Moderator's Introductory Comments

MVC, April 15, 2013

Seven Breakthroughs needed to create human-equivalent machines

1. Reduce cost of development & maintenance
2. ASR accuracy matches humans
3. Better SLMs & semantic interpretation
4. Find the right dialog framework
5. Real world knowledge
6. Domain adaptation
7. Efficient design/deployment architecture

Useful advanced dialog language constructs

- Powerful data extraction
- Derive rules from base set of rules
- Extract semantic information
- Seamlessly multimodal
- Uses context and user history

Advanced Dialog Workshop:

Hands-on experience with a conversational framework

Presented:

Mobile Voice Conference, San Francisco, CA, April 15, 2013

ejTalk, Inc.
www.ejTalk.com

Emmett Coin

Industrial Poet, CEO

The Problem

Voice development systems today are not very aware of the conversational context.

They DO NOT:

- Have easy ways to resolve pronouns. (“it” refers to?)
- Inherently adapt to repetitious tasks. (shared knowledge?)
- Automatically remember things. (we were talking about?)
- Coordinate multimodal activities with speech.

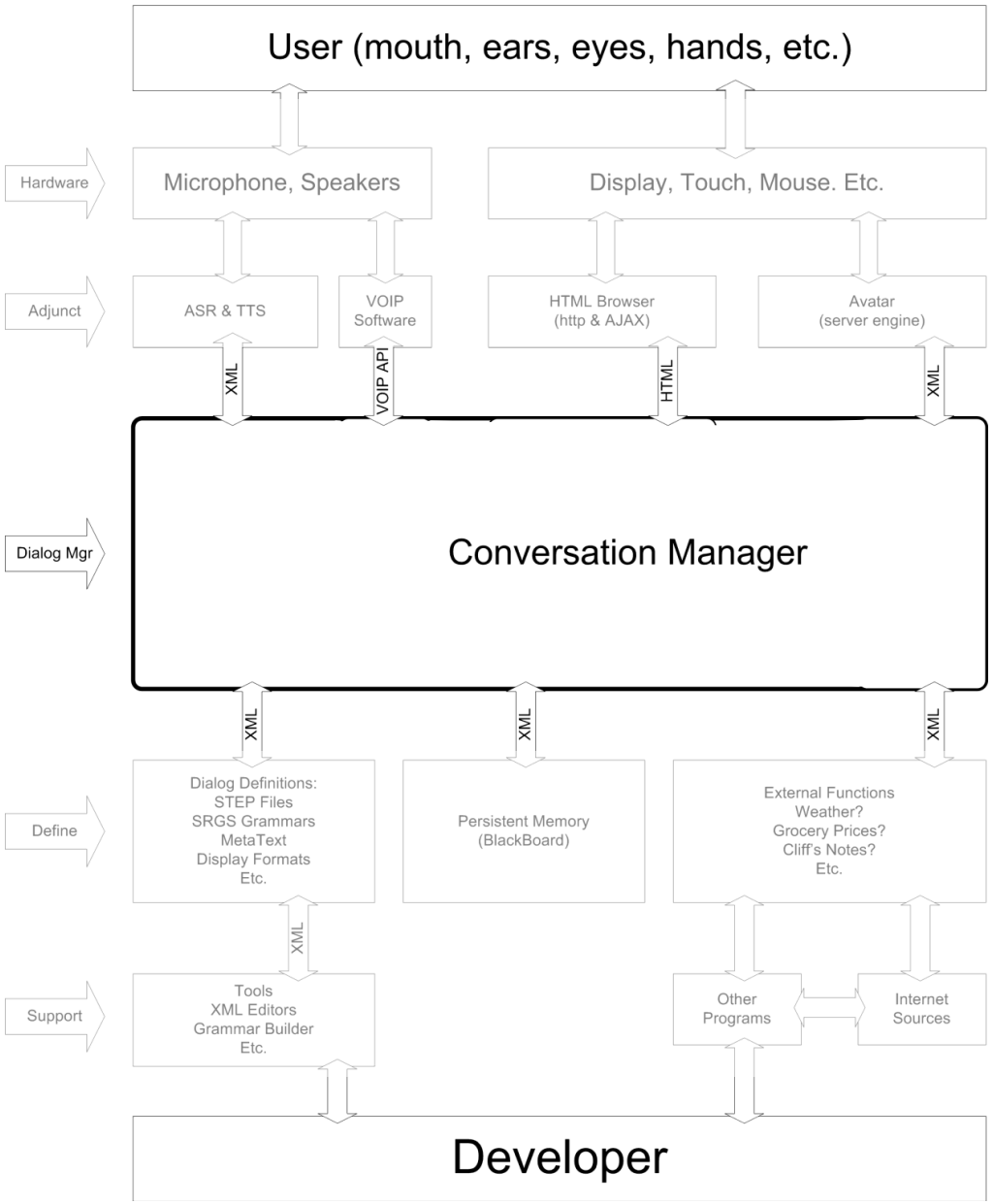
They are missing CONTEXT

The Solution

Integrate all of the application technologies into a single engine that WILL:

- Persist the details of a conversation
- Persist the details of a … relationship.
- Track the history of previous conversational paths.
- Understand “that one” is the “one” we just talked about.

An engine that is built on CONTEXT


User (mouth, ears, eyes, hands, etc.)
Hardware
Microphone, Speakers
Display, Touch, Mouse. Etc.
Adjunct
ASR & TTS
VOIP Software
HTML Browser (http & AJAX)
Avatar (server engine)
XML
VOIP API
HTML
XML
Dialog Mgr
Conversation Manager
XML
XML
XML
Define
Dialog Definitions: STEP Files SRGS Grammars MetaText Display Formats Etc.
Persistent Memory (BlackBoard)
External Functions Weather? Grocery Prices? Cliff's Notes? Etc.
XML
Support
Tools XML Editors Grammar Builder Etc.
Other Programs
Internet Sources
Developer

Is This Time The Time?

