Morfeo
PROJECT

*http://morfeo-project.org*

# Layered UI and Adaptation Policies for Ubiquitous Web Applications

Centering ideas for upcoming standards

25 October 2007

José M. Cantera .- **Telefónica I+D** *jmcf@tid.es*

Morfeo
PROJECT

# Introduction

# Introduction

- Developing applications for the Ubiquitous Web is hard. Main reason:
  - (X)HTML is a general purpose language designed to create hypertext documents in the web, **but not for describing user interfaces.**
- Scripting and server-side technologies “have filled the gap”, but:
- Developers have always been demanding **more powerful abstraction mechanisms**. As a result, the market has responded with solutions:
  - Ajax Toolkits
    - Dojo, Yahoo, GWT, ...
  - Propietary, tag-based, higher-level abstraction layers
    - JSF, XAML, XUL, Laszlo, MSXML
- What about open standards? Alternatives (**all of them insufficient**):
  - XHTML + XFORMS + Javascript and/or DIAL
  - HTML 5 + Web Forms 2.0
- **New standardization efforts around layered-UI are needed**

Morfeo
PROJECT

# Why existing standards are insufficient?

Morfeo PROJECT

# XHTML + XFORMS + Javascript

- Absence of a complete set of advanced UI components
  - grids, trees, menus, toolbars, progress bars, ...
- No rich set of containers and layout abstractions.
  - Developers end up using tables for layout → **Not mobile nor accessible**
- **No expression language** notation for addressing objects
  - Server-side scripts and Javascript → **Lots of code to maintain**
  - XFORMS only works with XML data models and XPath.
- They don't separate bindings, relevancy, formatting, validations.
- **No standard APIs** for
  - XFORMS elements, model mutation, creation of extensions
- **No sufficient** mechanisms for specifying **metadata** or hints needed for **adaptation to multiple delivery contexts**

# DIAL + DISelect

- "A profile of XHTML 2" (DIAL) + DISelect:
  - Goal: Content adaptation at server-side.
  - **It inherits all the problems** coming from (X)HTML and XForms.
- **Some UWA use cases where DIAL + DISelect fails**:
  - Different layouts for different delivery contexts
  - Date or time input component, rendered as a calendar or clock .
  - Cool menus that degrade gracefully
  - Select or menu component rendered as a popup list, or as list of links, or as a clickable map depending on the delivery context (device input mechanisms and browser capabilities).
  - Big table or menu with dynamic contents that need to be paginated
  - A big form which has to be paginated and divided in two or more chained subforms.

# HTML 5 + Web Forms 2

- Web Applications specification developed by the WHATWG and candidate to be adopted by W3C
- **Partial enhancements** on
  - Validations
  - Repetition model
  - Extended elements (table, range, etc)
- Problems. **All inherited from (X)HTML** and more
  - Tag-soup reinvented → **Not ready for enterprise development**
  - **Backwards compatibility toll** and browser vendor biases
  - A rich component set is still missing
  - **Imperative** against declarative : Scripting is encouraged
  - 400 members to agree on something :(

Morfeo
PROJECT

# What is the trouble with existing, propietary solutions?

# AJAX Toolkits

- Usage of AJAX toolkits is not transparent to the developer.
  - They **encourage imperative programming** to the detriment of declarative formalisms
    - Example: declarative styling of components (CSS) no longer used.
- **Tons of Javascript code** leads to
  - bad performance and maintenance.
  - applications not accessible nor friendly for mobile adaptation
- They do not provide advanced standard UI mechanisms and formalisms (data binding, validations, formatting).
- **Extreme dependency on the selected AJAX toolkit**
  - Knowledge reuse and standardization are severely compromised.
  - Writing new user interface components or extending existing ones is a difficult and tricky task.

