Building Multimodal Applications with EMMA

Michael Johnston
AT&T Labs Research
180 Park Ave
Florham Park, NJ 07932
johnston@research.att.com

ABSTRACT

Multimodal interfaces combining natural modalities such as speech and touch with dynamic graphical user interfaces can make it easier and more effective for users to interact with applications and services on mobile devices. However, building these interfaces remains a complex and high specialized task. The W3C EMMA standard provides a representation language for inputs to multimodal systems facilitating plug-and-play of system components and rapid prototyping of interactive multimodal systems. We illustrate the capabilities of the EMMA standard through examination of its use in a series of mobile multimodal applications for the iPhone.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation (e.g. HCI)**]: User Interfaces—*input devices and strategies (e.g. mouse, touchscreen),voice I/O. natural language, prototyping, standardization.*

General Terms

standardization

Keywords

multimodal, standards, speech, gesture, prototyping

1. INTRODUCTION

While numerous prototypes have been demonstrated over the years ([4, 13, 17]), building multimodal interfaces remains a complex and highly specialized task. Typically these systems involve a graphical user interface working in concert with a variety of different input and output processing components, such as speech recognition, gesture recognition, natural language understanding, multimodal presentation planning, dialog management, and multimodal integration. A significant source of complexity in authoring these systems is that communication among components is not standardized and often utilizes ad hoc or proprietary protocols. This makes it difficult or impossible to plug-and-play components from different vendors or research sites and limits the ability of authors to rapidly pull components together to prototype multimodal systems.

The new W3C EMMA standard [11] addresses this problem by providing a standardized XML representation language for encapsulating and annotating inputs to spoken and multimodal interactive systems. This paper explores the role of the EMMA standard in supporting multimodal applications for mobile devices, such as the iPhone, through examination of series of illustrative sample applications. These applications have been built in an authoring framework which uses EMMA as a communication language between a multimodal browser and network hosted services including a platform providing access to speech recognition and synthesis over HTTP ([5]).

Section 2 briefly summarizes the capabilities of the EMMA language. Section 3 describes the multimodal authoring framework. Section 4 discusses a series of different sample applications, and Section 5 concludes the paper.

2. EMMA: EXTENSIBLE MULTIMODAL ANNOTATION

In essence, EMMA is the glue which bonds together the disparate components of a spoken or multimodal interactive system. EMMA is an XML markup language which provides mechanisms for capturing and annotating the various stages of processing of users' inputs. There are two key aspects to the language: a series of elements (e.g. `emma:group`, `emma:one-of`, `emma:interpretation`) which are used as containers for interpretations of the user's inputs, and a series of annotation attributes and elements which are used to provide various pieces of metadata associated with those inputs, such as timestamps (`emma:start`, `emma:end`) and confidence score values (`emma:confidence`). Given the broad range of input types to be supported, a critical design feature of EMMA is that it does not attempt to standardize the semantic representation assigned to inputs, rather it provides a series of standardized containers for mode and application specific markup, and a set of standardized annotations for common metadata. The language also provides extensibility through the `emma:info` element, which is a container for application and vendor specific annotations on inputs. Note that individual EMMA documents are not intended to be authored directly by humans, rather they are generated automatically by system components such as speech recognizers and multimodal fusion engines. They are however intended to be manipulated and read by humans in logging

ICMI-MLMI'09, November 2–4, 2009, Cambridge, MA, USA.

```
<emma:emma version="1.0"
    xmlns:emma="http://www.w3.org/2003/04/emma"
    xmlns:xsi="http://www.w3.org/
    2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2003/04/
    emma http://www.w3.org/TR/2009/
    REC-emma-20090210/emma.xsd"
    xmlns="http://www.example.com/example">
  <emma:one-of id="r1"
     emma:medium="acoustic" emma:mode="voice"
     emma:function="dialog" emma:verbal="true"
     emma:start="1241035886246"
     emma:end="1241035889306"
     emma:source="smm:platform=iPhone-2.2.1-5H11"
     emma:signal="smm:file=audio-416120.amr"
     emma:signal-size="4902"
     emma:process="smm:type=asr&version=asr_eng2.4"
     emma:media-type="audio/amr; rate=8000"
     emma:lang="en-US" emma:grammar-ref="gram1"
     emma:model-ref="model1">
   <emma:interpretation id="int1"
    emma:confidence="0.75"
    emma:tokens="flights from boston to denver">
      <flt><orig>Boston</orig>
           <dest>Denver</dest></flt>
    </emma:interpretation>
    <emma:interpretation id="int2"
     emma:confidence="0.68"
     emma:tokens="flights from austin to denver">
      <flt><orig>Austin</orig>
           <dest>Denver</dest></flt>
    </emma:interpretation>
  </emma:one-of>
  <emma:info>
   <session>E50DAE19-79B5-44BA-892D</session>
  </emma:info>
  <emma:grammar id="gram1"
   ref="smm:grammar=flights"/>
  <emma:model id="model1"
   ref="smm:file=flights.xsd"/>
</emma:emma>
```

Figure 1: Sample EMMA document

and annotation tools. The specifics of the language are best explained by example. The EMMA document in Figure 1 is an example of the markup that might be produced by a natural language understanding component in a system for making air travel reservations. In this case, the user has requested information about flights from Boston to Denver.

