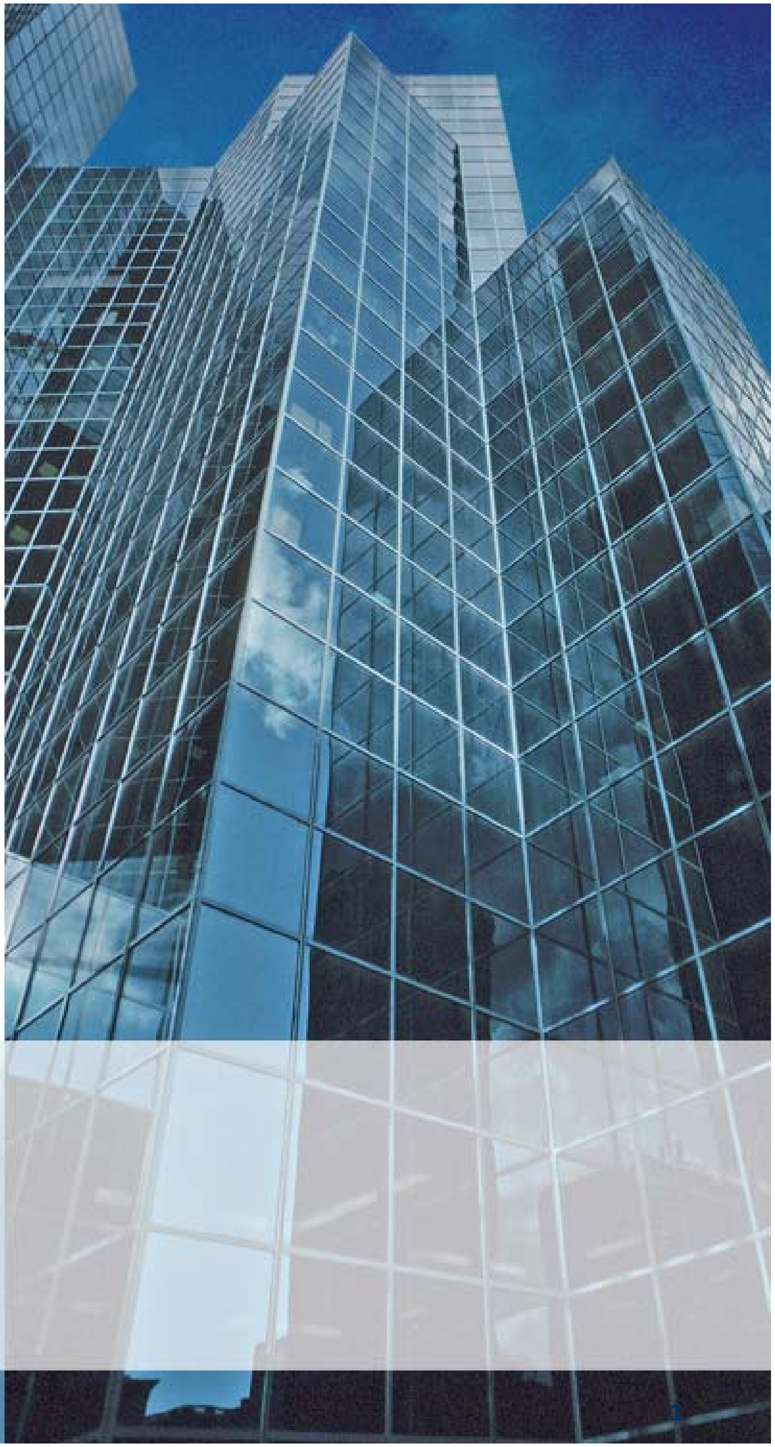
Multimodal Framework Proposal

Skip Cave
Chief Scientist,
Intervoice Inc.



