VoiceXML Tutorial

Felix Burkhardt, (fxburk@gmail.com),
written around 2002

VoiceXML Tutorial Contents

- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML: Concepts, Code Examples
- Voiceportal Architecture with VoiceXML and J2EE
- Application Development Tools (Example Speech Navigator)

VoiceXML Preliminary Outline

- Motivation
- Basic Technology
  - ASR
  - TTS

VoiceXML Preliminary Motivation: Why Phone?

- The web is spreading beyond the PC
- Phones are everywhere
  - Far more phones than PCs
  - More wireless phones than PCs, esp. in Europe
- Wireless phones are portable
  - Support location based services
- Hands-free, Eyes-free
- Instant-on

VoiceXML Preliminary Motivation Why Voice vs. Visual IO?

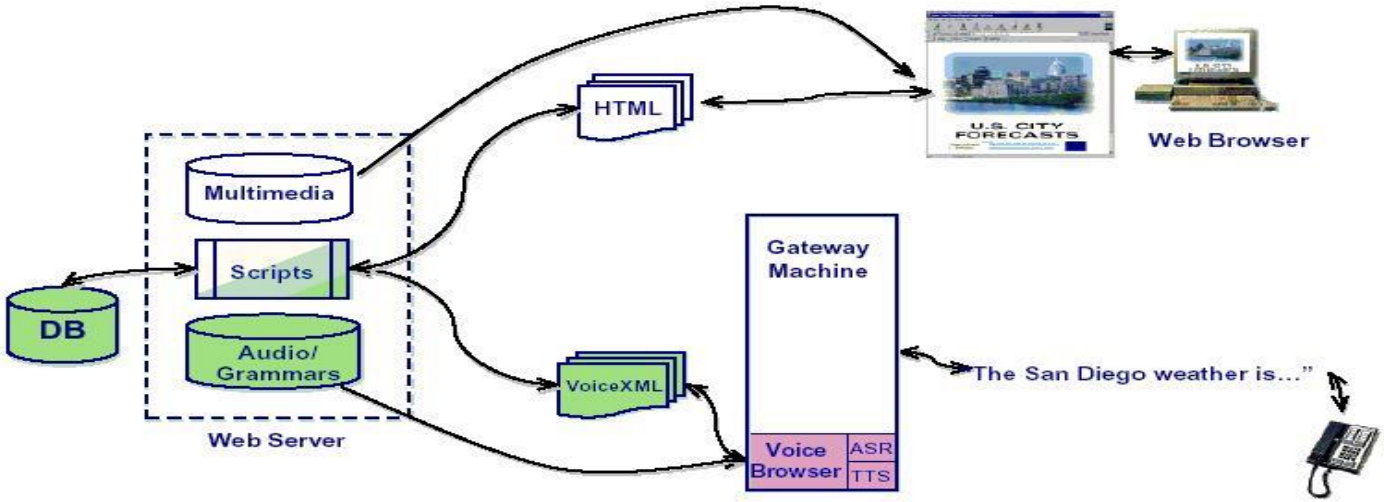
- Visual mobile IO (e.g. WAP) better for reading information, but
  - Not available on legacy phones
  - Small screens can be restrictive
  - Input difficult on small keypads
  - Not „eyes/hands-free“
- Next step: multimodal applications
- Use the right interface for the task!

VoiceXML Preliminary
Motivation Why VoiceXML?

- VoiceXML integrates Voiceportals into the web-paradigm
- Makes pervasive computing a reality
- High-level, domain specific language (vs. C++) simplifies programming
- Usage of standard language enables choosing „best of breed“ components
- Clean separation of service logic from user interaction (MVC)
- Leverage existing web-application development tools
- Many developers with VoiceXML-knowledge available

VoiceXML Preliminary

Motivation: Multi-Access Internet



www analogy

Voice	Web
VoiceXML	HTML
Voice Browser	Web Browser
Phone	PC

Source: voicexml_vbwc_tech_overview__18sep00_ferrans_.pdf

VoiceXML Preliminary

Motivation: What is VoiceXML?

- A language for specifying voice dialogs.

  Voice dialogs use audio prompts and text-to-speech (TTS) for output; touch-tone keys (DTMF) and automatic speech recognition (ASR) for input.

- Main input/output device (initially) is the phone.
- Standard language enables portability.
- High-level domain-specific language simplifies application development.

VoiceXML Preliminary Motivation: Aims

- Create a Language for voice applications that‘s based on Web-standards
- Responsiveness:
  - Allow specification of multiple interactions per page to reduce client/server network and processing delays.
  - State machine-based dialogs.
- Ease of use
  - Make common dialogs easy to write.
  - Enable complex dialogs.
- Error/event handling
  - Unexpected user response (inappropriate response, ‘help’ etc.)
  - ASR misrecognitions.
  - Network anomalies.
  - Hardware/software platform errors

VoiceXML Preliminary

Motivation: Advantages (vs. proprietary)

- Separate Development from Deployment
- Share content with HTML
- Build a „Voice Web“
- Enable a market of graded suppliers:
  - Core-tech
  - Platform
  - Development Tools
  - ASP
  - Application supplier
- Other advantages of standards, e.g. huge offering of third-party solutions, developers and knowledge

VoiceXML Tutorial Contents

- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML Code Examples
- Architecture VoiceXML/J2EE
- Application Building (Example Speech Navigator)

VoiceXML: Speech Interface Framework Outline

- Introduction
- Grammar: SRGML
- Speech Synthesis: SSML
- Call Control: CCXML

VoiceXML: Speech Interface Framework Introduction

- With the W3C Voice Browser Working Group and the development of VoiceXML 2.0, the standard became part of the Speech Interface Framework
- Suite of sub standards :
  - VoiceXML
  - SRGML Speech Recognition Grammar ML
  - Semantic Interpretation
  - SSML Speech Synthesis ML
  - CCXML Call Control
- Lower priority
  - Pronunciation lexicon
  - Stochastic grammars
  - Voice browser interoperation

VoiceXML: Speech Interface Framework

SRGML Speech Recognition Grammar Introduction

- Version 1.0 Candidate Recommendation
- Grammar format possible in two forms (semantically mappable)
  - XML-Form
  - ABNF-Form
- Scope: tell the ASR what it should listen for
  - Words
  - Patterns in which the words my be spoken
  - Language of the word
- May support N-gram (Stochastic Language Model)