All EMMA documents have the root element `emma:emma`. This has attributes indicating the version of EMMA and namespace and schema declarations. To simplify the presentation we will leave out the namespace information in the rest of the examples in this paper. The core of an EMMA document consists of a tree of container elements (`emma:one-of`, `emma:group`, and `emma:sequence`) terminating in a number of `emma:interpretation` elements. The `emma:interpretation` element is the main container for the semantic representation of a user input. In the example in Figure 1, the semantic representation is an XML element `<flt>` specifying an origin and destination for an airline flight query. In this case there are two possible N-best interpretations of the user input and the element `emma:one-of` is used as a container for the two competing interpretations, each contained within an `emma:interpretation`. The other main container elements in EMMA are `emma:group` for grouping inputs and `emma:sequence` for representation of sequences of inputs. Annotations appearings on `emma:one-of` are assumed to apply to all of the `emma:interpretation` elements it contains. The annotations `emma:medium` and `emma:mode` provide a classification of user input modality. In this case, the medium is `acoustic` and the specific modality is `voice`. Multimodal inputs will have multiple values within their medium and mode attributes. The `emma:function` annotation differentiates, interactive dialog (`dialog`) from other uses such as recording and verification. The attribute `emma:verbal` is a boolean indicating whether the input is verbal or not. This is used, for example, to distinguish handwriting (letters) from freehand drawing (lines, areas) in pen input. The attributes `emma:start` and `emma:end` are absolute timestamps indicating the start and end of the user input signal in milliseconds from Jan 1 1970. `emma:signal` is a URI (Uniform Resource Identifier) valued attribute which identifies the location of the input signal, in this case an audio file. The `emma:signal-size` attribute indicates the size of that file in 8-bit octets. `emma:source`, also URI-valued, provides a description of the device which captured the input. This attribute has an important use case for mobile multimodal applications, where it can be used to indicate the kind of device used to capture the input, e.g. iPhone vs. Blackberry, including the specific model and operating system. `emma:process` is a description of the processing stage which resulted in the current interpretation(s). In the example, it indicates that the process was speech recognition (`asr`) and specifies the recognizer version. The `emma:lang` attribute indicates the language spoken in the input signal, in this case US English. The `emma:media-type` attribute contains the MIME type of the signal, and provides a convenient location for specifying the codec and sampling rate (AMR encoded audio at 8000hz in this case). If a grammar is used in the processing it can be specified in an `emma:grammar` element under `emma:emma`. The attribute `emma:grammar-ref` on the interpretations or `emma:one-of` indicates which grammar resulted in that interpretation. The use of a `emma:grammar` element combined with the attribute `emma:grammar-ref` allows for multiple grammars to be specified and individually referenced for each interpretation. Similarly the element `emma:model` is used for in-line specification or reference to the data model of the semantic representation. More than one model may be specified, and the `emma:model-ref` attribute is used to associate interpretations with specific models. The `emma:info` element is used here to introduce a vendor specific session identifier. On the `emma:interpretation` elements, the attribute `emma:tokens` indicates the particular string of words that were recognized and `emma:confidence` contains a confidence score between 0 and 1 for each interpretation.

The ability to represent uncertainty using `emma:one-of` and `emma:confidence` is critical for multimodal systems utilizing natural modalities such as speech and gesture recognition where there may be multiple possible interpretations of a user input. In addition to `emma:one-of` for N-best the standard also provides an element for representation of lattice inputs from speech and other modalities. In Section 4.3

we illustrate the use of `emma:lattice` for representation of a gesture lattice associated with touch or pen input.

The `emma:derived-from` and `emma:derivation` elements can be used to reference and contain the previous stage of processing that an interpretation or set of interpretations is derived from. See Sections 4.1 and 4.3 for unimodal and multimodal examples.

The use of XML as the language for representing user inputs facilitates the generation and parsing of EMMA documents by EMMA producers and consumers since tools for XML processing and parsing are readily available in almost all programming environments. There is no need to write a specific parser for decoding input representations, as in the case of proprietary protocols. It also facilitates the creation, viewing, and manipulation of log files for interactive systems, since these can be manipulated and extended using general purpose XML tools such as XPATH and XSLT.

3. MULTIMODAL RAPID PROTOTYPING FRAMEWORK

The applications described in the next section were all built using a multimodal rapid prototyping framework which combines a multimodal browser with web services for speech recognition, speech synthesis, multimodal understanding, and database access. Figure 2 provides an overview of this framework and the communication among components.

