# Position Statement for Multimodal Workshop

**Stéphane H. Maes**
**stephane.maes@oracle.com**
**Director of Architecture**
**Mobile, Voice and Wireless and Advanced Technologies**
**Oracle Corporation**

## 1. Personal Interest

Stéphane has several years of experience and history with multimodal interactions. At IBM< He pioneered numerous activities, including speech and speaker recognition technology, voice browsing (SpeechML), embedded speech recognition, multi-channel and device-independent authoring and servers (XForms, XSLT, …), multimodal computing and conversational computing and their use in mobile or car environments.

In his current role at Oracle Corporation, Stéphane oversees strategy, research and development of tools and middleware (declarative and imperative, JSP and JSF, portlets) to support multi-channel applications, multimodal and multi-device applications as well as applications optimized for specific interaction channels including: J2ME, MS .NET Compact Framework, SALT, device-specific XHTML / Ecmascript MP, WML, XForms/XHTML, VoiceXML (browser specific and browser-independent), JSPs, JSFs and portlets.

During the years, Stéphane has created, chaired, actively participated, often as editor, or monitored related standard activities including:

- SVAP/SRAPI,
- W3C (XForms, XSLT, XHTML, MMI, Voice, DI, WS),
- OASIS,
- WS-I
- IETF (SPEECHSC, DSR),
- ETSI (DSR),
- 3GPP (SES, multimodal),
- ITU (DSR),
- WAP / OMA (Multimodal and multi-device enabler).

In particular, Stéphane is the champion for the multimodal and multi-device enabler activity at OMA. In addition, he is editor of the OMA requirement and architecture documents for multimodal and multi-device.

In the rest of this position statement, we present our view of the work done at OMA. It reflects the position of Oracle Corporation and it should not be construed as a liaison from OMA. OMA specifications may ultimately differ and address other objectives or rely on different design points than the considerations presented below.

## 2. Multi-modal and Multi-device support by Mobile Infrastructure

The OMA work focuses on enabling support by the OMA Service Environment (OSE) architecture of multimodal interactions. It strives to support:

- The different authoring approaches currently proposed in the industry including the W3C MMI work
- The different implementation and deployments models for multimodal browsers including sequential switches, thin and fat clients with local or distributed conversational engines.

## 2.1 Multimodal and Multi-device Execution Model

This section describes a basic flow for multimodal and multi-device interactions, believed to support the different use cases and requirements identified in the OMA multimodal and multi-device Requirement Document. It supports the fundamental execution mode of multimodal and multi-device applications.

It is envisaged that OMA multimodal and multi-device services satisfy this execution model. It is of key interest that the industry as a whole endorses such a model.

In this document, the term user agent is used loosely to designate the component that renders the presentation data into physical effects that can be perceived and interacted with by the user. For a given modality this may be a separate browser or platform or one or multiple components internal to a browser or platform.

The fundamental execution model of multimodal and multi-device applications is the following:

- A user interaction in one of the available modalities (user agent) results into interaction events
- These events are passed to a synchronization manager that handles a representation of the interaction event and determines the impact of the interaction based on the state of the application and synchronization rules.
- This in turns results into updates of the state of the application and update instructions sent to all the registered modalities (user agents) available to the user.

This is summarized in Figure 1, where each user agent may represent a different modality (e.g. VoiceXML browser and XHTML-MP browser or GUI and Voice Java applications) or different devices (e.g. smart phone and PDA or kiosk).