(For an truly conversational agent)

- Speech infrastructure is in place
- This generation is comfortable talking to “things”
- Voice search is getting more natural
- Understanding of “natural” speech is improving
- R&D departments experimenting with “conversation”
- Indications are that the public is seriously ready for more
- But most “conversational” apps are just “voice search”
  - Single query … no context
  - Lack adaptation to familiar tasks
  - No relationship with the user

However, context is a BIG PART of any real conversation

Workshop Goals

- Hands-on experience
  - View the specifications
  - Edit & Experiment in a simple development environment
- Exposure to Advanced Dialog Concepts
  - Mechanisms for adaptation and context
- Modify and test behavior on a live system
  - Edit on a laptop and test on an Android phone
- Experience “post search” speech app design
  - It is good if the app remembers things
  - It is better if it uses those memories to adapt automatically

Platform Technologies

(What we will use today)

- Speech ASR/TTS
  - All the usual suspects for recognition and synthesis
  - Google ASR & Native Android TTS
- Conversation Manager
  - VoiceXML?
  - State Charts?
  - ejTalker … more on the next slide
- Generic client device
  - HTML rendering, text & touch interaction
  - AJAX communications to the conversation manager
  - Android phones, laptop browsers

What Is “ejTalker”

This is the Conversation Manager that we will use today:

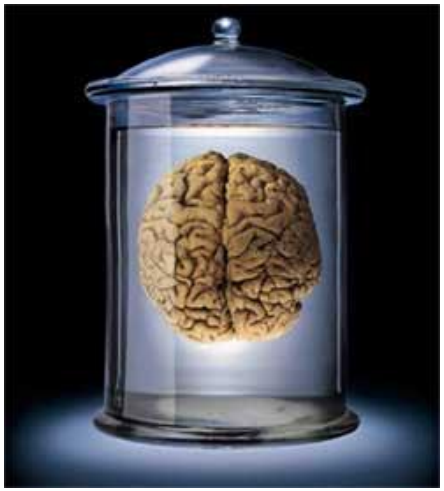
- ejTalker (pronounced: “edge” talker)
  - It coordinates all aspects of a conversation
  - It has been around for years
- Cloud based or embedded
  - Small executable footprint
  - Low compute loads
- A declarative language for conversations
  - XML formalism
  - Syntax reminiscent of rule based systems
- Invoke rich conversional behavior
  - Persistent interaction memory
  - Automaticity (automatic adaptation to usage)
  - Variability (non-static prompts)
  - Derivation of behavior

