SINGLE APPLICATION MODEL, MULTIPLE SYNCHRONIZED VIEWS

Rafah Hosn, Stéphane H. Maes, T.V. Raman

I.B.M
T.J.Watson Research Center, P.O.Box 218,
Yorktown Heights NY 10598
{rhosn,smaes,tvraman}@us.ibm.com

ABSTRACT

User interface is a mean to an end —its primary goal is to capture user intent and communicate the results of the requested computation. On today's devices, user interaction can be achieved through a multiplicity of interaction modalities including speech and visual interfaces.

As we evolve toward an increasingly connected world where we access and interact with applications through multiple devices, it becomes crucial that the various access paths to the underlying content be synchronized. This synchronization ensures that the user interacts with the same underlying content independent of the interaction modality — despite the difference in presentation that each modality might impose. It also ensures that the effect of user interaction in any given modality is reflected consistently across all available modalities.

We describe an application framework that enables tightly synchronized multimodal user interaction. This framework derives its power from representing the application model in a modality-independent manner, and by traversing this model to produce the various synchronized multimodal views. As the user interaction proceeds, we maintain our current position in the model and update the application data as determined by user intent, then reflect these updates in the various views being presented.

We conclude the paper by outlining an example that demonstrates this tightly synchronized multimodal interaction, and describe some of the future challenges in building such multimodal frameworks.

1. INTRODUCTION

The availability of a plethora of access devices such as wireless PDAs and smart phones has increased the value proposal of synchronized multimodal applications that allow ubiquitous information access.

Application authors are now faced with the challenge of delivering interactive content that can be made equally available to the various access devices. When deployed, the interaction needs to take advantage of the multiplicity of modalities that often supplement and complement one another, to provide a richer user experience.

Finally, the various access devices are themselves still evolving —this means that content that is being presently deployed will be accessed from devices not yet available.

These requirements are the driving force behind designing an application framework that facilitates the authoring, maintenance and deployment of multimodal applications.

The framework derives its power by separating application content from its presentation, and providing a controller that manages the interaction with the user while updating the application data. This follows the model view controller programming paradigm [KP88, SM00b]. The framework also leverages evolving XML-based industry standards for modeling application content, representing user interaction, as well as for communicating the results of user interaction among various components of the system [Con00b, Con00a, RC00, SM00a].

2. APPLICATION FRAMEWORK ARCHITECTURE

The framework is composed of three main components:

1. **Application Model**: application content and data.
2. **Transformation rules**: used to transform modality-independent content to the modality-specific views.
3. **Controller**: used to manage the user interaction, to update the application data and call appropriate presentation rules.

2.1. Application Model

Applications consist of a data model to be populated via user interaction, along with a high-level description of the user interaction to be carried out in populating this data model. The data model represents the type and structure of the fields that will be populated during the interaction. User interaction is composed of modality-independent building blocks called *conversational gestures* [Ram97, Ram98]. Examples of conversational gestures include input fields to be populated by the user, messages used to convey system response, as well as higher level abstractions such as *selecting* from a given set of choices.

Thus, authoring an application consists of:

- Defining the type and structure of the data items to be collected from the user,
- Declaring instances of these structures that will be populated by the user interaction,
- And authoring the application dialogs by expressing user interaction as a sequence of possibly nested conversational gestures.

The next section describes the various processing steps involved in synthesizing synchronized multimodal interaction from such applications.

2.2. The Controller

The *controller* implements the following processing model to produce tightly synchronized multimodal interaction for applications authored as described in 2.1:

- The data model definition is processed to produce appropriate data type (or class) definitions. These type constraints will be used to validate user input.
- The various components of the application state conforming to the data model are instantiated.
- The sequence of conversational gestures is represented as a tree structure and then traversed. As traversal progresses, user intent is processed to populate the relevant components of the application state.
- When the application state is updated, modality-specific transformation rules are applied to render or update the modality-independent interaction into the specific views of registered devices.

User input is captured by modality-specific devices and communicated to the controller. In turn, the controller updates the application state, and updates the views of all the other devices registered with it. The controller traverses the sequence of conversational gestures until all requisite components of the application state have been populated. The values collected from the user are validated against the type constraints specified in the data model. Invalid input results in the controller traversing appropriate portions of the user interaction to produce the necessary user interface to allow the user to correct erroneous input.

Note that the granularity at which synchronization is done can vary. For example the controller can synchronize views at event level, at input field level, per conversational gesture, per groups of conversational gestures or at application level.

3. ADVANTAGES

Devices Supporting new devices only requires the authoring of the appropriate device-specific presentation rules that enable the controller to render the interaction components.

Synchronization Synchronization in this framework can be done at various granularities as mentioned in section 2.2. In addition, the application author can specify within the conversational gesture whether a certain modality should be disabled, or whether this gesture should use a different set of transformation rules. This specialization step gives application authors the freedom to choose from tightly or loosely coupled synchronized views.

