VoiceXML Application Development Life Cycle: Building Usable Voice User Interfaces

OnOn Hong
10/04/04
Introduction

The Web has revolutionized how people communicate and share information. Businesses deploy millions of Web services to consumers with Internet access. Internet and telephony used to be two separate technologies requiring a specialized telecom expert to build applications accessible over the phone. VoiceXML bridges the gap; it leverages the existing Web infrastructure and enables Web developers to build voice-enabled Web applications accessible from any telephone, by anyone, anywhere, anytime.

Users interact with VoiceXML applications via a Voice User Interface (VUI) similar to the way they interact with traditional Web applications via a Graphical User Interface (GUI). Poorly designed VUI frustrate users quickly, resulting in operator assistance or disconnected calls. It does not matter how powerful the application is or how many features it supports, if users cannot or will not use it. Therefore, a well-designed VUI is essential to the success of any voice application.

VoiceXML is designed for the rapid development of voice Web applications, but it does not address usability. A quality VoiceXML application requires a well-designed VUI. This paper discusses the VoiceXML application development lifecycle focusing on application usability. The paper also describes key roles and skills required in each phase of the development cycle. It looks at how the HP VoiceXML tools help developers simplify the development process and improve the usability of their VoiceXML applications before deploying to the HP OCMP VoiceXML platform http://h20208.www2.hp.com/opencall/products/media/ocmp/index.jsp.

This paper targets Java/J2EE developers who want to put a voice interface on their existing Web applications or develop new voice applications leveraging their Web skills. For VoiceXML background information, please see http://devresource.hp.com/drc/protected/ocmp/vxml/vxvoice_enable.jsp.

VoiceXML Application Challenges

VoiceXML applications require humans to converse with a computer. Designing a usable VUI is more difficult than designing a GUI. Challenges include:

• VUI is invisible. Information is transient; callers forget what they just heard.
• Conversation is linear. Callers get very impatient listening to long prompts or hearing messages played repeatedly.
• Caller is “lost in space.” Callers get disoriented when they get distracted or when there are many layers of spoken menus.
• Caller can say anything. Humans make mistakes. The big challenge is to cue the caller with clear wording so that the expected responses are received.

Callers hang up when they get confused, become impatient with long prompts, or think the application is not working.

Speech recognition is another VUI challenge. Speech recognition technologies have advanced dramatically, but speech recognition is still not perfect. Background noise and poor voice quality from poor telephone connections or low-quality microphones may cause the speech engine to misinterpret the caller inputs, leading the application to perform incorrect actions.
VoiceXML Application Development Life Cycle

The VoiceXML application development life cycle is similar to that of a Web application but includes VUI design and speech recognition tuning. The development cycle consists of six phases:

1. Definition
2. Design
3. Development
4. Testing
5. Pilot/Tuning
6. Deployment/Monitoring

VUI design and usability testing are critical for developing a successful VoiceXML application. Iterative usability tuning is required throughout the development cycle to build useful VoiceXML applications.

Key Roles and Responsibilities

Although VoiceXML is easy to learn, building a successful VoiceXML application requires not only software development skills, but other skills like understanding human factors for the telephone interface, linguistics, speech recognition and audio production.

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business managers</td>
<td>Includes marketing manager and project manager. Responsible for defining business requirements, allocating resources and budget, and managing the project to completion</td>
</tr>
<tr>
<td>Application architect</td>
<td>Design the application</td>
</tr>
<tr>
<td>VUI designer</td>
<td>Understand speech sciences and linguistics, design, test, and refine dialogs and grammars</td>
</tr>
<tr>
<td>Speech scientist &amp; phonetician</td>
<td>Understands computational linguistics and phonetics, tunes the ASR</td>
</tr>
</tbody>
</table>
### Definition Phase

#### Tasks: Analyze and Define...
- Business requirements
- Customer profiles
- Application requirements

#### Roles:
- Business manager
- Application architect
- VUI designers
- Callers

#### Tooling:
- Market analysis tools
- Project management and planning tools

During the definition phase, the business managers and application architect collaborate to address the following key issues:

- What is the purpose of developing the application (e.g. provide new service, solve customer problem, reduce operational cost, etc.)?
- What is the business case?
- What is the callers’ profile (age, gender, general background such as immigrant, education level, etc.)? What is their usage profile? How will they use the application and what are their expectations?
- What are the use cases, features, and required functionality?
- What is the project scope?
- What are the performance, capacity, and reliability requirements?
- What platform, VoiceXML browser, hosting portal, speech recognition and Text To Speech (TTS) engine, and document server are they going to use? What back-end systems (business logic) will the application integrate?