VoiceXML: Speech Interface Framework
SRGML Speech Recognition Grammar Syntax 1

- A Grammar consists of a set of rules, that may be references

```
<ruleref  uri="http://grammar.example.com/cities.gram#argentina" xml:lang="es"/>
```

- Example: Philadelphia in the great state of Pennsylvania

```
<rule id="location">
  <ruleref uri="#city"/>
  <ruleref special="GARBAGE"/>
  <ruleref uri="#state"/>
</rule>
```

VoiceXML: Speech Interface Framework
SRGML Speech Recognition Grammar Syntax 2

- A Grammar may reference to an external pronunciation lexicon

```
<grammar>
      <lexicon uri="http://www.example.com/lexicon.file"/>
</grammar>
```

- A Grammar can have a DTMF mode

```
<grammar mode="dtmf" >
<rule id="digit">
 <one-of>
  <item> 0 </item>   <item> 1 </item>
</one-of>
</rule>
```

VoiceXML: Speech Interface Framework
SRGML Speech Recognition Grammar Syntax 3

- Rules can be sequenced:

the <ruleref uri="#object"/> is <ruleref uri="#color"/>

- Rule-Items can be weighted alternatives:

```
<one-of>
  <item weight=„.5">small</item>
  <item weight="2">medium</item>
  <item>large</item>
</one-of>
```

- Rule-Items can be repeated, given a probability:

<item repeat="0-1" repeat-prob="0.6">very</item>

VoiceXML: Speech Interface Framework
SRGML Speech Recognition Grammar Syntax 4

- Rule-Items can be language annotated

```
<rule id="yes">
  <one-of>
    <item>yes</item>
    <item xml:lang="fr-CA">oui</item>
  </one-of>
```

- A Rule can have a scope:

```
<rule id="city" scope="private">
 <rule id="command" scope="public">
```

VoiceXML: Speech Interface Framework

SSML Speech Synthesis: Introduction

- Working Draft available, Candidate Recommendation 2.Q 03
- The TTS-Engine transforms a textual input document into a waveform
- A SSML-conform TTS-Engine processes 6 steps:
  1. XML-parsing
  2. Structure analysis
  3. Text normalisation
  4. Text-to-phoneme conversion
  5. Prosody analysis
  6. Waveform production

VoiceXML: Speech Interface Framework

SSML Speech Synthesis Document Structure

- Speak : root element
- Xml:lang Language Attribute
- „paragraph“ and „sentence“

```
<speak version="1.0" xml:lang="en-US">
  <paragraph>I don't speak Japanese.</paragraph>
  <paragraph xml:lang="ja">Nihongo-ga wakarimasen.</paragraph>
</speak>
```

VoiceXML: Speech Interface Framework SSML Speech Synthesis Textprocessing

- „say-as“ element

<say-as type="acronym"> DEC </say-as>

Pope John the <say-as type="number:ordinal"> VI </say-as>

Proposals are due in <say-as type="date:my"> 5/2001 </say-as>

<say-as type="net:email"> road.runner@acme.com </say-as>

VoiceXML: Speech Interface Framework SSML Speech Synthesis Pronunciation

- „phoneme“ element

```
<phoneme alphabet="ipa" ph="t&#x252;m&#x251;to&#x28A;">
  tomato
</phoneme>
```

- „sub“ element

```
<sub alias="World Wide Web Consortium"> W3C </sub>
```

VoiceXML: Speech Interface Framework SSML Speech Synthesis Style

- „voice“ element (attributes: gender, age variant, name)

<voice gender="female" variant="2">It's fleece was white as snow. </voice>

- „emphasis“ element

That is a <emphasis level="strong"> huge </emphasis> bank account!

- „break“ element

Press 1 or wait for the tone. <break time="3s"/>

I didn't hear you!

VoiceXML: Speech Interface Framework
SSML Speech Synthesis Prosody

- „prosody“ element. Attributes:
  - **pitch** („high“, „medium“, „low“, „default“)
  - **range** (as Pitch)
  - **contour** (set of target frequency values)
  - **rate** („fast“, „medium“, „slow“, „default“)
  - **duration** (time value)
  - **volume** („silent“, „soft“, „medium“, „loud“, „default“)

```
<prosody contour="(0%,250Hz)(10%,+30%)(40%,+10)">
  good morning
```

VoiceXML: Speech Interface Framework
SSML Speech Synthesis Other Elements

- „audio“ element (text is synthesized if audio not available)

<audio src="prompt.au">What city do you want to fly from?</audio>

- „mark“ element (used for referencing)

We would like

<mark name="congrats">to extend our warmest congratulations</mark>

to the members of the Voice Browser Working Group

VoiceXML: Speech Interface Framework
CCXML Call Control Introduction

- CCXML is a XML-based language to control the setup, monitoring and tear-down of phone calls
- Working draft available, Version 1.0
- Some companies developed own languages (Voxeo‘s CallXML or Telera‘s TXML)

VoiceXML: Speech Interface Framework
CCXML Call Control

Why not integrate Call Control in VoiceXML?

- Needed because VoiceXML was never designed to include call-control but to specify voice-dialogs
- Some features included (e.g. „transfer“-tag), but too basic for many applications
- Difficult to integrate Call Control into VoiceXML because it‘s linear directed and not event based
- Companies wanted language usable outside VoiceXML

VoiceXML: Speech Interface Framework
CCXML Call Control Features

- Multi-party conferencing
- Richer event-handling
- Give each active line it‘s own VoiceXML interpreter
- Receive events from systems outside the platform
- Sophisticated multiple-call handling and control

VoiceXML: Speech Interface Framework

CCXML Call Control Example

```
<ccxml version="1.0">
 <eventhandler>
 <!– Lets handle the incoming call –>
  <transition event="connection.CONNECTION_ALERTING" name="evt">
   <log expr="'The called ID is ' + evt.calledid + '.'"/>
   <if cond="evt.callerid == '8315551234'">
    <accept/>
   <else/>
    <reject/>
   </if>
  </transition>
```