Cassandra is based on ejTalker

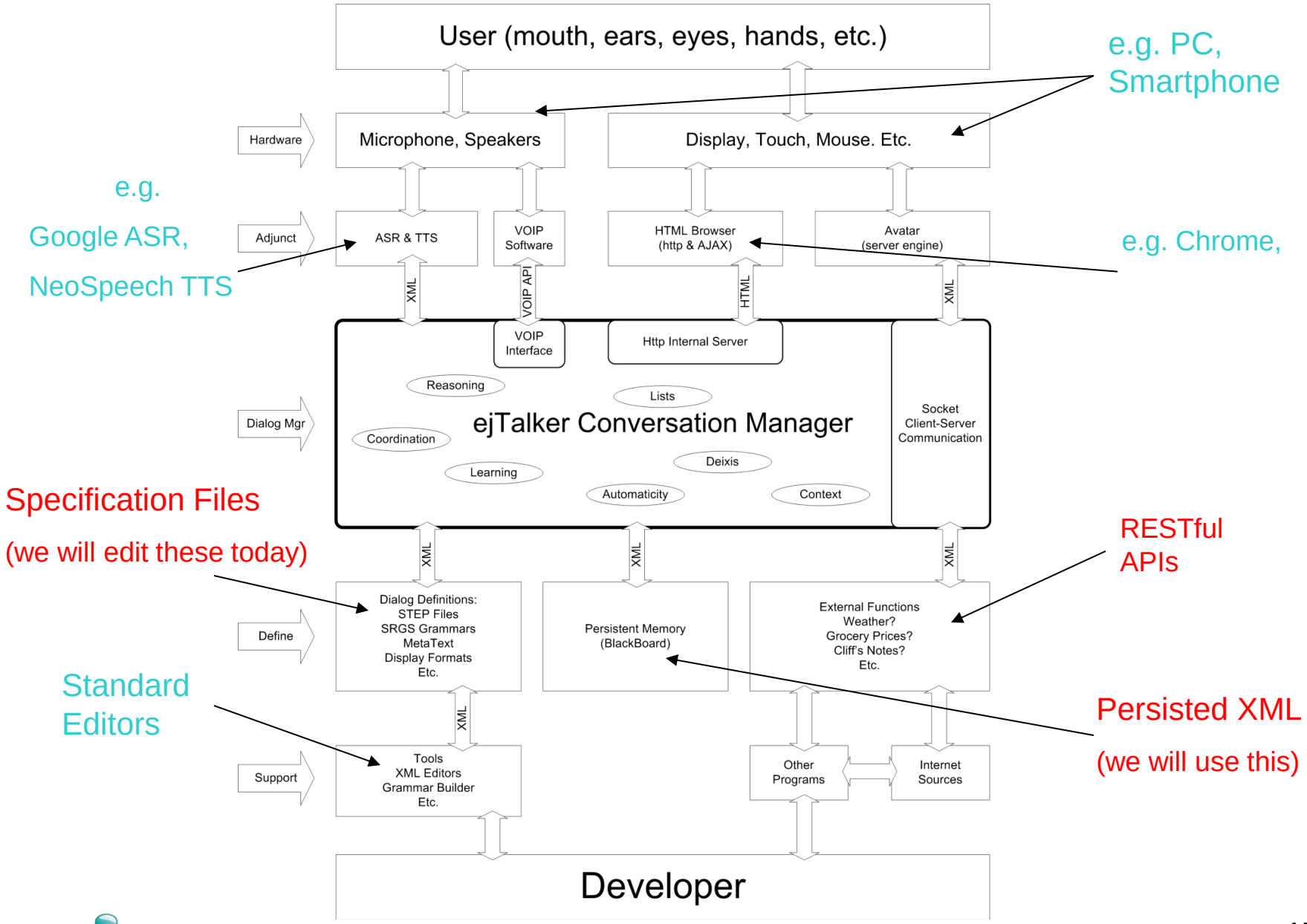
What we will do today

- Brief overview of the platform.
- Describe declarative engine methods
  - Transitions: Rules (in STEP files)
  - Adaptive behavior: metaText files
  - Variability: Production Grammars (prodGram files)
  - Memory: Persisting information
- Derivation
  - Use predefined behaviors (<derivedFrom> declaration)

