IBM Software Group

Rich Web Application Backplane

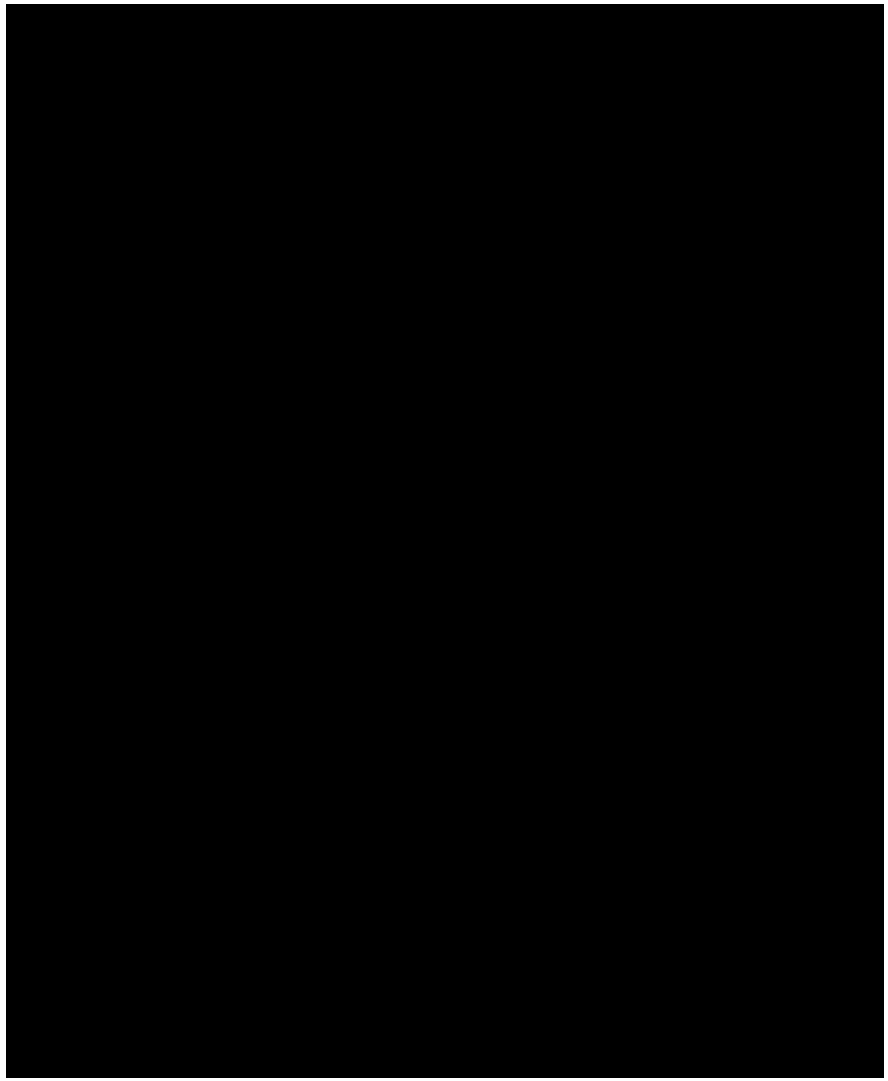
John Boyer

Lotus. software

e business software

One way to look at it…

- Markup delivered by a web application today must juggle hardening requirements…
- Layout and rendition of text, images, interaction controls
- Dynamic value calculation, content availability, datatype checking, process flow
- Enrich user experience with updates from web services
- Dynamic yet accessible
- So take a look inside and…