VoiceXML: Speech Interface Framework

CCXML Call Control Example continued

```
<transition event="connection.CONNECTION_CONNECTED">
   <log expr="'Call was answered. We are going to start a dialog.'"/>
    <dialogstart src="'hello.vxml'"/>
   </transition>

   <transition event="dialog.exit">
   <log expr="'We got a dialog.exit event.'"/>
    <disconnect/>
   </transition>
</eventhandler>
</ccxml>
```

VoiceXML Tutorial Contents

- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML Code Examples
- Architecture VoiceXML/J2EE
- Application Building (Example Speech Navigator)

VoiceXML Outline

- History
- What‘s new in VoiceXML 2.0?
- Key Concepts
- Code Examples

VoiceXML History

- Initially developed by VoiceXML Forum (started by AT&T, Lucent, Motorola, IBM) as consolidation from various ancestors
- March 2000 VoiceXML 1.0
- Then part of W3C Voice Browser Working Group
- Dec. 2001 first Draft of VoiceXML 2.0
- Jan. 2003 VoiceXML 2.0 is W3C Candidate Recommendation
- Future: V3; integration of multimodality

VoiceXML

Introduction VoiceXML Documents describe

- spoken prompts (synthetic speech)
- output of audio files and streams
- recognition of spoken words and phrases
- recognition of touch tone (DTMF) key presses
- recording of spoken input
- control of dialog flow
- telephony control (call transfer and hangup)

VoiceXML Introduction

What‘s new in VoiceXML 2.0?

- Logging: new „log“-tag
- Part of Speech Interaction Framework, special tags replaced by
  - Grammar Specification ML
  - Speech Synthesis ML
  - Call Control ML
- Finer grained control over caching
- Internationalisation

VoiceXML Introduction

What‘s new in VoiceXML 2.0?

- Speech Recognition Enhancement
  - N-Best
  - Weighted grammars
  - Application wide $lastresult
  - Different types of barge-in detection (type „speech“ or „hotword“)
- Telephony Changes
  - Transfer includes transferaudio-attribute
- Event Features
  - Passing a message

VoiceXML Tutorial

Key Concepts 1

- **Session** begins when user starts to interact and ends when requested by user, VoiceXML document or interpreter context
- VoiceXML describes state-machine, user always in defined **dialog-state**
- Dialogs include **forms** and **menus**, **fields** gather input
- **Application** is a set of documents that share the same **root-document**
- Dialog-state has a set of **grammars** associated that specify expected user input

VoiceXML Tutorial

Key Concepts 2

- Execution order determined by **FIA** (Form Interpretation Algorithm)
- **Subdialogs** are like subroutines, can be used to develop reusable dialog-fragments
- Named **variables** can be used for holding data (applicationwide through root-document)
- **Events** can be defined, thrown and caught
- **ECMA-Script** (like Javascript) is integrated

VoiceXML
Examples: Hello World

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0" lang="en">
        <form>
                <block>
                        <prompt bargein="false">
                                Hello World!
                        </prompt>
                </block>
        </form>
</vxml>
```

(examples inspired by Dave Raggett's VoiceXML tutorial at http://www.w3.org/Voice/Guide/)

VoiceXML Examples Menu

```
<vxml version="2.0">
    <menu>
        <prompt> Say one of: <enumerate/> </prompt>
        <choice next="http://www.weather.example/intro.vxml">
            Weather
        </choice>
        <choice next="http://www.news.example/news.vxml">
            News
        </choice>
        <noinput>Please say one of: <enumerate/> </noinput>
    </menu>
</vxml>
```

VoiceXML Examples Field

```
<form>
    <field name="city">
        <prompt>Where do you want to travel to?</prompt>
        <option>Edinburgh</option>
        <option>Stockholm</option>
    </field>
    <field name="travellers" type="number">
        <prompt>How many are travelling to <value expr="city"/>?</prompt>
    </field>
    <block>
        <submit next="http://localhost/handler" namelist="city travellers"/>
    </block>
</form>
```

VoiceXML Examples Count

```
<field name="travellers" type="number">
    <prompt count="1">
        How many are travelling to <value expr="city"/>?
    </prompt>
    <prompt count="2">
        Please tell me the number of people travelling.
    </prompt>
    <prompt count="3">
        To book a flight, you must tell me the number
        of people travelling to <value expr="city"/>.
    </prompt>
    <nomatch>
        <reprompt/>
    </nomatch>
```

VoiceXML Examples Value checking

```
<field name="travellers" type="number">
    <prompt>How many are travelling to <value expr="city"/>?</prompt>
    <filled>
        <var name="num_travellers" expr="travellers + 0"/>
        <if cond="num_travellers > 12">
            <prompt>
                Sorry, we only handle groups of up to 12 people.
            </prompt>
            <clear namelist="travellers"/>
        </if>
    </filled>
</field>
```

VoiceXML Examples Subdialog definition

```
<form id="ynconfirm">
    <var name="user_input"/>
    <field name="yn" type="boolean">
        <prompt>Did you say <value expr="user_input"/></prompt>
        <filled>
            <var name="result" expr="'false'"/>
            <if cond="yn">
                <assign name="result" expr="'true'"/>
            </if>
            <return namelist="result"/>
        </filled>
    </field>
</form>
```

VoiceXML Examples Subdialog usage

```
<field name="city">
        <prompt>Which city?</prompt>
        <filled>
                        <if cond="city$.confidence < 0. 7">
        <assign name="utterance" expr="city$.utterance"/><goto nextitem="confirmcity"/>
                        </if>
        </filled>
</field>
<subdialog name="confirmcity" src="#ynconfirm" cond="false">
        <param name="user_input" expr="utterance"/>
        <filled>
                        <if cond="confirmcity.result=='false'"> <clear namelist="city"/></if>
        </filled>
</subdialog>
```

VoiceXML Examples Mixed iniative

```
<form name="trader">
        <grammar src="trade.xml#command" type="application/grammar+xml"/>
        <initial name="start">
                <prompt>What trade do you want to make?</prompt>
                <nomatch count="2">
                        Sorry, I didn't understand your request.
                        Let's try something simpler.
                        <assign name="start" expr="true"/>
                </nomatch>
        </initial>
        <field name="company"> ... </field>
        <field name="action"> ... </field>
</form>
```

VoiceXML

Examples App-root Document