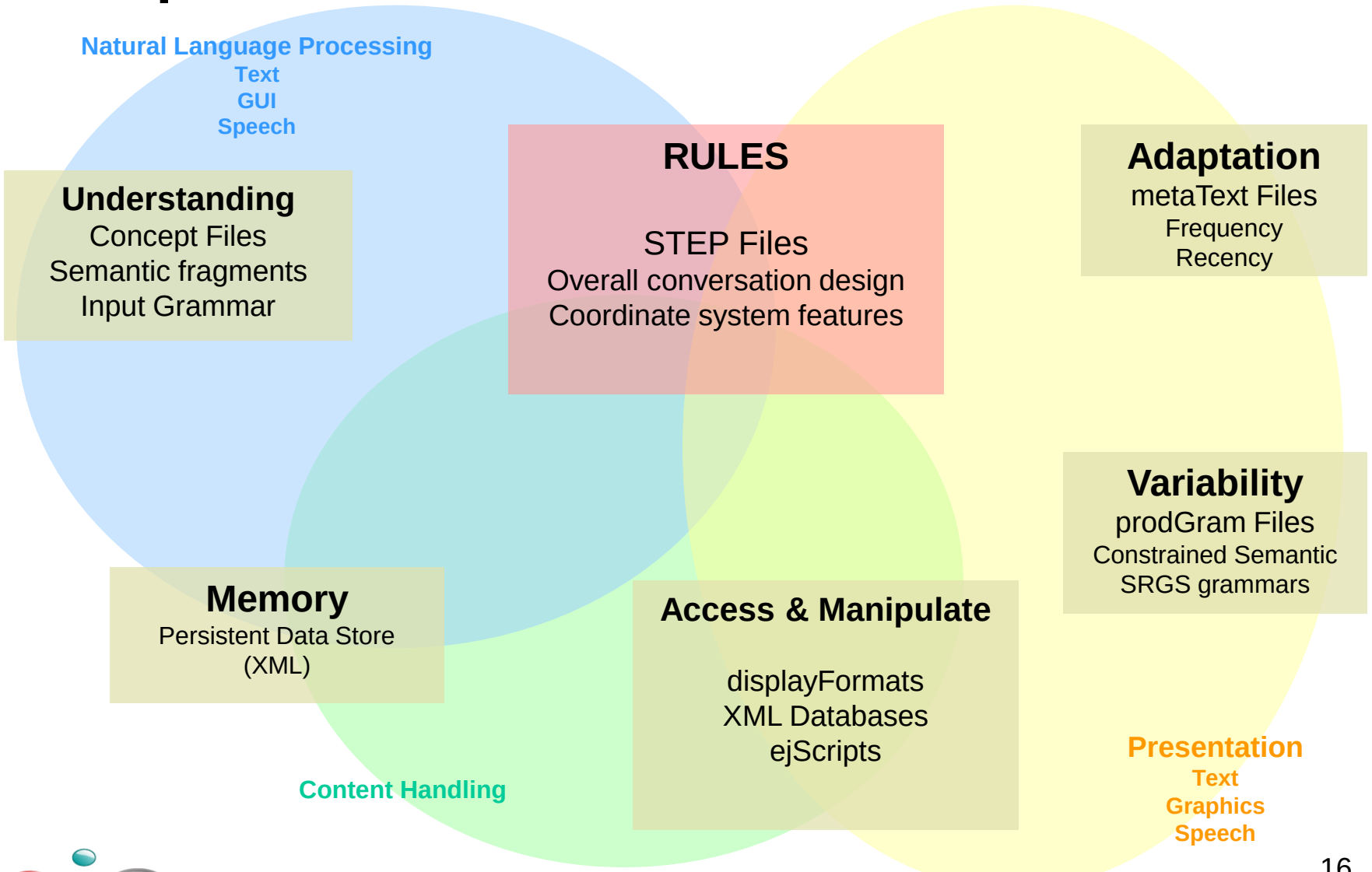
The Conversation Engine
(Just Thoughts)



Adding Input and Output


User (mouth, ears, eyes, hands, etc.)
e.g. PC, Smartphone
Hardware
Microphone, Speakers
Display, Touch, Mouse. Etc.
e.g.
Google ASR,
NeoSpeech TTS
Adjunct
ASR & TTS
VOIP Software
HTML Browser (http & AJAX)
Avatar (server engine)
e.g. Chrome,
XML
VOIP API
HTML
XML
VOIP Interface
Http Internal Server
Reasoning
Lists
Dialog Mgr
ejTalker Conversation Manager
Socket Client-Server Communication
Coordination
Deixis
Learning
Automaticity
Context
Specification Files
(we will edit these today)
XML
XML
XML
RESTful APIs
Define
Dialog Definitions: STEP Files SRGS Grammars MetaText Display Formats Etc.
Persistent Memory (BlackBoard)
External Functions Weather? Grocery Prices? Cliff's Notes? Etc.
Standard Editors
XML
Persisted XML
(we will use this)
Support
Tools XML Editors Grammar Builder Etc.
Other Programs
Internet Sources
Developer

Specification File Architecture



The ejTalker's declarative language.