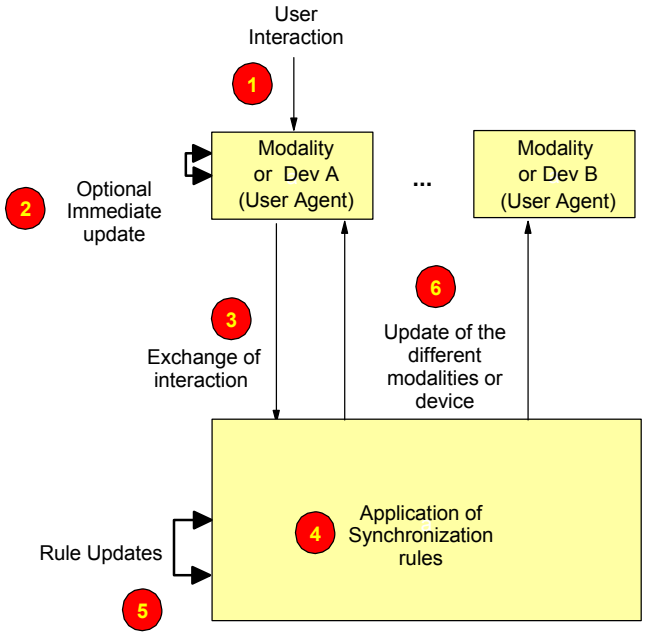


**Figure 1 – Fundamental execution model of multimodal or multi-device applications; independent of programming model or configuration.**

The user action may or may not result into an immediate update of the affected modality state prior to the synchronization (step 2). Immediate updates provide faster response in the modality that the user currently uses, but may lead to problems or confusing behaviors for example with composite inputs, concurrent or almost concurrent inputs in different modalities and conversational multi-modal application (i.e. where inputs are to be understood or disambiguated first).

Other events than user input may also trigger interaction events to transmit through step (3). When the service uses dialog management, it would typically be responsible for establishing and managing the multimodal synchronization.

The following basic architecture implements this execution model (Figure 2).

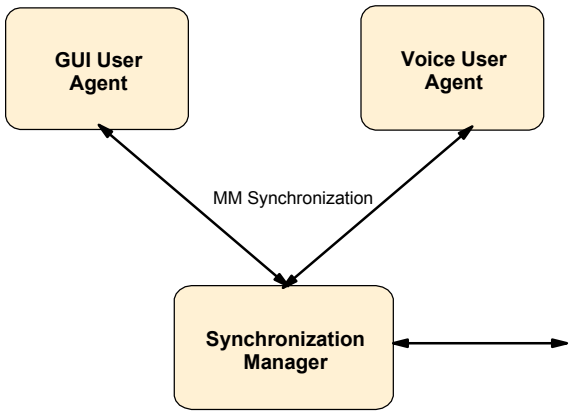


**Figure 2 – architecture to support multimodal and multi-device interactions illustrated for voice and GUI interaction.**

Each of these module or portions of these modules may be partitioned or combined on a single device or distributed across several devices or servers.

## 2.2 Associated Mobile Deployment Configurations

Figure 3 to Figure 8 present examples of multimodal or multi-device configurations that implement the multimodal and multi-device execution model and architecture discussed in section 2.1.

Except for the multi-device configuration described in Figure 8, the figures illustrate multimodal interactions with voice and GUI. Nothing imposes these modality or to limit the synchronization to two modalities or devices.

Except for the sequential configuration, they can support any synchronization granularity authorized by application, network or user.

The speech recognition framework (SRF) refers to a framework currently studied by 3GPP (3GPP TR 22.977 – Feasibility Study for Speech Enabled Services (Release 6). The speech recognition framework (SRF) enables to distribute the audio sub-system and the speech services by sending encoded speech and meta-information between the client and the server over a packed switched network. The SRF may use conventional codecs like AMR or Distributed Speech Recognition (DSR) optimized codecs.

The SRF can be deployed over a packet switched (PS) network. Over a generic PS network, SRF will require:

- Uplink and downlink transport of audio (e.g. RTP)
- Session establishment, signalling and control
- Codec negotiations
- Quality of service negotiation and provisioning

- Service administration.

The distribution of processing for other modality may require extension similar frameworks.

