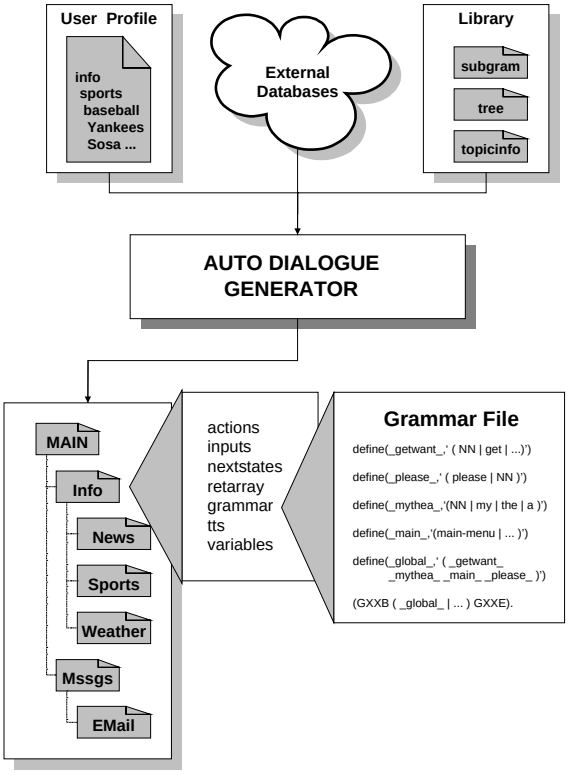

Figure 2. A block diagram of the Auto Dialogue Generator. At the top are the three data sources used to generate the dialogue states, each with its set of specification files, shown below.

2.3. Dialogue State Specification Files

Each dialogue state consists of a set of files that contain the dialogue and action components specifying how the system interacts with a user. The files for the Menu State, Info, are shown in Fig. 2. The files currently used are: actions (list of any actions to be performed), inputs (list of allowed commands

from the user), grammar (semantically acceptable phrases, vocabulary, and phonetic transcriptions), nextstates (list of possible dialogue states to go to based on user input), retarray (list of synonyms for input), tts (text string for the speech synthesizer), and variables (domain specific).

A key aspect of the ADG strategy is to characterize each dialogue state in a uniform and consistent manner. To this end, the dialogue components that define the Text-To-Speech (TTS) prompts and language models are built by using library templates and keywords. The ADG generates a text string that is sent to the TTS engine, which then generates synthesized speech. Keywords are combined with filler phrases to build the text strings. The Automatic Speech Recognizer (ASR) engine recognizes the user's response by using a language model with associated vocabulary. In our present study we use deterministic finite state representations for the language model where a set of sub-grammar templates is combined with key-phrase synonyms to create the language model. Synonyms allow the user to ask for the same thing in different ways. Sub-grammars presently include lists of query phrases, help requests, and connected digit phrases. Figure 2 shows part of a grammar file for the Info State. This was created from the subgrammar tables, where the global and local flow was determined from the state's position in the tree.

A mapping of possible user requests to actions and next-state transitions, including default states for exception and error handling, is defined. Methods for carrying out specified actions are also made available.

3. EXAMPLE: SPORTS SERVICE

Figure 3 is a schematic of an application requested by a single user. In this study, we are doing a detailed analysis of the various ADG features by concentrating on the Sports domain, expanded in Figure 3 to show some of the details. We are currently expanding the functionalities of the AG to include information services such as Email and News.

The user profile lists the user's preferences in desired services and, for Sports, the subtopics of interest, including keywords. In the case given in this report, the two subtopics and associated keywords were Baseball: (Sosa, Giants, Dodgers, Yankees) and Extreme Sports: (Daytona, Busch, Cup).

The ISM delivers documents daily to the user's mailbox. In order to do this the ISM must have the user profile and library information such as the URL to the site containing information requested by a particular service [10]. In the example given here, the ISM retrieves all documents from two sports sites, Baseball and Extreme Sports (which includes events such as triathlons, biking and skiing), and caches them in a main data library. Then, all documents in this main cache, containing at least one of the keywords specified in the user's profile, are copied into the user's cache.

As outlined in section 2 above, the ADG first generates the connected graph structure (see Fig. 2) along with associated dialogue flow. Next, the ADG generates the specification tables for each state. A major challenge is generating a grammar for an information service such as Sports. In addition to the general commands that allow the user to move between services, the grammar also includes important words contained in the documents selected that day, typically between 50 and 100 proper names, which is not too restricted for usage yet not too open to give poor performance. This allows the user to query their personalized database by asking for a subset of documents concerning topics such as "Boston Red Sox" or "Triathlon World Cup".

The user accesses the system by telephone. The DM first determines the user's name in the Entry State. Their application, including directory structure and associated dialogue component tables and files, is then accessed. The user may then ask for a particular service such as Sports. The user browses their database by asking for a list of titles or for a set of documents on a particular topic, for example: Could I hear more about Nikki Stone? In either case, the user may request a specific document by its number (they are numerically indexed) and then request to hear a brief summary of the document or browse the entire text content by paragraph. The barge-in technology is very useful here as it allows users to interrupt the information playback at any time, without the need to listen to an entire document that may be very long.

