# An Introduction to VoiceXML

**ART on Dialogue Models and Dialogue Systems**

**François Mairesse**
**University of Sheffield**
F.Mairesse@sheffield.ac.uk
http://www.dcs.shef.ac.uk/~francois

## Outline

- What is it?
- Why is it useful?
- How does it work?
- How to make it better?

# Brief history

- 1999: AT&T, IBM, Lucent Technology and Motorola formed the VoiceXML Forum
  - The goal was to for make Internet content available by phone and voice
  - Each company had previously developed its own markup language
  - Customers were reluctant to invest in proprietary technology
- 2000: release of VoiceXML 1.0
- 2005: VoiceXML 2.1 is a W3C candidate recommendation



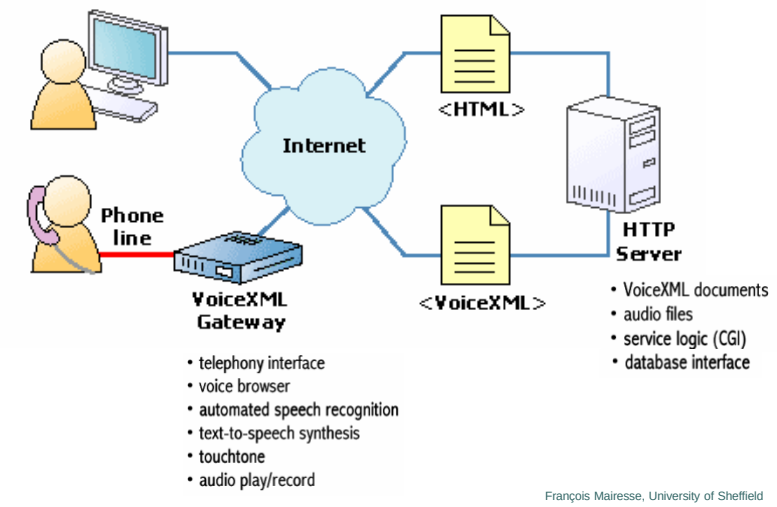
# What is VoiceXML?

- VoiceXML is a mark-up language for specifying interactive voice dialogues between a human and a computer
- Analogous to HTML
  - VoiceXML browser interprets .vxml pages
  - Can be dynamically generated by server-side scripts (JSP, ASP, CGI, Perl)
    - Can access external databases (e.g. SQL)
- Example

```
<?xml version="1.0"?>
<vxml version="2.0">
  <form>
    <prompt>
        Hello world!
    </prompt>
  </form>
</vxml>
```

- VoiceXML platform

# Architecture







# Voice User Interface (VUI)

Name:
Telephone:
Submit Clear

- Traditional web-based forms
- The purpose of a dialogue is to fill forms
- GUI vs. VUI
  - Fonts vs. prosody
  - Large menus vs. short utterances
  - Hypertext navigation vs. voice commands
  - Constraint on forms vs. recognition grammars
  - Global options always visible
    vs. only uttered at the beginning of the dialogue

# Why use VoiceXML?

Advantages of VoiceXML platforms

- Special-purpose programming languages
  - Reduces development costs
- Separation between dialogue system components
  - Portability of application
  - Flexibility: outsource or purchase equipment
  - Choose best-of-breed components
- Re-use of Internet infrastructure
- VoiceXML is becoming a standard



# The VoiceXML language

- XML structure

```
< element_name attribute_name="attribute_value">
    ......contained items......
< /element_name>
```

- Basic elements
  - *prompt*: specifies the system's utterance
  - *audio*: play pre-recorded prompts
  - *form*: set of fields
  - *field*: information needed to complete task
  - *grammar*: specifies possible inputs to a field

# Basic elements

- *filled*: what to do if user input is recognized
- *value*: return a field's value
- *goto*: go to another form or file
- *submit*: go to another file and keep field values

- Error handling
  - user says nothing: *noinput*
  - nothing matches the grammar: *nomatch*

- Many more elements: http://www.vxml.org




# VoiceXML document

```
<?xml version="1.0"?>
<vxml version="2.0">
<form id= "get_student_name">
  <field name= "student_name">
    <prompt> What's your name? </prompt>
    <grammar> john | mary | rob </grammar>
    <noinput> Please say your name. </noinput>
    <nomatch> I didn't understand that. </nomatch>
    <filled>
      Thank you, <value expr= "student_name" />
      <submit next="next_document.vxml" />
    </filled>
  </field>
</form>
</vxml>
```

What do we want to know? → `<field name= "student_name">`
Question → `<prompt> What's your name? </prompt>`
Possible answers → `<grammar> john | mary | rob </grammar>`
No answer? → `<noinput> Please say your name. </noinput>`
Wrong answer? → `<nomatch> I didn't understand that. </nomatch>`
Acceptable answer → `Thank you, <value expr= "student_name" />`
What's next? → `<submit next="next_document.vxml" />`

# Recognition grammars

- Key to successful recognition
- Many platform-dependent formats (JSGF, SGL, etc.)
- Inline grammar
- External file
  <grammar src="mygram.gram" type="application/x-jsgf" />

- Example with optional inputs (in brackets)

```
#JSGF V1.0;
grammar pizza;
public <pizza> = [I'd like a] <size> <type> [pizza] [please];
<size> = small | medium | large;
<type> = vegetarian | pepperoni | cheese;
```



