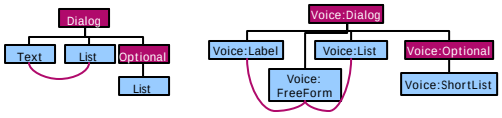
A Widget-Based Approach for Creating Voice Applications

Andreas Hartl
TU Darmstadt
andreas@tk.informatik.tu-darmstadt.de
MobileHCI Workshop P03
September 8, 2003

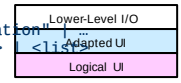
Voice Interfaces: State of the Art

- Current Approaches
  - Context Free Grammars
  - Statistics Based
- Both relatively Low-Level
- Limited reuse of existing solutions

Mapping Between UI Layers

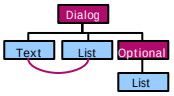



```
<dlg_label> = "New Task"
<ff_input> = <text>
<list> = „urgent" | „need preparation"
<dialog> = <dlg_label> <ff_input> | <list>
…
```



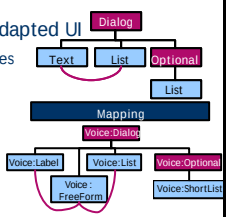
Logical UI/Abstract Widgets

- Object contents
  - Abstract information about content
  - Metadata
- Creating UIs
  - 1 Widget = 1 Object
  - Combination: combinatory widgets implement interfaces
  - Sample Widgets: free-form input, date/time input, confirmation, select 1 out of many, plain text output, structured text output, hierarchical navigation, …
- Almost modality-independent?
  - Low level widgets available e.g. Audio-Playback, Bitmap



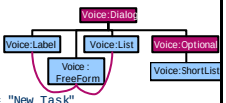
Logical → Adapted UI

- Mapping-Service for Logical → Adapted UI
  - → logical widgets don't map themselves
- Mapping dependent on
  - UI Paradigm
  - Device
  - Metadata-hints in logical widgets
- Other than 1:1 mapping possible
  - 1 logical : n physical – powerful logical widget to many physical ones
  - n logical : 1 physical – physical widget is multi-purpose



Adapted UI

- Widgets specific for
  - Interaction method
  - Device
  - I/O technology
- Widgets map themselves onto I/O technology
  - Generate CFG-Grammars
  - Generate N-Grams
- Acting as Controller
  - Widgets receive input
  - Convert them into networked events
  - Applications subscribe events




```
<dlg_label> = "New Task"
<ff_input> = <text>
<list> = „urgent" | „need preparation" | …
<dialog> = <dlg_label> <ff_input> | <list>
…
```

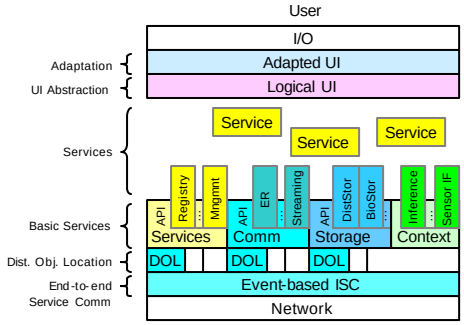
Conclusion

- Goal: provide higher level abstraction for voice
  - Support for multiple modalities, multiple technologies
- Separation of UI: Logical UI – Adapted UI – I/O
- Logical UI abstacts via widgets
  - Automatic mapping between logical UI & adapted UI
  - Logical UI may fusion events from realization
- Adaptated UI
  - modality & technology aware
  - Create I/O constructs necessary
  - Convert input into events

Spare Slides

(prepared for questions)

Setting within the Mundo-Project



Research Objectives

- Current Approaches
  - Grammar based
  - Statistics based
- Abstracting approaches with widgets
  - UIs are built combining standard controls
  - Predictability & Re-Use
  - Powerful
  - Localization & device association made easier
  - Embedded into group's MUNDO project
- Evolutional approach
  - Existing design-patterns (MVC) still valid
  - → Developers switch faster, QC easier

Widgets as Controller

- Adaptation Widgets receive input from I/O technology
- Convert into network events
  - Platform features event routing → Networking for free
- Logical widgets *may* refine events
  - Disambiguation by fusioning multiple events
- Applications subscribe to widget-events

Technical Details: How to map logical → adapted UI

- Adaptation widgets must be registered : placed in decision tree
- Decision tree layout:
  - highest level = logical widget to map
  - 2nd level = UI paradigm (e.g. voice)
  - 3rd-nth level = paradigm specific
- Example: Free form text input
  - w/ voice based interaction & JSGF: `Text/Voice/CFG/JSGF`
  - w/ SWT: `Text/GUI/SWT`

Technical Details: How to map logical → adapted UI

- Automatic mapping available for 1:1 & n:1
  - Existing logical widget
  - Mapping service searches tree bottom-up
  - Bottom-up works because by definition higher level nodes are not dependent on lower level features
  - n:1 easily possible: register more than once
- Example: find text widget for Server based ViaVoice

  `Text/Voice/CFG/JSGF/Server/ViaVoice`
- More specific adaptation widgets override general purpose ones
- Use OO features: if no mapping found → try to map abstract widget's base class

Technical Details: How to map logical → physical UI

- Semi-automatic mapping for 1:n & user defined mapping
- User defined mapping
  - via plug ins to mapping service
  - Define XPath expressions to select abitrary physical widgets
- Mapping 1:n
  - via Proxy-Object
  - Add physical widget that maps 1:1 to target
  - Physical widget instantiates several other physical widgets