Ubiquitous & Mobile Computing

Ease of Use

Dr. Erwin Aitenbichler
UC Interaction: State of the Art

- **WIMP User Interfaces**
  - Desktop metaphor now common form of interaction
  - Complex tasks require set of distinct commands, one after another

- **Mobile Devices: reduced subset of WIMP**
  - Menu + pointer and/or softkeys

- **Closed-world approach**
  - Device is self-contained, has full knowledge only about its own world -> ill-suited for UbiComp
  - Example scenario: Phone rings
    - Look at phone display for caller ID
    - Pause MP3-Player with Pause-Button
    - Change status in Instant Messenger
    - Answer phone with phone key

  ...now imagine interacting with a dozen of ubiquitous devices instead of just 3 appliances
UC UIs: Challenges

• Multiple devices, heterogeneous UIs
  - Design of appliance/tool/device/product interfaces
  - Distribute one interface to many devices → federated devices
  - Ubiquitous access to applications

• Limited interaction possibilities
  - Interaction on the move (hands&eyes-free interaction)
  - Small devices, high interaction costs
  - Different modalities → multimodal UIs, intelligent Uis, ambient Uls

• Ease of Use
  - Use affordances of softkeys / hardware buttons on real
  - Unify appliance interfaces (intelligent remote control) → intelligent UIs
    (Note: above two goals contradict!)

• Ease of Development
  - Design only a single user interface, on a higher level of abstraction
    → model-based UI development
UC UIs: Solutions

- Some solutions: (+ structure of this lecture)
  1. Use other *modalities* (than WIMP) for interaction: speech
  2. Multimodality: share devices that have different modalities
  3. Federated UIs
  4. Intelligent UIs
  5. Ambient UIs
Speech Interaction

- Sound suited not only for output, but also for input
  - Workers often have their hands busy while performing their tasks
  - Use of a keyboard or a mouse will force them to stop working

- Speech interaction
  - Freedom of movement: a headset has limited required space, little/no infrastructure needs
  - Acoustic channel still usable under extreme cases: darkness, etc.

- Speech is more natural?

  "Reports and thesis in speech recognition often begin with a cliché, namely that speech is the most natural way for human beings to communicate with each other"  
  (Melvyn Hunt, 1992)

  ... but that would mean:
  - Computers have the active and passive communication competence of humans
  - Assumes patiently listening, competent and intelligent system

- Speech UIs are way more than a speech recognizer
Speech: Properties

- **Speech is transient** („Once you hear it, it is gone”)  
  - Controlled by Short Term Memory (STM)  
  - Impossible to deliver large amount of data  
  - Advantages:  
    - Visual perception in parallel possible  
    - Deliver information without need to switch context

- **Speech is invisible**  
  - Often leaves the impression that the user does not control the system  
    - What actions can the user perform?  
    - What phrases/commands must be remembered?  
  - Lost in space problem

- **Speech is asymmetric** (with regard to multimodal UIs)  
  - People can  
    - speak faster than they type  
    - listen much more slowly than they read
Speech: Dialog Strategies

- **Form-based Interaction**
  - simplest and most common type
  - form items are executed exactly once in sequential order
  - Computer directs the conversation
  - Input with phone keypad

- **Command and Control**
  - Voice command menus that contains voice commands
  - Uses speech input, special vocabulary has to be learned
  - Does not scale; existing command sets (e.g., ETSI)

- **Mixed Initiative**
  - Requires some natural language understanding
  - Computer and human direct the conversation
  - User can speak to any active grammar
  - Input items are set and actions taken in response
Pick-by-Voice Example

1. Picker is directed to a particular location
2. Voice-based information about order item to pick
3. Picker confirms position by reading check number attached to the storage shelf location
4. System names the amount of merchandise to pick from the shelf
5. Picker confirms the actual removal
6. System removes the merchandise from the inventory list
7. Picker is directed to advance to the next picking location