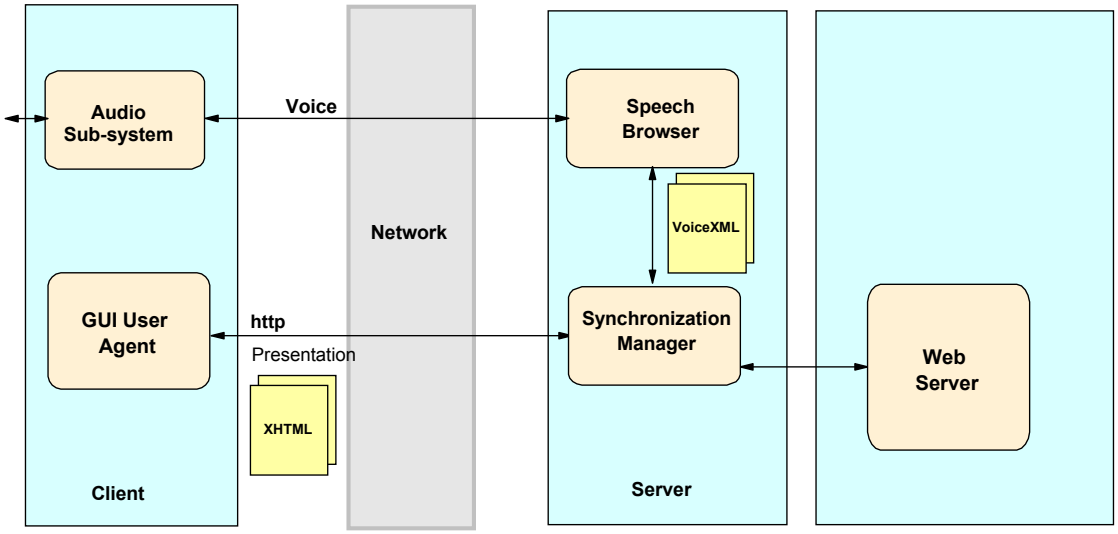


**Figure 3 – Example of sequential configuration (no voice and data support simultaneously) for voice and GUI interaction. This configuration does not require SRF: it can be deployed on 2G or 2.5G networks. Only one modality is available at a given moment. The user may switch at any time or when allowed or imposed by the application.**

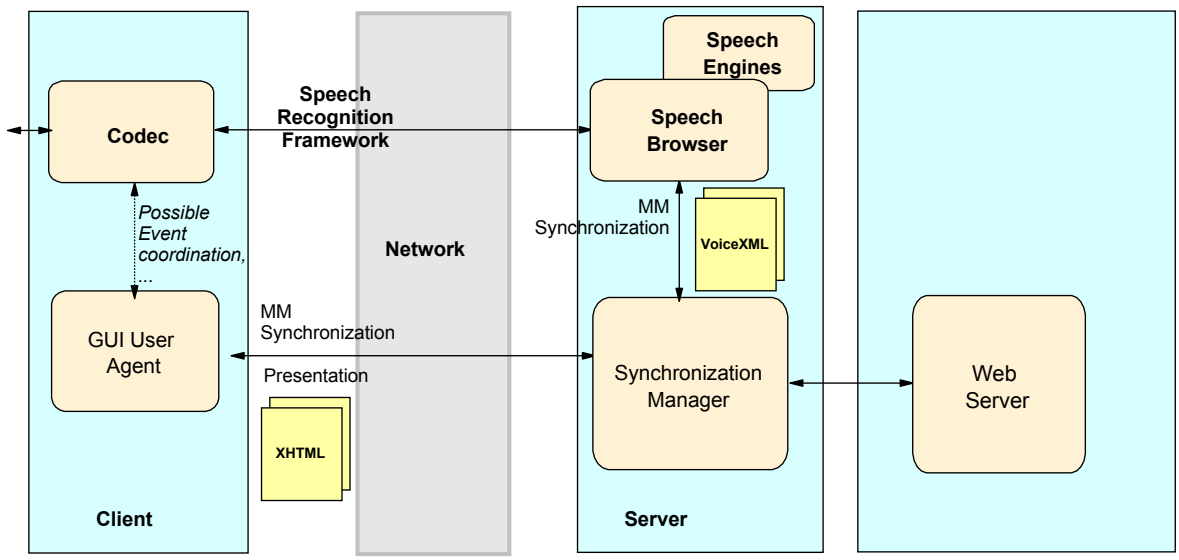


**Figure 4 – Example of Thin Client Configuration (voice and data support) with server-side speech engines local to speech browser for voice and GUI interaction.**

