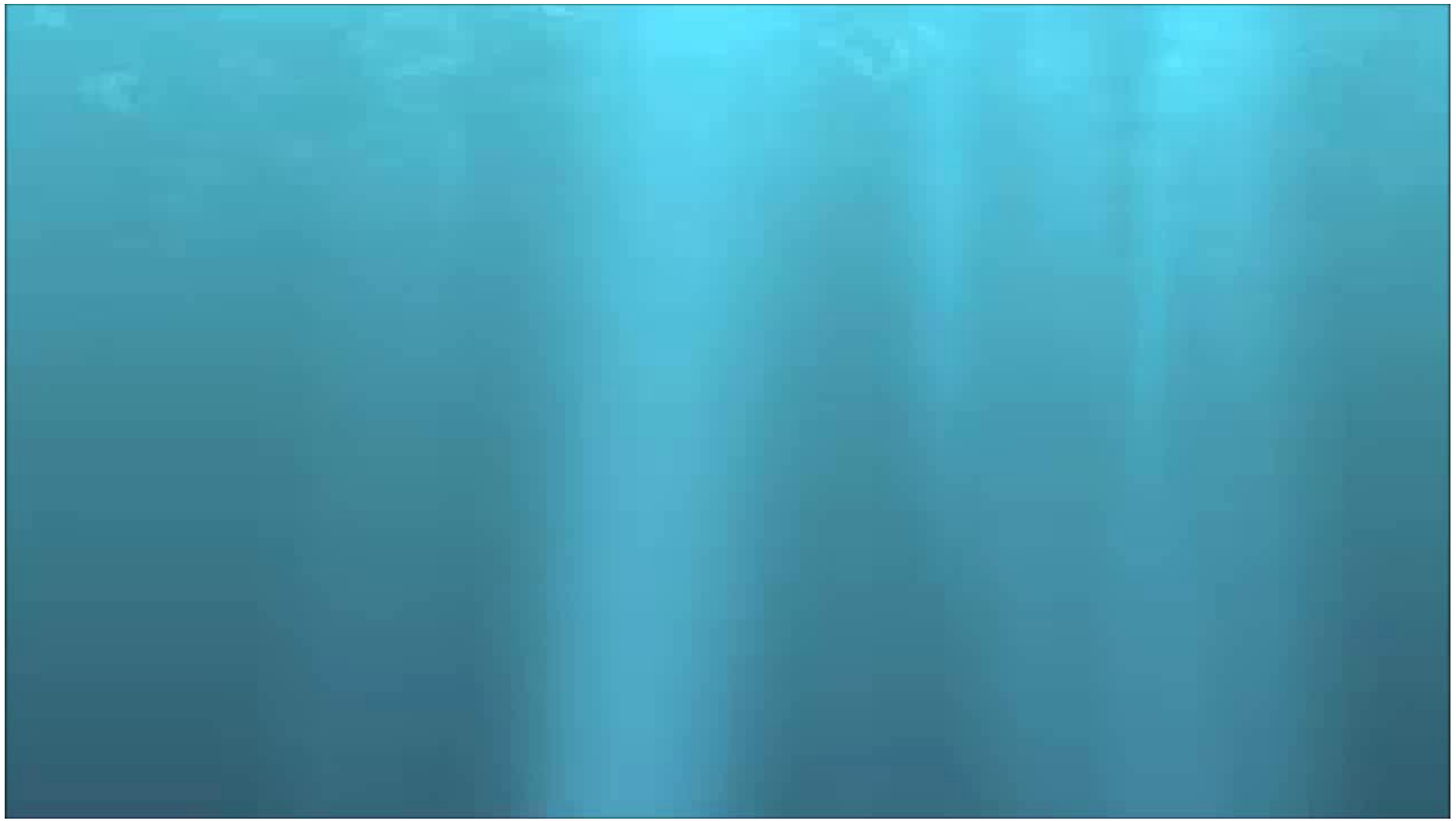
Creating an Advanced Dialog Application

An in-depth look at the process

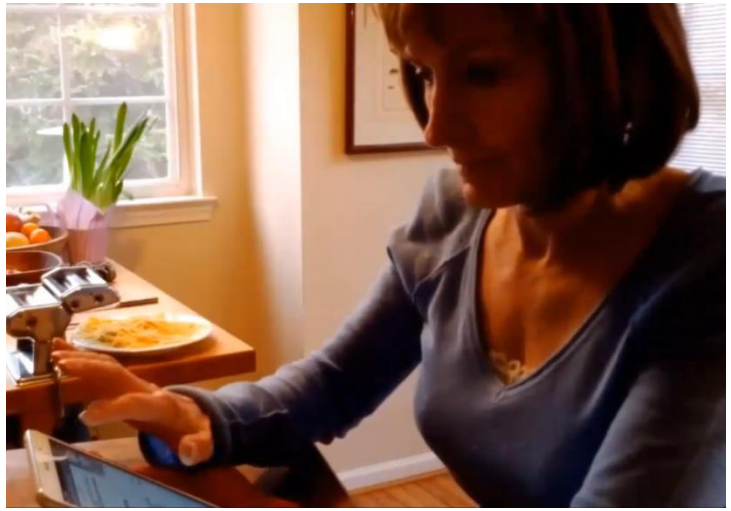
Lorin Wilde, CTO, Wilder Communications
(substituting for John Tadlock, Lead Principal Technical Architect, AT&T)
Marie Meteer, Adjunct Professor, Brandeis University
Emmett Coin, Speech Scientist and Founder, ejTalk

Introduction: K. W. (Bill) Scholz, President, NewSpeech, LLC

The Scenario



A Turn in an Advanced Dialog



- *discourse events*
- *flexible topic flow*
- *mixed initiative*

- *pronoun resolution*
- *multimodal input*

"But wait! Do I have enough for this?"

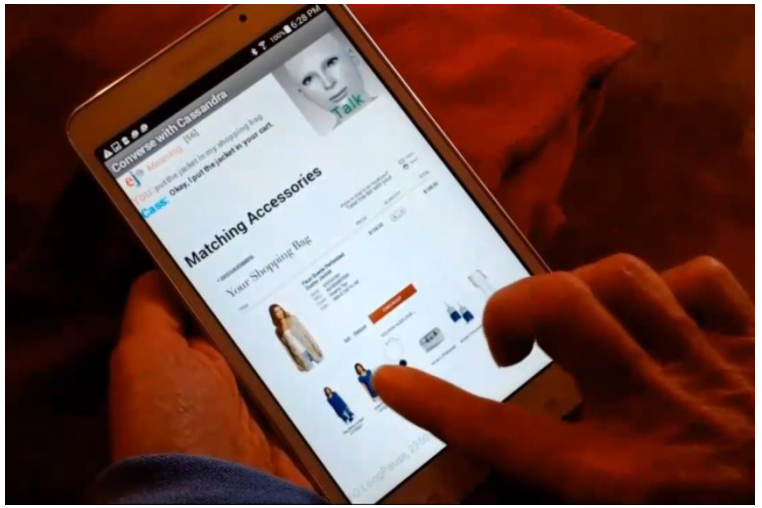
"No, but I could show you some tops that are on sale?"

"You're $8 over promotions. Want to save?"

"No, but there might be some, want to change the budget?"

"Not if you buy the suede jacket."

- *Personal information*
- *Problem solving*
- *Decision point*



A Preview of our Conclusions

- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected

Agenda:

Lorin Wilde (for John Tadlock)

- Project Development Methods

Marie Meteer

- Business Requirements
- Architectural Solution
- User Interface Design

Emmett Coin

- System Requirements
- Implementation

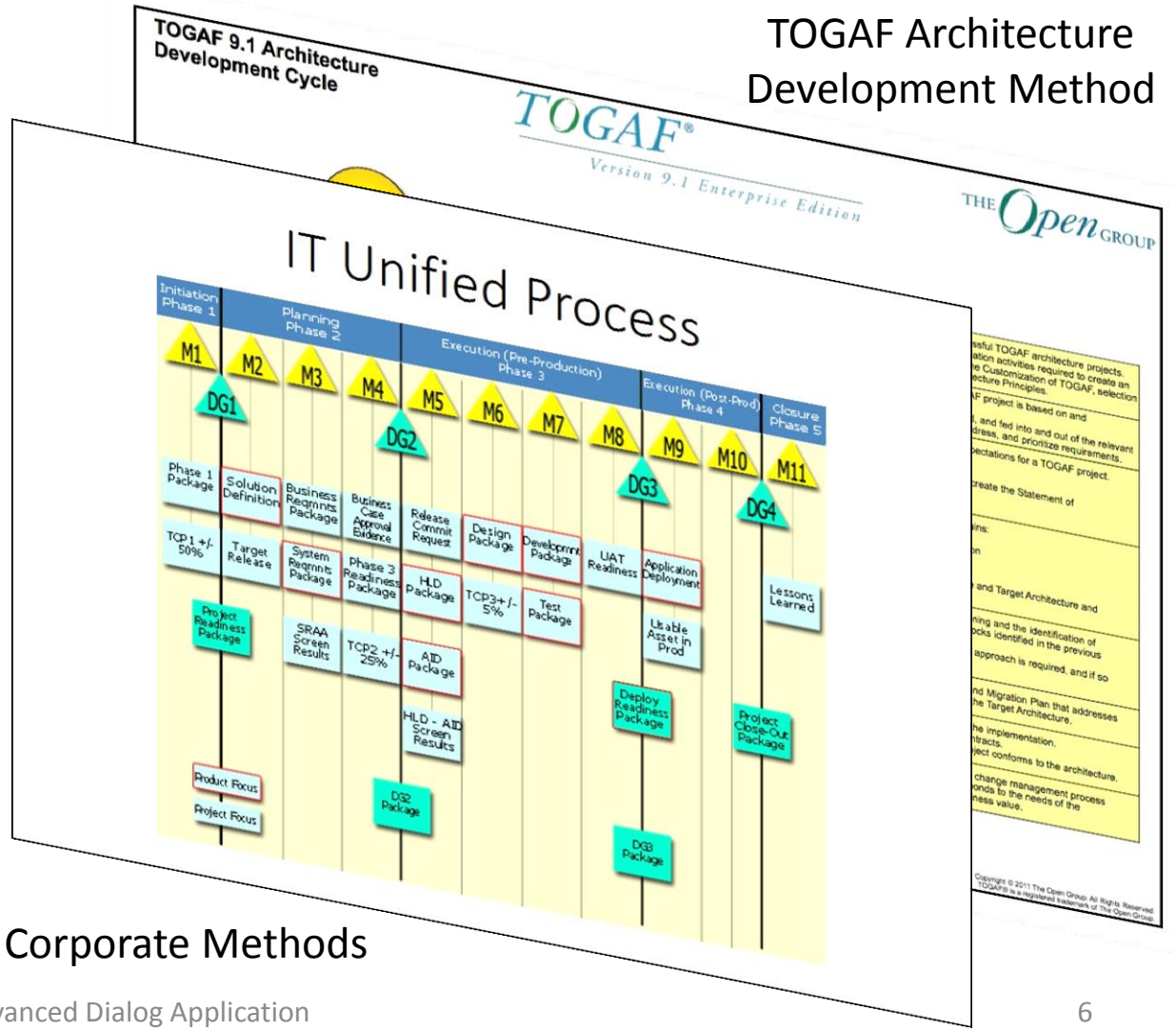
Project Development Methods

What are Project Development Methods?