Morfeo
PROJECT

# Tag-based abstractions

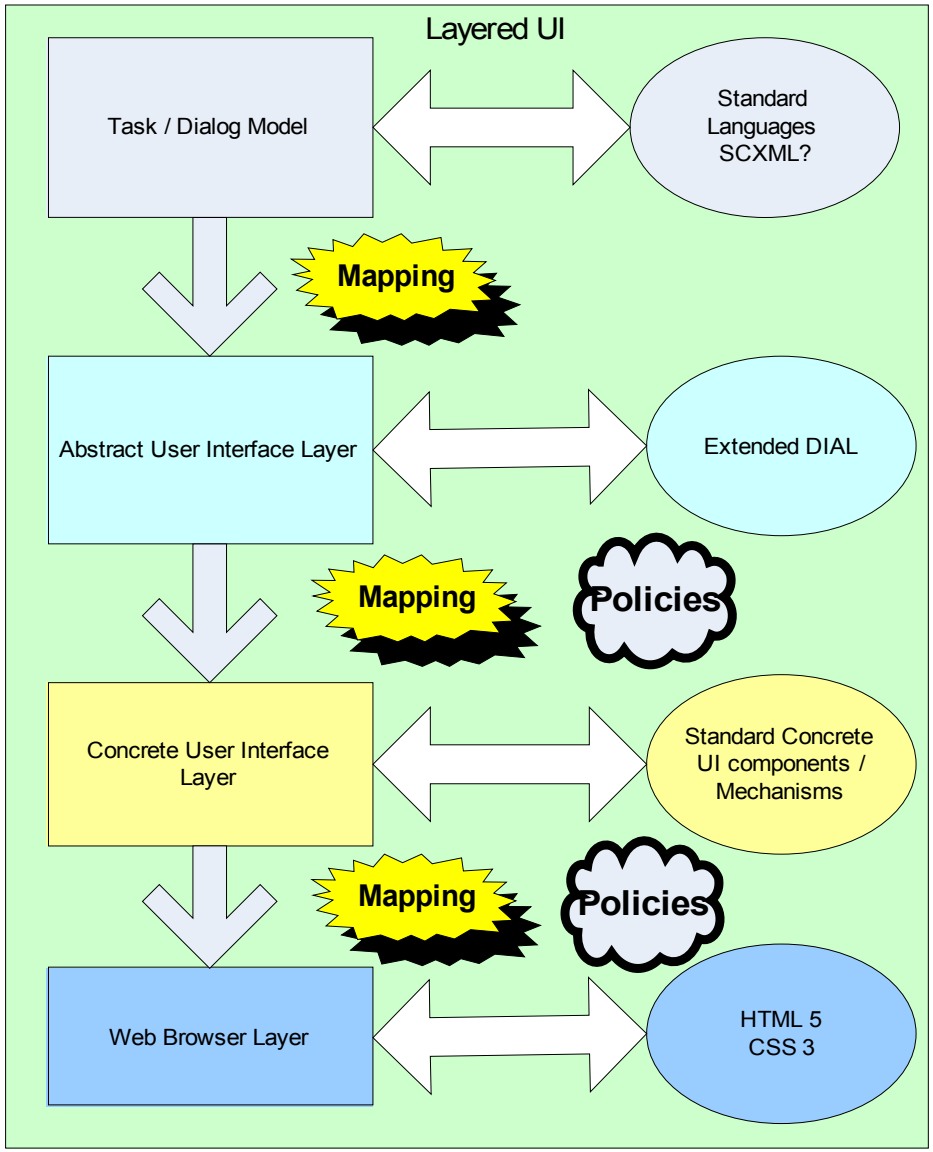
- A **myriad of technologies**, JSF, XUL, XAML, Laszlo, MXML ... **all suffering from the same problems:**
  - Platform dependency: Java, .NET, Flash
  - Openness : One implementation by one organization
  - Desktop-orientation: Device independency was not a design goal
  - Interoperability
  - Reuse of user interface components between the different languages.
- The open source community has started to understand the necessity of an open standard.
  - Apache XAP project, leaded by NexaWeb Technologies
    - It defines the XAL language to reduce JavaScript in AJAX applications.
    - It can work with different AJAX Toolkits (Component Bridge Pattern)

