End User Development for Interactive Multi-Platform Applications

Fabio Paternò
fabio.paterno@isti.cnr.it
http://giove.cnuce.cnr.it/~fabio/
ISTI-CNR
Pisa, Italy
Interacting with chaos
Universal Usability

- Ever-increasing introduction of new types of interactive devices
- Applications often need to be accessed through different interactive devices
- How to support designers and developers?
- How to obtain interfaces able to adapt to multiple devices (any device) while preserving usability?
Why Model-based approaches?

- Highlight important information
- Help to manage complexity
- Useful to support methods
Possible Views of an Interactive Systems

- Task and object – *I want to select a work of art*
- Abstract Interface – *Single selection object with high cardinality*
- Concrete Interface – *List Interaction object with X elements*
- Code – *List object in Java or XHTML or ....*
The Framework

EUD-Net Workshop - October 6-7, 2003
Use of Reverse Engineering

Platform X

Tasks and Objects
Abstract UI
Concrete UI
Final UI

Redesign

Transcoding

Platform y

Tasks and Objects
Abstract UI
Concrete UI
Final UI

EUD-Net Workshop - October 6-7, 2003
# Examples of Platform-dependent tasks

<table>
<thead>
<tr>
<th>Desktop system</th>
<th>Mobile System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparing prices of flights and making reservations.</td>
<td>Checking status of a particular flight.</td>
</tr>
<tr>
<td>Gathering background on a company, including maps.</td>
<td>Getting driving directions to a company—while on the road.</td>
</tr>
<tr>
<td>Browsing medical information.</td>
<td>Monitoring a medical condition.</td>
</tr>
<tr>
<td>Reading a movie review and/or watching a trailer.</td>
<td>Purchasing a cinema ticket to avoid the line.</td>
</tr>
</tbody>
</table>
Task-related issues in multi-platform environments

- Same task on multiple platforms in the same manner
- Tasks meaningful only on a single platform type
- Dependencies among tasks performed on different platforms
- Same task on multiple platforms but performed in different manner …
Same task on multiple platforms with different user interface objects
Same task on multiple platforms with different task decomposition
Same task on multiple platforms with different temporal relationships among tasks

- Example: Enter reservation data
Dependencies among tasks performed on different platforms
Design of Nomadic Applications Current Practice

- **Manual solutions**, expensive
- **Style sheets**, partial solution
Approaches to multi-platform interface development

- The User Interface Markup Language (UIML) (http://www.uiml.org/) developed by Harmonia and Virginia Tech.
- The eXtensible Interface Markup Language (XIML) (http://www.ximl.org/) developed by a forum driven by RedWhale software.
- PUC: Personal Universal Controller by Myers et al. (UIST’02)
- Xweb by Olsen et al. (UIST’01)
XForms

- Apply concepts from model-based design
- Separate presentation from content (form controls markup is separated from data-types and returned values)
- XForms 'native' form controls are device-independent
- Reduce need for scripting through client-side checking
- XML instance is returned allowing strong typing
Teresa Requirements

- Mixed initiative
- Model-based
- XML-based
- Top-down (complementing WebRevEnge)
- Different entry-points
- Web-oriented
- http://giove.cnuce.cnr.it/teresa.html
The Method

1. Specification of Nomadic Task Model
2. Filtering for deriving System Task Model for each platform
3. Identification of the corresponding abstract user interfaces
4. Refinement in the concrete user interface
5. Generation of the user interface code
The Structure of the Abstract User Interface

- Connections among presentations
- AIOs for interaction modalities supported by each presentation
- Different types of AIOs
  - interaction_aio (selection, editing, control)
  - application_aio (overview, grouping, etc.)
Communication-oriented Composition operators

- Grouping: a set of elements logically related to each other
- Ordering: existing of an order among AIOs (i.e. temporal)
- Relation: One AIO related to a group of AIOs (i.e. Disabling them)
- Hierarchy: a logical hierarchy among a set of AIOs
Operators platform-dependent implementation
Example of platform-dependent composition operator implementation

EXAMPLE: Grouping Operator

- Desktop Computers
  - Fieldset
  - Bullet
  - Background Color

- Mobile Phones
  - Unordered List On Column
  - Fieldset (only for medium-large phones)
**Example of platform-dependent CIO choice**

**EXAMPLE:**

Single choice AIO (*single_select_aio*)

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>Desktop Computers</th>
<th>Mobile Phones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low cardinality</td>
<td>Radio Button</td>
<td>Radio Button</td>
</tr>
<tr>
<td>Medium cardinality</td>
<td>List Box</td>
<td>Drop Down List</td>
</tr>
<tr>
<td>High cardinality</td>
<td>List with scrollbars</td>
<td>Drop Down List</td>
</tr>
</tbody>
</table>
Mapping AIOs -> Tasks
TERESA support in Development

- Choice of device type
- General settings
- Default settings for composition operators
- How to implement presentation components
- Summary of design choices and XHTML preview
- Recording of concrete aspects defined
Voice Interaction

- **Characteristics:**
  - Linear
  - Not persistent
  - More faster and natural some operations

- **Voice Guidelines:**
  - Provide feedback to check the status of application
  - Use specific error messages
  - Brief prompts and short lists of options to reduce memory capability
  - Management of no-input events
Specifying general parameters for all presentations
Speech user interface composition operators

- **Grouping:**
  - Insert a sound
  - Insert a pause
  - Use some keywords
  - Use a specific volume of synthesizer voice

- **Ordering**
  - Alphabetical order
  - Use some keywords

- **Relation**
  - Change context
    - (change type of menu)

- **Hierarchy**
  - Increase or decrease the volume of synthesizer voice
GUI vs VUI

- Welcome message
- Management of no input event
- Provide feedback
- Description Object
- Composition operators
Research Agenda

- Knowledge intensive tools
- Interactivity vs. automation
- Runtime uses of interface models
  - Adaptation, context-aware interaction
- Integration of forward and reverse engineering
- Natural Development for Multi-Device Applications
- Improving techniques for editing the relevant models
  - Vocal Interaction with natural language-to-model specification translation
  - Sketch-based input
  - Tangible Interfaces