Approaches for planning, designing, implementing, and governing complex projects.

Why are they needed?

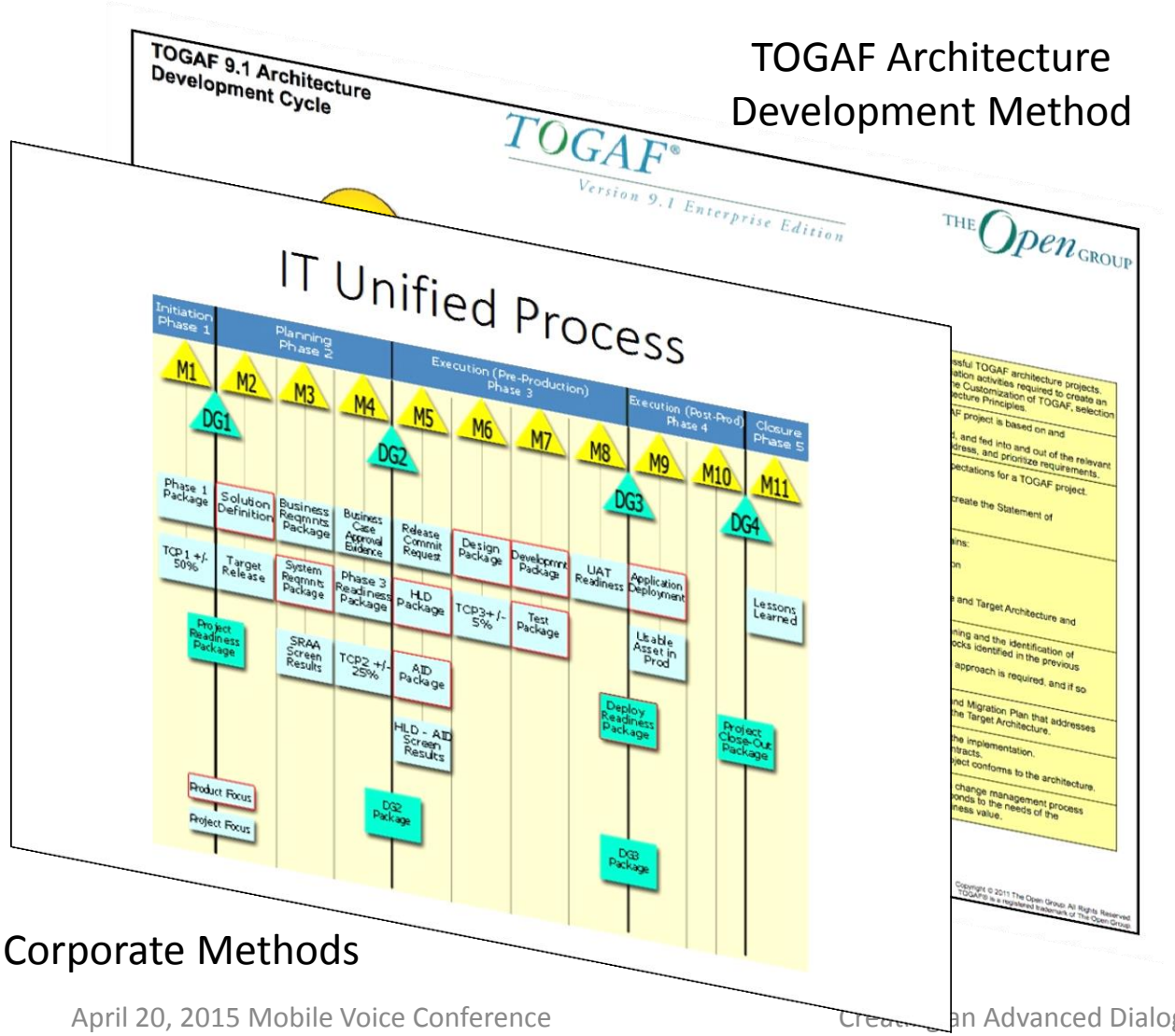
- Manage complexity
- Ensure completeness
- Support an enterprise architecture
  - Which supports strategy and planning
- Support change and growth
- Lower costs, reduces risk
- Improve time to market



TOGAF Architecture Development Method

Corporate Methods

Existing Methods Typical Artifacts Produced



TOGAF Architecture Development Method

Corporate Methods

1. Preliminary Project Request
2. Initial Architectural Solution

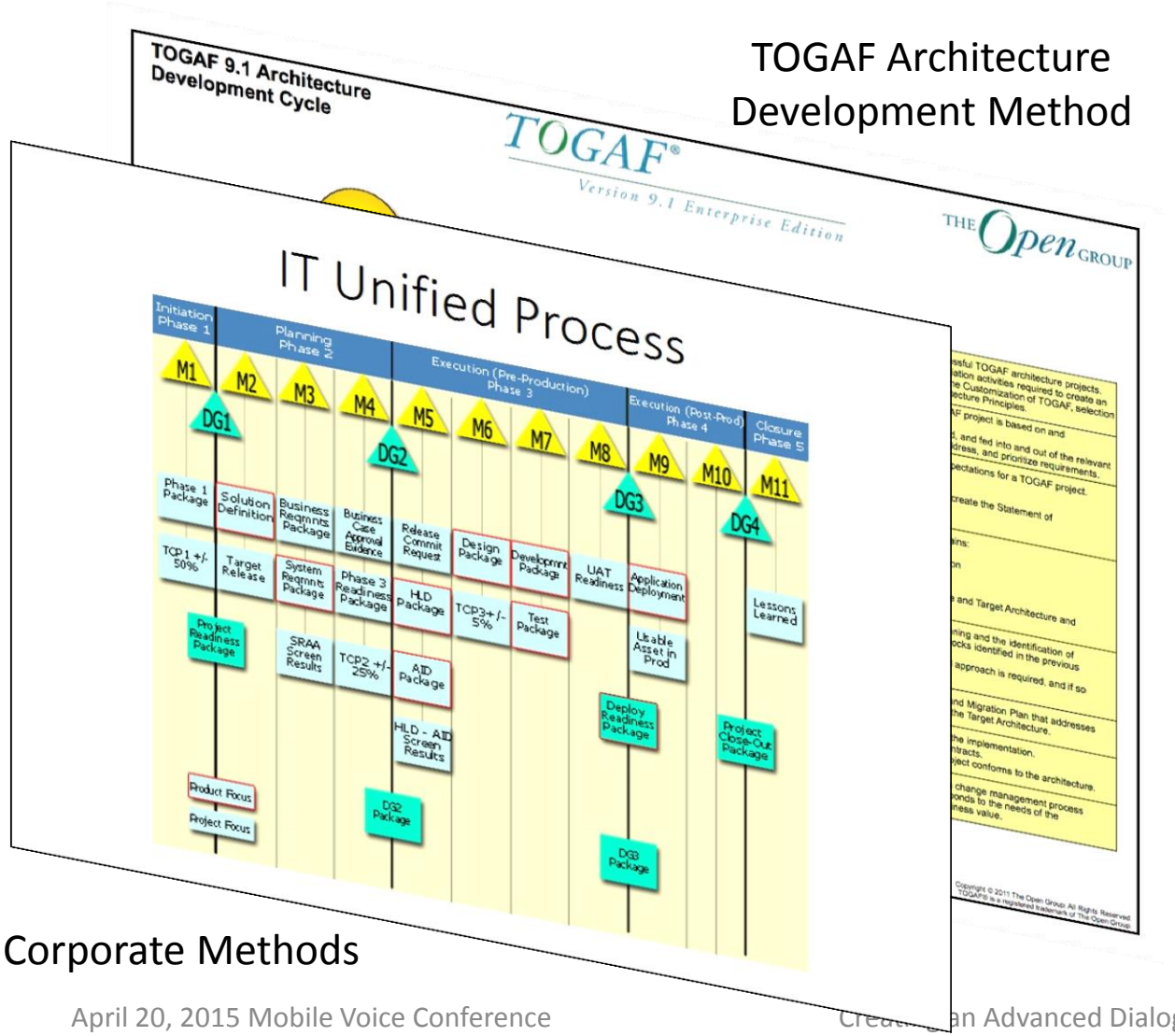
   Early Cost Estimate 
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case 
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination 
8. Deployment

   Project Close

Existing Methods Typical Artifacts Produced



TOGAF Architecture Development Method

Corporate Methods

1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate

3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case

5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination

8. Deployment

   Project Close

Agile Development

Advanced Dialog Applications Need Enhanced Methods and Enhanced Artifacts!

1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination
8. Deployment

   Project Close

Advanced Dialog Applications Need Enhanced Methods and Enhanced Artifacts!

What would these look like?