```
<form name="portal-commands" scope="document">
        <field name="action">
                <grammar src="http://buster/portal/commands.xml"
                        type="application/grammar+xml"/>
        </field>
        <block>
                <submit next="http://www.wl.com"/>
        </block>
</form>
<var name="portal-help" expr="To return to your portal home, say 'home page', or press 0."/>
<catch event="noinput">
Sorry, I didn't hear anything.
</catch>
```

VoiceXML Tutorial Contents

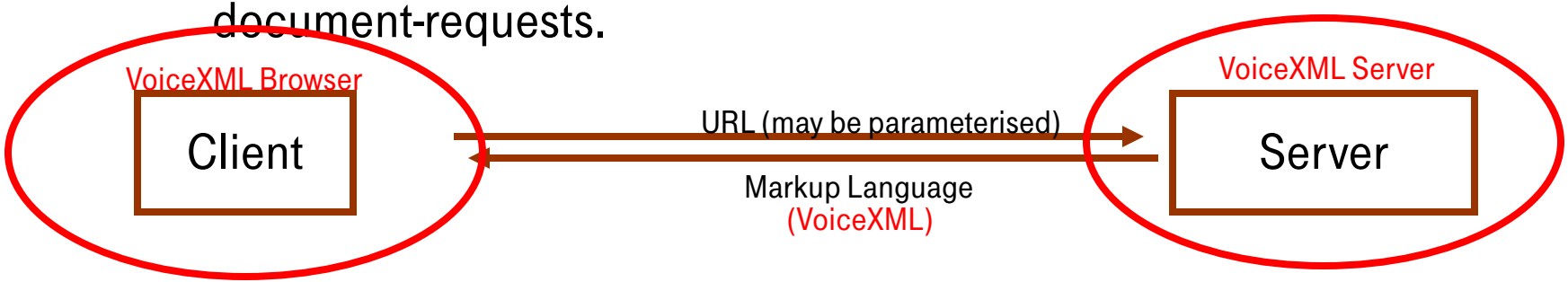
- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML Code Examples
- Architecture VoiceXML/J2EE
- Application Building (Example Speech Navigator)

Voice Portal with J2EE/VoiceXML Outline

- Key Concepts
- Architecture Voice Portal
- Database Connection

Voice Portal with J2EE/VoiceXML
Key Concepts 1

- The HTTP (Hypertext Transfer)-Protocol is a Communication-protocol. A Client asks a Server and gets a response
- J2EE (Java 2 Enterprise Edition) is a Java add-on, that introduces Servlet-, JSP and Bean technology
- This implies that a J2EE application is distributed by several document-requests.



Voice Portal with J2EE/VoiceXML
Key Concepts 2

- **Servlet**: a special Java-Class, that provides functionality for the HTTP-Protocol
- **JSP** (Java Server Page): a Document, that consists of a mixture of formatted content (e.g. VoiceXML, executed by the voice browser) and Java-Code (executed by the application server)
