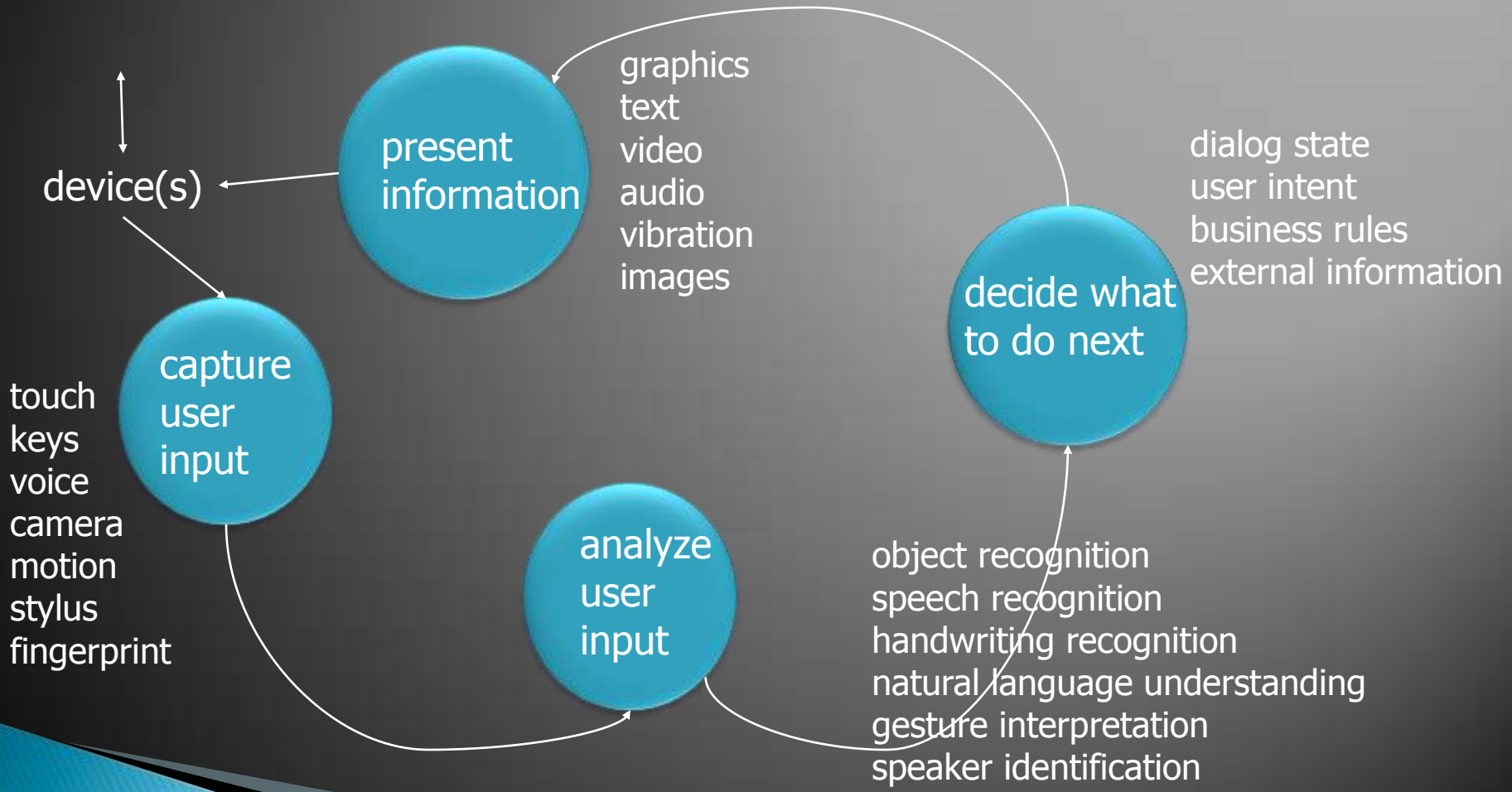
Distributed Multimodality in the W3C Multimodal Architecture

