Advanced Dialog Management

Emerging standards for managing advanced dialog on server and client

K. W. (Bill) Scholz
NewSpeech, LLC
Agenda

Advanced Dialog – working group charter and history

Advanced Dialog Architecture

Advanced Dialog Managers

- Frame Based Engines
- Rule Based Engines
- Plan Based Reasoning
- Statistical Dialog Management
- Case Based Reasoning
Advanced Dialog WG: Charter

- Advanced Dialog Working Group objectives:
  - Accelerating transfer of Advanced Dialog concepts from research to industry
  - Encouraging incorporation of Advanced Dialog methods into commercial applications

- Method:
  - Identify the best advanced dialog methods available
  - Form a model for incorporating these methods into practical development systems
  - Incorporate this strategy into speech community standardization efforts
Advanced Dialog WG: Activities & Goals

- Weekly conference calls
- Workshop series (Feb 07, Aug 07, Mar 08)
- Goals
  - Provide input to the metalanguage working group.
  - Investigate the field of speech application design and the next step beyond state-based methods.
  - Specify architecture for a complete voice application, including server-side logic and backend interactions.
  - Facilitate writing re-usable voice applications using common VoiceXML patterns.
  - Produce a specification or model for how to deal with AD that is defined with sufficient detail to incorporate into the metalanguage specification.
Evolution of Advanced Dialog

- VoiceXML enhancements to address growing complexity
  - Adding ‘guards’ (‘if’ condition for many VXML tags) – approximates a rule-based system
  - Add global variables to state-based dialog
  - Allow state transitions to be stochastic

- What makes a dialog advanced?
  - A dialog is not advanced if you can count the number of states.
Speech Application Architecture

Client

- Speech Recognition
- Speech Synthesis
- Pre-recorded Audio
- Standards Compliant

VoiceXML Interpreter

- Client-side Dialog Management
- Cache

App Server

- Dialog Manager
- Server-side Dialog Management
- Few Standards

Back-end System

- DB
- IDL

Standards

- SSML
- SISR
- SRGS
- VXML, SRGS, Javascript

Cache
Dialog Management: Client-side vs. Server-side

Dialog management done by VoiceXML interpreter, directed by mostly-static VXML data delivered by server, maintained on back-end DB.

Dialog management done engine running on server, consulting back-end database for essential data, dynamically generating VXML.

Dialog management shared between engine on server and scripting on client. ‘AJAX’
Advanced Dialog – Server-side

Dialog Manager may be active (driven by user action and user data), or passive (traffic cop: IDL to VXML).
Active app. server dialog management

App Server

Dialog Manager Engine

TAG Interpreter

SCXML Interpreter

DML Interpreter

Back-end Database Server and Transaction Processor

Service Creation Tools

Dialog Meta-Language

VXML Client

HTTP

VXML, Javascript

VXML

Ex.Tags

SCXML

DB, IDL

DB, DML

TAG Interpreter

SCXML Interpreter

DML Interpreter

VXML, Javascript

VXML

Ex.Tags

SCXML
Passive app. server dialog management

App Server

VDML/IDL Interpreter,
VXML Tag Generator

VXML Client

VXML, Javascript

HTTP

‘Back-end’ System or DM Server

Inference Engine,
Rules,
Plans
Frames,
Cases,
Probabilities

Transaction Processor

Database Processor

Dialog Manager

Service Creation Tools

Dialog Meta-Language

IDL/DML Interpreter,
VXML Tag Generator

VXML

Client

App Server

HTTP

VDML, IDL

Dialog Manager
Relation between DML and AD

- SCXML and VoiceXML specify explicit dialog states, so DML may not be appropriate for specifying advanced dialogs directly. **But ...**
- DML can pass submitted data to an external dialog manager, then pass prompts and grammars from the dialog manager back to a VoiceXML browser.
- Functionality required by AD that is **intrinsically** unsupported by VoiceXML (e.g. statistical language models) will need to be added to VoiceXML to be supported through DML.