Morfeo
PROJECT

# Towards the adoption of Layered UI Architectures

Morfeo PROJECT

# Abstract vs Concrete UI (I)



Morfeo PROJECT

# Abstract vs Concrete UI (II)

- The aforementioned tag-based abstraction technologies deal with the concrete UI representation but not with the abstract UI
  - This leads to problems in the presence of multiple delivery contexts
- DIAL might evolve towards an abstract UI language
  - We should think of what is missing in DIAL for being an abstract UI language
- UWA should work in standard mechanisms for mapping between the abstract UI and the concrete UI
  - Via adaptation policies
  - Setting up layers that are on top of web browser technologies
- In the long term, UWA group should think of the standardization of upper layers such us task-based models and dialogue-description languages

Morfeo PROJECT

# Mapping abstract and concrete UI (I)

- The mapping between abstract and concrete UI determines how an abstract component is finally 'rendered' in a delivery context
- For multiple delivery contexts it can be needed multiple mappings
  - Rendering / mapping / binding policies (a name should be chosen)
- In MyMobileWeb the mapping between the abstract and concrete user interface is done by means of a CSS property that can take different well-known values. Examples:
  - A select element (in the abstract UI layer) can be rendered as a set of radio buttons, as a pull down list, or as a list of links
  - A command element can be rendered as a link or as a button
- The mechanism is similar to the 'appearance' property specified in the CSS 3 Basic User Interface Module
  - http://www.w3.org/TR/css3-ui/

Morfeo PROJECT

# Mapping abstract and concrete UI (II)

- Changing the mapping between different delivery contexts is very simple
  - Just setting up different CSSs using Media Queries (executed at server side if necessary)
- The CSS-based approach is quite simple and useful but
  - It is not very flexible for specifying presentation properties at the level of the concrete UI, due to the lack of nesting in CSS (see example 1)
  - When the developer needs customized concrete UI representations it fails, although technologies like XBL can fill the gap
  - There is a mixing of layers (browser layer and UI definition layer)
- Example 1
  - If the command is mapped to a link I want the link font to be normal
  - If the command is mapped to a button I want the button font to be bold
  - This problem can be workarounded using CSS pseudo-classes but it is not very flexible

Morfeo PROJECT

# Mapping abstract and concrete UI (III)

- Proposal
  - Standardize common well-known mappings
  - Standardize how to create mappings with SVG, SMIL, etc.
  - Standardize how to extend common mappings in a flexible manner
  - Standardize how to create extended mappings
  - Standardize how to specify presentation properties at the level of the concrete UI
  - Standardize mechanisms for specifying mapping policies
- Issue:
  - Standardizing common mappings implies standardize concrete UI components

Morfeo
PROJECT

# Towards the creation of Adaptation Policies

# Adaptation Policies (I)

- Instructions given by the developer to guide the adaptation process through different delivery contexts
- Kind of policies
  - Styling policies
  - Layout policies
  - Rendering policies (mapping between the abstract and concrete user interface)
  - Content Selection policies
  - Pagination policies
  - ...

Morfeo PROJECT

# Adaptation Policies (II)

- For defining adaptation policies it is necessary to
  - Set up a common and extensible framework for adaptation policies
    - Issue: Should we follow a top-down approach or a bottom-up approach?
  - For each kind of policies define a “vocabulary of properties” that will be used for defining the policies
  - Have a language that allow to choose between different policies for different Delivery Contexts.
    - DISelect might  be the language

Morfeo
PROJECT

# Layout policies (I)

- An explicit work item present in the UWA charter
- Preliminary work studying
  - SMIL 2.0 Layout Module
    http://www.w3.org/TR/2005/REC-SMIL2-20050107/layout.html
  - NexaWeb's XAL declarative language (layouts) •
    http://dev.nexaweb.com/home/us.dev/index.html@cid=1446.html
  - CSS 3 Basic Box Module
    http://www.w3.org/TR/2007/WD-css3-box-20070809/
  - CSS 3 Grid Positioning
    http://www.w3.org/TR/2007/WD-css3-grid-20070905/
  - CSS 3 Multicolumn Layout http://www.w3.org/TR/css3-multicol/
  - CSS 3 Advanced Layout Module
    http://www.w3.org/TR/2007/WD-css3-layout-20070809/