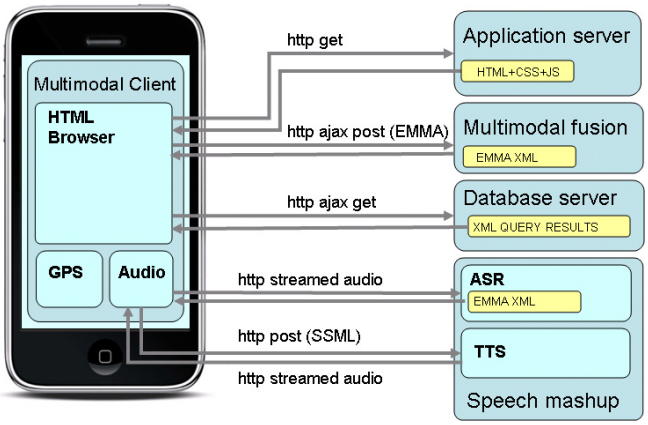

Figure 2: Multimodal architecture

The multimodal client is a native application running on the iPhone which combines a full HTML browser with an audio capture and streaming component and a GPS component which provides access to geolocation and device orientation. Communication between the HTML browser and the audio and GPS components is established through HTTP access to a set of dedicated URL types (e.g. 'watson://asr...'), which are captured by the containing application and responses are returned through Javascript callbacks. This mechanism performs a similar function to SALT [18] and X+V [1] in that it supports multimodality through voice enablement of HTML content. The approach is closer to SALT in that, unlike X+V, the dialog logic is not specified in markup but rather is handled either in client code or a separate dialog server. Notably though, unlike SALT and X+V there is no direct extension of the markup language. The application is authored in standard HTML, CSS, and Javascript and speech capabilities are accessed through Javascript commands. This has facilitated the creation of multimodal browsers for multiple platforms. In addition to the iPhone client, there is also plugin for Safari, and a control for speech enablement of Internet Explorer. The scope of the mechanism is also broader in that in addition to speech functions, in the mobile browser, this mechanism is also used to query the GPS, accelerometer and other device characteristics.

The developer of each multimodal application authors their application using a combination of HTML, Javascript, and CSS, hosted on an application server. As each application is accessed the multimodal client loads the relevant files from the application server. As a result, changes in the application can be made and tested rapidly without recompiling and downloading a new native application to the device. This is an important property for rapid prototyping and testing of new applications and services, especially for trials and bucket testing where it may not be feasible to constantly update participant devices. Once users have the multimodal client on their device, making a new multimodal prototype available to them is simply a matter of adding it to the application server. Using an HTML browser is also important for rapid prototyping since it allows easy access to all of the graphical interface elements, layout mechanisms, and capabilities of HTML, CSS, and Javascript. The sample applications described in this paper all make use of a combination of Javascript and CSS that simulates the appearance of a native application, including a navigation bar, spinnable lists of results, and animated navigation between interface sections.

The HTTP-based Speech Mashups platform ([5]) enables developers to easily add speech capabilities to web applications, in much the same way that mechanisms such as Googlemaps, Livemaps, and Yahoo! maps enable easy integration of dynamic mapping capabilities. The platform provides HTTP access to both speech recognition (ASR) and synthesis (TTS). In the case of ASR, in response to a Javascript command in the HTML browser, the audio component in the multimodal client collects audio and streams it over HTTP to the speech server, which uses the Watson recognizer ([6]) to perform ASR and return an EMMA document containing the recognition results. Parameters can be set in the HTTP request to the ASR to request N-best, in which case multiple results are returned in `emma:one-of`. In the case of TTS, an SSML document is posted from the multimodal client to the mashup server, and an HTTP stream of audio is returned and played on the client. Critically, in addition to HTTP access at runtime for recognition, the speech mashup platform also supports HTTP access for posting and compiling grammars, and a user portal where developers building prototypes can upload and manage grammar models. Both fixed deterministic grammars (SRGS[10], SISR[16]) and stochastic language models are supported. The portal also supports monitoring of ASR logs and provides tools for rapid online transcription of audio. See [7] for description of a similar framework and portal. Note that one of the significant differences here is the adoption of standards-based approach utilizing EMMA. See also [15] for discussion of the use of EMMA in the development of a mobile system for accessing the semantic web.

The multimodal fusion server utilizes finite state methods [12] for combination and understanding of speech with tactile gesture inputs, such as touch and pen. This server can

also be used for unimodal spoken language understanding, in which case the gesture input is empty. Access to this server is through HTTP. In the sample applications here an AJAX request is made from Javascript in the client application. An EMMA XML document containing the speech string and/or a gesture lattice is posted to the server and the server returns an EMMA XML document containing the combined interpretation. If no intepretation is available, then the server returns an empty `emma:interpretation` annotated as `emma:uninterpreted="true"`. The server also supports the use of finite state edit machines [2] for multimodal understanding, enabling the coupling of a stochastic language model for recognition with a deterministic finite state integration and understanding model.