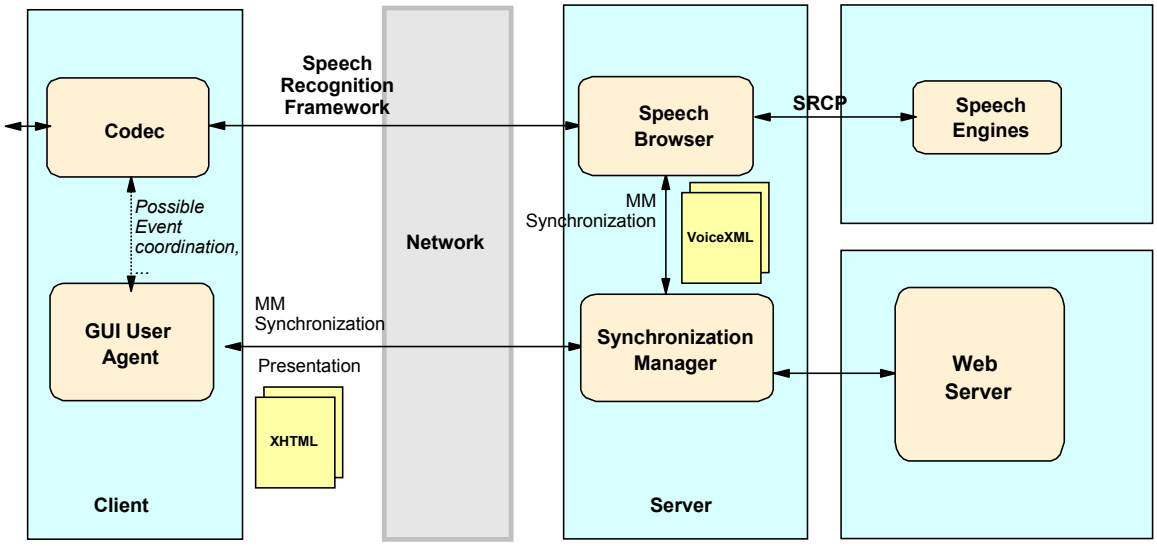


**Figure 5 – Example of Thin Client Configuration (voice and data support) with server-side speech engines remote with respect to speech browser for voice and GUI interaction.**

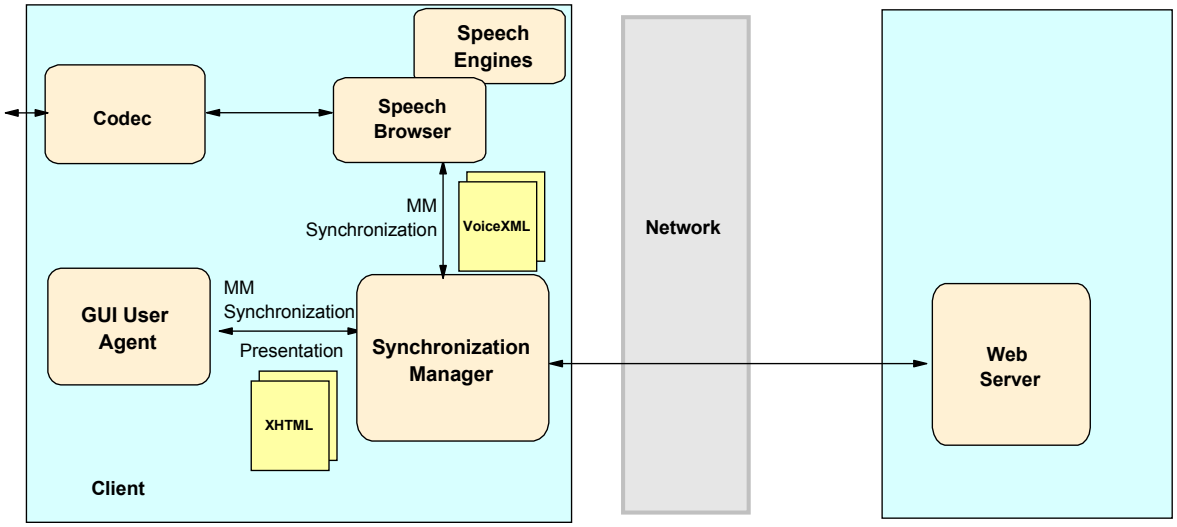


**Figure 6 – Example of Fat client configuration with local speech engines for speech and GUI interaction. This can be combined within a browser implementation.**