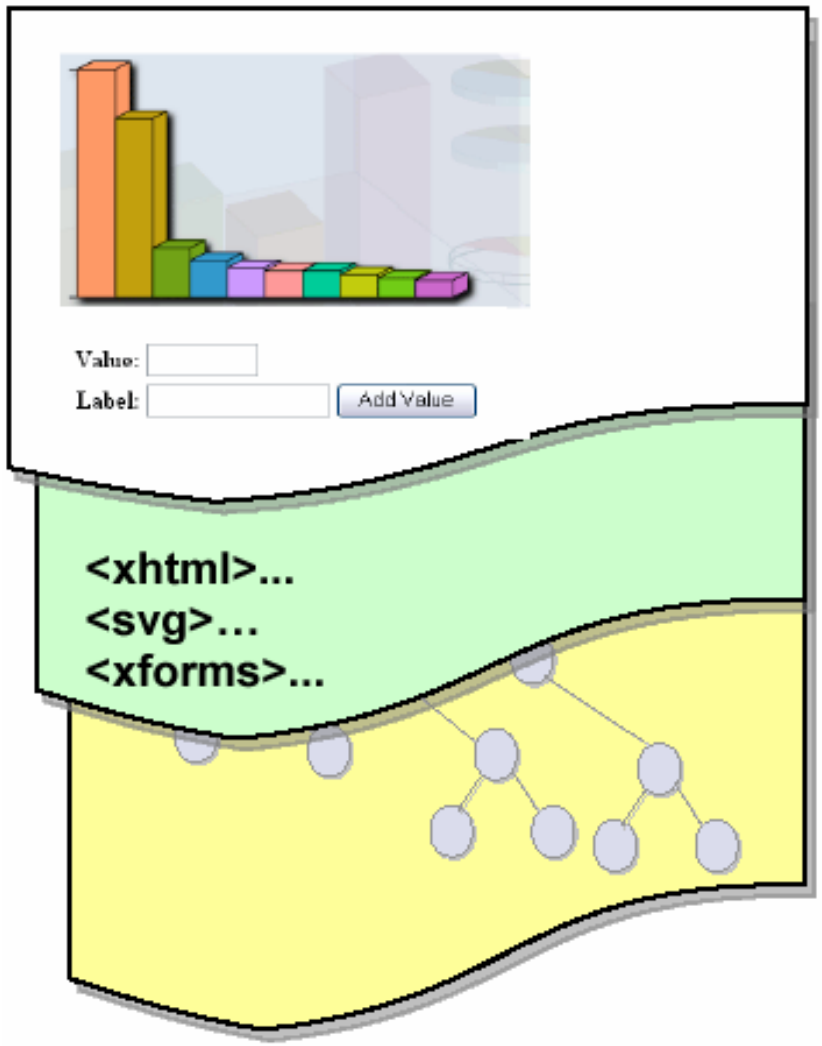


Current state of the art…

- Javascript/AJAX ad hoccinni in a tomato tag soup sauce with a sprinkle of herbs and CSS spices.
- Persisting this “assembly language of the web” programming model breaks the future of the web.
- To keep up, we need an upgraded methodology that can take us from mash-up to hook-up of componentized web applications