Content Re-use By decoupling the presentation from the application model, content re-use is optimized. Application authors need not worry about presentation when adding new content to their application, and in turn, content need not be modified when new devices are added. Furthermore, standard presentation rules are packaged with the framework; so for applications that do not require any specialization, the author need only create the application data model and conversational gesture.

4. IMPLEMENTATION APPROACH

The framework described in section 2 can be implemented in an imperative or declarative language.

4.1. Imperative Approach

In an imperative language, conversational gestures can be implemented as classes that can be grouped together to form more elaborate composite components which will effectively describe the application. Transformation rules would then be implemented as appropriate methods on these classes.

4.2. Declarative Approach

In a declarative language, conversational gestures can be easily serialized into XML elements. The application data can be mapped to an XML document compliant with XForms

[Con00b]. Finally for transforming into modality-specific views, one can use XSLT transformation rules [Con99].

4.3. Current Framework Implementation

The framework described in section 2 has been implemented using an XML-based declarative language called Interaction Markup Language (iML). iML is consistent with the current XForms specification. An iML application is made up of modality independent conversational gestures, grouped together to describe user interaction. Examples of such gestures are ⟨dialog⟩, ⟨message⟩,⟨input⟩,⟨submit⟩ etc...
Browsers register with a proxy that plays the role of the *controller* as defined in 2.2. iML documents are loaded, interpreted and transcoded for each of modality by this proxy. The proxy interacts with registered browsers through the Document Object Model interface (DOM2) [W3C01] to maintain application state and synchronize between modalities.

5. AN AIRLINE RESERVATION EXAMPLE

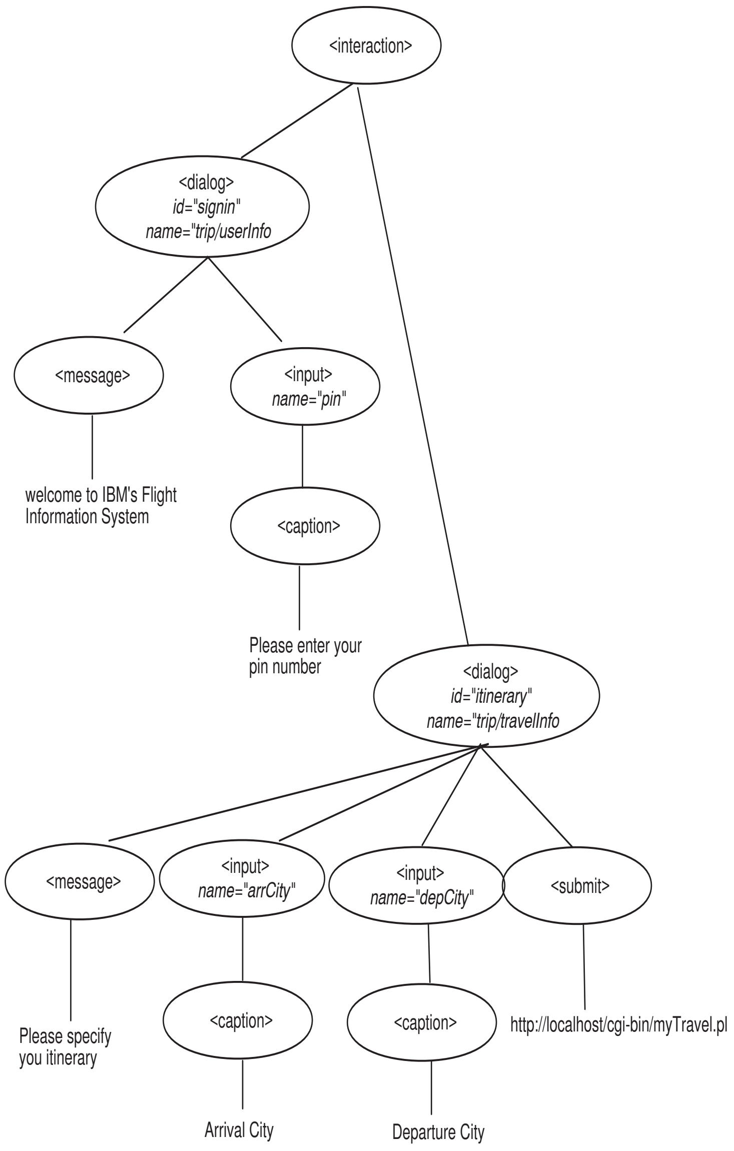
Figure 1 presents a portion of the traversal tree for a simple airline reservation system in which a user can sign in by entering his pin number, then query they system for a list of available flights matching his itinerary.