The final server used in the rapid prototyping framework is a simple database server using SQLite which provides access through AJAX to the underlying database used in the application. This server provides a URL syntax for specification of the database to be queried, specific constraints on fields, and the fields to be returned. The mechanism also allows specification of the format of the results to be returned, and the server returns results as an XML document containing a number of records. These XML results are then manipulated in the client Javascript code in order to dynamically create HTML content for displaying lists of results or details regarding items the user searches for. The database server is set up to be general purpose. Since the client can specify the data to be returned, it's format, and the database to be queried, it can easily be reused for a range of different applications (e.g. movies, books, restaurants, corporate directory) without modification of the database server itself.

In the architecture and sample prototype applications described here, access to resources, such as ASR, multimodal understanding, and database lookup are handled as separate HTTP queries. This increases the number of network roundtrips, though in testing we have not found this to be a significant source of latency. The advantage of this approach for prototyping is that it allows centralization of application logic in the client code, simplifying authoring, and allows easy implementation of feedback mechanisms on the user interface as each stage of processing takes place. Also, in several cases, such as presenting N-best recognition results, the interface designer may want to seek user input or confirmation between processing stages. The architecture also provides a number of mechanisms for tighter integration of processing stages which can avoid the multiple HTTP requests. The mashup platform allows for specification of a 'posturl' in an ASR query, so that instead of being returned directly to the client, the results of a speech recognition are passed to another server for processing. Also the Watson ASR engine supports specification of commands (in Python) to be executed on the speech recognition results. Both of these mechanisms enable implementation of single query HTTP access, where audio packets are sent to the ASR server and the result that comes back is a list of database entries. Note however that this can significantly complicate iterative development at the prototyping stage since the application logic is distributed over multiple different parts of the system.

4. SAMPLE APPLICATIONS

We consider four different application use cases. The first and simplest is a multimodal voice search interface for finding movies in a mobile video-on-demand application (iMOD). The second application, iDA, is a corporate directory application which illustrates how `emma:one-of` can be used to support visual display of N-best recognition hypotheses. The third example, iMatch, shows how the `emma:group` element and the `emma:derived-from` element are used for representing fusion of multimodal inputs. In this example, we also illustrate the use of `emma:lattice` for representation of gesture input. The fourth example considers the use of `emma:group` for grouping speech with GPS coordinates.

4.1 iMOD: Movie on Demand

As more and more content becomes available to consumers, through services such as IPTV, video-on-demand, and music download and subscription services, it is increasingly difficult for users to find the content they want. One way to address this problem is using a speech-enabled multimodal interface to search and browse for media content ([14, 19]).

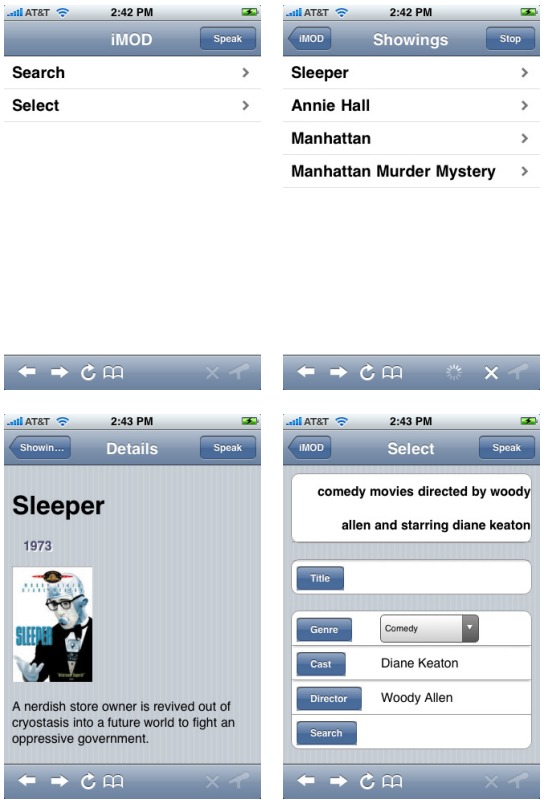

Figure 3: iMOD interface

The iMOD system, built using the framework described above, enables users to search and browse for movies on the iPhone based on range of different constraints, such as title, genre, cast, or director. Queries can be in natural language and can freely combine different constraints. From the initial screen (Figure 3, top left), the user hits the 'Speak' button and issues a spoken command, for example "comedy movies directed by Woody Allen and starring Diane Keaton". This results in display of spinnable list of titles (top right), and user can select individual titles to view details and play the content (bottom left).