In addition, the VUI requirements have to be clearly defined based on the caller’s profile, the usage and the expected caller’s experience:
• What is the persona for the application (gender, age, personality)?
• How should the application be perceived (serious, casual, humorous)?
• What are the spoken languages in the deployment region?
• What are the typical environments (e.g. in an airport, car, quiet office) in which the application will be used?
• How does the application work in a real world situation and what are the user scenarios?

Design Phase

The design phase can be broken into two stages, prototype and detailed design.

Prototype
The application architect and the VUI designers decide on the application dialog style (i.e., directed dialog\(^1\), mixed initiative\(^2\), natural language\(^3\) dialog) and create a prototype of the application. The prototype describes user scenarios, the initial high-level call flow, prompts, and grammars. It is important to start the usability analysis during this phase. A technique called “Wizard of Oz testing” involves using a human to play the role of the computer. This person reads the application prompts and processes the caller’s inputs to test the dialogs and the call flow before coding starts.

Detailed Design
The application architect and the VUI designers make detailed design decisions based on the requirements specification and the prototype. The detailed design includes:

• The application architecture, required components, types of implementation, and the back-end systems that will be integrated. The application is composed of the voice front-end (VoiceXML page) and back-end server components. The architect needs to consider performance when determining

---

\(1\) The application prompts the caller by asking a question and then waits for the caller to respond

\(2\) Both application and the caller initiate and direct the conversation

\(3\) “Natural” human like interaction between the application and the caller
front-end functionality. This helps the application avoid unnecessary remote HTTP requests to the back-end server, which create conversation delays.

- The call flows and detailed elements associated with each dialog state: actions, error handling, prompt and grammar definitions, input mode (voice, DTMF (touch tone) or both), universal commands, help, synthesized speech, recorded audio, and the interfaces for accessing the back-end logic and databases.

A good VUI design must consider:

- Prompt: Prompt should be simple and clear to intuitively lead the caller to an expected (anticipated) outcome.

- Memory load: Studies show that, under normal circumstances, callers have a short-term memory of approximately six words. Ideally, the number of choices for callers to select should be four or less. Otherwise, callers become confused and forget the choices presented to them.

- Service reachability: It is not pleasant for a caller to go through a large number of steps before he reaches a service. Callers start to get impatient with more than five steps. Minimize the number of steps a caller must take to reduce frustration.

- Navigation: Provide a way to navigate back and forth between various dialog steps. The caller should be able to go to different parts of the dialog easily.

- Phonetic similarity: Provide a clear set of choices for caller to select. Avoid choices with similar pronunciations.

- Grammar collision: Don’t give the same choices at different locations in the conversation. If the same grammar element appears in more than one context but has a different function, callers become confused.

- Help: Callers need help messages to explain things that they do not understand.

- Error handling: Humans make mistakes. Graceful error handling decreases dependency on operators.

- Confirmation: Confirm the caller’s response to ensure that the machine got exactly what the caller said and not a different selection. This gives the caller a feeling of confidence.

- User update: Let the user know what is going on and keep him engaged. Back-end service actions may be slow.

- Timeouts: Specify reasonable timeout values to mange the flow of the dialog and keep the user engaged.

- Educate ahead: Train callers on how to use the application and what they can expect. For example, provide a tutorial option for the first-time caller. This tutorial might play sample dialogs describing how to interact with the application. Another option is to send a caller a Web page link with dialog samples when they sign up for a voice service.

**Tooling**

These tools are useful during the design phase:
1. SUEDE is a speech interface prototype tool developed at UC Berkeley. SUEDE offers an electronic version of the Wizard of Oz technique that captures test data, allowing designers to analyze the VUI. See http://guir.berkeley.edu/projects/suede/.

2. Microsoft® Office Visio®, with Drag and Drop symbols, helps with designing high-level call flow.

Development Phase

The development phase involves grammar development, audio production, software development, and back-end integration. The VUI designer, audio production team and software developer work on different components of the application.

