State-, HTML-, and Object-Based Dialog Design for Voice-Web Applications

Gunar Fiedler¹, Thomas Schwanzara-Bennoit², Bernhard Thalheim¹, and Peggy Schmidt¹

¹ Computer Science Institute, University Kiel, Olshausenstrasse 40, 24098 Kiel, Germany
Email: fiedler|thalheim|pesc@is.informatik.uni-kiel.de
² Computer Science Department, Databases and Information Systems Group, Brandenburg University of Technology at Cottbus, PO Box 101344, 03013 Cottbus, Germany
Email: tsb77@web.de

Abstract. The user’s interaction with internet information services is usually based on dealing with a graphical interface, but because of the improvements in speech technology over the last years more and more phone speech information services appear. In March 2004 the W3C published the VoiceXML 2.0 recommendation to bring the advantages of web-based development and content delivery to interactive voice response information systems. Although classical modeling strategies for building graphical user interfaces can also be used for their voice based counterparts, there are certain aspects where voice response systems need special attention. For example, an intuitive navigation structure is crucial due to limited possibilities in expressing information to the user.

We present a systematic development process which integrates conceptual modeling, rapid prototyping, simulation and system documentation for voice based information services. Following this approach we have implemented a voice based interface for an e-Democracy portal which will be used as an example within this paper.

1 Introduction

In a normal man-machine conversation the partners change their roles between speaking and listening to perform a dialog. Interactive voice response information systems use natural speech to present requested information to the user and accept either Dual-Tone-Multifrequency (DTMF)¹ or spoken commands as the user’s input.

Processing natural speech is very complex and still an open field of research. Speech processing systems consist of four major components, shown in Fig. 1:

– The **speech recognizer** translates recorded wave input into text strings.

¹ Pressing a key (0-9, *, #) on a telephone produces a sequence of two well defined tones.
Fig. 1. Major components of speech applications

- The **string analyzer** checks the recognized input to extract the meaning of the input.
- The **dialog controller** performs previously defined actions according to the extracted input and produces the system’s answers.
- The **speech synthesizer** generates natural speech as wave output.

Every component encapsulates complex processings. In this paper we focus on designing the **dialog controller**. We do not address speech recognition, speech analysis, or speech synthesis but we have to know about restrictions made by these components which influence the system’s dialog design.

### 1.1 VoiceXML Based Information Services

VoiceXML (VXML) is a W3C recommendation for specifying dialogs featuring synthesized speech, digitized audio, recognition of spoken commands or DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. It defines constructs like forms, menus, and links and specifies the **Form Interpretation Algorithm** by which the dialogs are interpreted. A caller uses DTMF or speech as system input and gets synthetic speech or pre-recorded audio as system output.

The architectural model assumed by this paper is depicted in Fig. 2.

Fig. 2. Architectural model of VoiceXML applications
The central component within a VoiceXML application is the VoiceXML interpreter. It acts like the web browser in graphical environments. The interpreter is embedded within a VoiceXML Interpreter Context responsible for certain "administrative" tasks like detecting an incoming call or listening for special escape sequences. Dialogs are stored in VXML files on a document server which is typically a normal web server accessible via HTTP. The implementation platform is controlled by the VoiceXML interpreter as well as the VoiceXML interpreter context and generates events depending on user actions and system events.

1.2 Problems with Voice Based Man-Machine Communication

Although speech technology improved over the years there are some drawbacks and problems:

- **Finances**: Telecommunication providers, hardware phone systems, speech recognition components, VXML browsers and Text-to-Speech (TTS) voices cost a lot of money.
- **Naturalness**: Calling users expect to speak to a human. They do not want to talk to a computer.
- **Reliability**: Speech recognition components are not as good as a human agent in a call center.
- **Patience**: It takes time to listen to the system.
- **Navigation**: The information presented in a dialog scene is limited so navigation between dialogs becomes crucial.
- **Limited Conversation**: VXML dialogs contain only data and sentences they are programmed for.
- **Development Process**: Systematic approaches are just in the beginning.