EMMA documents are used both for communication with the ASR server and with the multimodal fusion server, which

in this case is used for unimodal spoken language understanding. In the example described above, after the user hits the 'Speak' button audio is streamed over HTTP to the ASR server through the speech mashup, and then it returns an EMMA document containing a single recognition result (Figure 4). For the sake of presentation we will leave out the full set of annotation elements and attributes, focusing on the core set pertinent to the presentation here.

```
<emma:emma>
  <emma:interpretation
    id="int1"
    emma:medium="acoustic" emma:mode="voice"
    emma:function="dialog" emma:verbal="true"
    emma:start="1241035986246"
    emma:end="1241035989306"
    emma:confidence="0.8" emma:lang="en-US"
    emma:process="smm:type=asr&version=asr_eng2.4"
    emma:media-type="audio/amr; rate=8000">
     <emma:literal>comedy movies directed by woody
     allen and starring diane keaton</emma:literal>
  </emma:interpretation>
</emma:emma>
```

Figure 4: EMMA response from ASR

This document is then sent from the multimodal client to the Multimodal fusion server as an HTTP post. The Multimodal fusion server returns the EMMA document in Figure 5 to the client. Here we see the capabilities in EMMA for representing the relationships between multiple stages of processing. The element `emma:derived-from` provides a reference to the resource `emma:interpretation` from which this new `emma:interpretation` was derived. The element `emma:derivation` is used as a container for the earlier stage of processing. Note that any annotations which appear on the earlier stage of processing (`int1`) are assumed to apply the later stage (`int2`) unless they are explicitly restated (such as `emma:process` and `emma:confidence`). The semantic interpretation from this document (`<query>`) is used by the client to build a URL expressing a database query and this is then issued to the database server. The database server returns an XML document containing details for each of the matching movies, and these are used by the client code to build an HTML list of results which is dynamically inserted into the results view (top right in Figure 3). In addition to issuing the database query when the EMMA document in Figure 5 is received the semantic representation is used to populate an interface Figure 3 (bottom right) which users can navigate to in order to repair errors or refine their query and reissue it. This can be done using either speech, keyboard input, or for some parameters using graphical widgets. For example, in our example case the user touches the 'Cast' button says "Mia Farrow", then hits 'Search' in order to issue the search "comedy movies directed by Woody Allen and starring Mia Farrow".

4.2 iDA: corporate directory application

The iDA prototype allows users to search a corporate directory using speech. For example, the user might say "John Smith" or "John Smith Florham Park New Jersey". The system provides details for each listing, and supports one touch

```
<emma:emma>
  <emma:interpretation
    id="int2"
    emma:tokens="comedy movies directed by woody
                 allen and starring diane keaton"
    emma:confidence="0.7"
    emma:process="smm:type=fusion&version=mmfst1.0">
    <query><genre>comedy</genre>
           <dir>woody_allen</dir>
           <cast>diane_keaton</cast></query>
    <emma:derived-from resource="#int1"/>
  </emma:interpretation>
  <emma:derivation>
   <emma:interpretation id="int1"
    emma:medium="acoustic" emma:mode="voice"
    emma:function="dialog" emma:verbal="true"
    emma:start="1241035986246"
    emma:end="1241035989306"
    emma:confidence="0.8" emma:lang="en-US"
    emma:process="smm:type=asr&version=asr_eng2.4"
    emma:media-type="audio/amr; rate=8000">
     <emma:literal>comedy movies directed by woody
     allen and starring diane keaton</emma:literal>
   </emma:derivation>
  </emma:derivation>
</emma:emma>
```

Figure 5: EMMA from fusion server

dialing, messaging, or mapping of the individual's location. In this example we will see how `emma:one-of` can be used to leverage the graphical aspect of a multimodal interface to allow user selection among N-best recognition hypotheses.

Proper name recognition over long lists of names can be error prone. A significant advantage of a multimodal user interface, compared to unimodal speech, is that multiple different (N-best) recognition hypotheses can be presented to the user. This is particularly effective on devices with a touch display where a spinnable list can easily be used to pick among a list of names. EMMA provides support for N-best recognition hypotheses through the `emma:one-of` container element which contains a sequence of interpretation elements. Consider the sample query 'John Smith'. In the HTTP request that initiates recognition, multiple N-best results can be requested. The resulting EMMA document returned from the recognition server contains these multiple results in an `emma:one-of` element as in Figure 6. In this case the grammar being used is an SRGS grammar and SISR is used to build the output representation as an Ecmascript object which is then serialized to XML.

This EMMA document is received by the multimodal client, which reads it into its local document object model. The client application iterates over the `emma:interpretation` elements to build the list representation for display to the user as in Figure 7 (left). If the user clicks on an item a panel slides into view showing the number, address etc for the particular contact (Figure 7, right).