Grammar Development
The grammar definition is a key part of the VUI and is tightly entwined with the dialog style and prompts. While creating the grammars, the dialog designers must collaborate very closely with the software developers and audio production team to iteratively test and refine the prompt wording and grammars. Well-tuned grammars can reduce many types of recognition errors.

Audio Production
The audio production team records the prompts and creates the audio files. The VUI designers often work with the voice talent to create the application persona, such as age (young or mature), personality (calm, peppy, romantic, trustworthy or authoritative), socioeconomic status, education level and etc.

Software Development
Based on the detailed call flow design, the software development team implements the VoiceXML application. They:
- Create the VoiceXML project and structure
- Develop code for complete call flow, such as output prompts (TTS and audio)
- Handle errors, help, and universal navigation commands
- Integrate with the back-end using jsp, servlets, cgi, and php
In addition, the team is also responsible for developing the server-side logic and integrating with other existing applications or databases. In general, the software development team consists of several sub-teams responsible for the front-end components, the back-end logic, and system integration.

Usability analysis and VUI refinement is very important during this phase to ensure that the implementation closely follows the call flow and detailed design.

**Tooling**
HP OpenCall provides VoiceXML development tools [http://www.hp.com/go/voicexml](http://www.hp.com/go/voicexml) for both Eclipse and BEA WebLogic Workshop. These tools accelerate the development of VoiceXML applications that run on the HP OCMP VoiceXML platform. In addition to assisting the developer in creating, editing, and validating static and dynamic JSP VoiceXML and grammar documents, the tools provide Dialog Analysis capability to help detect usability issues. The Dialog Analysis displays potential usability problems related to memory load, service reachability, grammar collision and phonetic similarity. The tool also displays statistics, broken links, warnings, and a metric summary of the VoiceXML document.

**Testing Phase**

All components should undergo independent unit testing. After the individual components have passed unit testing, the application logic, expected functionality, and dialog flow are tested. Testing can be performed in a simulated environment with Text to Speech (TTS) and Automatic Speech Recognition (ASR) support so developers can actually “dial up” the application.

Usability testing is required for all dialogs, actions, navigation, and help prompts associated with each dialog. The caller’s response patterns can indicate the trouble spots, for example, using words like “uh,” and “umm,” and pausing for a long time after a prompt. The usability engineers evaluate the human interaction experience, identify ambiguous and inconsistent direction, make sure services can be reached in a reasonable number of steps, and look for missing universal commands. The usability engineer’s primary objective is to make sure the application provides the required service.
**Tooling**

The HP OpenCall development tools tightly integrate with HP OCMP VoiceXML simulated test environment, HP OCMP-VoiceXML SDK. Developers deploy the application to the simulated test environment, and dial up the VoiceXML application using a simulated soft phone. The HP OCMP-VoiceXML SDK allows developers to use the logging technique `<log>` to log to Call Data Record (CDR) and test the application logic, the conversation flow, and the speech recognition. The event logs and CDR capture user-computer interaction and call control events, and contain detailed information for each call to the application. This includes HTTP requests, fetched documents, VoiceXML events, properties (i.e., timeout, barge-in, fetchtimeout), and waveform recognition results.

---

### Pilot/Tuning Phase

In this phase, the application runs on a real telephony network with a sample group of live callers. The goal is to capture real data, analyze application behavior, identify and tune problems in the grammars and speech recognition before the final deployment. Tuning may require only minor changes to prompts or grammars, or a modification or addition to the recognizer’s phonetic data, but it can be repetitive.

#### Dialog Tuning

Analyzing log file data helps to understand the application’s behavior:

- What are the task completion and dropout rates?
- Which state of the dialog flow causes the most dropouts?

Post-test interviews are a good way to capture usability feedback:

- Why did a caller abandon certain tasks or not use a specific feature such as universal commands, barge-in, or help?
- Were the prompts confusing?
- Did the caller think some options were unnecessary?

Refine the prompts, grammars, parameters, and other dialog elements, and test again!

**Speech Recognition Tuning**
Tune speech recognition accuracy and grammar, including:
- ASR parameters
- Acoustics models
- Confidence thresholds
- Phonetic dictionary (adding alternate pronunciations)

Successful tuning requires extensive knowledge speech recognition techniques and of the speech recognition engine being used. Speech scientists should be involved in the speech recognition tuning.