Morfeo
PROJECT

# Layout Policies (II)

- Conclusions of the preliminary study
  - Too many CSS 3 overlapping specs, a nightmare
  - SMIL 2.0 introduces good ideas but it is very limited
- Design goals for layout policies
  - As declarative as possible, while leaving doors open for complex situations
  - Authoring simplicity for common layout situations
    - Grid, horizontal, vertical
  - Independent of future browser support of CSS layout specs

Morfeo PROJECT

# Layout Policies (III)

- Options for progressing
  - Standardize a set of CSS properties similar to those already present in MyMobileWeb
    - Issue: We might need to deal with CSS WG
  - Make layout policies independent of CSS and define a new formalism
    - Simple and extensible
    - Issue: What will be the role of CSS here?
- Option 1 provides a fast lane for layouts in DIAL
- Option 2 might take more time to develop and standardize
- In both cases, We will need to define the relevant layout properties
  - We do not need to reinvent the wheel, adopt something

Morfeo PROJECT

# Layout Policies based on CSS

- Create a new 'layout' pseudoelement to deal with layout
  - http://www.w3.org/2007/uwa/wiki/Layout-containers
-

Morfeo PROJECT

# Layout Policies based on an external mechanism

- Define a standard mechanism for assigning layout to a DIAL authored unit
  - Ex: A processing instruction or a <link rel="layout" ... />
- Define a standard format for declaring layout policies based on standard layout properties
- Define a mechanism for selecting a layout depending of the delivery context
- Issue: What properties apply to the layout and what apply to an specific component?
- Issue: What would be the role of CSS in this approach?

Morfeo
PROJECT

# Towards the separation of selection conditions from page structure

Morfeo PROJECT

# Separating DISelect from DIAL

- Initial ideas
  - http://www.w3.org/2007/uwa/wiki/DISelect-Separation

Morfeo
PROJECT

# Proposal of work items for UWA

Morfeo PROJECT

# Work items proposal (I)

- DISelect
  - Decoupling mechanisms between DISelect and DIAL
  - DISelect as the basic policy specification language
  - Expression Language extensibility in DISelect / XForms
- DC XPATH Functions
  - Enrich the set of functions for advanced adaptation policies
- Abstract UI
  - Evolving DIAL towards an abstract UI language
  - Leverage XForms validations and binding (ex. allow JSON)
- Concrete UI
  - Languages for specifying the concrete UI layer
  - Standard set of concrete UI components
  - Standard mechanisms for concrete UI layer extensibility

Morfeo PROJECT

# Work items proposal (II)

- Adaptation Policies
  - Standard framework for adaptation policies (extensible to allow others to create customized-policies)
  - Layout policies spec for UWA layouts
  - Pagination policies
  - Policies for mapping between the abstract and concrete UI
  - Styling policies. How to change the skin of the Web App in different delivery contexts
  - Content Selection policies
    - Leveraging DISelect
- User interface behaviour
  - Standardization of a language for specifying the behaviour of the UI
  - Leveraging / Extending SCXML

Morfeo
PROJECT

# Work items proposal (III)

- Events
  - Mappings between events at different layers
  - Standard events in different layers
- APIs
  - DIAL DOM-style API
  - Behaviour component (SCXML) API
  - Model APIs (setValue, getValue)
- Future
  - Task model languages
  - Dialog model languages
  - Mapping between the task model and the abstract UI

# Conclusions

# Conclusions

- There is a **gap** wrt open, standards-based declarative models for UWA and ,in particular, **in the user interface area**
- **Existing open standards are insufficient**.
- **AJAX and propietary tag-based** abstractions are more and more popular but create and **extreme dependency on specific toolkits.**
- There is an opportunity for pushing forward the layered-UI approach exploiting the advantages that it presents when dealing with multiple delivery contexts
  - This should be done incrementally, first introducing the abstract UI vs the concrete UI approach and then going beyond, introducing task and dialog models for UI (three-layer approach)
  - Issue: What happens in those cases where people want to develop at the concrete level?
- There are a bunch of technologies that might be standarized by the UWA WG
  - We do need to set up a roadmap and prioritize

# Thank you for your attention!