An architecture for natural language speech applications

Florian Metze
Technische Universität Berlin, Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7, D-10587 Berlin
florian.metze@telekom.de

Frank Oberle
T-Systems Enterprise Services GmbH, Systems Integration,
Project&Design, Business Unit Telco, Line of Business Product & Services, Goslarer Ufer 35, 10589 Berlin
frank.oberle@t-systems.com

Abstract

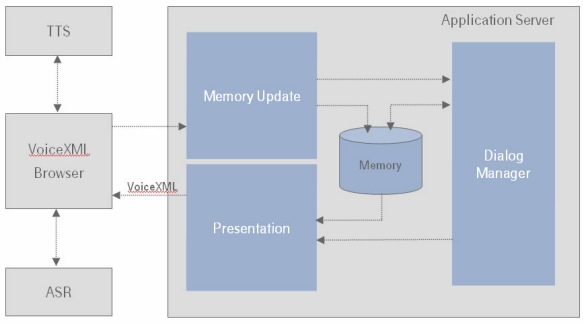
Traditional VoiceXML applications follow the form-based dialogue modelling approach of VoiceXML and distribute the control tasks of an application across several forms. This architecture is sufficient in order to realize simple command and control tasks or more advanced concepts like mixed initiative, but it fails as soon as natural language concepts like overloading of answers, interruptibility or elliptical expressions are required.
To overcome this deficit, we propose to concentrate all elementary control tasks of an application in one single component on the server. When combined with a session memory, this component can draw on the whole session when deciding about the next dialog step.

Introduction

The dialog control of a traditional VoiceXML2.x application is distributed between several VoiceXML forms and is therefore restricted to the limited capabilities of a VoiceXML form. The idea is to concentrate all important control tasks of the application in one component. By adding a memory component, the dialogue control is no longer restricted to the form level view, but it is able to keep track of the whole session information.
Furthermore, VoiceXML 2.x applications show a low degree of reusability because they don't separate control from presentation. To overcome this deficit SCXML (State Chart Extensible Markup Language) was designed by the W3C's Voice Browser Working Group. The proposed architecture is focused on a clean separation of data, presentation logic and flow control, thus following the MVC (model, view, control) paradigm that has proved valuable in many web applications. A refactoring of VoiceXML is currently under discussion in order to keep the presentation logic separate from the flow control and the global application data.
Thus a well designed architecture for natural language speech applications has to provide both:

- a clean separation of control and presentation,
- the possibility to integrate an intelligent dialog management component, e.g. following approaches like ISU (information state update).

Components of Natural Language Interaction Management



Natural language processing, i.e. the interpretation of user utterances, is usually provided by the ASR server. Common voice browsers thus are able to submit already the semantic interpretation of the user utterance to the server. Especially for statistical language models

(SLMs) it might be an alternative to send the unprocessed user utterance to the server. Thus natural language processing can be located on the server and can be closely linked with the rules of the dialog management, i.e. the session memory update and the calculation of the next dialog step.
The memory update component, located on the application server, has to handle all incoming requests from the client and, based on rules, has to update the memory of the session.
An update of the memory triggers the dialog manager who determines the next task. The calculation of the next task is based on the state of the memory component and updates the memory again.
The memory component has to store at the very least the current task and the state of the slots, which have to be filled during the interaction. Depending on application-specific requirements, it might be necessary to augment the memory, e.g. by a stack, list or set of current tasks. In [Fernandes, Endriss] an approach based on DFA (deterministic finite automata) is discussed. The DFA approach is enhanced by adding a memory in form of an instance of an abstract datatype such as a stack, a set, or a list, e.g. to handle interruptability or embedded tasks.
According to the task determined by the dialog manager as the next dialog step, the presentation has to render the according VoiceXML page. This comprises the delivery of the prompts (as wave files or textual) as well as the ASR grammar (e.g. textual). When rendering the VoiceXML page for the next task, the presentation has to take into account all the previously stored information in the session memory, related with this task.

Asynchronous Client Server Communication for an Intelligent Distribution of Control Tasks

As mentioned above, there is no need to concentrate all control tasks on the server. The idea of an intelligent distribution of control tasks is to handle only those calculations on the server, which require the knowledge of the session memory. The following definition might help: a task covers not only a prompt and a grammar but also the escalations for the events “nomatch”, “noinput” and “help” or, in case of mixed initiative, the inclusion of more than one <field> and an <initial> tag.
The idea is then to calculate the decision for the next task serverside whereas the internal control of a task, i.e. the handling of the <initial> and the filling of more than one <field> as well as the nomatch-, noinput- or help-handling can be executed clientside as before. An asynchronous client server communication thus may help to distribute control tasks more flexible between client and server and to deliver more complex VoiceXML pages to the client in order to limit the client server data traffic.
During the execution of a task on the VoiceXML client, asynchronous communication supports a continuous update of the session memory on the server. Furthermore, the execution of the task on the client can be aborted immediately, if a user utterance is meant to initiate a new task. Together with the request for a new page, the client has to submit the slots defined so far, in order to save them in the session memory for usage later in the dialogue, no matter whether this information is related with the current task or otherwise. A task which was aborted and has to be finished later has to be saved in the session memory too, e.g. by adding a stack to the session memory. When rendering the VoiceXML page for the next task, e.g. a new task or a previously aborted task, the presentation has to consider all the information already collected in the session memory, which is related with this task. In this way, natural language concepts like overloading of answers, interruptibility, embedded tasks, or elliptical expressions can be handled by the dialog management.

Conclusion

For the modelling of natural language speech applications many different approaches have been discussed. Plan based approaches assume that attitudes like knowledge, belief, desire, and intention play a role in conversational behaviour. Other approaches are focusing on the intentional attitudes of the caller. Furthermore the current discussion focuses also on the ConcurTaskTree (CTT) approach of [Paterno et al.] to model the human machine interaction.
Those approaches will result in a very complex dialog management, neither easy to learn nor easy to understand for the usage in practice. At the same time, [Kronlid, Lager] propose, that the dialog management could be based on a simple Harel statechart ([Harel]), slightly extended to implement the Information-State Update (ISU) approach. Because the W3C has selected SCXML as the basis for future standards in the area of voice and multimodal dialogue systems, this seems to be an interesting approach.
If it could be demonstrated that all important and essential concepts of enhanced natural speech interaction could be covered by an SCXML based interaction management, enhanced by adding a simple structured memory component, this could be an encouraging approach for future voice dialog systems.

References

Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. In Science of Computer Programming 8, North-Holland.

Fernandez, R. ; Endriss, U. (2007). Abstract Models for Dialogue Protocols. In Journal of Logic, Language and Information vol. 16, no. 2, pp. 1221-140, 2007

Kronlid, F.; Lager, T. (2007). Implementing the Information-State Update Approach to Dialogue Management in a Slightly Extended SCXML. In Ron Artstein and Laure Vieu (Eds.) *Proceedings of the 11th International Workshop on the Semantics and Pragmatics of Dialogue (DECALOG)*, Trento, Italy. pp. 99-106, 2007.

Paterno, F.; Mancini, C.; Meniconi, S. (1997): ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. Proc. InterAct 1997, 363—369, 1997.