Mash-ups Considered Harmful! Composition and Choreography of Web Components

Charlie Wiecha, Rahul Akolkar, Rafah Hosn, Thomas Ling
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598,USA
{wiecha, rhosn, akolkar, ling}@us.ibm.com

ABSTRACT
We review several compositional-approaches to web application design – including multimodal programming models such as XHTML+Voice and common AJAX design patterns used in Mash-ups today. We propose a factoring of Mash-ups in order to support a looser coupling between independently-authored components along with a new XML-based controller language under design in the W3C based on State Charts from UML 2.0. Finally, we propose a formalization of web components, leveraging XForms, allowing the packaging of mash-ups into reusable web application components.

INTRODUCTION

The Web originated as a means for publishing hypertext documents; using HTTP and HTML, it created a universal platform for deploying user interaction. As the Web moves out of our desktops and into our pockets, we have seen increased fragmentation; today, Web authors are forced to choose among client-side technologies such as Flash or complex scripting when deploying to *rich* clients and customized applications optimized for a particular mobile device.

The primary motivation behind our work is to regain some of the lost ground with respect to deploying *universally accessible* Web content, where universal access includes *all* of the Web's original advantages: accessible to markup authors, accessible from a variety of end-user devices, and accessible to users with ever-changing needs and abilities.

Thus, we aim to enable Web applications that are easy to author and deploy, and create rich end-user interaction that degrades gracefully. Toward this end, we propose an extension of XForms, called XML Application Components (XACs), that collects the concepts of composition, integration, and late binding into a concrete proposal for a simplified and more functional web-based programming model. Along the way, we bring back some of the Web's core principles such as easy to author markup that is universally accessible creates the show-source network effect.

CURRENT WEB COMPOSITIONAL MODELS

Building web applications by composition has a long history, of course, including multimodal interaction. Figure 1 shows the XHTML+Voice programming model which composes GUI and Voice widgets using XML Events to synchronize state and focus changes between the two components. Targeting current XHTML markup, and hence lacking a shared data model, X+V applications will suffer the $O(n^2)$ complexity of direct widget to widget wirings.

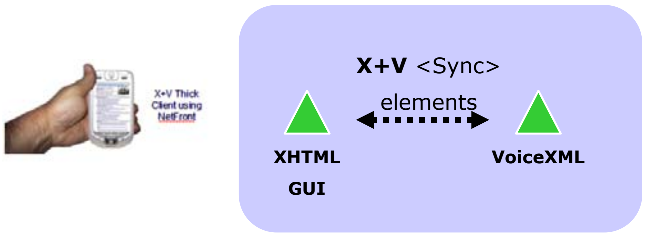

Figure 1: Multimodal widget to widget wiring

Mash-ups, such as the Google maps overlay with Dublin real-time commuter train data shown in Figure 2, below, have gained significant attention recently as illustrating the potential for highly interactive UIs on conventional web platforms. Notice, however, that train data is still passed to the map component by direct widget to widget wiring – hence suffering the same $O(n^2)$ blowup as in the X+V example above. Further, the mash-up component is tightly coupled to the map component – with its own interaction logic intermixed with the train to map wiring logic. We seek a looser coupling programming model in which each component can be separately authored and then brought together and coordinated independently.

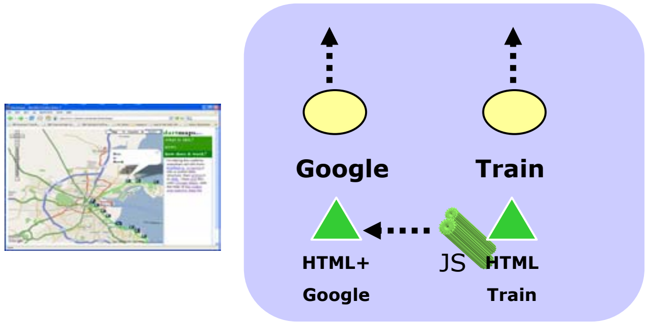

Figure 2: AJAX widget to widget wiring

Copyright is held by the author/owner.
UIST'06, October 15–18, 2006, Montreux, Switzerland.

TOWARD A NEW WEB COMPONENT MODEL

XML Application Components (XAC's) is a framework for authoring, composing and deploying XML Web applications. It builds on existing Web infrastructure including XForms, XHTML and XML Web services to enable component based Web applications. XML Application Components (XACs) encapsulate fragments of XHTML and XForms markup that create and bind a custom user interface to an XML data instance. XACs can be instantiated and configured to author higher-level applications rapidly. XACs can be composed to create larger applications. XACs publish an XML structure to their caller, and the type (i.e., the shape of the XML structure returned by that component) is exposed via the component's public data models. Public data models are designed to be both machine and human consumable and serve as the primary component API.