Deborah Dahl
Conversational Technologies
Chair, W3C Multimodal Interaction Working Group

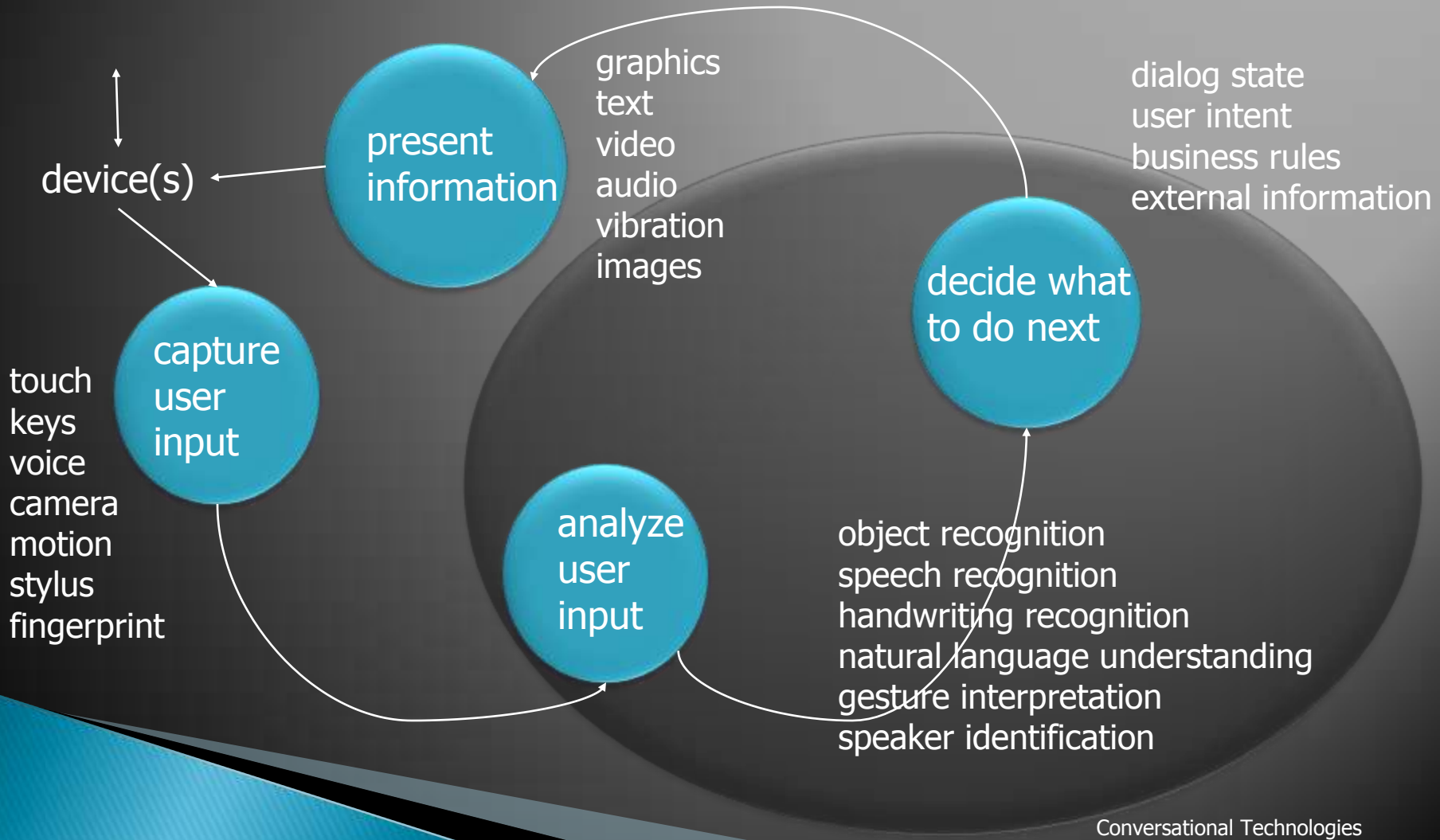
Making Multimodality Happen

- What does an application do?
- present information to user
- capture user input
- analyze user intent
- decide on next steps

Components of a Multimodal Application



Components of a Multimodal Application



Where are Components?