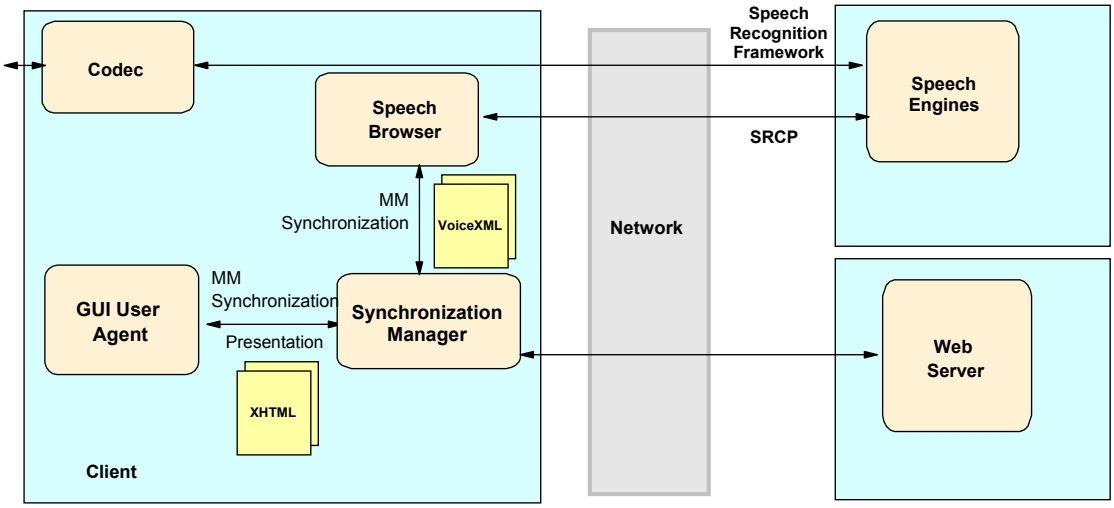


**Figure 7 - Example of Fat client configuration with server-side speech engines for speech and GUI interaction. The speech engines are remote controlled by SRCP.**

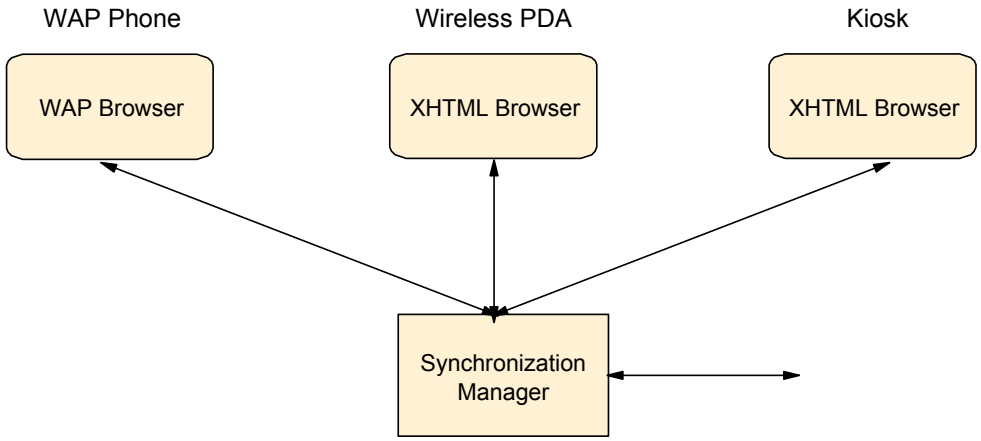


**Figure 8 – Example of Multi-device configuration.**

Configurations as illustrated in Figure 5 and Figure 7 require remote engine remote control APIs or protocols, as developed by IETF as part of SPEECHSC. Other modalities may require modality-specific extensions.

In the multi-device configuration illustrated in Figure 8, the synchronization manager may be located on the server or on a mobile device.

In all cases, the configuration may change dynamically and it may be of interest for the user to support seamless transition between these configurations, for example with mechanisms like discovery, negotiation, replication, etc…

The different configurations and functions is a function of the devices / channels, tasks and environment. Configurations may require registration and negotiation or dynamic provisioning of the device when accessing a multimodal or multi-device service. It also depends on having appropriate mechanisms to query, or examine the device capabilities and configuration modes.

## 2.3 Multimodal and Multi-device Authoring

Methods for authoring of multimodal and multi-device applications can be divided into several types as summarized in Figure 9.