- **Bean:** A special Java-class with standardized interface (Getter- und Setter-methods).
- **Enterprise Bean**: denotes a Bean that runs on different server than the main application. Enables distributed applications.
- **Entity Bean:** A special Bean, that stores data persistent (longer than application). Encapsulates Database-Functionality
- **SOAP** (Simple Object Access Protocol): W3C-Proposal for a XML-Protocol for the communication in distributed Systems (Part of WebServices)

Voice Portal with J2EE/VoiceXML

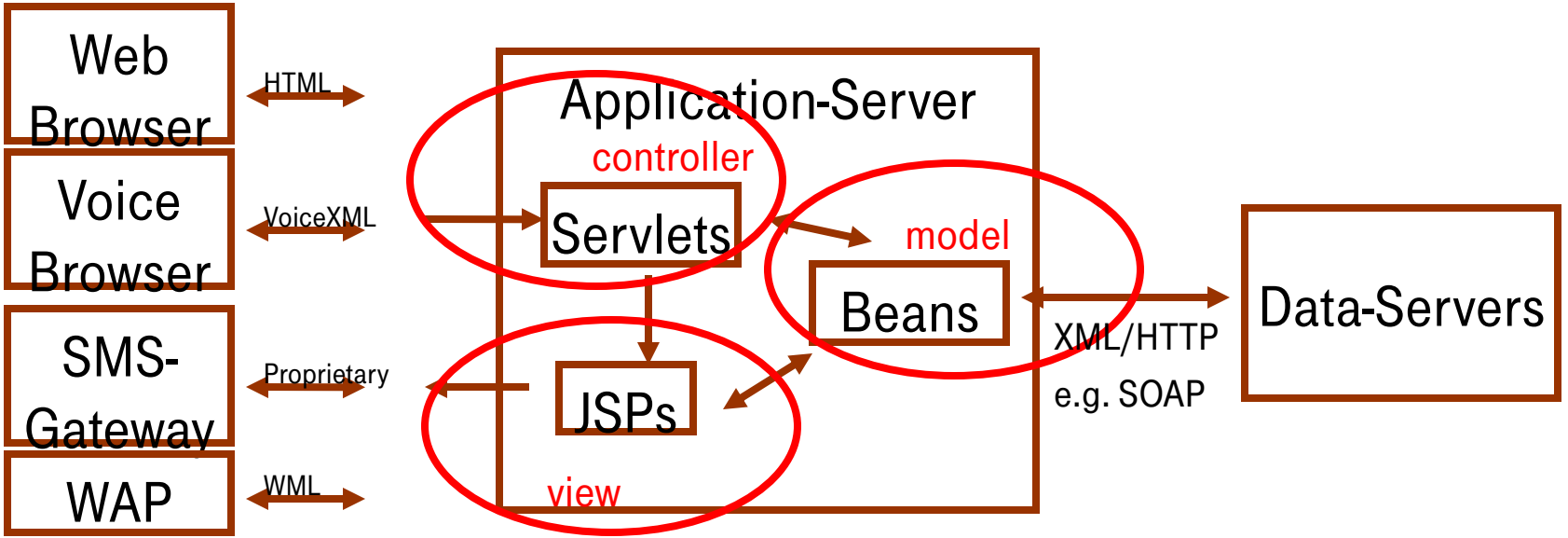
Portal Architecture Introduction

- A Portal is a web-based system that enables a user to interact in order to gather information
- It consists of the following components:
  - Access point, Browser
  - Interaction control, Application server
  - Content, Database

Voice Portal with J2EE/VoiceXML

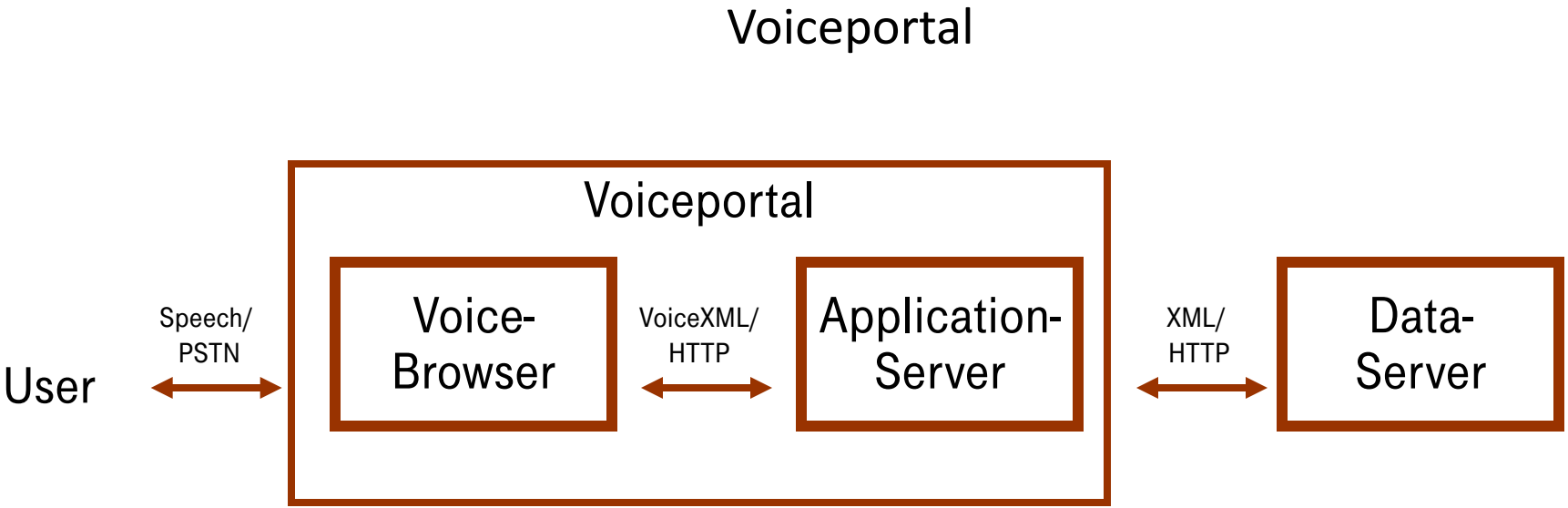
Portal Architecture

Multi-Access Portal



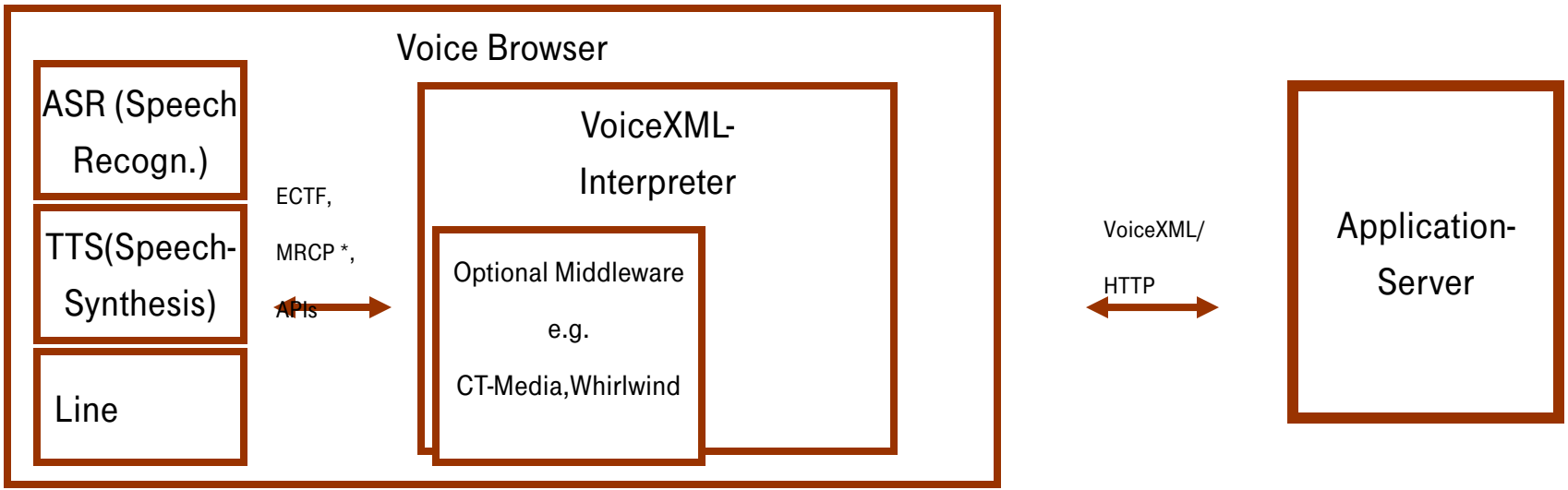
Voice Portal with J2EE/VoiceXML

Portal Architecture



Voice Portal with J2EE/VoiceXML

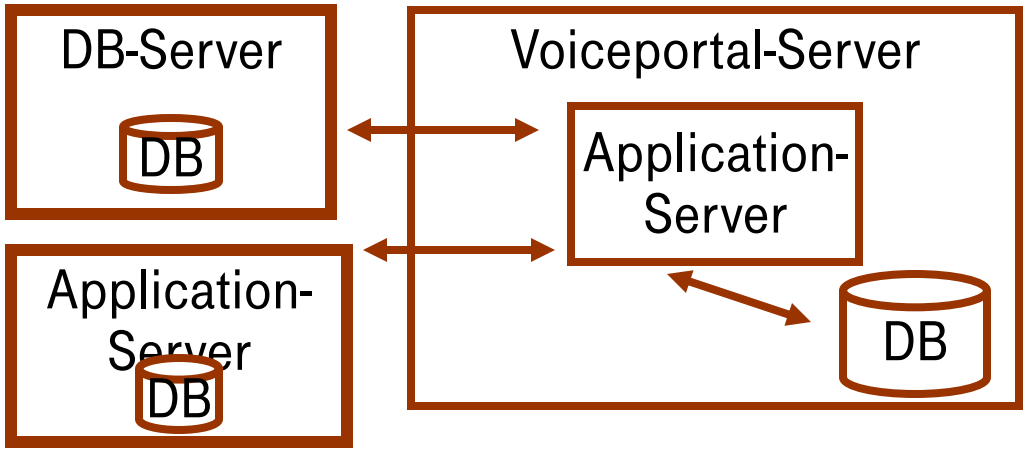
Portal Architecture Voice Browser



* MRCP (Media Ressource Control Protocol) IETF

Voice Portal with J2EE/VoiceXML Database connection

- XML over HTTP: XML (might be SOAP, Simple Object Access Protocol -> WebServices)
- Java-Beans
  - Entity Beans
  - Remote Entitiy Beans
- Direct with JDBC

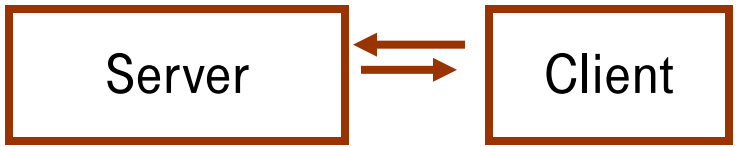


Voice Portal with J2EE/VoiceXML

Data in XML via HTTP

http://datenserver.com/servlet/GetUser?userId=32142