1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination
8. Deployment

   Project Close

Conclusions

- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected

1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination
8. Deployment

   Project Close

Conclusions (and Ramification #1)

- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected

Agile development needs to be extended
to apply to Business Requirements and Architecture.

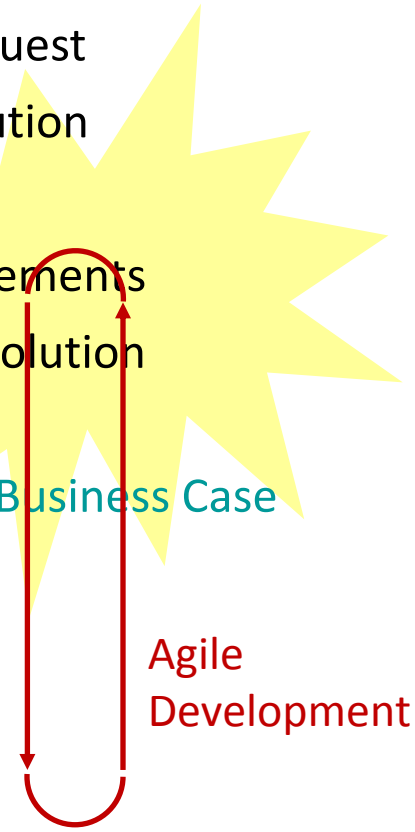
1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination
8. Deployment

   Project Close



Conclusions (and Future Work)

What makes this difficult?

- Iterative business requirements ***delay determining final costs***
- Often, ***separate organizations*** are responsible for Business Requirements, Architecture, and Development.
- Even within Development, ***separate teams*** are responsible for separate applications.
- Commonly, key aspects of the ***work is outsourced*** to vendors.

"Agile" development needs to be extended to apply to Business Requirements and Architecture.

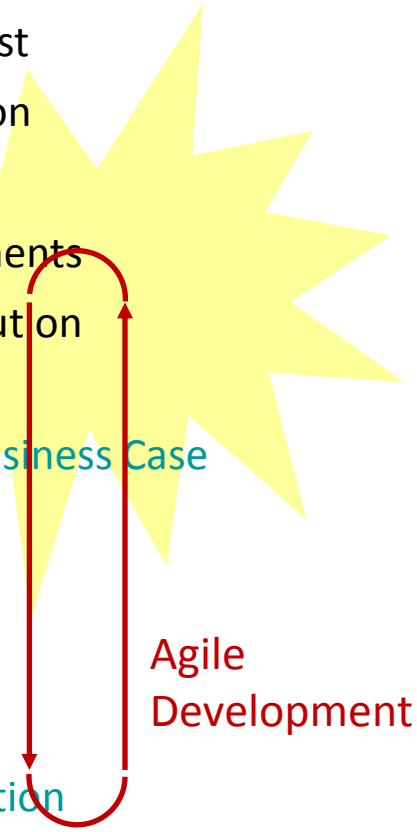
1. Preliminary Project Request
2. Initial Architectural Solution

   Early Cost Estimate
3. Formal Business Requirements
4. Detailed Architectural Solution
   - System Requirements

   Detailed Costs and Business Case
5. High Level Design
   - User Interface Design
6. Development
7. Testing

   Readiness Determination
8. Deployment

   Project Close



Business Requirements

Business Requirements for our Dialog Turn

- ~~"Provide a multimodal voice user interface to a catalog ordering system"~~
- Provide a user experience on mobile devices that the allows customers to
  - Browse and select products from an extensive catalog
  - Comparison shop
  - Manage a shopping cart
  - Respect a budget provided by a financial application
- Provide a user experience that
  - Maximizes the value of the customer and
  - brings the customer back

What makes this difficult?

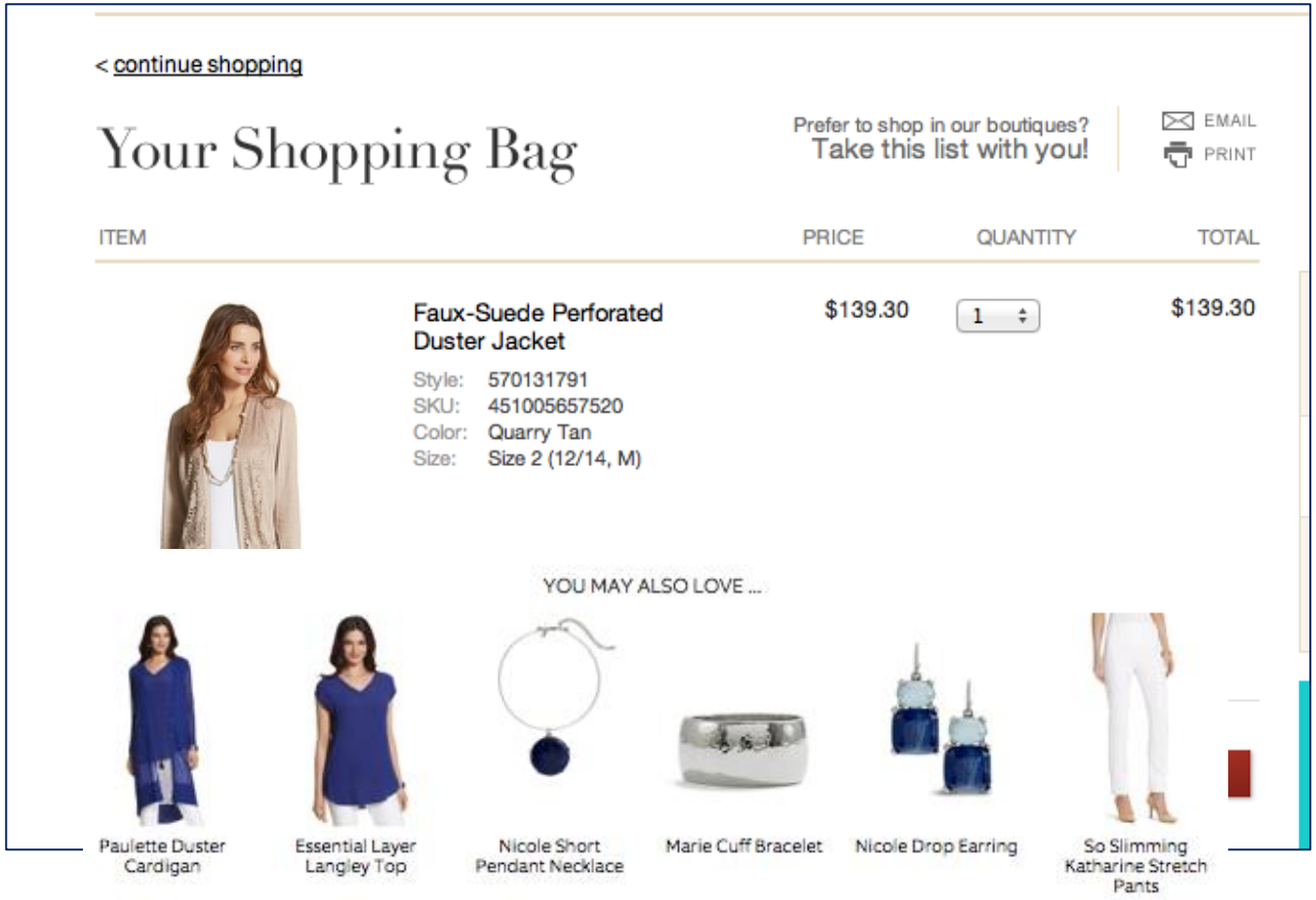
- How do you describe the WHAT without the HOW?
- "Use cases" focus on general functionality
- Need scenarios that drive advanced development

Scenario :

"I need a jacket and I don't want to spend more than $200.00."

- Dialog so far results in the choice of jacket.
- *You may also love* opens up new opportunities.

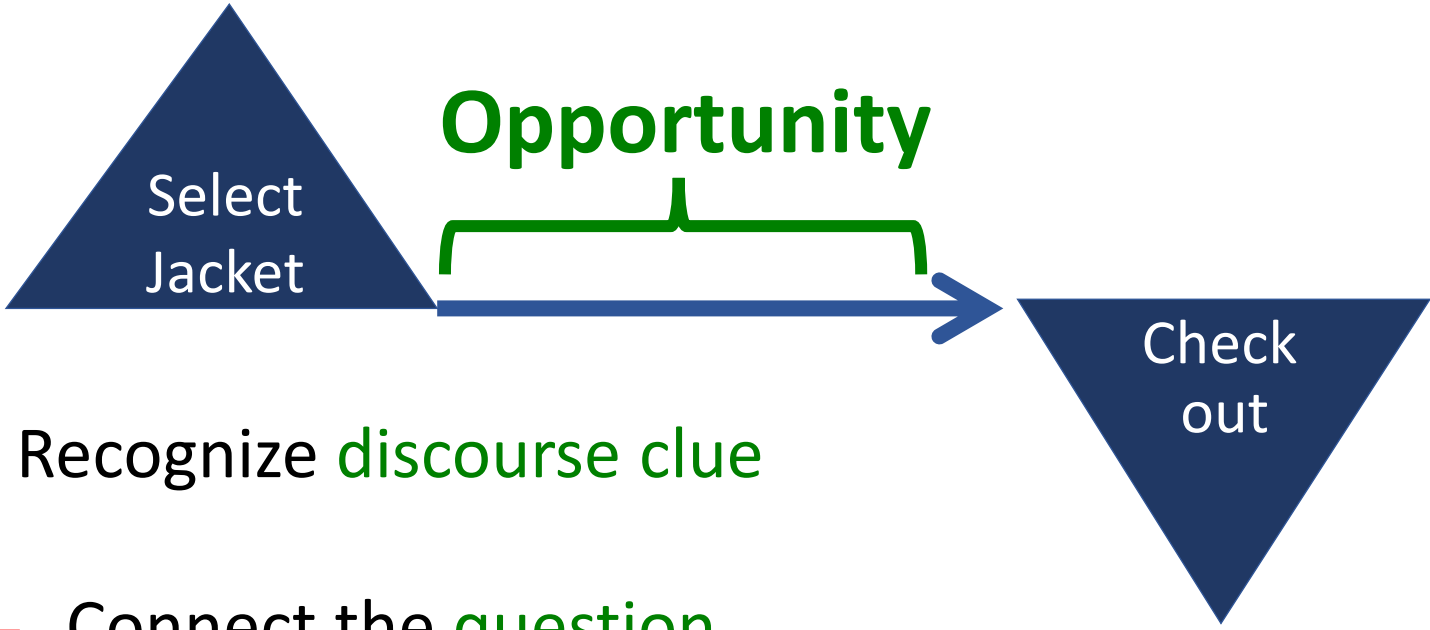
"Oh, show me the blue top."