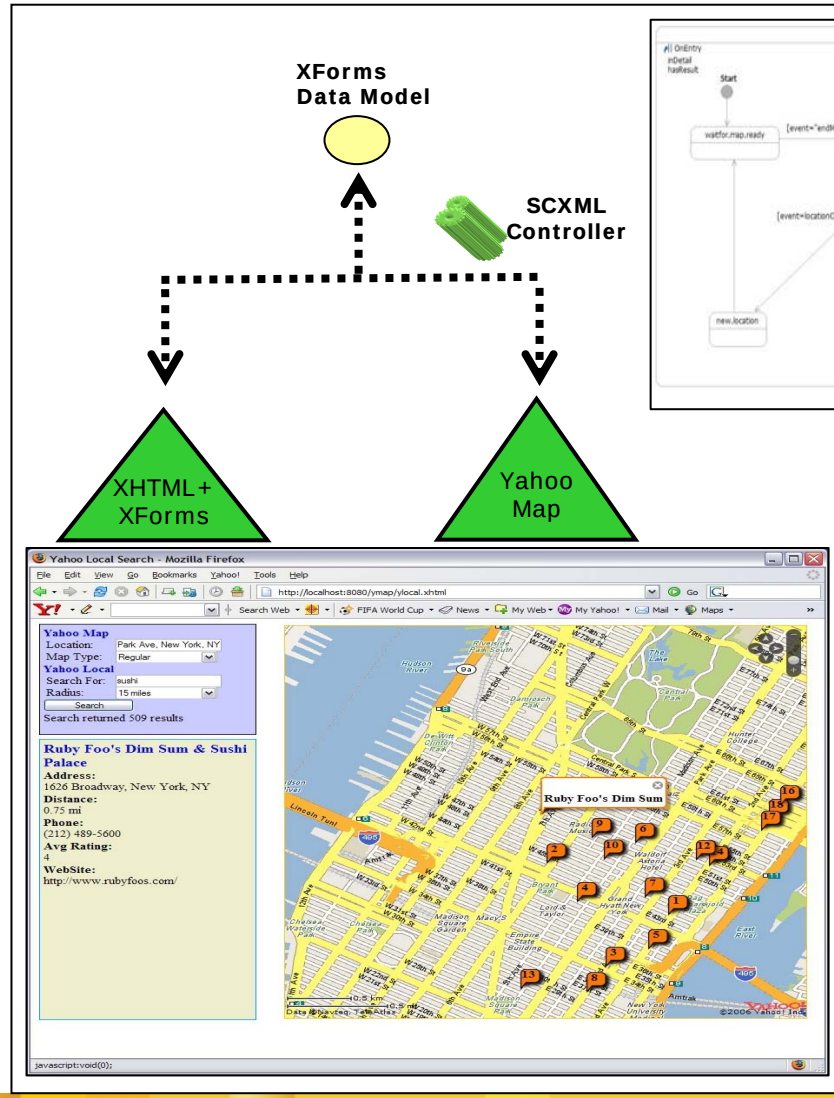
Rich Web Application Backplane

- Create a consistent model for mixed markup authoring and reliable component hook-up
- Reduce duplication and non-interoperability of the same functionality in many markups
- Maximize information flow among components to address interactivity and responsiveness requirements
- Address accessibility needs of dynamic web content



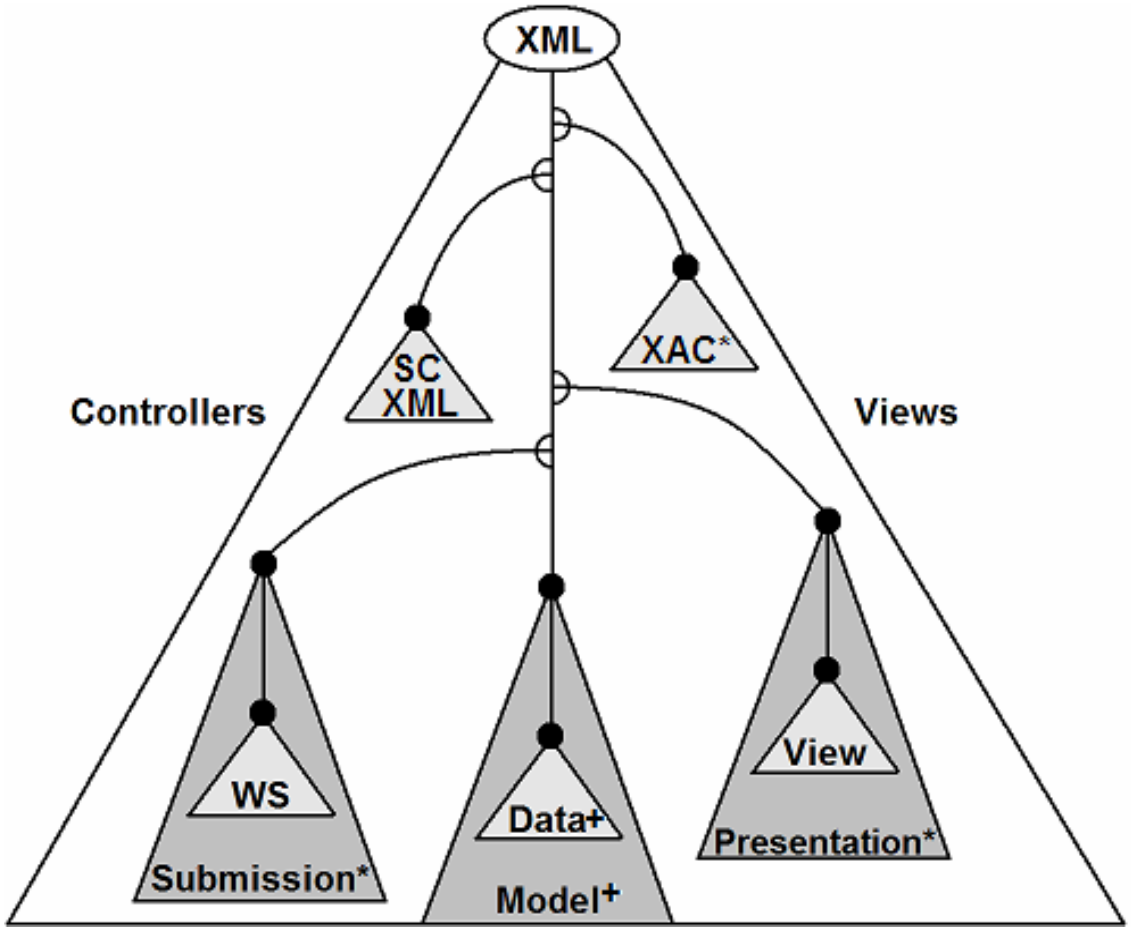
The “Hello, world!” of the Backplane

- Search box elements and Yahoo map component “hook-up” to backplane – vs. mash-up to each other
- State Chart XML Controller can manage cross-component coordination