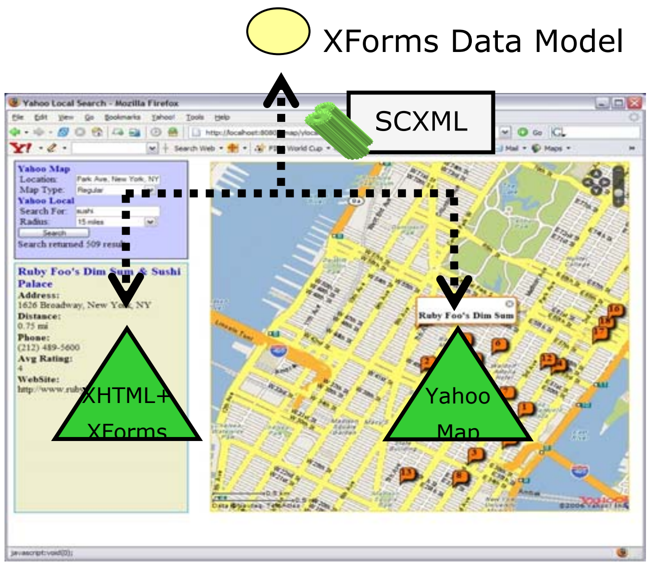

Figure 3: Widget to model binding with SCXML Controller

In the figure above, all UI components – whether built-in atomic XForms widgets or composite XACs, are wired to corresponding data model elements rather than cross-wired to each other. Immediately, this reduces the complexity of UI wiring to be linear in the number of widgets rather than quadratic. Further, each UI component is independently authored and assembled as a data model listener -- perhaps with additional XML event-based choreography as described in the next section.

We explore multiple languages for cross-component choreography, including the State Chart XML language emerging from the W3C and co-edited by one of us [1, 2]. A visual editor for SCXML building from IBM's Rational Software Architect tools platform, is shown in Figure 4, below.

Leverage Legacy: XACs can leverage existing Web applications such as Yahoo Maps or GMail without regard to their underlying implementation technique; existing Web applications can be adapted by defining a light-weight XAC adapter that publishes an appropriate data model for the underlying application and implements the XAC contract. The Yahoo map component in Figure 3 is one such "legacy" web component – and functions in our environment by means of an XBL wrapper assuring it responds to and raises appropriate XForms events to function as an XAC component.

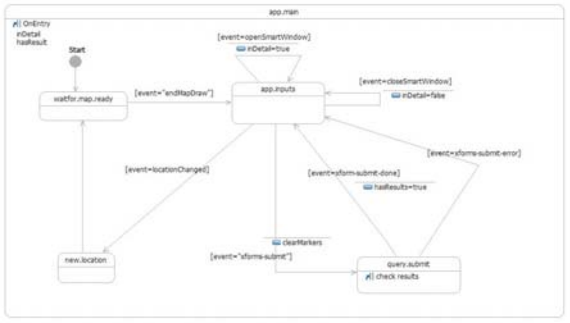

Figure 4: State chart XML (SCXML) controller

Enable Late Binding: XACs encourage late-binding; final presentation (including visual or auditory styling), as well as interaction behavior (i.e., the event handlers that are active in a given interaction environment) may be augmented and configured on the end-user device. Interpreting high-level, intent-based XML markup at the final step enables dynamic behaviors not normally possible in approaches that rely on server-side render kits for device adaptation.

Componentizable: XACs are themselves componentizable; i.e., once authored, an XAC application can itself be re-used as a XAC component by exposing an appropriate public data model. Thus, XACs close the loop between Web Services and Web Applications; XACs once published to the Web can be accessed as Web Services, and available Web services can be exposed as XAC components.

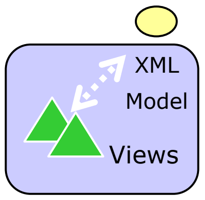

Figure 5: XAC republishes two views and public model as new component

REFERENCES

1. State Chart XML (SCXML): State Machine Notation for Control Abstraction, W3C Working Draft 24 January 2006: http://www.w3.org/TR/scxml/

2. Apache Jakarta Commons SCXML: http://jakarta.apache.org/commons/scxml