Workshop Goals

- Identify & prioritize requirements for changes, extensions, and additions to the MMI architecture to better support Speech, GUI, Ink, and other Modality Components

Agenda

intervoice

- Current Lifecycle Events
- Rationale for New Functionality
- Paradigm-Breaking Examples/Use Cases
- Elucidating Questions on Framework Limitations
- Proposed New Lifecycle Interaction Modes/Events
  - Basic
  - Modify
  - Parallel
- Example Diagrams
- New Functionality Objectives
- Proposals
- Issues

Current Life Cycle Events

- New Context Request
  - MC -> RF
- Prepare
  - RF -> MC
- **Start**
  - **RF -> MC**
- **Done**
  - **MC -> RF**
- Cancel
  - RF -> MC
- Pause
  - RF -> MC
- Resume
  - RF -> MC
- Data
  - RF -> MC or MC -> RF
- Clear Context
  - RF -> MC
- Status Request
  - RF -> MC

Rationale For New Functionality

- What if the application developer wants to modify a specific executing MC script without stopping the execution of that current script?

- What if the application developer wants to initiate a concurrent operation to a specific Modality Component? The concurrent operation in the MC would share the same User, I/O devices, Media streams etc., running in parallel with the initial MC process.

Paradigm-Breaking Examples – Use Cases

- Modify
  - Volume Up (Touch Screen Button)
  - Change Audio Playback Speed (Keyboard)
  - Bold Text (Voice Command)
  - Pause or change volume of video in one window of multi-window screen (Voice Command - “Louder on video one”)
- Parallel
  - Oral Test
    - Concurrent Audio Recordings (System & User) (Graphical PDA buttons)
  - Digital Music Store
    - Concurrent Audio Playback (Annotation) (Graphical PDA buttons)
  - Multiple-concurrent-window displays
  - Single Screen/Multi-user GUI Interactions (Multiplayer Games)