```
<?xml version="1.0"?>
<!DOCTYPE voicePortalLogin SYSTEM "voicePortalLogin.dtd">
<voicePortalUser version="1.0" >
        <user>
                <firstName>Hans</firstName>
                <lastName>Mustermann</lastName>
                <sex val="m"/>
        </user>
</voicePortalUser>
```



VoiceXML Tutorial Contents

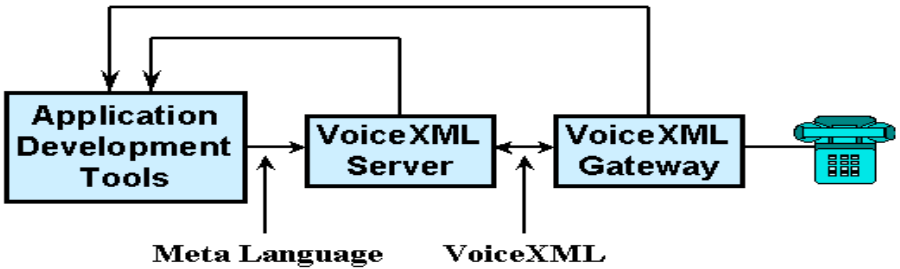
- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML Code Examples
- Architecture VoiceXML/J2EE
- Application Building (Example Speech Navigator)

VoiceXML Application Development Outline

- Overview
- IDE Examples
- Example: Speech Navigator: J2EE-based Application Development Toolkit

VoiceXML Application Development Overview

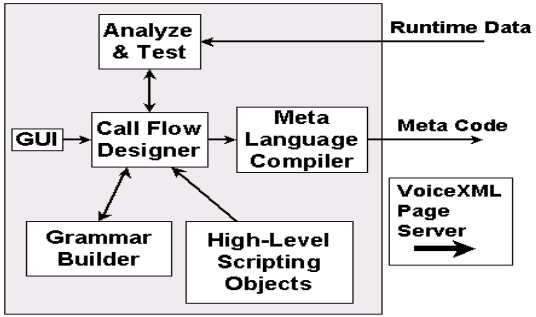
- VoiceXML Application consists of set of VoiceXML-documents and data-interface
- Platform independency problem: VoiceXML is generated dynamically by Metalanguage ->VoiceXML Tools Committee is working on methods to improve uniformity



Picture-source: VoiceXML-Review Vol. 2 Issue 7

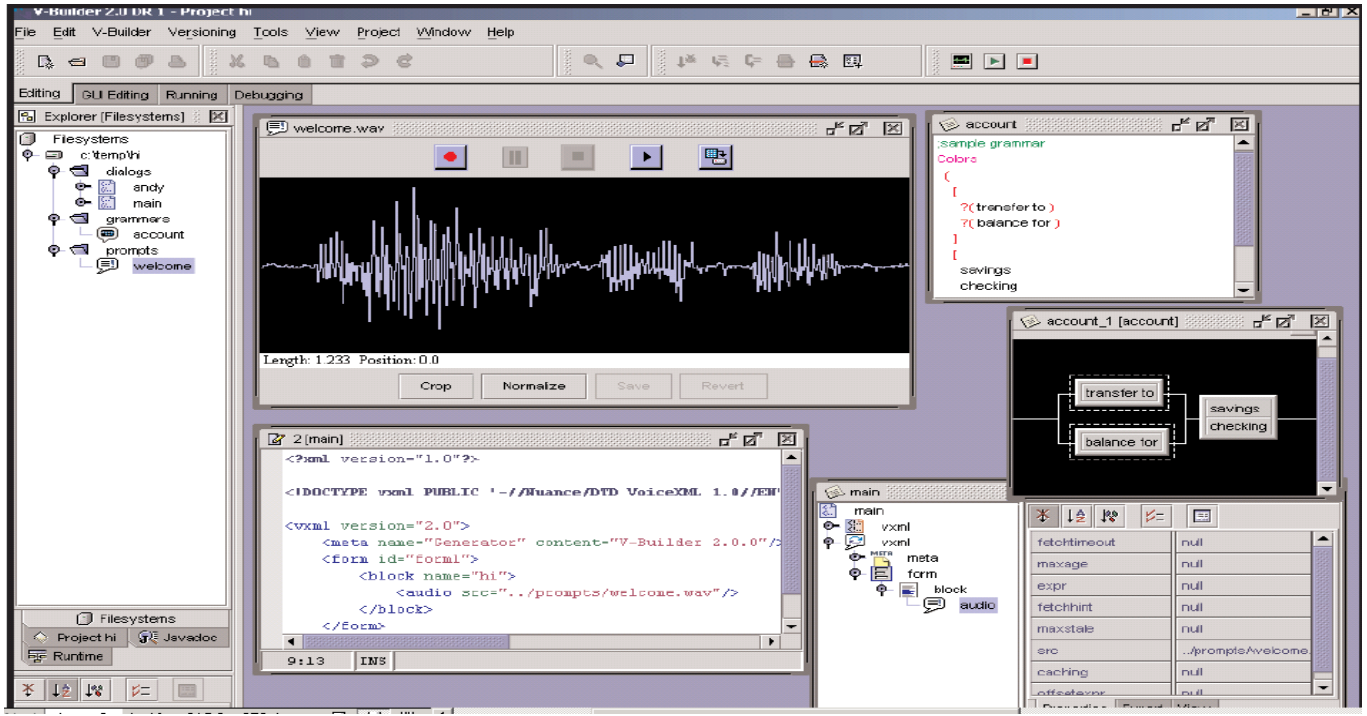
VoiceXML Application Development Overview

- Development of VoiceXML-Documents
  - Usage of favourite text editor
  - Usage of libraries, templates, sub dialog-collections
  - Usage of Generator-library
  - Usage of graphical Development-Environment
    - Desktop-based
    - Web-based

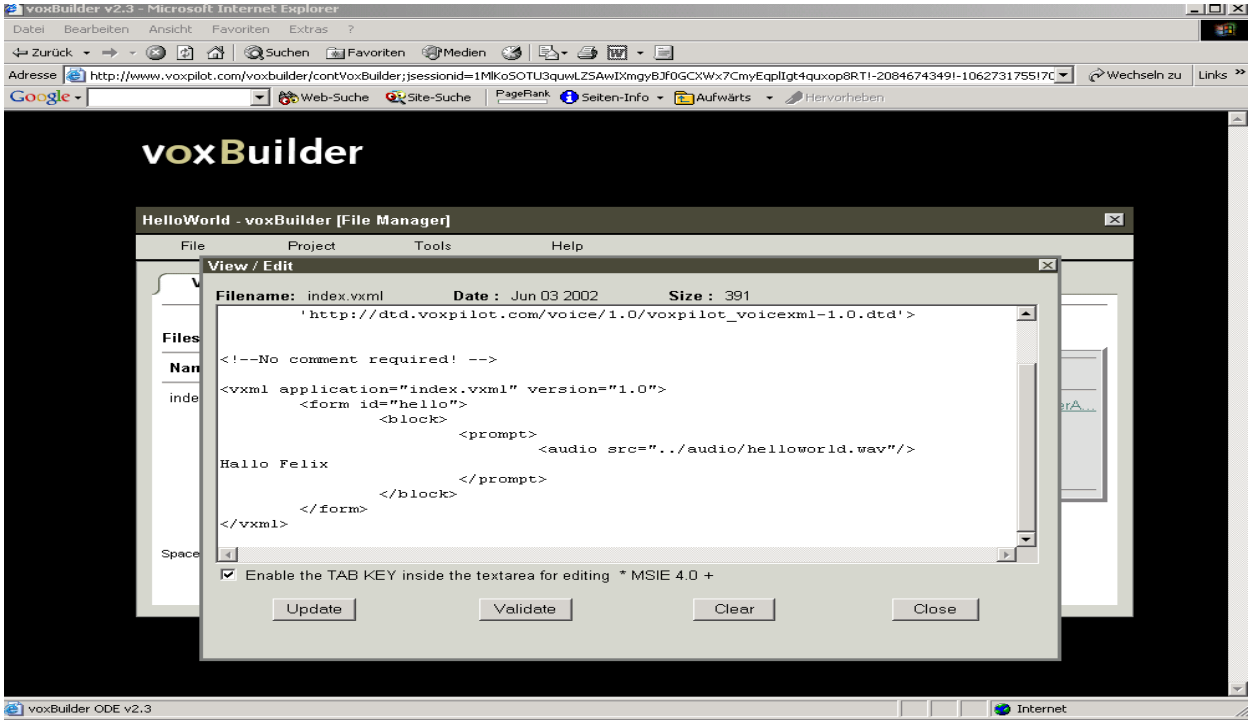


Picture-source: VoiceXML-Review Vol. 2 Issue 7

VoiceXML Application Development Desktop Based IDE: Nuance V-Builder



VoiceXML Application Development
Web Based IDE: VoxPilot VoxBuilder