Scenario:
The Key Moment

- **"But wait! Do I have enough in my budget for this one?"**
- ***What should the response be?***

Scenario:
The Key Moment



But wait ← Recognize discourse clue

Do I have enough ← Connect the question

in my budget ← with the app (budget management)

for this one ← Identify the object and Pick out the appropriate facet ($$$)

- ***What should the response be?*** ← **How to make the most of this opportunity?**

From a dialog management point of view

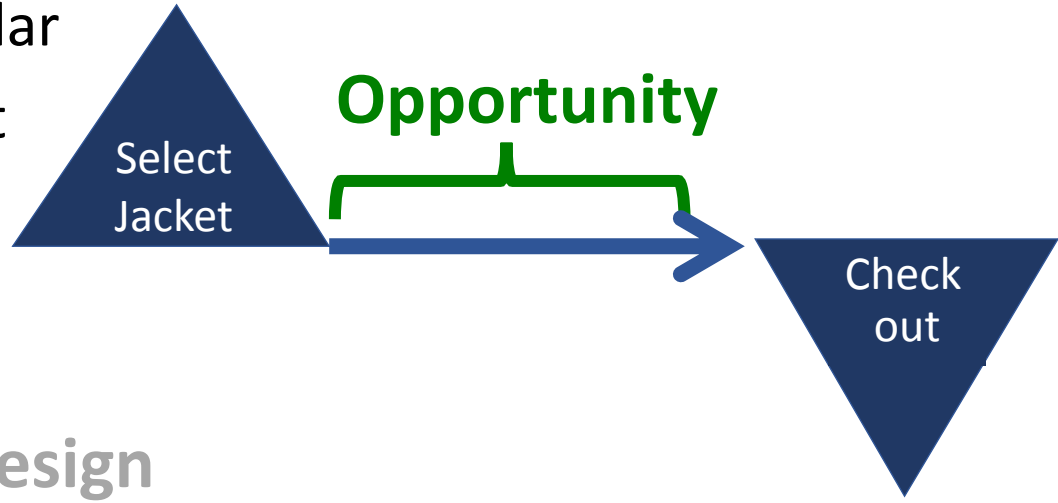
- Where would alternative replies lead the dialog?
  - **Not if you buy the Suede jacket.**
  - **No, you will be $8.00 over budget**
    - Is that OK?
    - There are some promotions available. Let me check whether one would apply.
  - **No, but we have some similar tops on sale.**
    - Would you like to see those?
- A dialog manager needs to determine which direction
  - Best matches the goals of the user
  - And best matches the business requirements

What makes this difficult?

- User goals need to be identified and tracked
- Business requirements need to be translated so they can guide dialog choices

Principles (and Ramification #2)

- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected

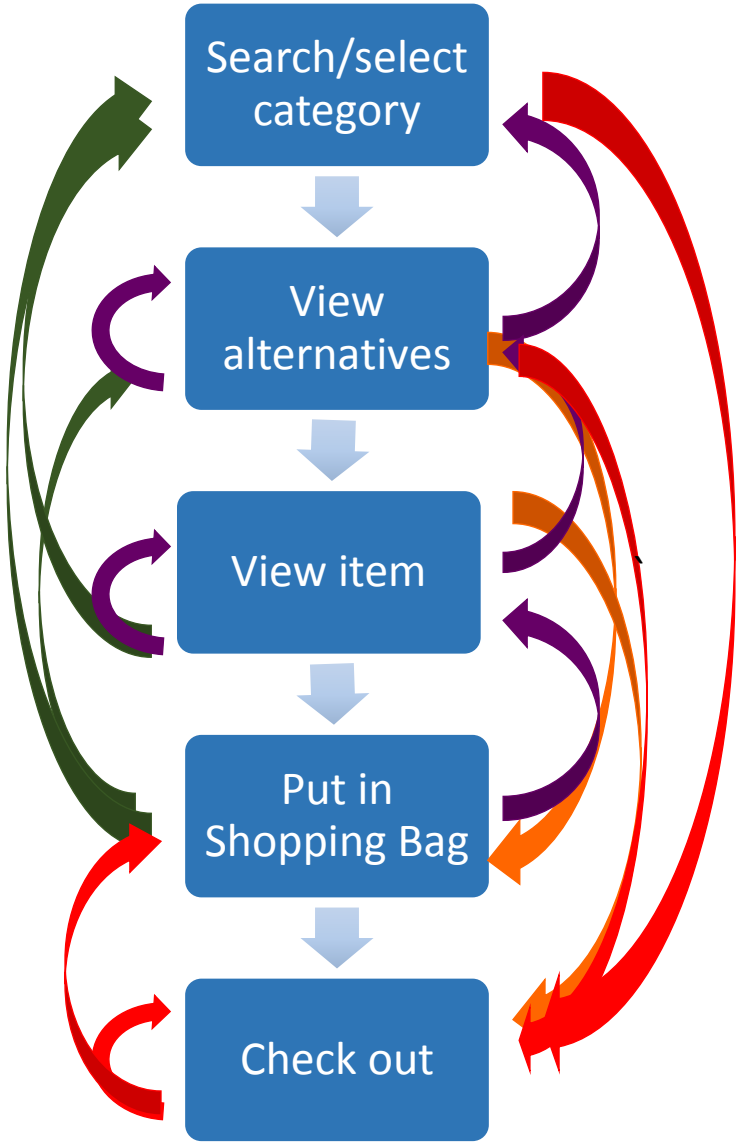


Dialog management and customer management are intimately connected

Architectural Solution

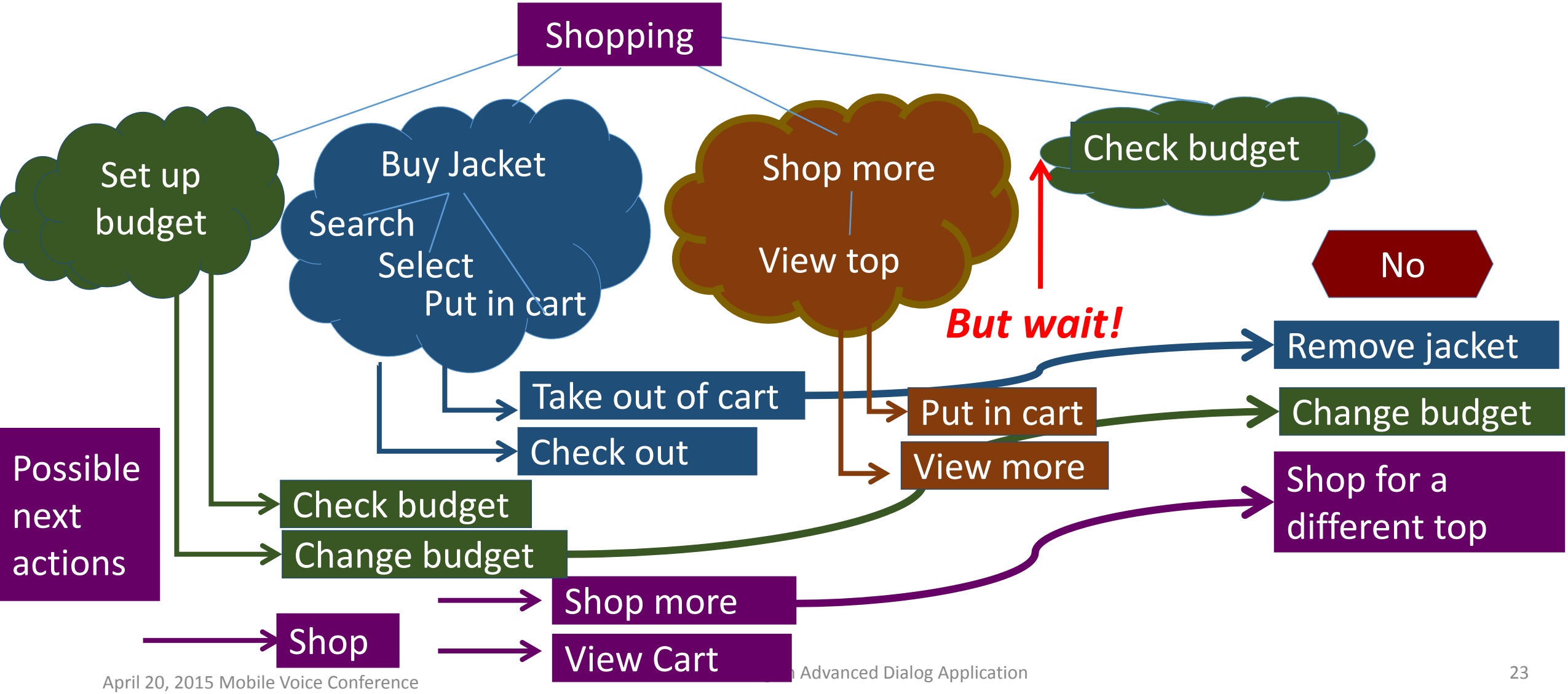
Dialog and Task Flow

- We think of a task as a sequence of steps
- But wait!
  - The task flow allows loops and repetitions
  - Each box can be multiple “turns” in the dialog
  - Can leave a box, then need to return to the same place in that box
- Lots of possible paths, but
  - Not all paths are possible
  - Not all are equally likely
  - Not all meet users goals and business requirements