4.3 iMATCH: Local search with multimodal integration

The iMATCH prototype provides a natural language interface for access to restaurant information. This prototype

```
<emma:emma>
<emma:one-of id="one-of1"
  emma:medium="acoustic" emma:mode="voice"
  emma:function="dialog" emma:verbal="true"
  emma:lang="en-US" emma:start="1241641821513"
  emma:end="1241641823033"
  emma:media-type="audio/amr; rate=8000"
  emma:process="smm:type=asr&version=watson6">
<emma:interpretation id="nbest1"
  emma:confidence="1.00" emma:tokens="jon smith">
  <pn>jon</pn><ln>smith</ln>
</emma:interpretation>
<emma:interpretation id="nbest2"
  emma:confidence="0.99" emma:tokens="john smith">
  <fn>john</fn><ln>smith</ln>
</emma:interpretation>
<emma:interpretation id="nbest3"
  emma:confidence="0.99" emma:tokens="joann smith">
  <fn>joann</fn><ln>smith</ln>
</emma:interpretation>
<emma:interpretation id="nbest4"
  emma:confidence="0.98" emma:tokens="joan smith">
  <fn>joan</fn><ln>smith</ln>
</emma:interpretation>
</emma:one-of>
</emma:emma>
```

Figure 6: EMMA N-best list of ASR hypotheses

is in a similar application domain to the MATCH (Mobile Access To City Help) system [13], and like several other multimodal prototypes, combines voice and gesture input with a dynamic map display [4, 3, 9, 8]. The system supports search for restaurants based on business name, food type, and location. For example, in Figure 8 (left), the user has said "italian restaurants near the empire state building". The results can be viewed either as a spinnable list of results or as icons plotted on a map. Like iMOD, iMATCH provides an interface for multimodal error correction and iterative refinement for queries. iMATCH also supports multimodal integration. In this section, we show how EMMA can be used to represent multimodal combinations of speech with gesture input that are then processed and combined by the multimodal fusion server.

In order to combine speech and gesture inputs, the multimodal fusion server uses finite state language processing techniques [12]. The inputs to this process are a speech string and a lattice representing the the gesture input, in this case touch gestures on the display. As an example of a multimodal command, consider the query "italian restaurants near here" accompanied by a touch gesture on the map display (Figure 8 (right)). The diamond symbol indicates the location of the touch gesture. The speech input is sent to the ASR in the mashup server for processing and an EMMA interpretation document for the ASR result is returned, similar in form to Figure 4. At the same time as the audio is being transmitted the multimodal client also captures touch gestures on the display and these are represented in an EMMA lattice. In the finite state approach to multimodal integration [12], gestures, like speech, are represented as sequences of symbols. Sequence of gesture symbols can be represented as a series of `emma:arc` elements within an `emma:lattice`, which itself appears within `emma:interpretation`. The multimodal client uses the element `emma:group` as a wrapper for the multimodal combination of a touch gesture and the speech input (Figure 9). Note that the `emma:group-info` element can be used to provide information about the characteristics of the group, such as the temporal relationship between the elements to be combined. The `emma:group` element is used to show that the inputs are related to each other, but does not yet show their combined interpretation. This EMMA document represents a 'package' of multimodal input and is posted to the multimodal fusion server, which then returns an EMMA document capturing the combined interpretation (Figure 10). Note that in this case there are two `emma:derived-from` elements each pointing to the individual interpretation of one of the combining modalities within `emma:derivation`. This is used to build a database query which is then issued to the database server. In this application the database is an search engine supporting geographic queries and the set of italian restaurants closest to the specified coordinates are returned.