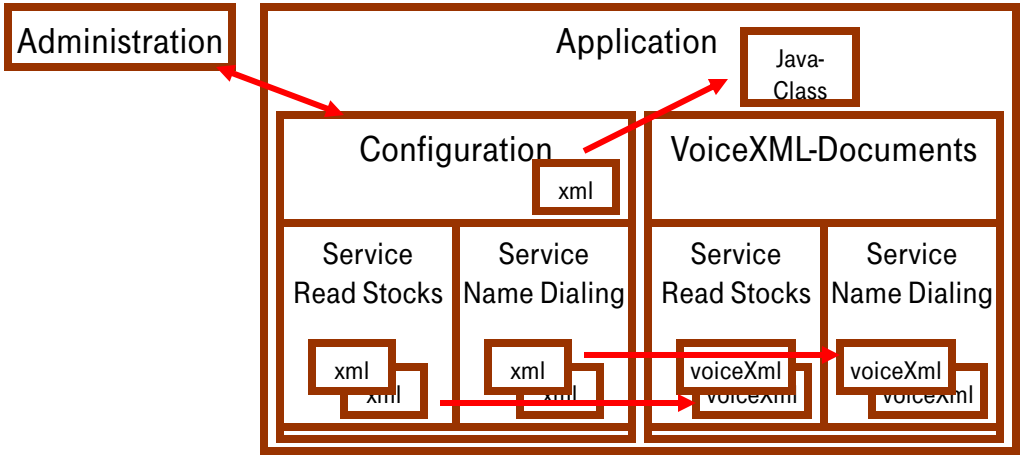
VoiceXML Application Development Example: Speech Navigator

Overview

- Server-based System (except PushService)
- Framework based on J2EE
  - all configuration data is centrally stored as XML-data and can be changed on runtime
  - Utility Classes Java-Library
- Modules
  - Encapsulation of complete functionality, e.g. PushService, Administration, Fax-sending
- Templates and Subdialogs
  - Reuse of VoiceXML-code by adapting template and subdialog library
- Custom-Taglibs
  - Kind of a macro to generate VoiceXML, e.g. formatted dates, escalating help-strategy, ...

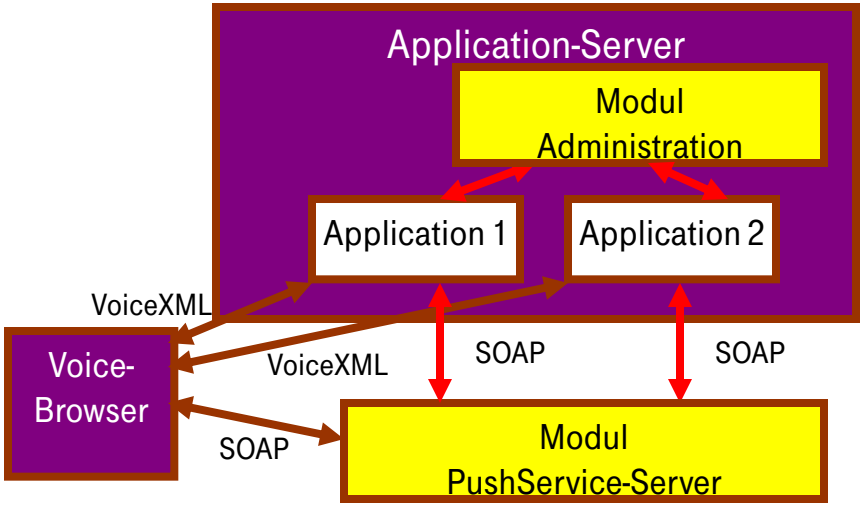
VoiceXML Application Development Speech Navigator: Framework

- Configurable data (prompts, internal grammars, data-server-URLs, timeouts, ...) stored in XML-file hierarchy
- Web-based on runtime changeable
- Java-Utility library (grammar-generation, dates, user-modelling, http-communication, ...)



VoiceXML Application Development Speech Navigator: Modules

- Independent services to encapsulate reusable features
- Examples
  - Administration
  - Push-Service
  - Fax-Service
  - Audio-recording
- Advantages
  - No need to reinvent
  - Usage of one service by several applications



VoiceXML Application Development
Speech Navigator: Taglibs

- Taglib: collection of formatting-tags, similar to html
- Examples:
  - Integration of prompts and grammars
  - Formatting of Dates and nat. numbers
  - VoiceXML-constructs
  - Logging/Reporting
- Advantages:
  - Code gets clearer
  - Look & feel uniform
  - reuse