1.3 The Running Example: An e-Democracy Portal

We designed a VXML based interface component for an existing e-Democracy information system called SeSAM which will be used as the running example within this paper. The system is designed as a community portal where citizens and members of municipal parliaments can access information about the current political work like agendas of actual meetings, bills and motions, messages, and contact information.

Although SeSAM itself is a running application the voice portal is intended to be a prototype for exploring current possibilities of speech technology. We made use of open standards and standard software. The application is written in Java and runs on a Tomcat web server. The dialogs follow VoiceXML 2.0. For simulation the OptimTalk VoiceXML interpreter with DTMF as primary input and speech recognition as secondary input is used.
2 The Development Process for Voice Applications

The development process for voice-web applications has to deal with some specialities. The amount of information presentable within a single dialog is limited. For example, using DTMF keys as a source of input restricts the length of a menu to a maximum of 12 items. In practice the menus have to be even smaller due to human’s receptivity and patience. This leads to a crucial "deep" navigation and causes stronger needs for system adaptation to the user’s profile and portfolio. Designing a menu structure transporting the right information at the right time while not straining the user can often only be handled in a handicrafted way. Following some kind of "trial and error" strategy the designer itself or a group of test persons have to evaluate the application under usability aspects. Their remarks are used to improve the dialog structure until a feasible result is reached, until the dialog "sounds nice".

The big problem with this kind of strategy is the long design cycle. Listening to a voice dialog sequence takes a lot of time. While even complex graphical information can be perceived within only a few seconds, especially if the designer can focus on local improvements, the dialog sequence test takes place always in real time and lasts for many seconds or even minutes.

To deal with these limitation we propose a navigation centered application development process with different stages of precision. We use Finite State Machines (FSM) for modeling the dialog transitions; static HTML pages for defining the ornamentation of dialog scenes, for visual simulations, for documentation, and as executable specification; dynamic HTML pages for coupling visual simulations and real application data as well as fixed transformation rules for producing VXML dialog files from the dynamic HTML prototype.

The development process is depicted in Fig. 3. It is structured in several phases:

1. **System analysis**: an informal or semi-formal description of the application’s goals is created.
2. **Process analysis**: the underlying business processes are analysed.
3. **Use case analysis**: the user’s interaction with the system is identified.
4. **Designing navigation**: the interaction flow is specified by creating a finite state machine.
5. **Designing dialog appearance**: the states of the FSM are filled with content.
6. **Defining media objects**: the application’s business data is created together with views on this data representing the data’s appearance within the dialog scenes.
7. **Coupling dialogs and data**: the interface logic is coded, e.g. accessing media objects in a database.
8. **Creating VXML output**: the HTML tags are translated to corresponding VXML tags to form valid VXML dialogs.

The following sections describe the different stages of the development process in detail.
2.1 System Analysis and Story Specification

A story describes the set of all possible intended interaction of users with the system. It is often defined generally in a product specification document in an informal or semi-formal way.

A brief description of our SeSAM application as a story space could be the following:

Story {
  Visitor {Group Information,
             Public Messages,
             Public Political Appointments}
  RegisteredUser { Group Information, Messages, Appointments}
}

"The system is used by registered users like party members or politicians and visitors, mostly the citizens of the town. The user will get information about members and parliamentary groups, messages and appointment information by phone."

The proposed story specification method is not rigid, so everyone can use his/her favorite form.
2.2 Process Specification

A process could be anything that operates for a period of time, normally consuming resources during that time and using them to create a useful result.

A process model is a general approach for organizing a work into activities, an aid to thinking, and not a rigid prescription of the way to do things. It helps the software manager to decide what work has to be done in the target environment and in what sequences the work is performed.

A process specification is useful for systems where voice interfaces are intended to support the processes or goals of the organization. An employee who answers customer questions by phone could be assisted by an automatic voice system which offers the desired information by a structured FAQ.

Process specification costs time for analysis and design, but it is a good documentation and a useful basis for further requirement specification.

2.3 Use Case Specification

Based on intuitive goals, a story, and a business process description we model the requirements of the application with use cases. A use case describes a set of actions of a system watched by the user which causes some results. In our case it describes the set of scenarios and is started by an actor. A scenario is a specific set of actions that clarify the behavior of the voice system and represent the basis for abstraction and specification. An actor represents one role outside the system.