# Built-in grammars

- Boolean
- Currency
- Date
- Digits
- Number
- Phone
- Time

- Example: <field name="get_digits" type="digits">

- Can add additional constraints
  <field name="get_digits"
  type="digits?minlength=3;maxlength=9">

# Events

- Similar to exceptions
- Thrown by
  - Platform: ASR misrecognition
  - Application: <throw>
- Handler
  - Specific: <noinput>, <nomatch>, <help>
  - General: <catch event=…>
  - Can count number of event occurrences
    - Successive ASR errors with different repairs
      <nomatch count=3> What did you say ? </nomatch>



# VoiceXML properties

- Can be modified using the <property> element
  - Confidence level of ASR
  - Barge-in
  - Time-out
  - Voice/DTMF
- Properties can be defined at all levels: for the whole application, document, or a specific field

# Mixed-initiative dialogues

- VoiceXML allows for simple mixed-initiative
  - More flexible
  - More room for errors
- A form-level grammar that can recognize multiple fields at once
  - E.g. “Please tell me a departure day and a destination”
  - Grammar needs to account for all possible orderings
    - “I’m going to DEST on DATE”
    - “I’m leaving on DATE to go to DEST”
  - What if we don’t have all required information at once?
    - Back to directed dialogue
    - Need traditional fields
  - How to know what fields remain unfilled?




# Form Interpretation Algorithm

- Defines how control flows through a VoiceXML application as it executes
- Makes VoiceXML declarative
  - Just specify utterances, fields and grammars
  - Define what happens, not how
  - FIA deals with procedural details
    - Keeps querying undefined fields
    - Throw events and loop until field is filled by user
      - → <nomatch> or <noinput>
      - → <filled>

# FIA - confirmations

- If a field value isn't confirmed by user, set it to undefined and the FIA will ask for it again

```
<field name="confirm" type="boolean">
 <prompt> Do you want details on <value expr="student_name" />?
 </prompt>
 <filled>
   <if cond="confirm">
      Looking up details on <value expr="student_name" />
   <else />
      Let's try again
      <clear namelist="student_name confirm" />
   </if>
 </filled>
 </field>
```




# Limitations

- Simple mixed initiative
- How to retrieve information from a database?
- What about more advanced dialogue system features?
  - Content summarization
    - Multiple database entries
    - Find alternatives answers
  - Dynamic grammars
    - If the database changes, the recognition grammar must adapt
  - Generate VoiceXML pages dynamically

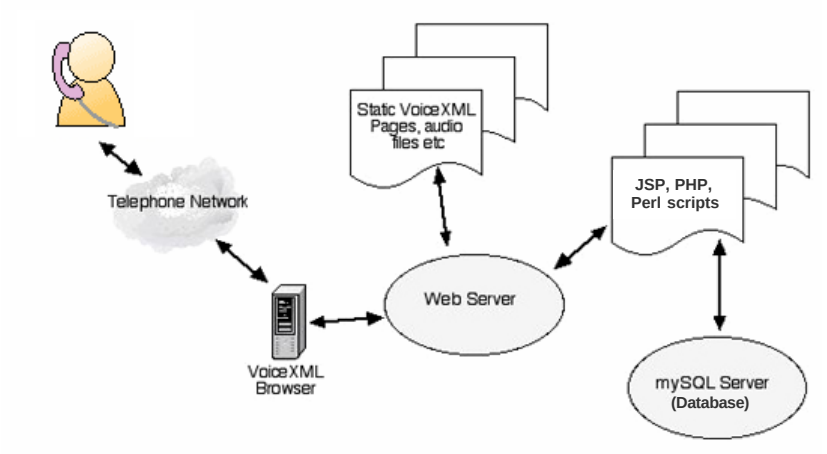
# Dynamic VoiceXML

- Similar to dynamic HTML pages
- Content isn't stored on the server, but created on-the-fly based on the user's parameters and a database

- Typical interaction:
  - A static VXML page collects information from the user
  - Submit the fields to a server-side script (JSP, PHP, ASP, Perl, etc.)
  - The script queries the database and processes the results
  - The script outputs VXML code which is interpreted by the browser



# Dynamic VoiceXML



# Implementation in Perl

- When form is filled, send fields value to the server-side script for processing

```
<filled> Thank you
<submit next="http://mywebserver/script.perl">
</filled>
```

- The Perl script collects information

```
$q = new CGI;
$name = $q->param('student_name');
```

- Connect to the SQL database

```
$handler = DBI->connect("DBI:mysql:$db", $user, $password);
```

- Query the database for the student's name

```
$query = $handler->prepare("SELECT * FROM students
                            WHERE name = \"$name\"" );
$query->execute;
```

- Output beginning of VoiceXML document (<xml>, <voicexml>, <form>, <prompt>)
- Output result, i.e. the student's phone number

```
@row = $sth->fetchrow_array;
print "The phone number of $name is $row[2]\n";
```

- Output end of VoiceXML document (</form>, </voicexml>, etc.)




# Dynamic grammars

- What to do when the recognition vocabulary is not known in advance?
- Rewrite a grammar at each database update
- Better, use a server-side script to
  - Retrieve patterns from database
  - Write grammar to an external file
  - Call a VXML page using this grammar

# Conclusion

- VoiceXML has become a standard
  - All-in-one solutions available
  - Reduces dialogue system development time
  - Comes with limited dialogue management and language generation capabilities
  - Additional functions can be easily implemented
  - Develop your own dialogue system with free VoiceXML browsers!

    https://studio.tellme.com