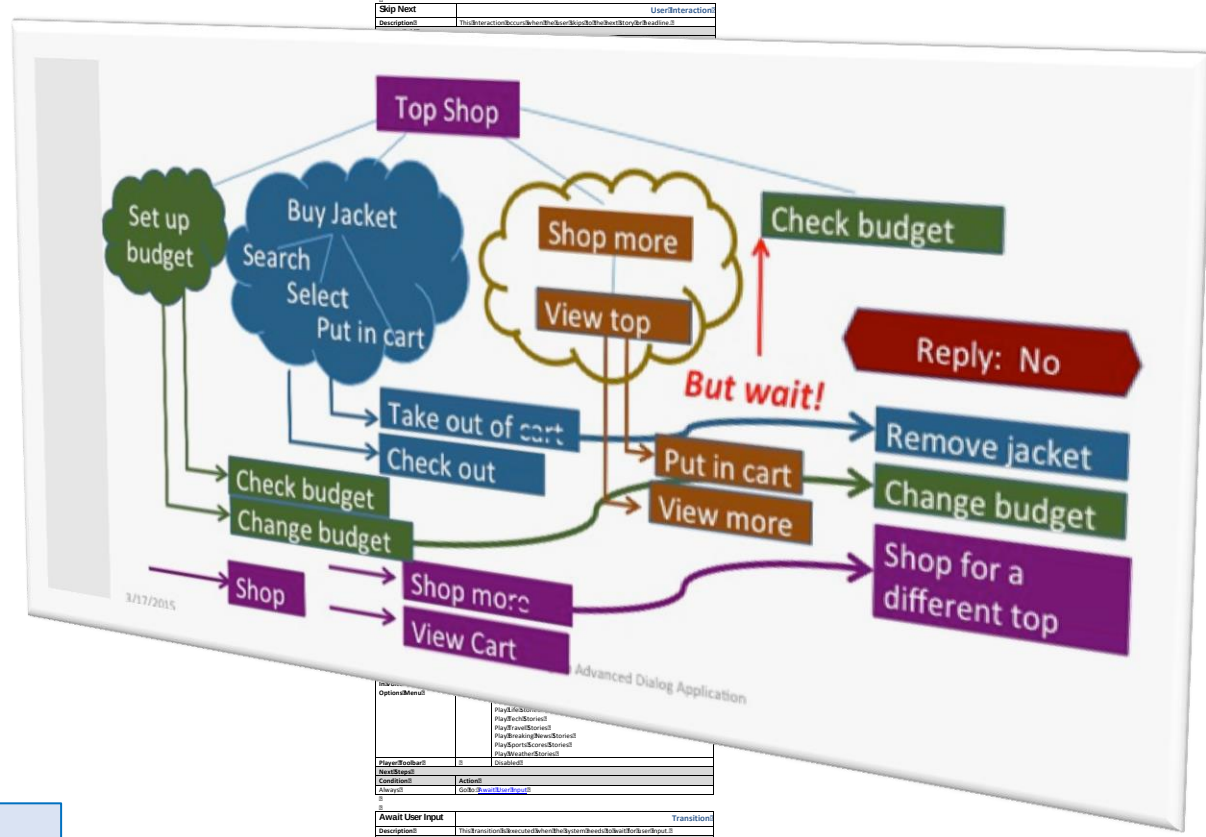
Dialog is doing.

It's not what to say next, but what to do next



Conclusions (and Ramification #3)

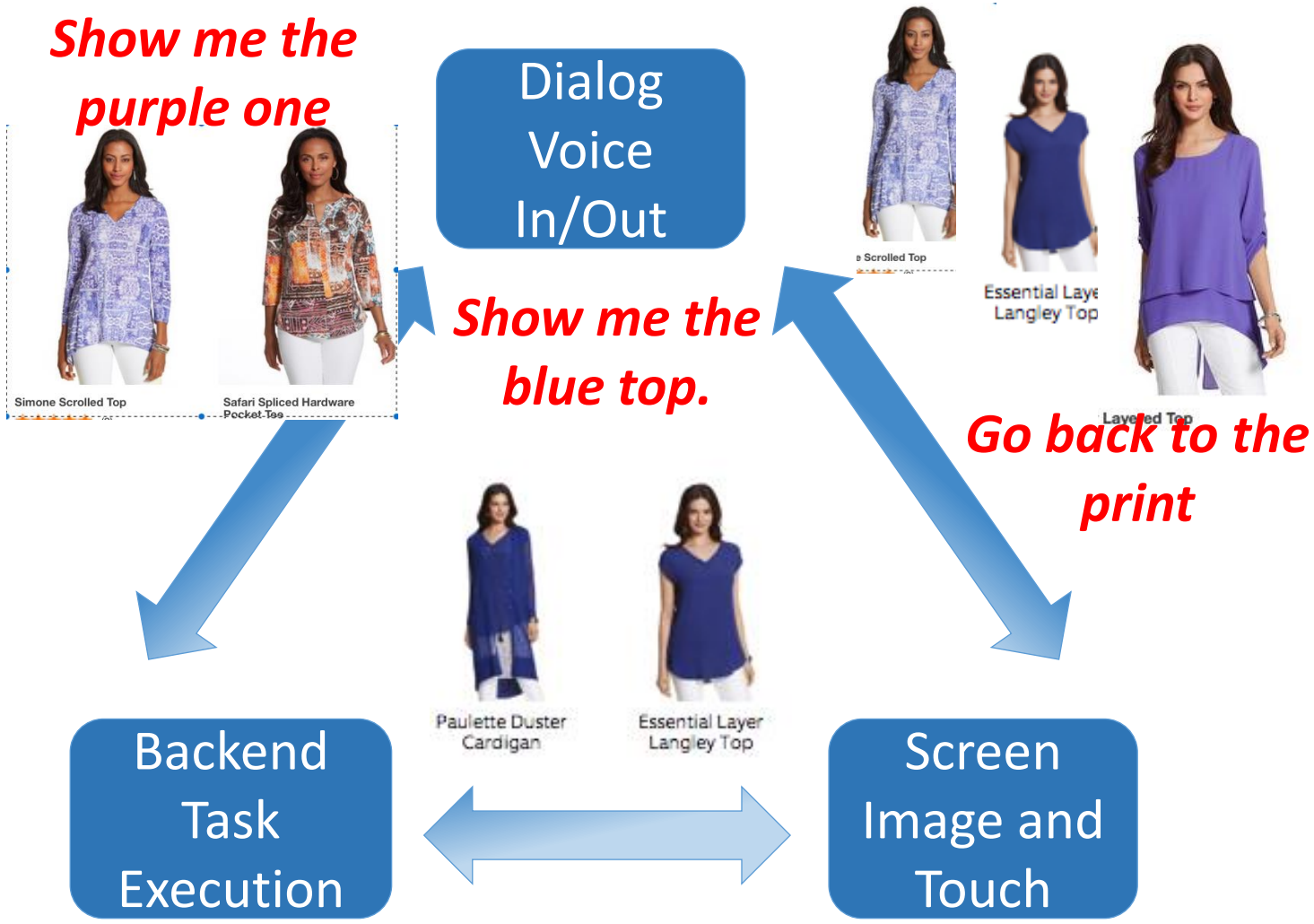
- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go.
  - Task based organization is required for effective dialog management
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected



Dialog design needs to be driven by the "doing" not the sequence of "saying"

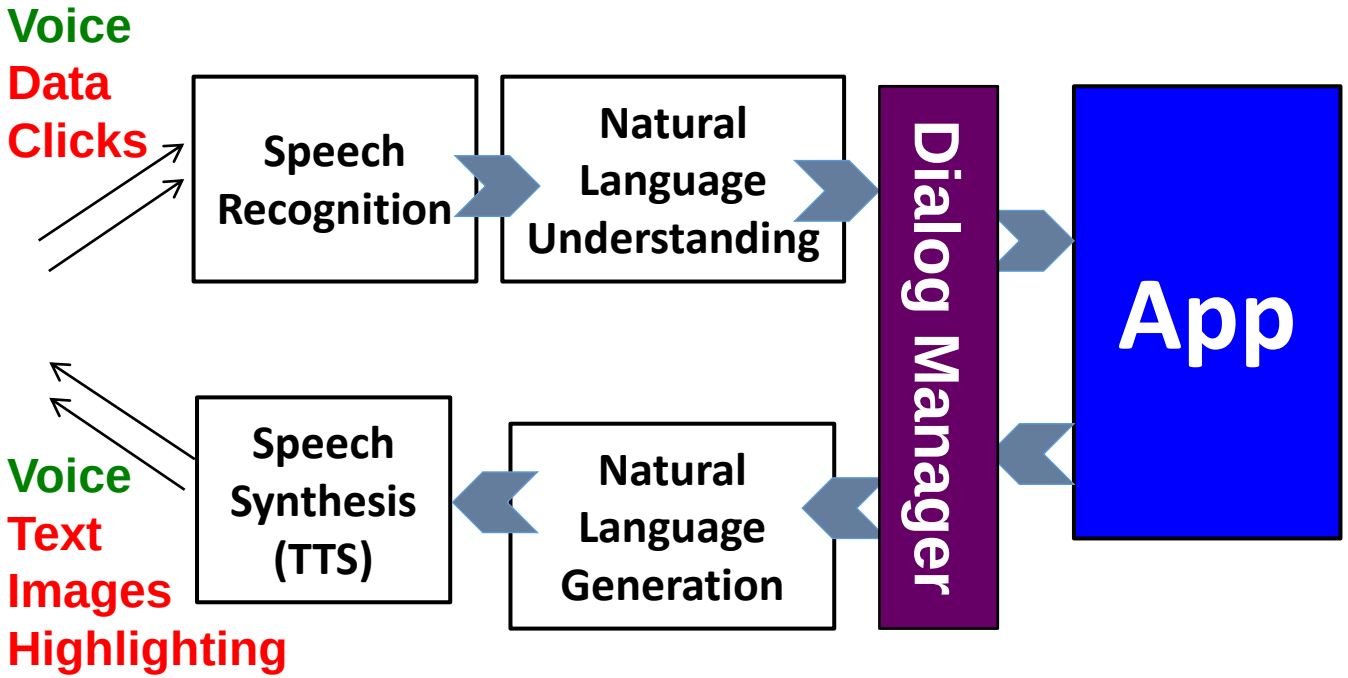
User Interface Design

The representation, both in the dialog and in the underlying data, is critical



- **Show me the blue top**
  - People have different vocabularies for color and objects
  - The product features in the database need to include color and variants
- **Show me the purple one**
  - "One" refers to an object in the context
  - There are only two images on the screen—one has to fit the description
- **Go back to the print**
  - History has to track objects and descriptions
  - Description can change to differentiate objects

Dialog can't be an appendage built by the speech guys



Conclusions (and Ramification #4)

- **Dialog decision points are strategic opportunities**
  - A system response takes the dialog in a particular direction and we can make that direction count
- **Dialog design is modular, not linear**
  - We multitask, our dialog systems need to as well
  - The linear user interface design doc has to go
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management are intimately connected

Dialog cannot be done “After the fact”. The design cannot be separated from implementation

What makes this difficult?