**Performance and Load Testing.**
Performance and load testing ensure that the system can handle the expected call volume within the expected response time.

**Tooling**
The HP OCMP-VoiceXML SDK and Dialog Verification tool allow developers and usability engineers to perform detailed analysis of the application behavior. This helps developers locate trouble spots for recognition errors. Once trouble spots are identified, the application can be “tuned.”

**Deploy/Monitoring Phase**

**Tasks:**
- Publish applications
- Continuous monitoring
- Periodic tuning

**Roles:**
- Application architect
- System administrator
- Callers

**Tooling:**
- HP OCMP vXML Toolkit for Eclipse
- HP OCMP VoiceXML Extension for BEA WebLogic Workshop
- HP OCMP-VoiceXML SDK

The final stage is to deploy the application. System administrators need to continuously monitor logged calls and reports, periodically tune and enhance the application to adapt to changing caller profiles and usages.
Tooling

The HP VoiceXML development tools provide the ability to package and publish the VoiceXML application to the document server. HP OCMP-VoiceXML SDK provides event logging, online reporting, and CDR to continuously monitor the application.

Summary

The most important element of a voice application is the VUI usability. Callers abandon applications quickly in response to a poor VUI. Usability must be addressed in all stages of the development lifecycle. In particular, callers’ involvement, iterative testing, and VUI tuning are critical when building quality VoiceXML applications.

Using development and testing tools reduces both the complexity and time required to develop a VoiceXML application with a good VUI.

The HP OCMP vXML Developer Toolkit and HP OpenCall VoiceXML Extension for BEA WebLogic Workshop provide integrated Java and VoiceXML development environments and help developers implement the front-end VoiceXML VUI and the back-end service components. The tools are specifically designed to address the common VUI issues, allowing developers to iteratively analyze and catch usability problems at development time. The OCMP-VoiceXML SDK and the Dialog Verification tool help in the testing and tuning of the call flow, dialogs, and speech recognition problems.

HP OpenCall provides developers with development tools and testing environment, which help improve the application usability, simplify the development process and reduce the development time for building VoiceXML applications that run on the HP OCMP VoiceXML platform.
For more information

Books

- Voice User Interface design by Michael H. Choen, James P. Giangola, Jennifer Balogh
- VoiceXML—An Introduction to Developing Speech Applications by James A. Larson
- Voice Enabling Applications: VoiceXML and Beyond by Kenneth R. Abbott
- VoiceXML: Professional Developer’s Guide by Chetan Sharma, Jeff Kunins
- Voice Application Development with VoiceXML by Rick Beasley, Kenneth Michael Farley, John O’Reilly, Leon Squire, Kenneth Farley

Technical Papers

- Developing voice-enabled Web services by OnOn Hong, HP
  http://devresource.hp.com/drc/protected/ocmp/vxml/vxvoice_enable.jsp
- VoiceXML: Changing the landscape of voice services
- A Taxonomy of Usability Factors in Telephony Voice User Interfaces by Sherif Yacoub, HP.
- Recognition of emotions in interactive voice response systems by Sherif Yacoub, HP
- Standardizing VoiceXML Generation Tools by David L. Thomson:
- Testing VoiceXML Applications by Bryan Michael and Mukund Bhagavan:
  http://www.voicexmlreview.org/Jul2001/features/testing_voicexml.html
- Choosing the Right Test Method by Peter Leppik
  http://www.voicexmlreview.org/May2003/features/May2003_test_method.html
- A Case for Improved Dialog Traversal (Call Flow) Testing and Analysis by Stuart Harding
  http://www.voicexmlreview.org/May2003/features/May2003_testing.html

VoiceXML Web Sites

- HP VoiceXML developer resources: http://www.hp.com/go/voicexml
- Building manageable apps for the adaptive enterprise: tools and technique
  http://www.presentationselect.com/hpinvent/detailsa.asp#485
- Voice Extensible Markup Language (VoiceXML) 2.0: http://www.w3.org/TR/voicexml20/
- VoiceXML Forum: http://www.voicexml.org
- Speech Technology Magazine: http://www.speechtechmag.com/

© 2004 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.