```
<vxml>
...
<t:EscalatingHelp key="myExample"/>
...
</vxml>
```

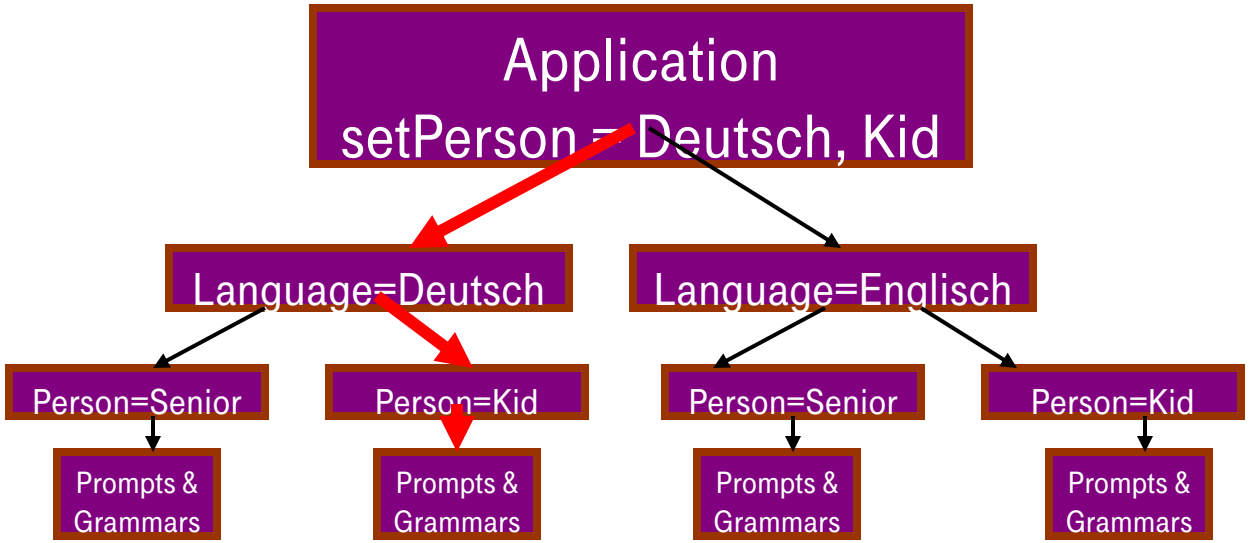
```
<vxml>
...
<nomatch count=„1"> ... </nomatch>
<nomatch count=„3"/> ... </nomatch>
<noinput count=„1"/> ... </noinput>
<noinput count=„3"/> ... </noinput>
...
</vxml>
```

VoiceXML Application Development
Speech Navigator: Change Management

- Administration-interface for developers (editable JSPs)
- Special web-interface for non-technicians
- Hierarchic prompt & grammar editing

VoiceXML Application Development
Speech Navigator: Languages, Personalities

- One application – multiple personalities
- Multiple configuration-file set
- Changeable for each session
- Advantages:
  - Complete separation from wording and app-development



VoiceXML Tutorial Retrospect

- Preliminary (Motivation, Basic Technology)
- Speech Interface Framework Overview
- VoiceXML Code Examples
- Architecture VoiceXML/J2EE
- Application Building (Example Speech Navigator)

VoiceXML Tutorial
Thanks

- Thanks for listening
- Any Questions welcome!