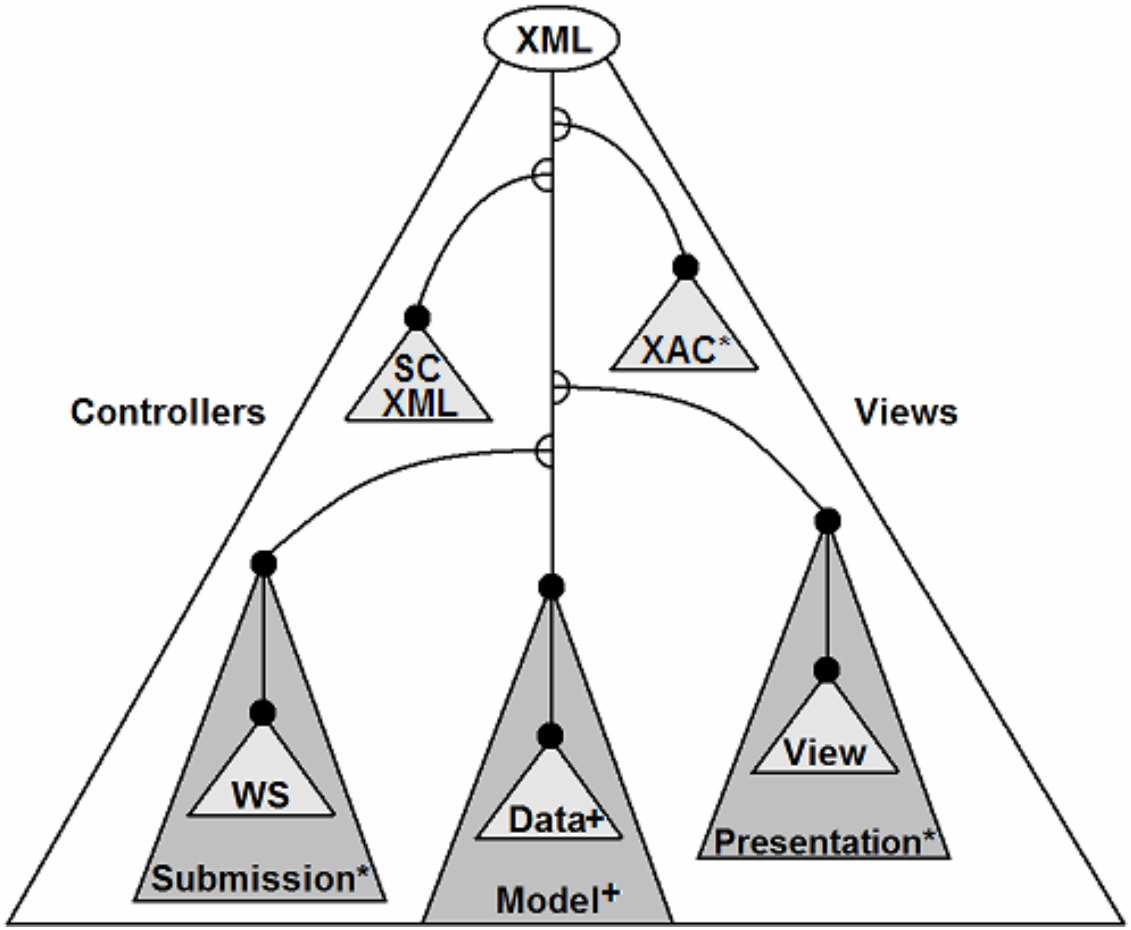
Intelligent, Componentized Interaction with Events, I

- Data access bindings allows views and controllers to obtain and manipulate data in a common way
- Results of mutations are communicated by events to bound components.
- Components may also learn of and respond to mutations from events bubbling up from data.