The portion of the iML application for the example shown in figure 1 is:

```
<?xml version="1.0" encoding="UTF-8"?>
 <iml name="travel" version="1.0">
  <interaction id = "travel" name = "trip"
               model_ref="TravelDetails">
   <dialog id="signin" name = "trip/userInfo">
    <message>
      Welcome to IBM's flight information system
    </message>
    <input name="pin">
     <caption> Please enter your pin number </caption>
    </input>
   </dialog>
   <dialog id="itinerary" name = "trip/travelInfo"
          action=''submit''>
    <message> Please specify your itinerary </message>
    <input name="arrCity">
     <caption> Arrival City </caption>
    </input>
    <input name="depCity">
      <caption> Departure City </caption>
    </input>
    <submit target="http://localhost/cgi-bin/myTravel.pl''>
   </dialog>
  </interaction>
 </iml>
```

The user interaction is described using the conversational gestures shown in figure1, extended of course to reflect all the other fields we need as airlines, dates etc... Suppose the application instance data consists of the strings: $pin, $arrivalCity, $departureCity, $arrivalTime, $departureTime, etc...

The controller conducts user interaction by traversing the nodes of the tree in-order, selecting a set of gestures and rendering the selected set to the modality specific views.



<interaction>
<dialog> id="signin" name="trip/userInfo
<message>
<input> name="pin"
welcome to IBM's Flight Information System
<caption>
Please enter your pin number
<dialog> id="itinerary" name="trip/travelInfo
<message>
<input> name="arrCity"
<input> name="depCity"
<submit>
Please specify you itinerary
<caption>
<caption>
http://localhost/cgi-bin/myTravel.pl
Arrival City
Departure City

Figure 1: iML interaction tree

The manner in which gestures are selected determines dialog strategy. For example, we can traverse the entire tree and generate the modality specific views thus simulating a mixed-initiative dialog. Alternatively the controller might traverse the tree sequentially to produce directed dialog. Note that the controller can synchronize application state across the various views independent of the dialog strategies used.

Consider the following application dialog flow: the user is presented with the welcome message and is asked to login. Upon successful login, he is presented the flight reservation dialog. We achieve this by visiting and rendering the subtree rooted at the *signin* dialog. Once the input field $pin number has been acquired and verified, we then proceed to the *itinerary* dialog node where we render all nodes in this subtree. DOM events are sent to the controller for each focus and input field update. The controller uses these DOM events to maintain the application state. Once all input fields have been acquired, the controller submits the instance data model to the application backend.

6. CONCLUSION

The framework described enables the rapid development and deployment of synchronized multimodal applications. Providing a high-level description of the user interaction allows us to define modality-independent conversational gestures that can be appropriately combined to express the application's dialog flow. These gestures are used to populate the various components in the application data model. The interaction with the user is conducted through the controller that maintains and updates the data model. It is also responsible for reflecting these updates on all available views.

There are many challenges in developing such frameworks, such as

Multi-level Synchronization Devices vary in sizes and shape. Thus, presenting information in one modality may be inappropriate in another. For example, selection rules differs depending on the device in use; for speech, listening to more than three choices becomes tiresome, for wireless devices presenting more than two or threes choices becomes cumbersome, while listing all choices on a desktop browser is easier and intuitive. Hence, synchronization between modalities may happen at different points in the application model depending on the type of device it's being presented on.

Generic Vs Specific Presentation Rules Certain applications many need to override the default presentation rules. For example, images are not typically rendered on a small device, but this needs to be overruled for interactive map applications.

Legacy Applications Converting legacy applications that have been authored without separating presentation from interaction, for example today's HTML pages, requires significant effort.

7. REFERENCES

[Con99] World Wide Web Consortium. Xsl tranformations. Technical report, W3C, 1999.

[Con00a] World Wide Web Consortium. Multimoal requirements for voice markup language. Technical report, W3C, 2000.

[Con00b] World Wide Web Consortium. Xforms:the next generation web forms. Technical report, W3C, 2000.

[KP88] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. Technical report, 1988.

[Ram97] T. V. Raman. *Auditory User Interfaces –Toward The Speaking Computer*. Kluwer Academic Publishers, August 1997.

[Ram98] T. V. Raman. Conversational gestures for direct manipulation on the audio desktop. In *Third Annual ACM Conference on Assistive Technologies*, pages 51–58, 1998.

[RC00] T.V. Raman R.B. Case, S. H. Maes. Ibm position paper for the w3c/wap workshop on the web device independent authoring. In *W3C/WAP joint Workshop on Web Device Independent Authoring*, October 2000. Bristol.

[SM00a] T.V. Raman S.H. Maes. Ibm position paper for the w3c/wap workshop on the multi-modal web. In *W3C/WAP joint Workshop on the Multi-modal Web*, September 2000. Hong Kong.

[SM00b] T.V. Raman S.H. Maes. Multi-modal interaction in the age of information appliance. In *ICME 2000*, July 2000. New York.

[W3C01] World Wide Web Consortium W3C. Document object model (dom) level 2. Technical report, W3C, 2001.