Overview: Multimodal Architecture and Interfaces

Deborah Dahl

W3C Workshop on Multimodal Architecture and Interfaces

16-17 November 2007

W3C/Keio, Fujisawa, Japan

MMI Architecture

A loosely-coupled, event-based architecture for integrating multiple modalities into applications

- Modality components can be local or distributed
- All communication is event-based
- Based on a set of standard life-cycle events
- Components can also expose other events as required
- Encapsulation protects component data
- Encapsulation enhances extensibility to new modalities
- Represents user inputs in a standard way (EMMA)

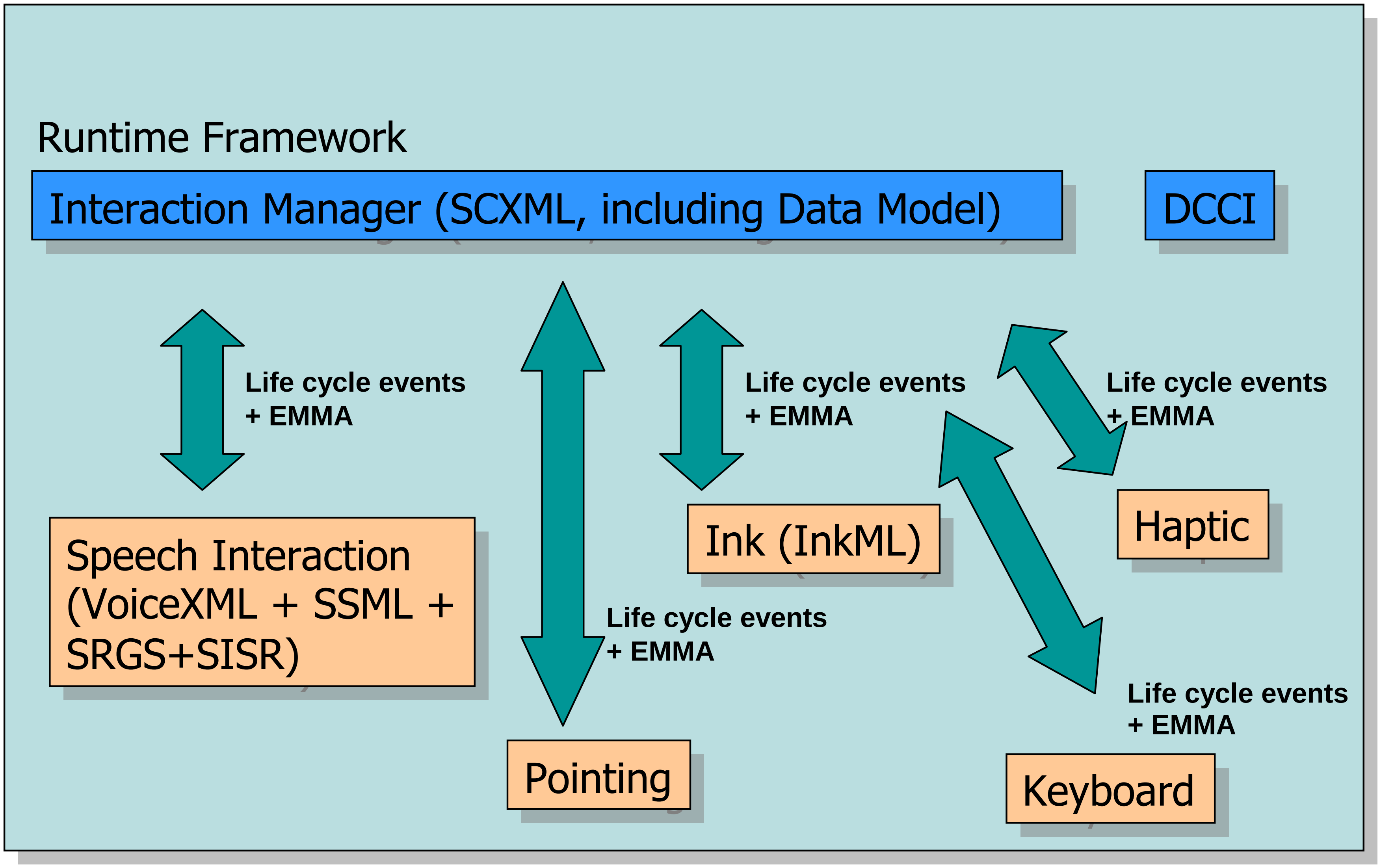
Constituents

- Defines five basic components of an MMI system
  - Runtime Framework or Browser: initializes application and runs markup
  - Interaction Manager: coordinates modality components and provides application flow
  - Modality Components: provide modality capabilities such as speech, pen, keyboard, mouse
  - Data Model: handles shared data (part of the Interaction Manager)
  - DCCI (Delivery Context Client Interfaces): device properties and user preferences
- Most recent Working Draft, 11 December, 2006 http://www.w3.org/TR/mmi-arch/

MMI Architecture Principles

- Runtime Framework communicates with Modality Components through asynchronous events
- Modality Components don’t communicate directly with each other but indirectly through the Runtime Framework
- Components must implement basic life cycle events, may expose others
- Modality components can be nested (e.g. a Voice Dialog component like a VoiceXML <form>)
- Components need not be markup-based
- EMMA communicates users’ inputs to IM

Instantiated MMI Architecture



Life Cycle Events

Event	From	To	Purpose
NewContextRequest	Modality	Runtime Framework	Request new context
NewContextResponse	Runtime Framework	Modality	Send new context id
Prepare	Runtime Framework	Modality	Pre-load markup
PrepareResponse	Modality	Runtime Framework	Acknowledge Prepare
Start	Runtime Framework	Modality	Run markup
StartResponse	Modality	Runtime Framework	Acknowledge Start
Done	Modality	Runtime Framework	Finished running
Cancel	Runtime Framework	Modality	Stop processing
CancelResponse	Modality	Runtime Framework	Acknowledge Cancel
Pause	Runtime Framework	Modality	Suspend processing
PauseResponse	Modality	Runtime Framework	Acknowledge Prepare
Resume	Runtime Framework	Modality	Resume processing
ResumeResponse	Modality	Runtime Framework	Acknowledge Resume
Data	either	either	Send data values
ClearContext	Runtime Framework	Modality	Deactivate context
StatusRequest	Runtime Framework	Modality	Check status of MC
StatusResponse	Modality	Runtime Framework	Report status

Summary

- MMI Architecture provides a general, clean interface to a wide range of modality components
- EMMA provides a standard and general way of representing user inputs
- Very easy to integrate new modalities
- Loose coupling and lack of access to internal modality data improves security