Rules
Adaptation
Variability
Memory

- STEP files
  - Initial instructions for what to say and do
  - Sets of **Rules** that specify how to react.
- MetaText files
  - Automatic adaptation specs
    - How often and how recently have we done something
- ProdGram
  - Specs for automatic variability
    - Choosing parts of output phrases from semantically equivalent alternatives.
- setMEM & {V:some/path/to/data:default}
  - How to persist a simple memory: setMEM
  - How to access that memory: {V:a/path/:no memory found}

Let's Log On Now!

- Set your wireless network to use
  - Network Name: "Dialog"
  - Password: "myinitial1"
- Open your browser
  - (Chrome, Firefox, etc.)
- Type in address bar:
  - http://192.168.1.119:15400/br1/ejLogon.html
  - You should see the Logon Page with Cassandra
- Type your assigned User Name: e.g. "user73"
- Type (or verify) your Starting STEP File: "myMovie.step.xml"
- ONLY THEN click on the "LOGON: Text I/O"

STEP Files: Top level domain descriptions

Rules
Adaptation
Variability
Memory

- Some administrative parameters
  - <head>
    - Similar function to the <head> in an HTML file
- Content and Interaction Components
  - <body>
    - Similar to the <body> in an HTML file
  - What to do when we enter this domain
    - <introduction>
      - Say and display things
  - Instructions on how to “listen” etc.
    - <attention>
      - Select a recognition environment (optional)
  - Rules that decide what course we take
    - <response> & <rule>
      - Test the input, context, data, etc. then react
      - Where most of the interesting stuff happens!

STEP Files: A first look

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> Say something about  Lincoln. </text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text> Lincoln was a good movie! </text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

STEP: A Prompt

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> Say something about  Lincoln. </text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text> Lincoln was a good movie! </text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

STEP: A Rule

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> Say something about  Lincoln. </text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text> Lincoln was a good movie! </text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

STEP: A Reply

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> Say something about  Lincoln. </text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text> Lincoln was a good movie! </text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

The ejTalker's declarative language.

Rules
Adaptation
Variability
Memory

- STEP files
  - Initial instructions for what to say and do
  - Sets of **Rules** that specify how to react.
- MetaText files
  - Automatic adaptation specs
    - How often and how recently have we done something
- ProdGram
  - Specs for automatic variability
    - Choosing parts of output phrases from semantically equivalent alternatives.
- setMEM & {V:some/path/to/data:default}
  - How to persist a simple memory: setMEM
  - How to access that memory: {V:a/path/:no memory found}

metaText: A Prompt

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie1.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> {M:movie.metaText.xml#promptForLincoln:Say something about  Lincoln.}</text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text> Lincoln was a good movie! </text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

metaText: What does it look like?

Rules
Adaptation
Variability
Memory

Cassandra says:
{M:myMovie.metaText.xml#promptForLincoln:Say something about Lincoln.}

She uses the following specification:
(this is an XML snippet from the file “myMovie.metaText.xml”)

```
<promptForLincoln>
        <int>
            <val>So, do you have any comments or thoughts about the movie Lincoln?</val>
        </int>
        <tut>
            <val>Why don't we talk about the Lincoln film.</val>
        </tut>
        <beg>
            <val>Say something about Lincoln.</val>
        </beg>
        <nor>
            <val>Comments on Lincoln?</val>
        </nor>
        <exp>
            <val>Okay, sure, one more time! Please share your thoughts about Lincoln.</val>
        </exp>
</ promptForLincoln >
```

metaText: What does it do?

Rules
Adaptation
Variability
Memory

- Track frequency and recency of usage
  - Uses an impulse/amplitude and decay/half-life methodology
- Quantify the concept of familiarity
  - How many times has it been done
  - How recently has it been done
- Abstract the adaptation to repeated behavior
  - Repeated sub tasks can automatically become more concise
- Accommodate dynamically and in the moment

The ejTalker's declarative language.