Ian Sutherland
Dialog Management

Inference engines - candidates

- Frame-based
- Rule-based
- Plan-based
- Stochastic
- Case-based reasoning
Frame-Based Systems

- A frame is a traditional way of organizing related information as a set of (possibly nested) slots and values.
- Example: A sandwich
  - Bread: whole wheat, white, rye, kaiser roll, bagel ...
    - If bagel: type could be ‘plain’, ‘poppy seed’, ‘egg’...
    - Toasted: yes/no
  - Filling: chicken salad, roast beef, pastrami...
  - Extras: tomato, lettuce, onion...
- In VoiceXML: frames and slots are familiar as forms and the fields within a form.
- Convenient to map dialog states to frames
Rule-Based Systems

Forward Chaining

1. Rule Base
2. Working Memory
3. Rule Fit?
   - If No, Next rule
   - If Yes, Resolve Conflicts
4. Exit

Example: Medical Diagnosis

Backward Chaining

1. Goal
2. Rule Base
3. Goals Match?
   - If Yes, recursively backchain
   - If No, Resolve Conflicts
4. Working Memory
5. Exit

Example: Medical Epidemic

If <condition> then <action>
Plan-Based Dialog Systems

Observation:
• humans don’t produce communicative actions randomly
• actions are planned to achieve various goals
• dialog acts are part of a plan
• listener’s job to uncover the plan and react accordingly

Example: “It is cold.”
Plan-recognizer should determine:
 a) information
 b) request to close the door, or switch on heat
 (depending on context)
Stochastic Dialog System

Objectives
Failure: -100
Success: +100
Each turn: -1

Optimization process

Notes:
- Designers provide the “palette” of actions and provide high-level goals
- The optimization works out the details
- Designer can also incorporate constraints or start optimization with an existing design

Statistical approach
Dialog state
- Power-state
- Network-on
- Username-ok
- Password-ok
- Can-surf-ok

Dialog state is a distribution over possible states of the world - not one guess

Dialog manager
Designer provides high-level goals for the dialog manager; details worked out by optimization process
Case-Based Reasoning

Definition...

- A CBR system uses **Case Libraries**. Each case:
  - Describes a **problem** and its **symptoms**
  - Provides a **solution** to the problem (often a **script**)

- Evidence obtained (from the caller, from tests, from data dips, etc.) matches symptoms and can activate each case at differing levels.

- Current activation levels for all cases create patterns that activate **rules** that drive the dialog by suggesting the next prompt, further tests or additional data dips.

- Through this interaction, eventually one case’s activation level crosses a threshold and its solution is triggered.

- CBR systems are capable of adaptation over time by adding new cases, new rules and by adjusting the weights for evidence.
Knowledge Base Development

Knowledge engineers can develop and improve the knowledge base (the library of cases, rules and solution scripts) based on the experience of agents.

**Tier 2 Agents**
- Build and improve the case library
  - Knowledgeable and experienced agents

**Tier 1 Agents**
- Use the case library to facilitate dialog
  - Less experienced or offshore agents
- Use the case library to drive dialog during self-service automation

John Tadlock
Summary and Conclusions

- AD Working Group – Charter and Mission
- AD Architecture
- AD Engine candidates
  - Frame Based
  - Rule Based
  - Plan Based
  - Stochastic
  - Case Based Reasoning

Diagram:
- VXML Client
- DML & TAG Interpreter
- App Server
- Inference Engine CBR
- Service Creation Tools
- Dialog Server
Thank You

NewSpeech, LLC.
Advanced Speech Solutions
Karl W.'Bill' Scholz, Ph.D.
President
203 Vassar Circle
Stratford, PA
19087

cell: 610-212-8016
cell: 610-989-0989
bill.scholz@comcast.net
http://www.newspeechsolutions.com/