MORFEO PROJECT

IDEAL

TSI-020301-2008-25

A Language for Authoring Context-Aware Ubiquitous Web Applications & Content

José M. Cantera Fonseca
jmcf@tid.es

Introduction

- Traditional user interface development approaches are insufficient for supporting the new generation service front-ends.
  - Oriented to specific platforms, devices or modalities
    - normally a PC device and GUI modality (big screen mouse & keyboard)
  - Imperative, platform and language-driven development style
    - increases the effort and time to market suffers
    - UI designers cannot fully concentrate on the real application requirements.
  - Do not support the development of context-aware UI / contents
- Example : AJAX-based UIs
  - HTML
    - neither device nor modality independent
  - Javascript
    - both device dependent and imperative
  - Impossible to adapt to multiple devices / modalities
  - Accessibility is also an issue
- There is a big yet-to-be-explored potential for declarative authoring languages for UI
  - Applying existing research results on model-based UI dev.

State of the Art

- 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, ...
      - Vendor-Locking, imperative, oriented to an specific problem
  - Proprietary, tag-based, higher-level abstraction layers
    - JSF, XAML, XUL, Laszlo, MSXML
    - They normally reinvent the wheel for the 80% of the functionality offered
    - There is a remaining 20% of functionality that might be interesting
    - Deal with the concrete UI (typically for PCs) representation but not with the abstract UI
- Existing standards are insufficient but
  - can be “wisely” extended with additional functionalities
  - they can be a good starting point for new languages
  - Promote reuse and interoperability

Authoring vs Delivery

- Developers often don't understand the difference between
  - Authoring formats
    - A format intended for humans that provides as many abstraction levels as possible
  - Delivery formats
    - A format to be interpreted directly by the client platform (the web browser, for example)
  - HTML 4 / 5, CSS, etc are delivery formats
    - Unfortunately in many occasions are used as authoring formats
- IDEAL is an Authoring Format easy-to-be learned by typical Web Authors
  - Simple things should be easy (80%)
  - Complex things should be possible (20%)

IDEAL Language (Overview)

- A Declarative ***Authoring Format*** for creating applications / content on the Ubiquitous Web
- Defined under the Eureka-CELTIC MyMobileWeb
  - After 4 years of engineering experience
- Main characteristics
  - Modular and Extensible
    - “XHTML Family Module Conformant Language”
    - Extensions must follow XHTML Modularization 1.1
  - Not too abstract nor too concrete (good compromise)
  - There is no wheel reinvention
    - Reuse an existing standard whenever it is feasible
  - Initially thought for visual modality but
    - prepared to evolve to support any kind of modality
  - Targeted both to content and applications
  - IDEAL is **not** XHTML 1, XHTML 2, HTML 4, HTML 5 …
    - A superset of W3C’s Device Independent Authoring Language, DIAL

IDEAL Language (History)

- IDEAL v 1.0 was designed to deal with the problem of creating web applications that adapt to multiple Delivery Contexts
  - Based on simple containers and interactors
    - Select, menu, table,
  - Prepared to be used mainly over the Java Platform
  - CSS was used both for theming and guiding the adaptation process
- IDEAL v1.0 has a rendering engine that generates markup code for the Mobile Web platform
- IDEAL v1.0 lessons learned
  - The need of modularization
  - The need of extensibility (at the language and at the toolkit level)
  - The need to be language neutral
  - Whenever is possible reuse existing markup
  - CSS cannot be sufficient to deal with adaptation policies
  - The need to support at the same time content-driven and application-driven features
- IDEAL v2.0
  - The new generation of the IDEAL language
  - Result of our previous experience

IDEAL Language 2.0 (II)

- It clearly separates
  - UI Structure
  - UI Components / Binding with the target toolkit, for example, DoJo
  - UI Behaviour
  - UI Data / Content Model Definition & Restrictions
  - UI Data / Content Binding (including repetitions)
  - UI Layout
  - UI Theming (Look & Feel)
  - UI Adaptation / Selection Policies (to deal with multiple DCs)
  - UI Accessibility
  - UI Content / Data Semantic Annotations
- Standards integrated
  - XForms
  - XMLEvents
  - Role Attribute / Access Module
  - DISelect
  - RDFa
  - XBL 2

MORFEO IDEAL Language

IDEAL Specification 2.0

- IDEAL Overview
  - Overview of the specification
- **IDEAL Core Language**
- IDEAL Concrete UI Language (reusing ARIA work and taxonomy)
- IDEAL Layout Definition
- IDEAL Adaptation / Transformation Policies
- IDEAL Themes & Look & Feel (leveraging CSS)
- IDEAL Semantic Annotations (extending RDFa)
- IDEAL Toolkit Bindings
- IDEAL Extension Guidelines (based on XHTML 1.1 Modularization)
- IDEAL Primer
  - General tutorial for developers with lots of examples
- IDEAL for AJAX developers
  - Tutorial targeted to typical HTML, Javascript, CSS developers
- IDEAL for Mobile Web Developers
- IDEAL for …
- ***Licensed under a Creative Commons 3.0 License (Attribution Non-Commercial)***

IDEAL Core Language 2.0

- A Working Draft is available at
  - http://mymobileweb.morfeo-project.org/specs/ideal
- Modules
  - UI Structure
    - Based on DIAL
  - UI Components
    - XForms + Extensions
  - UI Behaviour
    - XML Events
    - XForms Actions + Extensions
  - UI Data / Content Model Definition & Restrictions
    - XForms + Extensions (models based on objects are permitted)
  - UI Data / Content Binding
    - XForms + Extensions (Dotted expressions are permitted)
  - UI Accessibility
    - The XHTML Access Module
  - Selection to allow context-aware UIs
    - Based on DISelect