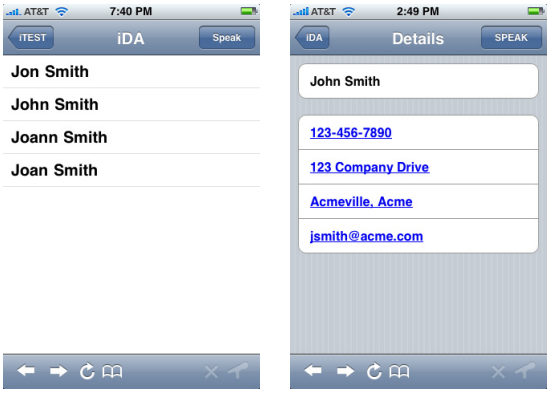

Figure 7: iDA interface

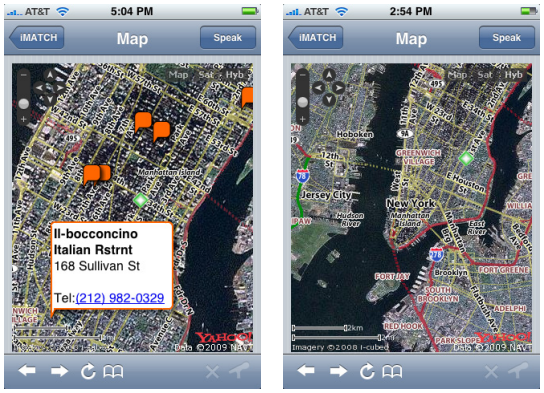

Figure 8: iMATCH interface

In this example, language understanding and multimodal integration are part of a single process. EMMA also provides a mechanism to facilitate implementation of two stage multimodal language processing. The attribute `emma:hook` can be added to an element in the semantic representation in order to indicate that content from another modality needs to be combined with the interpretation (Section 4.2.12 [11]).

```
<emma:emma>
  <emma:group
   emma:medium="acoustic,tactile"
   emma:mode="voice,touch" emma:function="dialog">
   <emma:interpretation id="speech1"
    emma:confidence="0.9" emma:verbal="true"
    emma:start="1241035886246" emma:end="1241035889306"
    emma:medium="acoustic" emma:mode="voice"
    emma:confidence="0.8" emma:lang="en-US"
    emma:process="smm:type=asr&version=asr_eng2.4"
    emma:media-type="audio/amr; rate=8000">
    <emma:literal>italian restaurants near here
    </emma:literal>
   </emma:interpretation>
   <emma:interpretation id="touch1"
       emma:confidence="0.8"
       emma:medium="tactile" emma:mode="touch"
       emma:start="1241035886250"
       emma:end="1241035886500">
       <emma:lattice initial="0" final="4">
        <emma:arc from="0" to="1">G</emma:arc>
        <emma:arc from="1" to="2">sel</emma:arc>
        <emma:arc from="2" to="3">coords</emma:arc>
        <emma:arc from="3" to="4">
        SEM([40.729567,-73.986053])</emma:arc>
       </emma:lattice>
   </emma:interpretation>
   <emma:group-info>temporal</emma:group-info>
  </emma:group>
</emma:emma>
```

Figure 9: Multimodal input using emma:group

4.4 Annotating geolocation using emma:group

Another use case for EMMA is for capturing the combination of user input with geolocation. In some systems there may be a constant stream of GPS updates from the mobile device to a dialog or interaction manager and GPS information can be directly accessed when a user input reaches the dialog manager. In many cases though, it may not be desirable to use data bandwidth for this constant stream of GPS and will be more effective to pass GPS information along with user inputs. This is also important for cases where the device location changes rapidly, such as while driving, since it may be important to know where the user was when they issued a command, rather than where they are when the dialog manager processes their input. The `emma:group` element can be used for this purpose. Spoken or multimodal input can be grouped with the geolocation of the device when that input was received, as in the example in Figure 11. Note that in this instance `emma:group-info` is used to indicate that the reason for the grouping is `geolocation`.

5. CONCLUSION

EMMA provides a highly expressive language for representation of inputs to both multimodal and unimodal systems, and simplifies plug-and-play of different system components and modalities. We have illustrated the practical use of EMMA within an architecture for rapid prototyping of multimodal applications and shown how features of EMMA can be used to support important features of multimodal interaction for a broad range of domains. We presented sample applications spanning media search, local search, and corporate directory access. Some of the key features of EMMA for building multimodal interfaces are the ability to represent uncertainty, using the `emma:one-of` element and `emma:confidence` and the use of the `emma:group` element to represent 'packages' of multimodal input for integration by a fusion server, combining speech input with deictic gesture input or multimodal sensor data. The derivation mechanism using `emma:derived-from` and `emma:derivation` is also critical since it enables representation of the relationships among the multiple stages of processing that are characteristic of multimodal systems.

Both standards and authoring frameworks are notoriously difficult to quantitatively evaluate. One measure for EMMA is the number of groups adopting the standard. In the implementation report phase of the W3C process, 11 separate implementations of the EMMA standard were reported from

```
<emma:emma>
  <emma:interpretation
   emma:medium="acoustic,tactile"
   emma:mode="voice,touch" emma:function="dialog"
   emma:process="smm:type=fusion&version=watson6"
   emma:start="1241035886246" emma:end="1241035889306"
    <query>
     <cuisine>italian</cuisine>
     <location>[40.729567,-73.986053]</location>
    </query>
   <emma:derived-from resource="#speech1"/>
   <emma:derived-from resource="#touch1"/>
  </emma:interpretation>
  <emma:derivation>
     <emma:interpretation id="speech1"
       emma:confidence="0.9"
       emma:start="1241035886246"
       emma:end="1241035889306" emma:verbal="true"
       emma:medium="acoustic" emma:mode="voice"
       emma:confidence="0.8" emma:lang="en-US"
       emma:process="smm:type=asr&version=asr_eng2.4"
       emma:media-type="audio/amr; rate=8000">
       <emma:literal>italian restaurants near here
       </emma:literal>
     </emma:interpretation>
     <emma:interpretation
       id="touch1"
       emma:confidence="0.8"
       emma:medium="tactile" emma:mode="touch"
       emma:start="1241035886250"
       emma:end="1241035886500">
       <emma:lattice initial="0" final="4">
        <emma:arc from="0" to="1">G</emma:arc>
        <emma:arc from="1" to="2">sel</emma:arc>
        <emma:arc from="2" to="3">coords</emma:arc>
        <emma:arc from="3" to="4">
        SEM([40.729567,-73.986053])</emma:arc>
       </emma:lattice>
     </emma:interpretation>
   </emma:derivation>
</emma:emma>
```