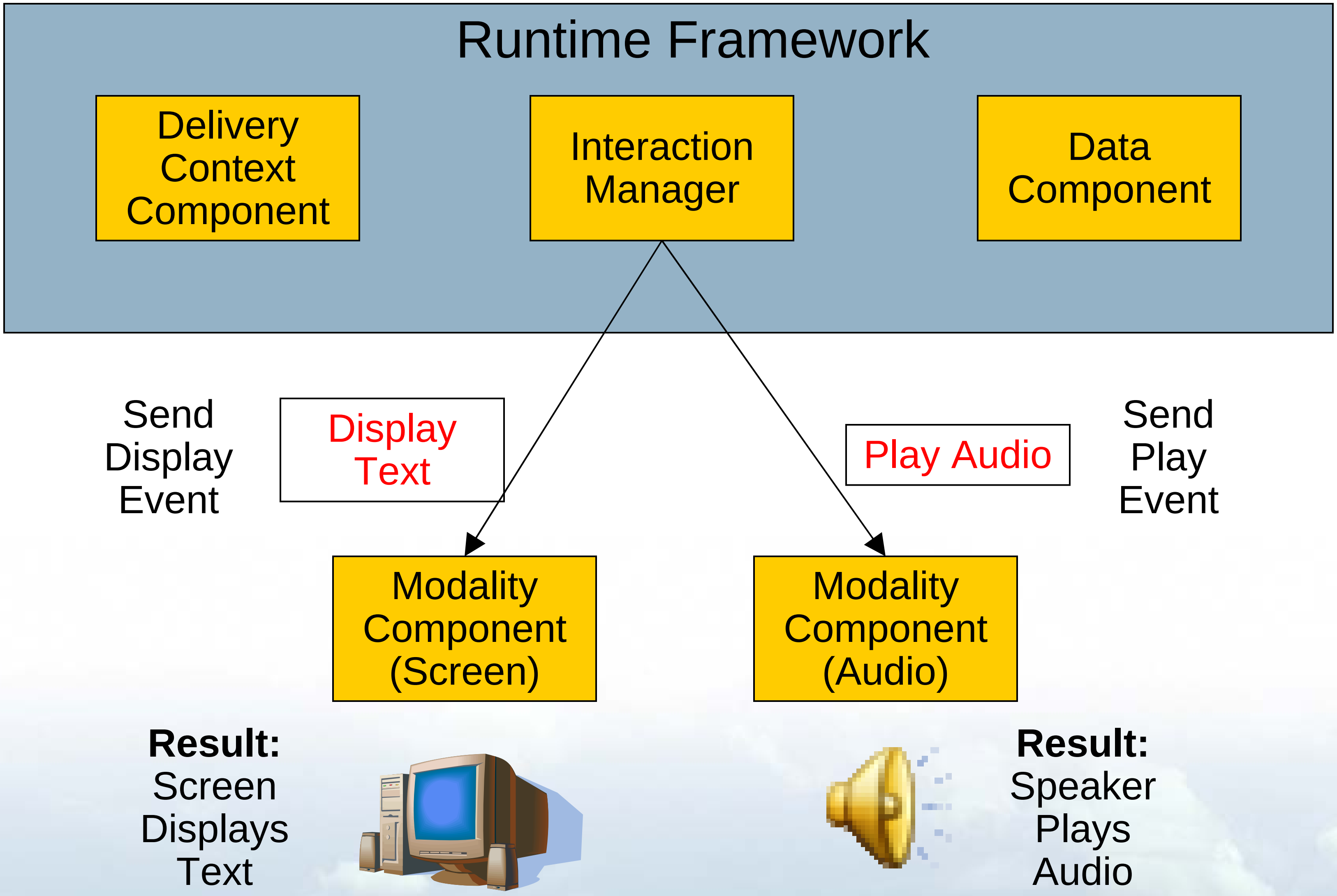
Questions

- How can the Interaction Manager indicate a modification to an ongoing Modality Component interaction or script without stopping and re-starting the MC?
- How can the Interaction Manager initiate a parallel process within a MC without stopping and re-starting the current script process within the MC? A parallel MC process would utilize the same MC, and user, as well as the same media streams and I/O devices.
- How does the IM identify the specific parallel process it is addressing, when sending events to an MC?

Possible New Lifecycle Interaction Modes

- Standard Event
  - Invokes markup for MC execution, either via URL or inline
- Modify Event (Data Event?)