We used the UML notation for defining SeSAM use cases in different levels of abstraction, please refer to Fig. 4. There are other use case diagrams for the actions coordinate, communicate and get member information which should clarify and describe the interaction between the user and the system.

2.4 Dialog Structure Specification

The result of the story definition and requirement analysis is a specification of the typical interaction of a user with the computer voice system. But this documentation contains no information about the navigation and the dialog structure of the VXML pages. Based on the nature of discussion each VXML page contains the data spoken by the TTS system and data about the reaction if the user says something.

We model navigation, the structure of the whole story, and the structure for each individual use case with finite state machines \( M = (F, A, Q) \) where:

1. \( Q \) is a finite set of conversation states
2. \( A \) is a finite set of user input symbols
3. \( F \) is a function of the form \( F: Q \times A \rightarrow Q \)

Every state of the FSM is regarded as a conversational state. In a conversational state the system says something, asks the user something for an answer
Fig. 4. Use case modeling

and goes to another conversational state, depending on the user’s input. Fig. 5 shows a simplified version of the dialog structure specification for the use case communication of our SeSAM application. The system begins to read the start dialog at the start page and asks the caller what to do. The caller could decide by speaking or using DTMF to go to services where all available information services will be read. From the service overview the caller could go to message services, selects "my messages", gets a message list and selects then his/her desired message which will be read.

It is very important to clarify the transitions for every state to other states. Normally, like in HTML pages, the user gets different standard options, e.g. "go back" or "go back to services" on every page. These transitions are not listed in Fig. 5 but they are part of it and reduce the fan-out of every state available for task specific transitions.

At the end of this stage you should have a generic FSM for the whole story and special FSMs for every single use case. All use case FSMs have to be integrated into one story FSM.

The dialog structure specification as a FSM defines the navigation structure of the voice application but it contains no data about spoken sentences. This is done by a dialog step specification in static HTML.

2.5 Dialog Step Specification in HTML

The FSM specification is now added with data of spoken sentences and prompts through the HTML dialog step specification. An HTML page contains conversational data, prompts and answers for one conversational state and links to other
Fig. 5. Dialog structure specification through FSMs

conversational states. We map every state of a FSM to one HTML page. Figure 6 shows a dialog step definition of the state services from Fig. 5.

Every transition of the FSM (a user input which results in a new state) is mapped into a hyperlink to another page. It is free to the developer to use graphical mouse hyperlinks or JavaScript user input functions. On each page the machine says something and expects input from the user to go to another page. It is important to keep in mind that the specification is the basis for the voice application and text and menus should be as short and objective as possible. If there is any dynamic information, e.g. from a database, then it will be modeled with static example data.

At the end of the dialog step definition you should have a HTML prototype of your VXML application which behaves similar to the VXML application. One advantage of this specification is the graphical definition of the dialogs, software developers are used to this. On the other hand you get a prototype as a set of HTML files which could be simulated to analyse the dialog and to improve the conversation. As a third outcome of this stage the static HTML prototype can be used as the basis of the application’s documentation.

2.6 Media Object Integration

The static HTML prototype of the voice application contains navigation structure and the ”adornment” of the scenes. It is filled with sample data and contains no dynamic information, e.g. from databases.
You have selected information services.
Please select now your desired service:

- [ ] Member information
- [ ] Messages
- [ ] Appointments

---

Now we design the media objects of our application to add dynamic functionality to our prototype. Generally, a media object is a view on a database expressing the aspects that are of interest in a dialog scene. From the object oriented point of view a media object is the member of a class with certain attributes and methods. In our approach media objects carrying the data the system and the caller are talking about. The design of business and media objects is straightforward, the developer is free to use his/her favorite method. In the case of a multichannel information system all interfaces share the same business objects so this stage is reduced to the definition of views over an existing database.

Once media objects are defined they can be integrated into the static HTML prototype. This integration is done by replacing the sample data in the HTML pages with the media objects and their functionality. At this point you have to choose a programming language and you have to implement the functions of your classes. Fig. 7 shows HTML code within a Java Server Page which is enriched by the media object message.