- Silos
- The people that build the app, design the interface, and design the dialog all
  - Work in different organizations
  - Use different vocabularies

System Requirements

One possible approach

What might we need?

System Requirements

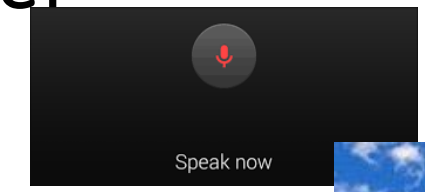
Issues: Latency, MM, Platform, Hardware and Services

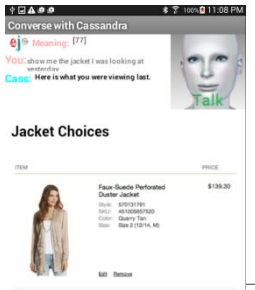
- Input
  - **Speech[ASR]** -> Semantics[NLU] -> Dialog[**C**onversation **M**anagement]
  - **Touch[deictic, gesture]** -> Semantics[contextual] -> Dialog[CM] -> Resp[TTS, Display]
  - Is it **Platform** provided? 3rd party? Stream? Batch? …
- Process
  - **Semantics[Understand]** NLP/NLU, Context, Relevance, Appropriateness
  - **Dialog** Conversation Management: Declarative, OO, platform agnostic, MM integration
  - **Data**
    - Contextual[anaphora, previous action]
    - DB[catalog, cart, budget] select, sort; RESTful API
  - **Scope** Single/Multi store? Relationship span? Linguistic register? …
- Output
  - **Response Modes[TTS, audio, video, images, text, vibration, etc.]** Quality vs. speed/cost
- Performance
  - **Time[sub-second cycle time e.g. “end of speech” to “display of item”]**

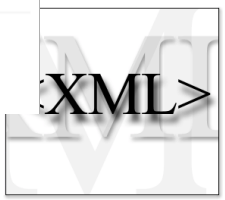
Implementation: A Functional Prototype

This Implementation: **Tech and Tools**

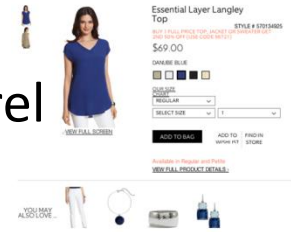
[Things used for this prototype]



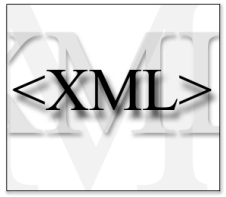


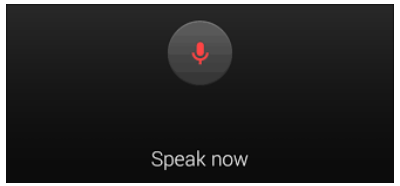


- Platform:
  - Android [phone, tablet]
    - Speech [Google cloud **ASR**, device based TTS]
    - ejTalk Android client, **MIT AI2** based [browser-like UI]
    - **C**onversation **M**anagement in the **cloud engine**
- Data Access (cart, catalog, financial, images)
  - XML database suitable for direct/built-in ejTalk access
- Interaction Scope (store, product type, anything)
  - Single clothing/accessory **online store** selling women's apparel
  - Focus on the "Wait, do I have enough for this?" turn
- Dialog Design Formalism: ejTalk engine
  - Graphic and text-based **Declarative** development tool
  - **Encapsulate** the Speech, NLU, tactile I/O, context, db, etc. elements
  - **Runtime** interpretation/execution of the language



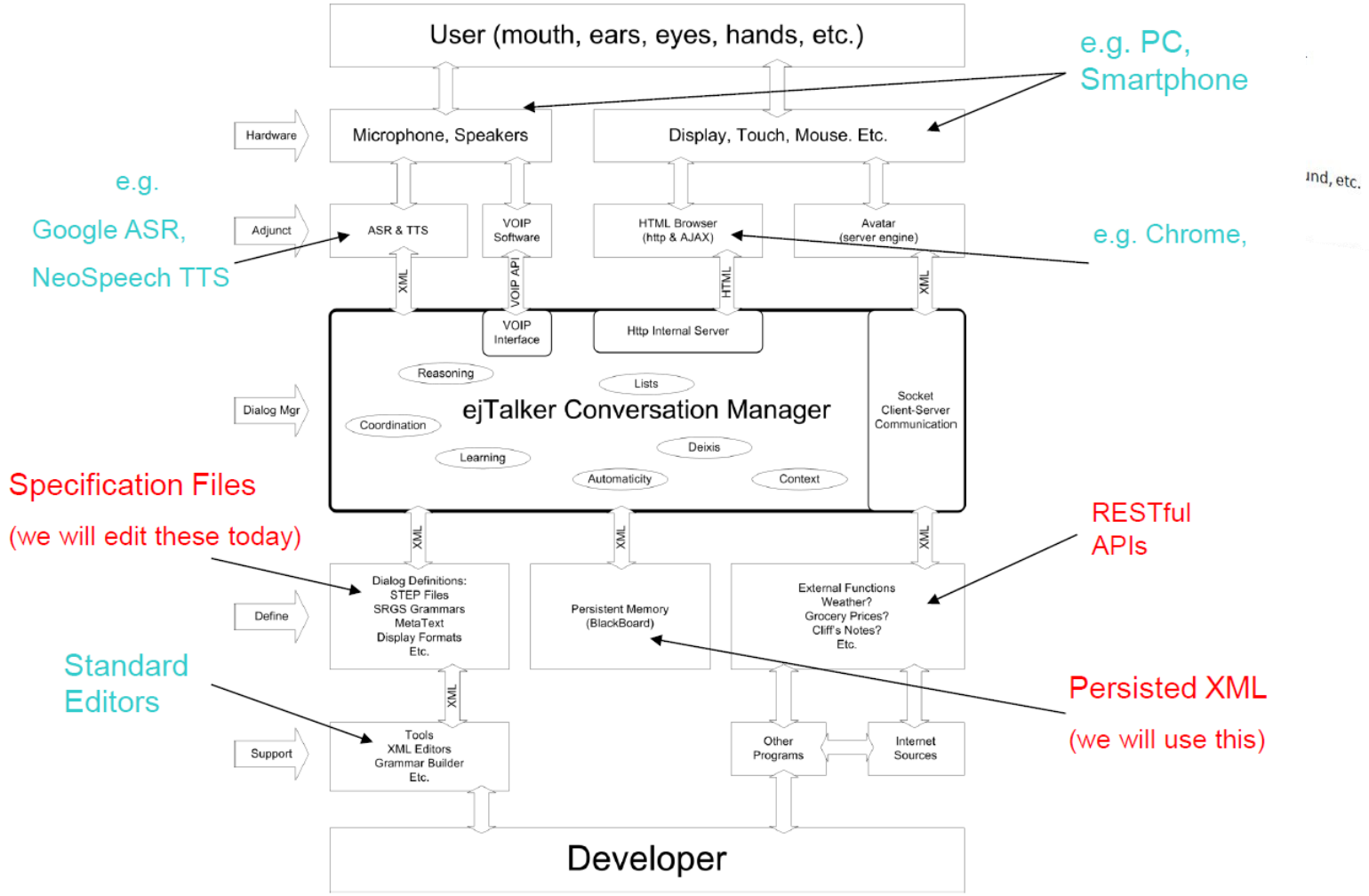
The Client Side



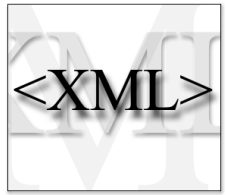


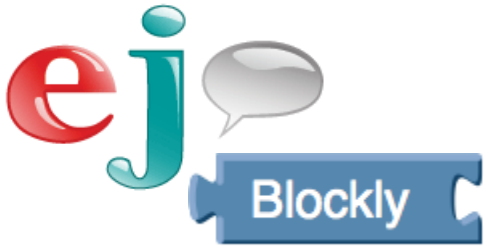
- AI2 [MIT AppInventor 2]
  - Manage input/output technologies
  - Pack inputs & transmit to cloud engine
  - Unpack responses from cloud & present to user
- Detect touch input [via AI2 HTML]
- Process speech input
  - Start/Stop cloud ASR
  - Gather utterance result
- Present responses to TTS, HTML display, sound, etc.