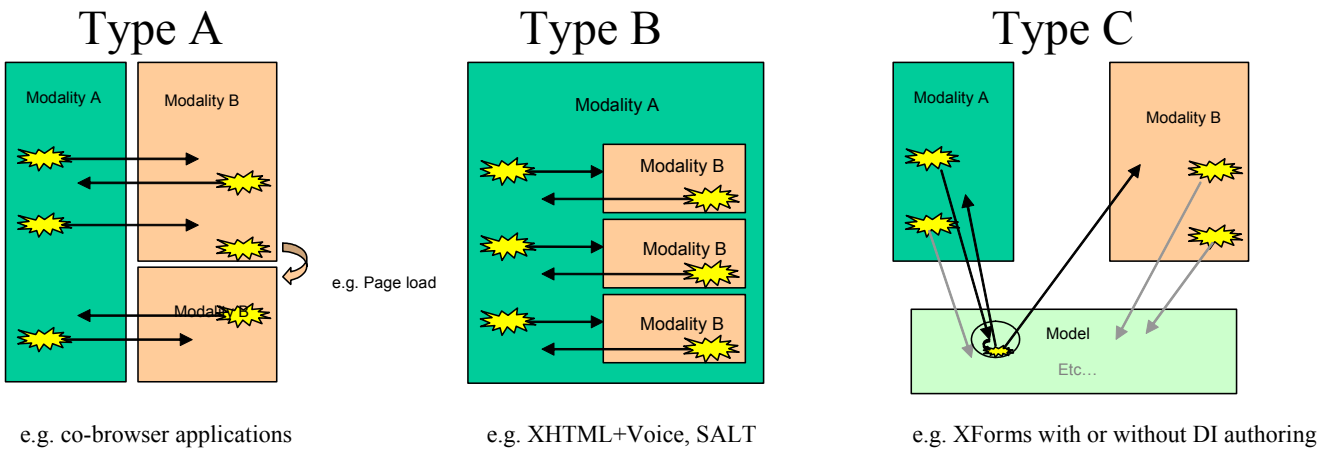


**Figure 9 – Different types of multimodal or multi-device authoring methods**

Different authoring approaches have been proposed so far:

- **Type A** multimodal and multi-device authoring where the application is authored as stand alone presentation for each modality with different data models and synchronization or co-visit tags (e.g. Co-browser authoring).

- **Type B** multimodal and multi-device authoring, where the application is authored for one modality with events and event handlers that specify what to do in the others. Approaches like XHTML + Voice (http://www.w3.org/TR/xhtml+voice/) and SALT (http://www.saltforum.org/) follow this approach. The presentation associated to one modality or device may or may not share at authoring a common data model.

- **Type C** multimodal and multi-device authoring, where the applications are authored at the level of the data model (e.g. XForms in XHTML container, JSP, …) and the presentations for each modality or device are bound to the data model and manually authored or automatically generated from the data model (CSS). Synchronization results from the binding to the data model.

The execution model and architecture proposed supports the three authoring types, for example by transforming at runtime type A and type B into type C. Indeed, the module responsible for the application of the synchronization rules can interprets multimodal applications and executes the synchronization. This renders the proposed execution model independent of the authoring language and compatible with the different approaches that have been proposed so far, including XHTML+Voice, SALT and Xforms-based synchronization, with binding to pre-compiled presentation or to a device-independent representation.

Note also that Figure 2 does not address the steps internal to the user agents. For example, a voice or handwriting user agent will interface with speech engines to process input and generate outputs.

## 2.4 Items to specify for Mobile support

- Synchronization mechanisms of user agents by exposing as part of the Browser enhancement work item:
    - Access to interaction events from the GUI browser (e.g. DOM UI events, XML events)
    - Mechanisms to update the presentation in the GUI browser (e.g. DOM manipulation, Xupdate)

- Distribution of these exchanges (.e.g based on SOAP):
    - Discovery, registration and deregistration of modalities with synchronization manager:
        - Addresses
        - Capabilities
    - Security
    - Privacy
- Fat client browser enhancements part of the Browser enhancement
- Authoring of multimodal applications:
    - Evaluation of use of  type A, B and C possible with mobile profile selections
    - Possible non-declarative specifications
- Possibly additional options like supporting: TBD
    - Session management and synchronization to support suspend and resume and switches between thin and fat configurations etc…
    - …