

## Promoting Portability in Dialogue Management

Joseph Polifroni and Grace Chung<sup>1</sup>

**Introduction:** One of the hardest parts of designing and building a conversational system is configuring and coding the dialogue manager. Components such as speech recognition and language understanding have both been modularized, with domain-specific information contained in external files and models. SPEECHBUILDER [1] was designed to make these two components even easier to configure, with a Web-based graphical interface to help developers write grammars and create language models for recognizers. However, dialogue management has resisted this push towards portability and modularity, since its role in planning and response generation was considered ultimately too domain-dependent.

In the course of building dialogue managers for each of our separate systems (e.g., weather [2], air travel [3], flight status, urban navigation [4], and task delegation [5]), we have noticed that the basic functionality repeats itself across domains. For example, each system must gather information from a user and prompt the user for critical missing pieces, and each system must filter responses from the database to insure that they match user-specified constraints. Furthermore, certain categories of information, such as dates and times, recur in multiple domains. Users can ask for flights on “Tuesday,” or about the weather “the day after tomorrow,” or for the estimated landing time of a flight scheduled for “late this afternoon.”

We have recently begun an effort to develop a domain-independent dialogue manager that can be used as part of the SPEECHBUILDER framework. This dialogue manager is being designed to enable developers to construct more complex conversational systems without modifying underlying code. We are also incorporating pre-specified grammars for semantic concepts such as dates and times, along with servers for interpreting and canonicalizing these concepts. Our purpose is to give SPEECHBUILDER developers a pre-compiled way of understanding and representing generic concepts, drastically reducing the work required to quickly configure and deploy a conversational system.

**Approach:** Turns in a conversational system can be thought of as a process of form-filling. For example, in a flight reservation domain, the user needs to specify, or the system needs to elicit, at a minimum, a source, a destination, and a date before any answers can be provided. The underlying “form” for such a domain would contain slots for these keys, as well as optional keys (e.g., airline, time of day, price). Forms vary from domain to domain, but maintaining a meaning representation as a set of key/value pairs in a form-like structure is constant throughout all our domains. Dialogue management involves acquiring the necessary values for these keys to complete a particular transaction.

An individual dialogue turn can be viewed as divided into four phases, some of which are not visited in each turn. In the first phase, *pre-retrieval*, the system must verify the input, check confidence scores, if available, and interpret fragmentary responses in context. During pre-retrieval, the system also determines whether a query can be sent to the content provider, i.e., whether sufficient constraints have been elicited from the user to obtain data. The second phase is *retrieval*, where a query is formatted and dispatched to the content provider. After retrieving database tuples, the dialogue manager must *filter* the response based on the constraints from the user (e.g., find the cheapest) and order the resulting set for presentation to the user. Finally, in *response construction* a response frame is produced and comments are added where necessary (e.g., to explain why a particular piece of information is unavailable and offer an alternative). Response construction in our systems also involves adding follow-up queries, called *system initiatives* or *continuants* that help guide the user. This phase of dialogue management is also responsible for providing help messages, which are usually conditioned on the dialogue state.

Control during the phases of dialogue management is maintained by an external, domain-specific table specifying a set of operations and the key/value pairs that control the order of execution of these operations. In the course of a given turn, the dialogue manager works its way through this *dialogue control table*, evaluating keys and their values to see if a particular operation should activate. Each operation is called with a dialogue structure containing all the necessary information from the history, the current utterance (along with alternative hypotheses), and user-specified preferences. Each operation is able to modify the values for particular keys and then return one of three possible moves: Continue, Stop, or Start Over. Multiple rules can apply during a single turn.

**Progress:** We have implemented a dialogue manager that is able to perform the basic functionality required by the four phases of dialogue management in a domain-independent way. Domain specific information is specified in an external

---

<sup>1</sup>Grace Chung is currently a research scientist at CNRI, Reston, VA

table in a standardized format. This generic dialogue manager has been used successfully to answer basic queries in four very different domains: hotels, including prices and amenities; television schedules and descriptions of shows; personnel in the Laboratory for Computer Science; and financial planning.

We have also begun an effort to standardize certain semantic sub-domains, in particular dates and times. We have extracted parse rules for these concepts from multiple, domain-specific grammars, and organized them centrally in separate grammars for inclusion, in much the same way that libraries of code are included in SPEECHBUILDER developers' grammars. We have also developed a "canonicalizing" server, which can take the meaning representation from the parse of dates and times and return standardized values. In this way, a SPEECHBUILDER developer will not need to worry about writing parse rules for phrases such as "next Thursday" or "mid-afternoon," or interpreting these phrases. Dates and times will automatically be transformed from their English surface forms to canonical representations such as "FEB 07, 2002" or "begin\_time: 1300 end\_time: 1600".

We are currently in the process of developing a text-based user interface to the dialogue control table. The interface will enable developers to examine the control structure and manipulate the order of rules and placement of variables. We have a prototype in place and are ready to test it with naive system developers. We are currently using the hotel information domain to test most extensively the functionality of the generic dialogue manager, the canonicalizing server, and the new format for specifying domain-specific dialogue functionality. To insure that our dialogue manager can handle the sorts of interactions that arise with real, dynamic data, we are using an online content provider for up-to-date information about hotels. User interactions within this domain, using the generic dialogue manager to plan responses, should yield useful data for further development and testing of all system components.

**Future:** We plan on making the generic dialogue manager part of the SPEECHBUILDER distribution. In doing so, we expect to discover areas in which we must expand and enhance the server. We would also like to make the GUI interface to the dialogue control table accessible to SPEECHBUILDER developers and build on it based on that experience. We are also planning to explore ways of automatically and generically configuring a dialogue, based on the result of queries to a content provider.

**Research Support:** This research was supported by DARPA under contract N66001-99-1-8904 monitored through Naval Command, Control and Ocean Surveillance Center, by an industrial consortium supporting the MIT Oxygen Alliance and by a contract from NTT.

#### References:

- [1] J. Glass and E. Weinstein, "SPEECHBUILDER: Facilitating spoken dialogue system development," in *Proc. Eurospeech, 2001*, Aalborg, Sept. 2001, pp. 1335–1338.
- [2] V. Zue, S. Seneff, J. Glass, I. L. Hetherington, E. Hurley, H. Meng, C. Pao, J. Polifroni, R. Schloming, and P. Schmid, "From interface to content: Translingual access and delivery of on-line information," in *Proc. Eurospeech, 1997*, Rhodes, Sept. 1997, pp. 2227–2230.
- [3] S. Seneff and J. Polifroni, "Dialogue management in the MERCURY flight reservation system," in *Proc. ANLP-NAACL 2000 Satellite Workshop*, Seattle, May 2000, pp. 1–6.
- [4] J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue, "Multilingual spoken-language understanding in the MIT VOYAGER system," *Speech Communication*, vol. 17, no. 1-2, pp. 1–19, Aug. 1995.
- [5] S. Seneff, C. Chuu, and D. S. Cyphers, "Orion: From on-line interaction to off-line delegation," in *Proc. International Conference on Spoken Language Processing, 2000*, Beijing, Oct. 2000, pp. 142–145.