- Must be local
  - capture – mouse, touchscreen, voice
  - presentation – video, audio, images
- Could be distributed
  - analysis
  - decision-making
  - infrequently used
  - resource-intensive
  - need maintenance
  - external information

Candidates for Distributed Components

- resource-intensive
- require maintenance and updates
- infrequently used
- require access to external resources

Candidates for local components

- interact directly with the user
- may be used off-line

Why distributed model?

- support for thin clients with limited processing resources
- fewer client requirements – just sensors, presenters and connectivity – make application more device-independent

W3C Multimodal Architecture

- Modality Components encapsulate modality functions
- Interaction Manager coordinates interaction among components to perform an application
- Communication is based on Life Cycle Events with EMMA representing user inputs

How does the MMI Architecture support distribution?

- Standards-based modality components can be located locally or remotely
- Anyone can develop components and make them available as a general resource on the web
  - Speech recognition, text to speech or natural language processing for a language with relatively few speakers
  - Developers can be assured that their components will work as part of others' systems
- Communication via standard protocols, such as HTTP

Example: startRequest event

- Sent by the Interaction Manager to start a component running
- The modality component is just referenced by a URI, so it can be anywhere
- The markup that it will run is also referenced by a URI

startRequest

```
<mmi:mmi xmlns:mmi="http://www.w3.org/2008/04/mmi-arch"
  version="1.0">
  <mmi:startRequest source="IM-URI" target="MC-URI"
  context="URI-1" requestID="request-1">
  <mmi:contentURL href="someContentURI" max-age=""
  fetchtimeout="1s"/>
</mmi:startRequest>
</mmi:mmi>
```

Example: doneNotification

- Sent by a component with EMMA results when it's finished processing

doneNotification with EMMA

```
<mmi:mmi xmlns:mmi="http://www.w3.org/2008/04/mmi-arch"
version="1.0" xmlns:emma="http://www.w3.org/2003/04/emma">
    <mmi:doneNotification source="someURI"
    target="someOtherURI" context="someURI" status="success" requestID="request-1" >
            <mmi:data>
                    <emma:emma version="1.0">
                            <emma:interpretation id="int1"
                              emma:medium="acoustic"
                              emma:confidence=".75"
                              emma:mode="voice"
                                      emma:tokens="flights from boston to denver">
                                <origin>Boston</origin>
                                <destination>Denver</destination>
                            </emma:interpretation>
                    </emma:emma>
            </mmi:data>
    </mmi:doneNotification>
</mmi:mmi>
```

Wheel reinvention in speech API's

Many speech API's exist or have been proposed, some standard and some proprietary

- Sun JSAPI 1.0 and 2.0
- IETF MRCP
- W3C VoiceXML
- Microsoft SAPI
- ATT Speech Mashup
- X+V
- SALT
- MIT WAMI
- Chant SpeechKit
- Nuance Dragon SDK

Commonalities

- Setting properties (timeouts, confidence thresholds, grammar, etc.)
- Starting and stopping
- Getting results
- Communicating with calling programs

MMI Architecture addresses all of these except setting properties, because that's modality-specific

Summary

- Distributed modality processing can simplify applications
  - support for thin clients
  - maintenance of grammars and UI is simplified
  - processing resources are more available in the cloud
- The MMI Architecture supports distribution by
  - providing a standard way to reference remote or local modality resources
  - Standard API's encourage developers to make a variety of modality resources available

More Information

- W3C Multimodal Architecture
- http://www.w3.org/TR/mmi-arch/
- EMMA