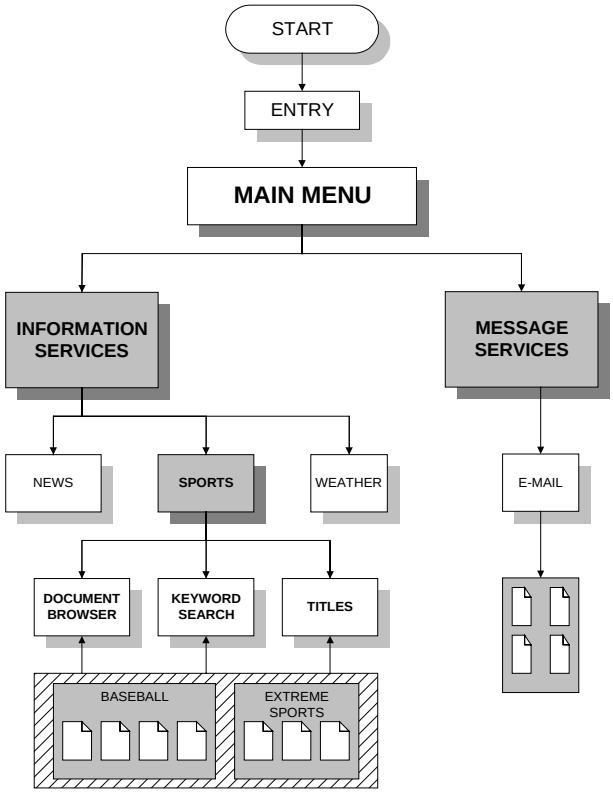

Figure 3. A block diagram of an application requested by a particular user. The Sports component is expanded.

4. RESULTS AND DISCUSSION

The main goal of this project has been to match user expectations with system capabilities and then automatically generate applications with associated dialogue flows. We developed an architecture that generates uniform finite state dialogues by specifying the task description in tables. The ADG dynamically adapts the dialogue to changes in the user profile and system resources.

4.1. Major Advantages of the ADG

There are several advantages to using an ADG to generate dialogues, instead of generating the code for a domain specific dialogue by hand. They are:

- Prompts and grammars are generated in a consistent manner
- Prompts and grammars are generated dynamically
- The application is organized visually as a directory tree
- New, user-specified applications, can be quickly generated

Generating prompts and grammars that are consistent eliminates confusion for the user where some options might be inadvertently omitted at certain points in a hand-generated application. In addition, any changes in a library subgrammar, such as including more natural ways for a user to request a given service, immediately become incorporated into *all* the dialogue specifications for *all* of the users. Grammars and prompts must also be generated dynamically for information services where the data content is continuously changing. Both the outgoing TTS prompts and grammars for ASR engine need to be updated as the service's data content changes.

The visual organization of the dialogue flow makes it easier for an application developer to design new applications. All of the specification tables for each dialogue state are immediately available simply by clicking on the appropriate text file.

The ADG makes it possible to modify existing services and build entirely new applications by merely altering a few library files or a user's profile. A new user's application is generated in a few minutes once their profile has been created.

4.2. Future Research Topics

There are certain types of states that might perform in a more satisfactory manner if they were not predefined but were generated during the dialogue session. This requires devising a means of automatically classifying states in order to model different types of dialogue phenomenon. For example, context sensitive states that are affected by the dialogue history should take into account the user's specific situation at that point in the dialogue. This would include Help states, which might be more useful if they were dynamically constructed, considering such things as number of times the user has attempted a particular transaction, required information not yet supplied by the user, number of misrecognitions, etc.

5. SUMMARY

We have successfully built a platform that automatically builds dialogues approximated by a finite state specification. The individual states are generated in a consistent and uniform manner that is portable to other domains. The user's preferences are defined in a user's profile and the ADG combines this with information about the system resources. The resultant application is dynamic and updates the dialogue specifications when either the user's profile or system resources change.

6. REFERENCES

1. H. Aust, O. Schroer, "Application Development with the Philips Dialog System", *ISSD 98*, Sydney, Australia, 30 November 1998, p. 4-1
2. L. Devillers, H. Bonneau-Maynard, "Evaluation of Dialog Strategies for a Tourist Information Retrieval System", *ICSLP'98*, Sydney, Australia; 01–04 December 1998
3. S. Seneff, et al., "Galaxy-II: A Reference Architecture for Conversational System Development", *ICSLP'98*, Sydney, Australia; 01–04 December 1998
4. C. Kamm, S. Narayanan, D. Dutton, R. Ritenour, "Evaluating Spoken Dialog Systems for Telecommunication Services", *5th European Conference on Speech Communication and Technology*, Rhodes, Greece; 22-25 Sept 1997
5. M. Walker, et al., "Evaluating Competing Agent Strategies for a Voice Email Agent", *5th European Conference on Speech Communication and Technology*, Rhodes, Greece; 22-25 Sept 1997
6. S. Issar, "A Speech Interface for Forms on WWW", *5th European Conference on Speech Communication and Technology*, Rhodes, Greece; 22-25 Sept 1997
7. A. Pargellis, J. Kuo, C.-H. Lee, "Automatic Application Generator Matches User Expectations to System Capabilities", *Interactive Dialogue in Multi-modal Systems*, Kloster Irsee, Germany; 22– 24 June 1999
8. J. Kuo, A. Pargellis, C.-H. Lee, "Information Services Manager for Customized Interaction with Dialogue System", *Interactive Dialogue in Multi-modal Systems*, Kloster Irsee, Germany; 22–24 June 1999
9. Q. Zhou, C.-H. Lee, W. Chou, A. Pargellis; "Speech Technology Integration and Research Platform: A System Study"; *5th European Conference on Speech Communication and Technology*, Rhodes, Greece; 22-25 Sept 1997
10. A. Pargellis, Q. Zhou, A. Saad, C.-H. Lee, "A Language for Creating Speech Application", *ICSLP'98*, Sydney, Australia; 01–04 December 1998