The Cloud Engine

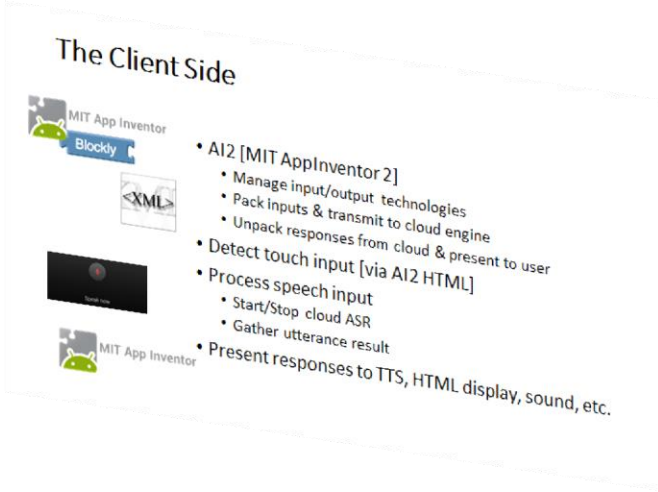


The Cloud Engine

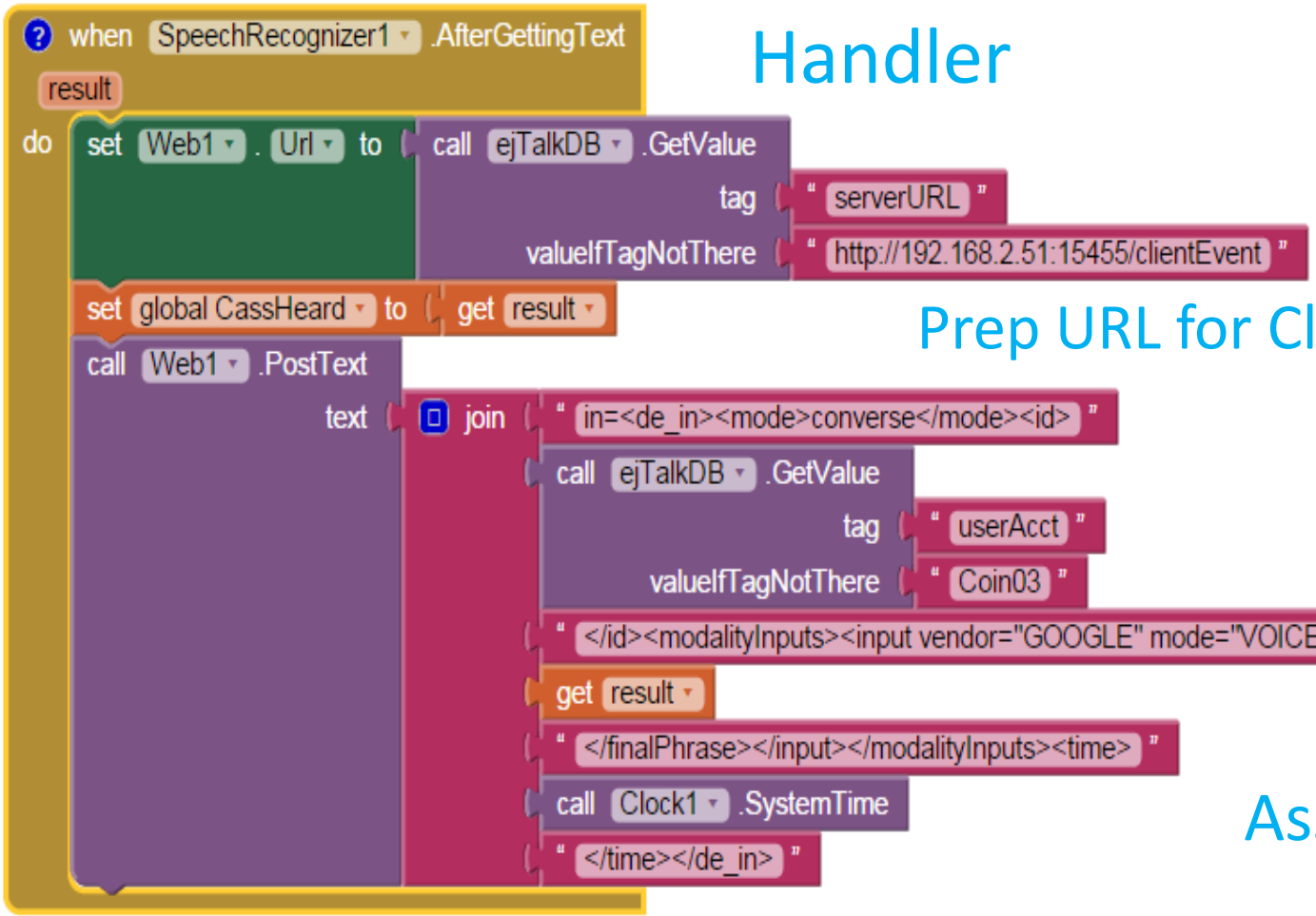




- Modified HTTP **server** supporting ejTalk Engine
  - Communicate with the client side
  - Only XML strings between Server & Client
- ejTalk **Engine** [Execute conversation definition]
  - Process inputs for understanding
    - Words, Semantics, Context
  - Shift the Domain focus
    - Was “looking” now “buying”
  - Remember relevant context
    - Implicit & Explicit
  - Generate response directives
    - Speech, Text, Sound, Video, Avatar, HTML display, etc.



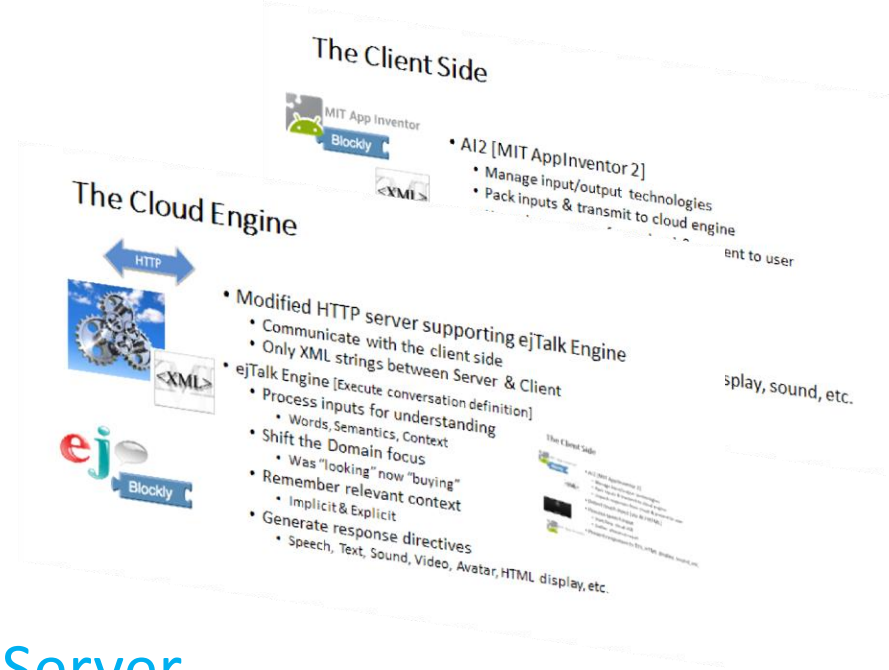
Client: Get ASR Result [AI2 code]



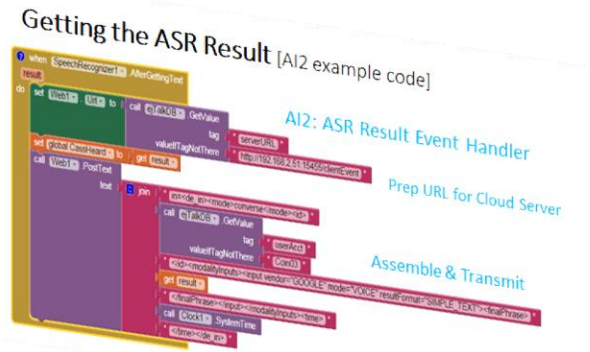
AI2: ASR Event Handler

Prep URL for Cloud Server

Assemble & Transmit



Cloud: Extract Concepts[ejTalk code]



ejTalk Browser Converse ...

Session: ReLogon | End Sess

Debug: Transcript | Concepts | Editor

Human: But wait, do I have enough for this

Understood: [12]

Said: Not if you buy the suede jacke

Concept Name	finalScore	span
ejThirdPerson	1.505	this
ejDO	1.380	do
ejEnoughFor	1.255	have enough fo
ejInteruption	1.255	wait
ejConjunctio	1.079	for
ejFirstPerson	0.602	i
ejPrep	0.000	for
ejPronoun	0.000	this
ejBad	-0.301	
ejNumberCardinal_0_10	-0.301	for
ejLogoff	-0.313	i have
ejExpertiseQuestion	-0.500	do
ejRequest	-0.522	do
ejDislike	-0.931	do
ejWorkFor	-1.045	do
ejCorrection	-1.045	do
ejFamiliarWith	-1.142	do
ejNameDeclare	-1.221	this
ejAskTimeQuestion	-1.957	do

Engine Extracts Concepts

Blockly "Concepts"