  - Invokes markup for MC execution which will modify the current script execution, either via URL or inline.
  - Will not stop the execution of current MC user interaction as modifications are made
- Parallel Event (Concurrent Start?)
  - Invokes markup for MC execution which will cause parallel operations within the target MC, either via URL or inline. Same user, same media streams, same I/O devices
  - Will not stop the execution of current MC user interaction

Basic Interaction Mode – Output Example



Runtime Framework

Delivery Context Component

Interaction Manager

Data Component

Send Display Event

Display Text

Play Audio

Send Play Event

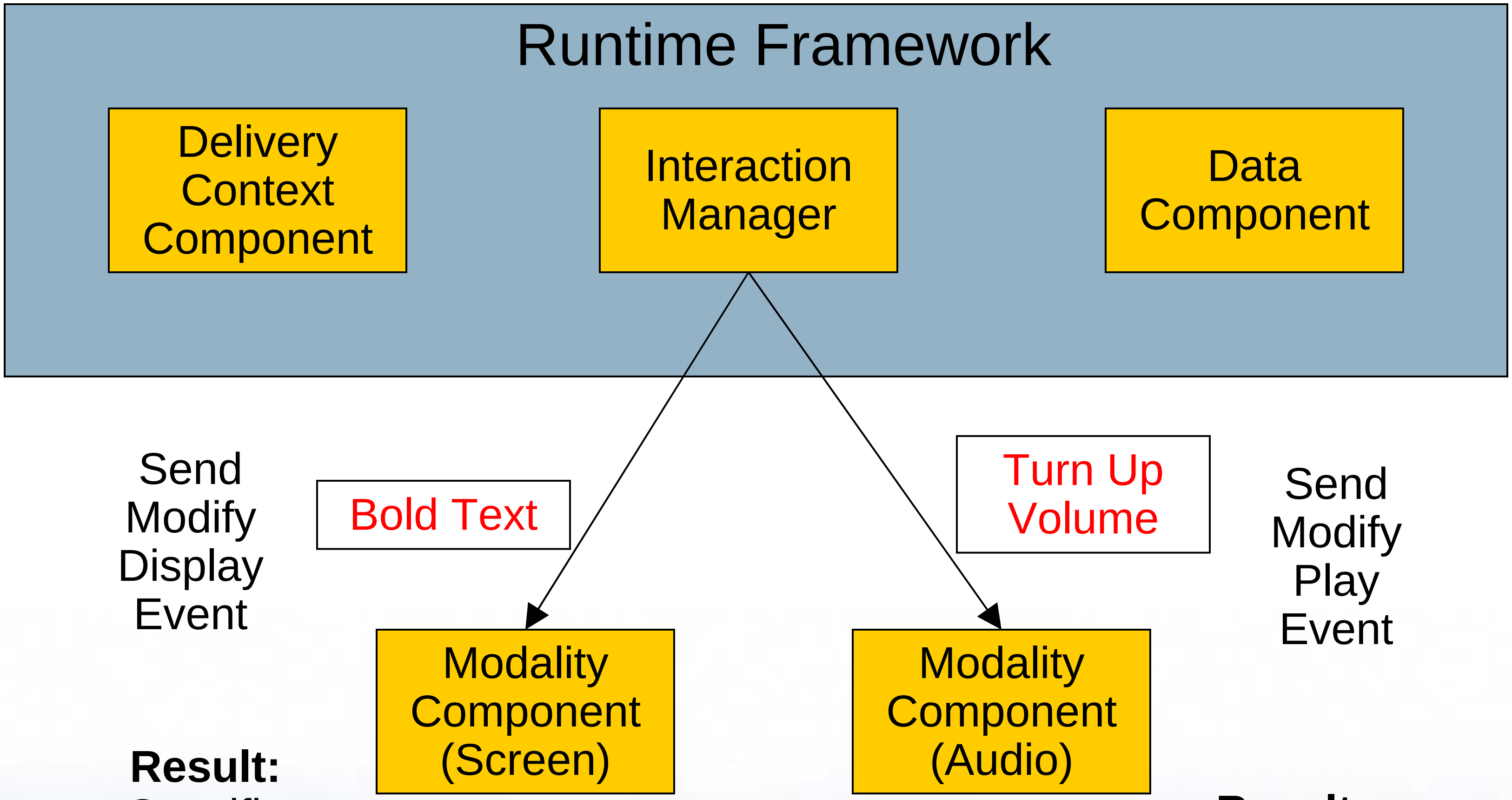
Modality Component (Screen)