Rules
Adaptation
Variability
Memory

- STEP files
  - Initial instructions for what to say and do
  - Sets of **Rules** that specify how to react.
- MetaText files
  - Automatic adaptation specs
    - How often and how recently have we done something
- ProdGram
  - Specs for automatic variability
    - Choosing parts of output phrases from semantically equivalent alternatives.
- setMEM & {V:some/path/to/data:default}
  - How to persist a simple memory: setMEM
  - How to access that memory: {V:a/path/:no memory found}

prodGram: Mix it up!

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie2.step.xml</name>
  </head>
  <body>
    <introduction>
      <action>
        <presentation>
          <text> {M:movie.metaText.xml#sayLincoln:Say something about  Lincoln.}</text>
        </presentation>
      </action>
    </introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">
        <pattern>*lincoln*</pattern>
        <action>
          <presentation>
            <text>Lincoln was a {G:ejCore.prodGram.xml#great:good} movie!</text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

What does prodGram look like?

Rules
Adaptation
Variability
Memory

Cassandra says: …{G:ejCore.prodGram.xml#great:good}…

She uses the following specification:

```
<rule id="great" scope="public" ex="That joke was GREAT.">
    <one-of>
        <item>great</item>
        <item>terrific</item>
        <item>wonderful</item>
        <item>amazing</item>
        <item>fantastic</item>
        <item>grand</item>
        <item>marvelous</item>
    </one-of>
</rule>
```

Production Grammars

prodGram

Rules
Adaptation
Variability
Memory

- What does it do?
  - Adds variability via a simple semantic reference
  - Abstracts variability in the dialog
- What is it?
  - Rules written in exactly the same SRGS form as used for recognition grammars
    - It is a familiar syntax
  - A library of prodGram rules
    - Common phrases
    - Simplifies composing
    - Encourages persona consistency

The ejTalker's declarative language.

Rules
Adaptation
Variability
Memory

- STEP files
  - Initial instructions for what to say and do
  - Sets of **Rules** that specify how to react.
- MetaText files
  - Automatic adaptation specs
    - How often and how recently have we done something
- ProdGram
  - Specs for automatic variability
    - Choosing parts of output phrases from semantically equivalent alternatives.
- setMEM & {V:some/path/to/data:default}
  - How to persist a simple memory: setMEM
  - How to access that memory: {V:a/path/:no memory found}

setMEM: Remember

Rules
Adaptation
Variability
Memory

```
<step>
  <head>
    <name>myMovie3.step.xml</name>
  </head>
  <body>
    <introduction>…</introduction>
    <attention></attention>
    <response>
      <rule name="mentionLincoln">…</rule>
      <rule name="mentionSkyfall">
        <pattern>*skyfall*</pattern>
        <action>
          <setMEM>
            <v>movie/dislike=Skyfall</v>
          </setMEM>
          <presentation>
            <text>Bond? I didn't like {V:movie/dislike:that movie} very much.</text>
          </presentation>
        </action>
      </rule>
    </response>
  </body>
</step>
```

What does setMEM look like?

Rules
Adaptation
Variability
Memory

ejTalker <action> element that sets a persistent memory.

It uses the following specification:

```
<setMEM>
   <v>movie/dislike=Skyfall</v>
      … (set as many as you want) …
      e.g.
   <v>movie/genre=action</v>
   <v>some/other/variable=127</v>
</setMEM>
```

Use them later like this:

"I think the genre is {V:movie/genre:who knows what}."
… (it is remembered forever) …

Remember

setMEM

Rules
Adaptation
Variability
Memory

- What does it do?
  - Saves strings in a hierarchical data store
  - Persists data over the relationship
- What is it?
  - String assignments set within <action> elements
    - Arbitrarily deep pathing
    - Hierarchy promotes ontologies
  - String value evaluation wherever “constant” string can be used
    - Common evaluation syntax for all system values

Contributors

(members AVIOS and/or Advanced Dialog Group)

- **David Thomson**, PMTS - AT&T Labs
- **Emmett Coin**, CEO & Industrial Poet - ejTalk Corp.
- **Deborah Dahl**, Principal - Conversational Technologies
- **John Tadlock**, Lead Principal Technical Architect - AT&T Services
- **K. W. 'Bill' Scholz**, President - NewSpeech LLC
- **Lorin Wilde**, CTO - Wilder Communications, Inc.
- **Marie Meteer** – Brandeis University
- **Ria Farrell Schalnat**, Counsel – Dinsmore & Shohl LLP

Thank you

Emmett Coin
ejTalk, Inc
emmett@ejTalk.com