Cassandra's **Text**Editor

ejTalk

User Name: coin10

File Type: steps | concept | grammar | displayFormat | metaText

Files: application | online_Shopping |

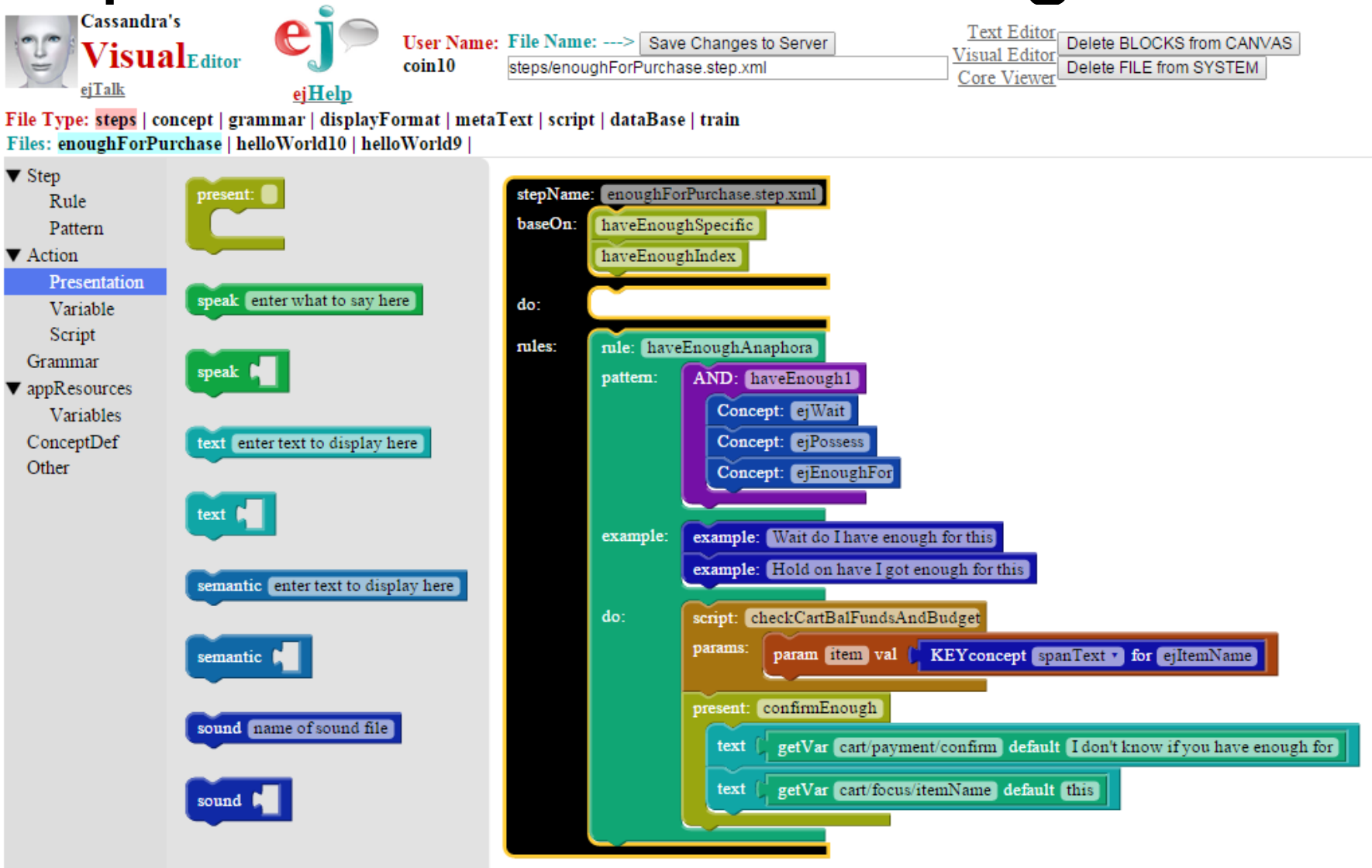
```
1  <concept name="online_Shopping.concept.xml">
2      <class name="ejEnoughFor">
3          <expr val="have enough for"/>
4          <expr val="got enough for"/>
5          <expr val="have enough to"/>
6          <expr val="got enough to"/>
7      </class>
8      <class ...</class>
14     <class ...</class>
20 </concept>
21
```

XML for Engine

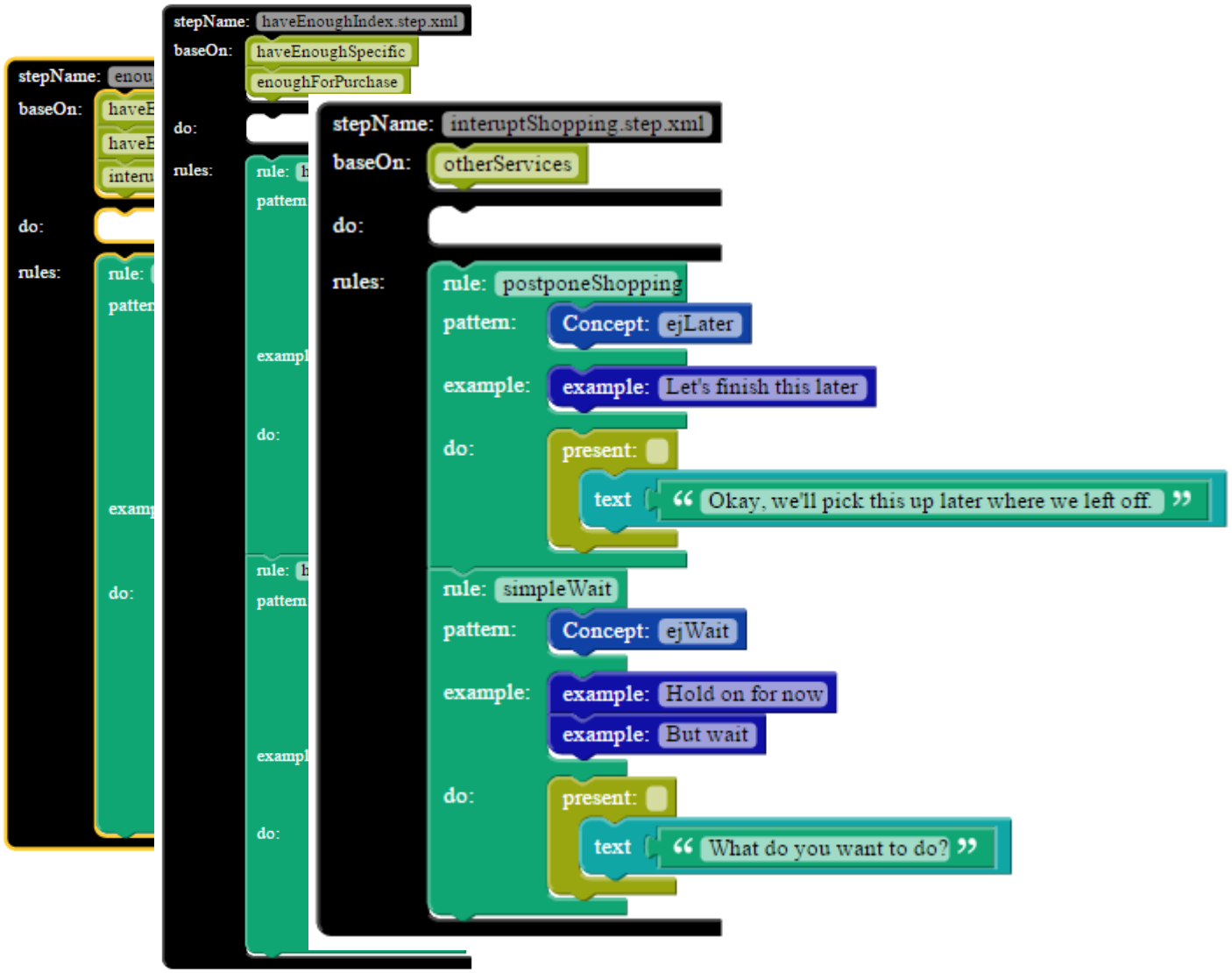
See Notes

Very Quick Look at Blockly

Capture the “Do I have enough” event

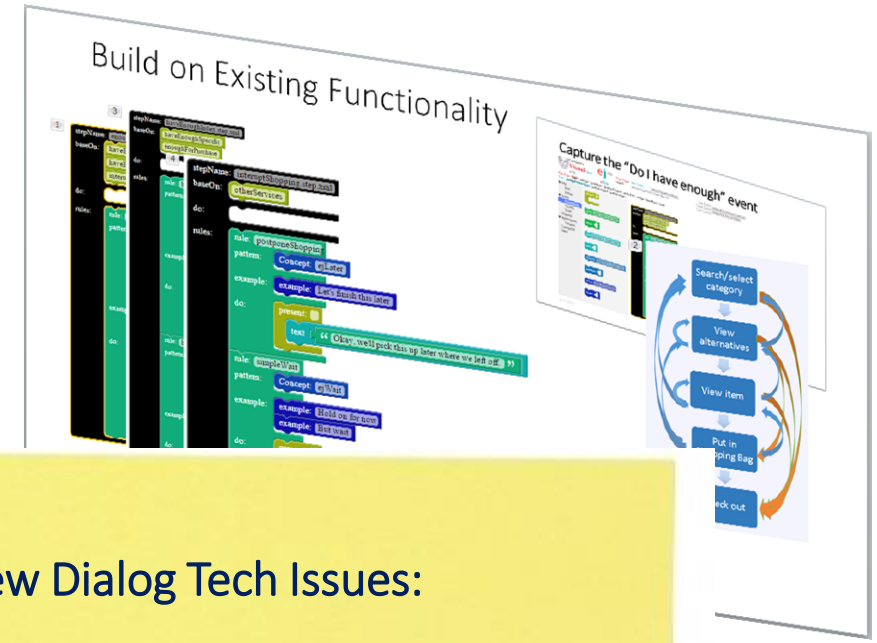


Build on Existing Functionality



The Challenge: How to…

- How to define functionality in clear and open terms
  - **Modular** components
  - **Interoperable** modalities
  - **Infinite** pathways
- Maximize reuse
  - M**inimize** development EFFORT
  - M**aximize** CONSISTENCY
- Leverage automaticity
  - **Requires** TRUST
  - **Necessary** for naturalness



New Dialog Tech Issues:

- Naturalness is **non**-linear (only predictable short-term)
- **Multi**-task dialog -> IVR
  Multi-thread -> **Single**-thread
- Business wants **MAX** control
- But, Automaticity means:
  Delegation of control

Conclusions (and Ramification #5)

- **Dialog decision points are strategic opportunities**
  - We are building an *encounter* **not** a *call flow*
- **Dialog design is modular, not linear**
  - Multitasking dialogs will need new formalisms
  - Concise flow charts of the experience are now longer possible
- **Requirements cannot be separated from design**
  - The words, the visuals, the data representations, and dialog management (response time, accuracy, adaptation) will be tethered to the available technology.
  - Tech choices propagate back to the business case

- The technology that a user experiences as well as the tools used to build the experience both constrain the behavior possible.
- Defining this new kind of experience will inevitably lead to new paradigms and formalisms.

What makes this difficult?

- New tools are needed
- Tighter coupling of the "idea to product" cycle
- The product will be more autonomous
- Subtle but different
- Revolutionary not evolutionary

Final Comments

Conclusions:

- Decision points are **strategic opportunities**
- Dialog design is **modular, not linear**
- Requirements **cannot be separated** from design

- **Agile development extends across Business Requirements and Architecture.**
- **Dialog management and customer management are intimately connected**
- **Dialog design needs to address “doing”, not just “saying”**
- **Dialog design cannot be separated from implementation**
- **Technology and Tools used to build experience constrain the possible behavior**
- **This new kind of experience will lead to new paradigms and formalisms**
- **Today there is no manageable formalism to define large, context-driven conversations**