Figure 10: Combined multimodal input

```
<emma:emma>
  <emma:group>
   <emma:interpretation
    emma:tokens="gas stations"
    emma:confidence="0.9"
    emma:medium="acoustic" emma:mode="voice"
    emma:start="1241035886246"
    emma:end="1241035887346"
    emma:process="smm:type=asr&version=asr_eng2.4"
    emma:media-type="audio/amr; rate=8000"
    emma:lang="en-US">
    <emma:literal>gas stations</emma:literal>
   </emma:interpretation>
   <emma:interpretation
    emma:medium="sensor" emma:mode="gps"
    emma:start="1241035886246"
    emma:end="1241035886246">
       <lat>40.777463</lat><lon>-74.410500</lon>
       <alt>0.2</alt>
   </emma:interpretation>
   <emma:group-info>geolocation</emma:group-info>
  </emma:group>
</emma:emma>
```

Figure 11: Geolocation and emma:group

companies and universities. The authoring framework has rapidly decreased the time and resources needed to prototype multimodal applications. In addition to the three applications described in the paper, a total of ten prototypes have been built over a few month period, some of them taking as little as 3 or 4 hours to build and deploy for testing.

6. ACKNOWLEDGMENTS

Thanks to Thomas Okken, Pino DiFabbrizio, Simon Urbanek, and Patrick Ehlen.

7. REFERENCES

[1] J. Axelsson, C. Cross, J. Ferrans, G. McCobb, T. V. Raman, and L. Wilson. Mobile X+V 1.2, 2005. http://openstandardswork.net/specs/multimodal/x+v/mobile/12/.

[2] S. Bangalore and M. Johnston. Robust understanding in multimodal interfaces. *Computational Linguistics (Forthcoming)*.

[3] A. Cheyer and L. Julia. Multimodal Maps: An Agent-Based Approach. *Lecture Notes in Computer Science*, 1374:103–113, 1998.

[4] P. Cohen, M. Johnston, D. McGee, S. L. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. Multimodal interaction for distributed interactive simulation. In M. Maybury and W. Wahlster, editors, *Readings in Intelligent Interfaces*. Morgan Kaufmann Publishers, 1998.

[5] G. D. Fabbrizio, J. Wilpon, and T. Okken. A speech mashup framework for multimodal mobile services. In *Proceedings of ICMI*, Boston, USA, 2009.

[6] V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar. The AT&T WATSON Speech Recognizer. In *Proceedings of ICASSP*, Philadelphia, PA, 2005.

[7] A. Gruenstein, I. McGraw, and I. Badr. The WAMI toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces. In *Proceedings of ICMI*, pages 141–148, 2008.

[8] A. Gruenstein, J. Orszulak, S. Liu, S. Roberts, J. Zabel, B. Reimer, B. Mehler, S. Seneff, J. Glass, and J. Coughlin. City browser: developing a conversational automotive HMI. In *Proceedings CHI '09 Extended Abstracts*, pages 4291–4296. ACM, 2009.

[9] J. Gustafson, L. Bell, J. Beskow, J. Boye, R. Carlson, J. Edlund, B. Granström, D. House, and M. Wirén. AdApt - a multimodal conversational dialogue system in an apartment domain. In *Proceedings of ICSLP*, pages 134–137, Beijing, China, 2000.

[10] A. Hunt and S. McGlashan. Speech Recognition Grammar Specification Version 1.0, March 2004. http://www.w3.org/TR/2004/REC-speech-grammar-20040316/.

[11] M. Johnston, P. Baggia, D. C. Burnett, J. Carter, D. A. Dahl, G. McCobb, and D. Raggett. EMMA: Extensible MultiModal Annotation markup language, February 2009. http://www.w3.org/TR/2009/REC-emma-20090210.

[12] M. Johnston and S. Bangalore. Finite-state multimodal integration and understanding. *Journal of Natural Language Engineering*, 11(2):159–187, 2005.

[13] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An architecture for multimodal dialog systems. In *Proceedings of ACL*, pages 376–383, Philadelphia, 2002.

[14] M. Johnston, L.-F. D'Haro, M. Levine, and B. Renger. A multimodal interface for access to content in the home. In *Proceedings of ACL*, pages 376–383, 2007.

[15] N. Reithinger, S. Bergweiler, R. Engel, G. Herzog, N. Pfleger, M. Romanelli, and D. Sonntag. A look under the hood: design and development of the first SmartWeb system demonstrator. In *Proceedings of ICMI*, pages 159–166, Trento, Italy, 2005.

[16] L. V. Tichelen and D. Burke. Semantic Interpretation for Speech Recognition (SISR) Version 1.0, April 2007. http://www.w3.org/TR/2007/REC-semantic-interpretation-20070405/.

[17] W. Wahlster. SmartKom: Fusion and fission of speech, gestures, and facial expressions. In *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, pages 213–225, Kyoto, Japan, 2002.

[18] K. Wang. SALT: A spoken language interface for web-based multimodal dialog systems. In *Proceedings of ICASSP*, pages 2241–2244, 2002.

[19] K. Wittenburg, T. Lanning, D. Schwenke, H. Shubin, and A. Vetro. The prospects for unrestricted speech input for TV content search. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 352–359, New York, NY, USA, 2006. ACM.