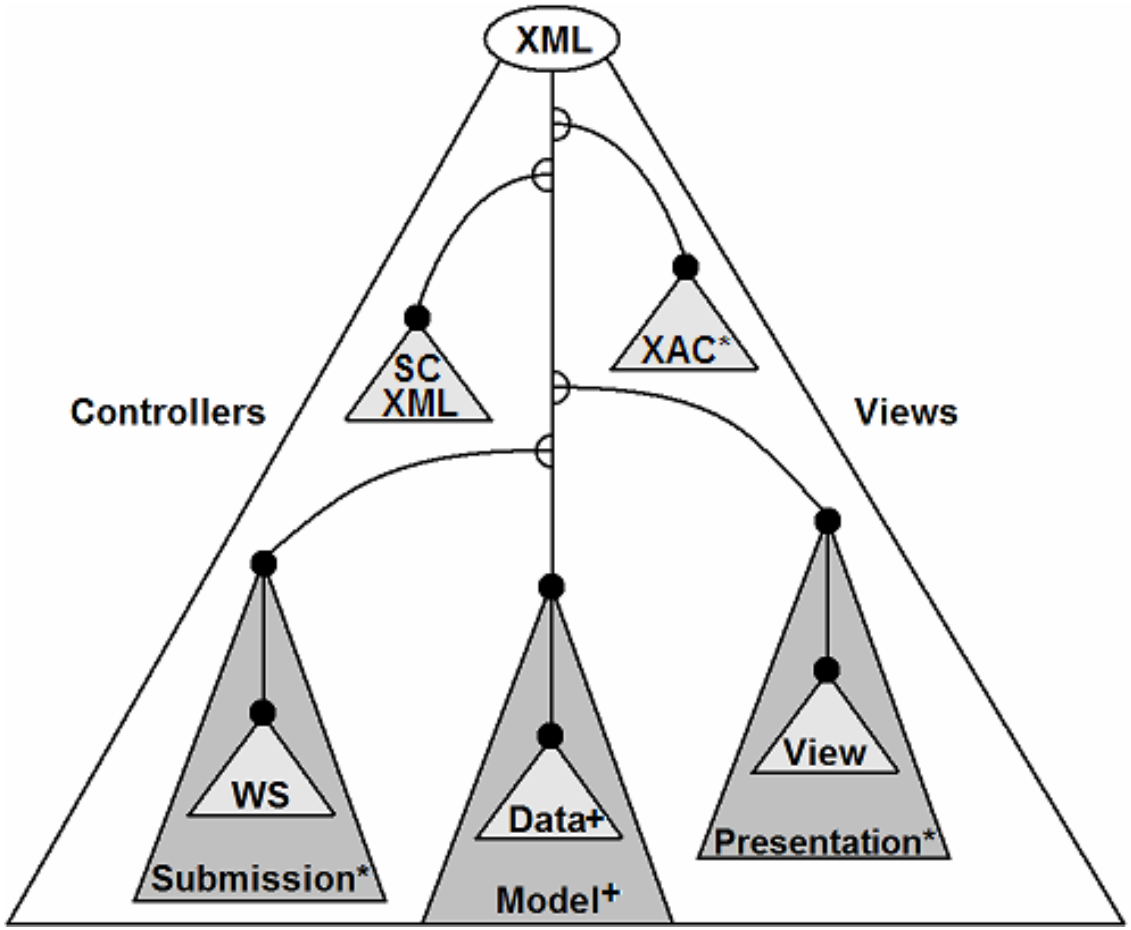
Intelligent, Componentized Interaction with Events, II

- Controllers like SCXML may trigger changes to presentational views and XACs based on data changes
- Controllers like web service submissions may trigger massive changes to data, which then trigger changes to presentational views, XACs and even other controllers like SCXML



Example: Model Property Broker

- Model may annotate data with useful properties, e.g. data type, constraints, patterns, validity, relevance, and mutation locks
- Model properties consumable by controllers and views
- Model properties mutated by procedural *or* declarative means



Another Example: Submission Object

- Ability to send XML data to a service that expects structure according to some schema.
- Ability to select a portion of the data being manipulated by the web application
- Ability for data model annotations or properties to affect submission, e.g. data constraint conformance/validity
- Ability for data to affect the destination URI and other parts of HTTP submission like verb and headers.
- Ability to use XML result to replace data content within the current application
- Ability to send non-XML, receive non-XML, and even place the result within XML data.

Making Lasagna with the W3C Technology Stack

Interaction Namespaces and Frameworks

XHTML/CSS	SVG	VoiceXML	SMIL	XForms	XBL2	AJAX-based Composite Apps

Rich Web Application Backplane

Compound Document Framework	XML Data Objects And Bindings	XML Events And Handlers	XML Data Model Property Broker	Submission Object (XMLHTTP++)	Choreography (SCXML)	Application Component Model

Platform technologies

DOM Parser	DOM Events	XPath	Schema Datatypes And Patterns	HTTP, URL, IRI	Security	JS/AJAX