José M. Cantera

Telefónica I+D

MORFEO MyMobileWeb
http://mymobileweb.morfeo-project.org

Model-Based UI W3C XG

Telefónica I+D's Input

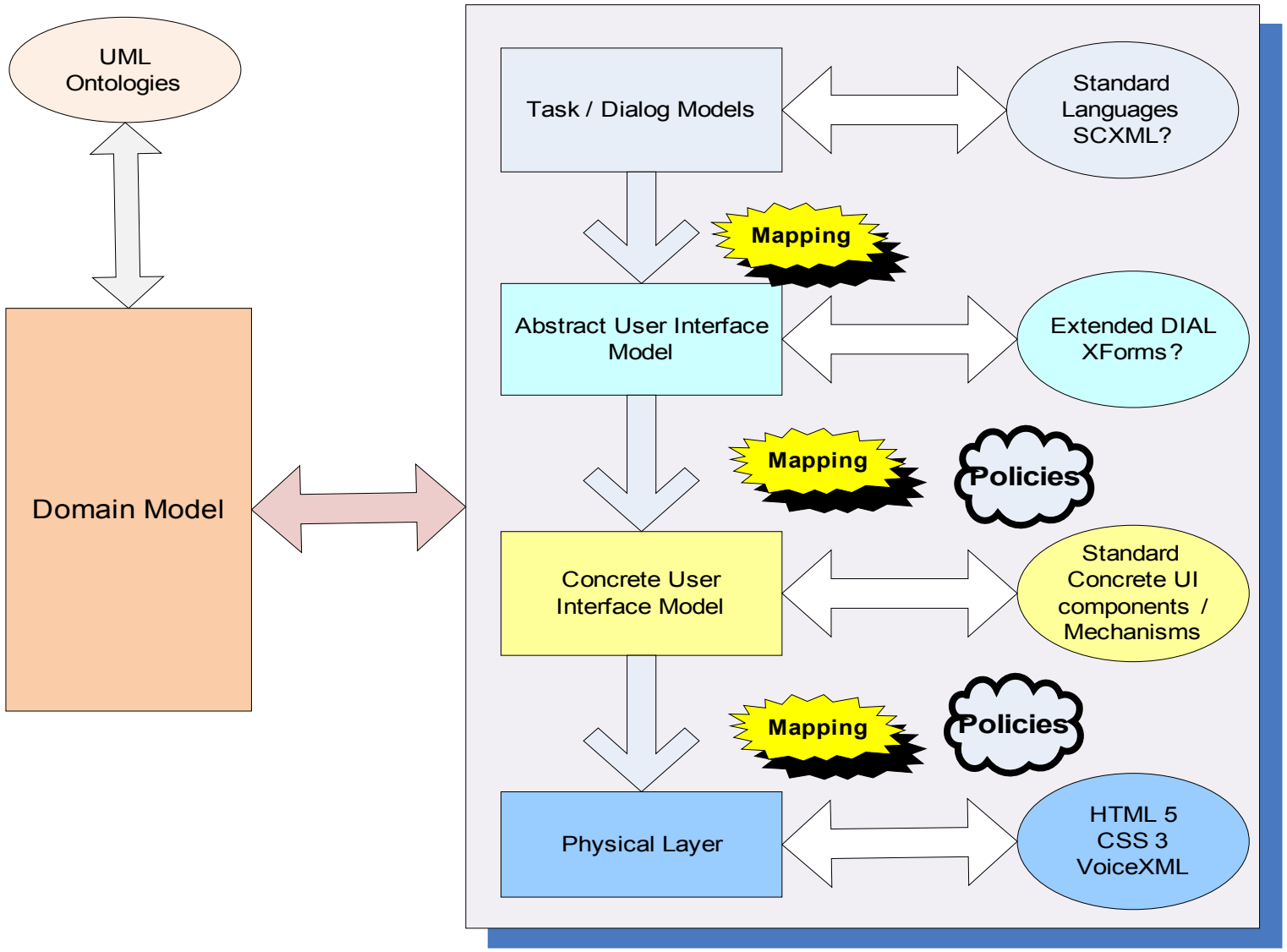
MINISTERIO DE INDUSTRIA, TURISMO Y COMERCIO

FIT-350405-2007-1
FIT-350401-2006-2

Background

- 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 intended to work on multiple devices or modes of interaction**
- Developers have always been demanding **more powerful abstraction mechanisms**. As a result, the market has responded with declarative and imperative 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
- There is a big yet-to-be-explored potential for declarative authoring languages for UI
  - Applying existing research results on model-based UI dev.

Model-Based UI .- Overview



MORFEO MyMobileWeb

Abstract vs Concrete UI (I)

- 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 be the starting point towards an abstract UI language
  - We could think of what is missing in DIAL for being an abstract UI language
  - DIAL modularization can save us the day
- We can 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, we should think of the standardization of upper layers such us task-based models and dialogue description

Mapping Abstract - Concrete UI (II)

- The mappings 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/

Mapping Abstract - Concrete UI (III)

- 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

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 MyMobileWeb

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

Possible work items for the XG

- Brainstorming
  - Make XForms more abstract
  - 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
  - Reuse ARIA work?

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 model-based 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

MORFEO MyMobileWeb

Thank you for your attention

http://mymobileweb.morfeo-project.org