Modality Component (Audio)

Result: Screen Displays Text

Result: Speaker Plays Audio

Modify Interaction Mode – Output Example



Runtime Framework

Delivery Context Component

Interaction Manager

Data Component

Send Modify Display Event

Bold Text

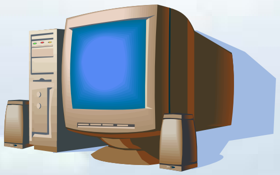
Turn Up Volume

Send Modify Play Event

Modality Component (Screen)

Modality Component (Audio)

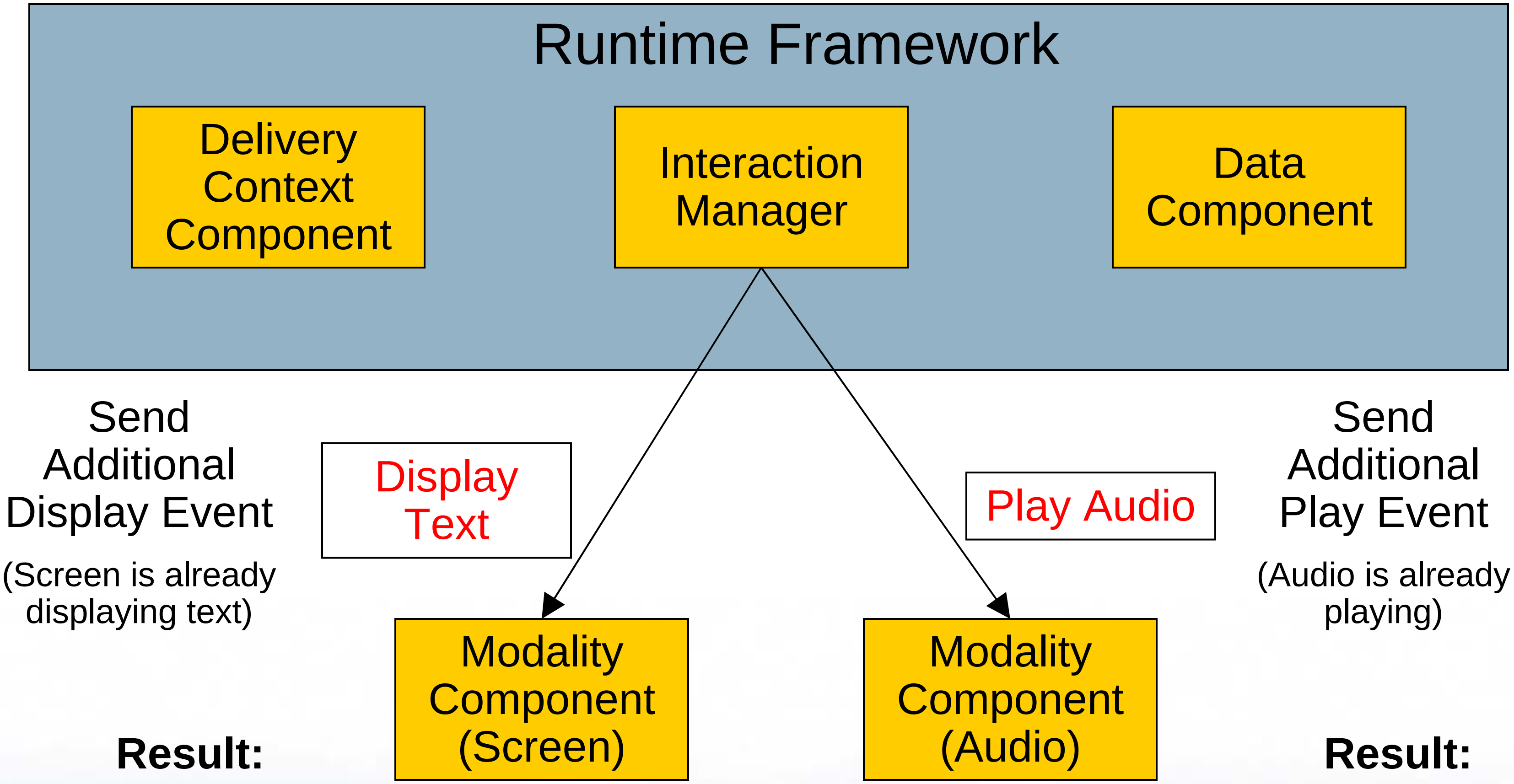
Result: Specific Text on Screen is Made Bold



Result: Audio Volume is Raised

Parallel Interaction Mode – Output Example

intervoice



Runtime Framework

Delivery Context Component

Interaction Manager

Data Component

Display Text

Play Audio

Modality Component (Screen)

Modality Component (Audio)

Send Additional Display Event

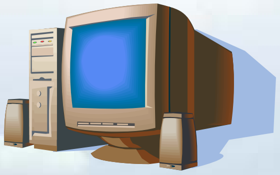
(Screen is already displaying text)

Send Additional Play Event

(Audio is already playing)

Result:

Display Additional Text on Screen in Another Window



Result:

Second audio stream is mixed with original, and both streams are heard from speaker

Objectives of Proposal

- Make simple modifications and parallel invocations to MCs easy for developers to implement
- Allow embedded markup in events for immediate execution
- Avoid requiring developers to write Asynchronous event handlers on Modality Components
- Allow Granular Operations within MCs controlled from IM

Proposal

- Define a “Modify” LC command for initiating modifications to existing running processes on an MC.
- Allow multiple Start commands to be issued before the first “Done” command is received from an MC.
  - Start commands issued before a Done terminating the initial “Start”, will cause the target MC to start a second parallel instance sharing the same media streams and I/O devices.
  - Additional Start commands will cause additional “done” commands to be returned, one for each Start.
- Pause-Resume-Modify and other LC commands must be addressed to a specific Start-Done process, and will operate within that specific start-done scope

Issues

- How to identify specific start-done processes/command pairs?
- How to send suspend-resume-modify and other lifecycle events to a specific start-done process?
- How to handle the sharing of media streams with concurrent operations. The intuitive approach is to automatically replicate input, and sum output.
  - Modern OS functionality
    - Audio Output: DVD player and MP3 player
    - Audio Input: Speech Reco App (Transcription) and Podcast recording

Thank You!

Questions?

Skip Cave
Chief Scientist
Intervoice Inc.
skip.cave@intervoice.com

intervoice