At the end of this stage we have a dynamic graphical HTML prototype which has the same dialog flow and functionality as the VXML application and which uses the same data from the database. Now the HTML web application can be simulated and reviewed under consideration of conversational and speech aspects.

The listing shows HTML code with JSP scripting elements. Other programming languages like PHP or Perl could also be used for implementation.

**2.7 Translation to VXML**

In this step each HTML page is translated to a VXML page. The VXML page uses the same media objects but some output functions have to be modified to get VXML output instead of HTML. This process can be automated by software.

---

2 e.g. a graphical and a voice interface coexist
tools. Table 1 shows the most important VXML elements which were used in our application.

The result of this translation is a ready VXML application which could easily be run on a normal PC or a VXML based telecommunication system. Listing 8 shows a typical SeSAM VXML document.

2.8 Simulation and Test

The simulation and run of the application is important to validate and check the behavior, the data and the functionality of the system. We suggest the following points to perform rapid development and validations:

1. Specify and test FSMs for navigation and input validation
2. Specify and test a static HTML dialog for an use case
3. Specify and test a static VXML dialog for an use case
4. Add dynamic media objects to HTML files and test them
5. Translate HTML to VXML and test them in a web browser with XML support and XML validation
6. Test each VXML file in your voice browser (OptimTalk, Elvira)
### Table 1. HTML to VXML translation

<table>
<thead>
<tr>
<th>HTML tag</th>
<th>VXML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;...&lt;/html&gt;</td>
<td>&lt;vxml&gt;...&lt;/vxml&gt;</td>
</tr>
<tr>
<td>&lt;p&gt;text&lt;/p&gt;</td>
<td>&lt;block&gt;text&lt;/block&gt;</td>
</tr>
<tr>
<td>&lt;ol&gt;&lt;li&gt;...&lt;/li&gt;&lt;/ol&gt;</td>
<td>&lt;field&gt;&lt;option&gt;...&lt;/option&gt;&lt;/field&gt;</td>
</tr>
<tr>
<td>&lt;a href=&quot;...&quot;&gt;text&lt;/a&gt;</td>
<td>text&lt;goto next=&quot;...&quot; /&gt;</td>
</tr>
</tbody>
</table>

7. Test the whole use case completely without visual support

Application simulation starts in the design phase. Any FSM, static or dynamic HTML page, or VXML page should be simulated with suitable tools to validate the design and the specification of the application.

At any time it is important to interact with the VXML application and to simulate it - without a visual interface. The tester has to listen to it because the real application environments are (mobile) phones or IP-telephony. Through a simulation you can also validate the FSM, HTML, and media object specification since the VXML programmed code is based on these specifications.

### 3 Conclusion

In this work we presented a specification for the rapid development process of voice application services. The specification is part of the documentation and basis for implementation. Due to the codesign of a HTML prototype and a VXML application the specification is easily understandable and is based on a robust finite state machine navigation structure. This specification approach speeds up implementation time since you can implement and simulate the first prototypes graphically instead of listening to an application.

### 4 Future Work

The proposed development process carries great opportunities for automation. Once a FSM is specified a tool can generate stubs for the static HTML prototype. Because the structure of these files is limited to a subset of HTML the reverse way is possible, too. The translation between HTML and VXML is nearly a one-to-one mapping, only VXML specialities like audio recording are not directly representable in HTML.

Another point of further research is the integration of the development process of graphical and voice interfaces under the aspects of component or code reuse.
<jsp:useBean id="member" class="SeSAM.VXML.Member" />
<% member.setSession(session); %>
<xml version="1.0"/>
<vxml version="2.0">
<form>
  <block>You chose member information services.</block>
  <field name="choiceMember">
    <prompt>Please choose the desired group or
    other services</prompt>
    <option dtmf="<%= member.getDTMFCounter()%>">
      value="services">back to services</option>
    <%= member.getTopGroupsOptions()%>
    <filled>
      <%= member.getTopGroupsFilledIfs()%>
      <if cond="choiceMember=='services'">
        <goto next="/servlet/services" />
      </if>
    </filled>
  </field>
</form>
</vxml>

Fig. 8. VXML document with JSP media object

References