Adaptation Policies

- Instructions given by the author to guide the adaptation process through different Contexts
- Kind of policies
  - Styling / Theming policies
  - Layout policies
  - Rendering policies (mapping between the abstract and concrete user interface)
  - Content Selection policies
  - Pagination policies
  - Context-Aware Reordering Policies
    - Examples
      - Reorder a menu to give preference to user's tastes
      - Reorder a table depending on the location

Adaptation Policies

- For defining adaptation policies it is necessary to
  - Set up a common and extensible framework for adaptation policies
  - 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 Contexts.
    - DISelect might be the starting point for such language

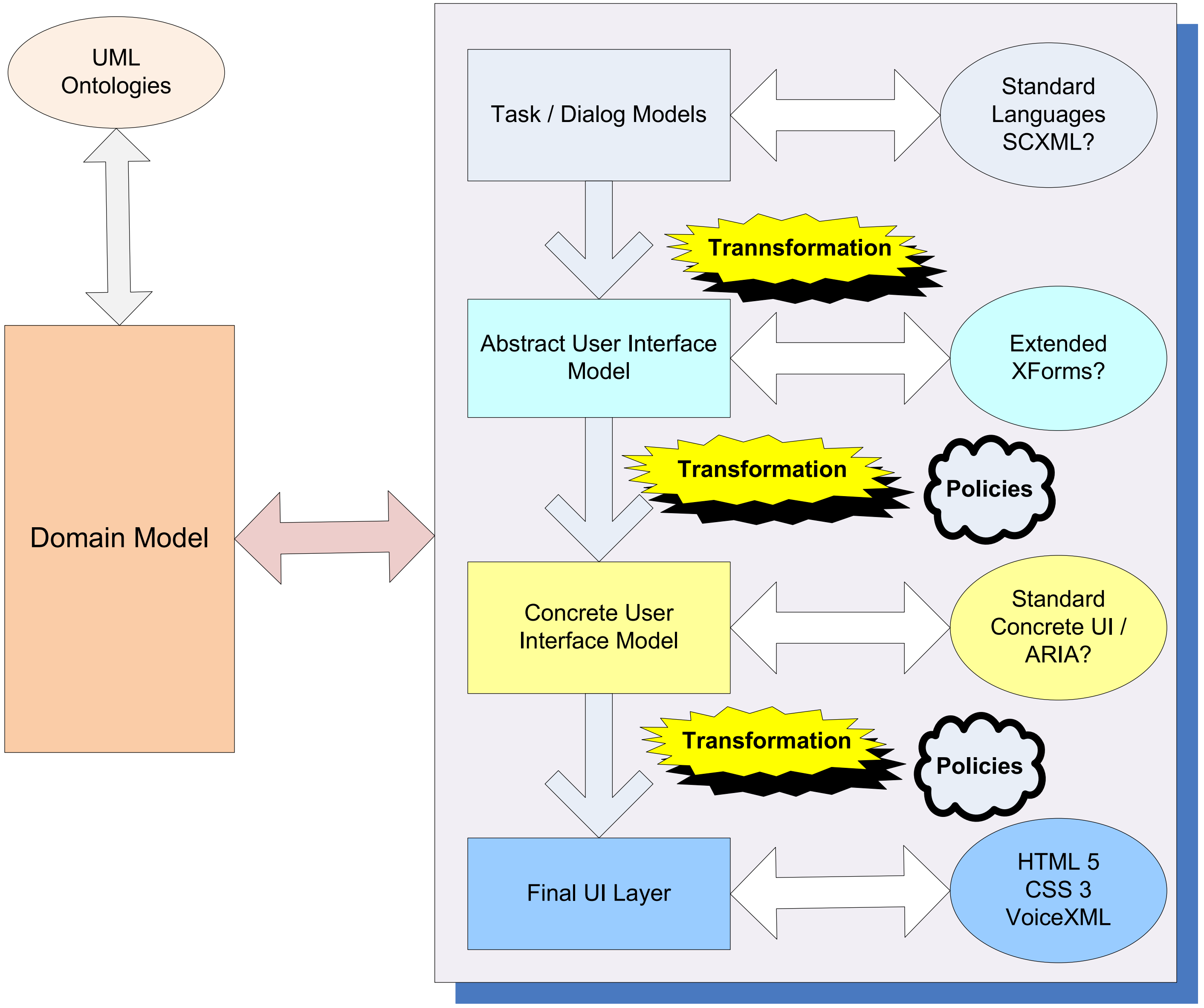
IDEAL 2.0 Examples

- Repeating structures
  - XML instance (e.g. table control)
    - View source code
  - JSON instance (e.g. ordered list control)
    - View source code
- Binding and validation
  - View source code
- Formatting
- Fragment inclusion
  - View source code
- Context-Aware Selection
  - View source code
- Events
  - View source code

IDEAL Roadmap

- Short Term Actions
  - IDEAL Core
    - A complete and stable version of the spec will be available by October 31st
  - The rest of the IDEAL specs will be available by the end of the year
    - For some subjects it is needed to investigate more
  - Tutorial Materials are expected for Q1 2009
  - Implementation plans
    - A subset of the IDEAL Core Language is already implemented in MyMobileWeb v 3.2
      - The rendering engine generates markup for mobile web-enabled devices
- Future Targets
  - Re-Modularize following a layered approach
    - Task Models vs Abstract UI vs Concrete UI
  - Research towards Model-Based UI that allows to create compelling context-aware UIs

IDEAL Roadmap (Model-Based UI)



IDEAL Language

MORFEO