Optional, after step 5:
- If only a few items are left, the system names the number of items left
- The picker can confirm or correct the inventory information
Multimodal Interfaces

- „Multimodality is Multimedia which the computer understands“
- User input is gathered from several sources
  - Famous example: „Put-that-there“ (Richard Bolt, 1980): Voice+Gesture input
- Benefits
  - Convenience: User may choose most natural/obvious modality to interact
  - Better performance: computer can use input from several channels to resolve recognition errors and ambiguities
- Examples of Modalities
  - Large visual displays
    - Pro: Users can quickly scan to find data relevant to their current needs
    - Con: not appropriate for blind users and mobile users who are not able to stand or sit in one place or to carry a display
  - Speech interaction
    - For people with limited typing ability to increase data entry speed
    - Hands and eyes not needed for interaction
    - Command and control: “open word”
  - Combination of display & speech
    - Spoken command “color this red” + touchscreen/mouse
### Multimodal Interfaces

- Most of the five classical senses (sight, hearing, touch, smell, taste) can be used for both the input and output sides

<table>
<thead>
<tr>
<th>Sense</th>
<th>Input Types</th>
<th>Output Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sight</td>
<td>Eye tracking</td>
<td>Visual displays (small and large) of graphics and text</td>
</tr>
<tr>
<td></td>
<td>Retinal scan</td>
<td></td>
</tr>
<tr>
<td>Hearing</td>
<td>Speech, Speaker identification, Humming</td>
<td>Text-to-speech, Recorded audio, Auditory icons and earcons, Sonification</td>
</tr>
<tr>
<td>Touch/Gesture</td>
<td>Keyboard, Mouse/pointing device, Handwriting, Gesture and motion, Pressure sensing, Heat sensing, Fingerprint recognition</td>
<td>Tactile display, Braille display, Haptic/force feedback display</td>
</tr>
<tr>
<td>Smell</td>
<td>Chemical odor-detecting sensors</td>
<td>Olfactory output</td>
</tr>
<tr>
<td>Taste</td>
<td>(currently none)</td>
<td>Gustatory output</td>
</tr>
</tbody>
</table>
Purposes for using Multimodality

- **Complementary Modalities (Multimodal Fusion)**
  - natural interaction; good mapping to task
  - modalities are chosen that are the best fit to different parts of the task
  - avoid “clutter”/overload on any one channel (output)
  - disambiguation (input) - non-mobile

- **Replacement Modalities**
  - replace one modality for another
  - improve efficiency
  - reduce screen clutter
  - compensate for limited I/O capabilities of devices
  - compensate for hands- or eyes-busy tasks

- **Redundant Modalities**
  - provide users with the ability to choose from different interaction options
  - accessibility
  - improve efficiency
Example: QuickSet

- DARPA-funded project for multimodal input
  - Current application: map-based maneuver planning

- Integrates
  - Voice recognition
  - Gesture recognition
  - Natural language recognition
  - Text-to-speech
  - VR
  - Web

- Uses statistical unification for combining interpretations
- Output uses agents interpreting feature structures
Input Fusion

- Combination of inputs from several different modalities; distinguish:
  - **Early fusion**
    - Typically at a low syntactic level; granularity: single input elements
    - Example: „Put-that-there“
      - one single grammar for all modalities
  - **Late fusion**
    - Typically at the semantic level; granularity: whole dialog steps
    - Separate grammars/appearances for each modality
CoScribe: Pen and Paper Interfaces

Annotieren

Verlinken

Taggen
Federated Devices: Multimon++

- **Multimon** = Multiple monitors for one desktop

- → Maintain WIMP

- **Gloss** (Global Smart Spaces)
  Université Joseph Fourier, Grenoble
  - Multiple monitors
  - Geometric discovery
  - Supports handheld devices

http://iihm.imag.fr/projects/Gloss/
Augmented Surfaces
Pick-and-Drop

Jun Rekimoto, Sony Computer Science Laboratories
• Drag and Drop *between two devices*

http://www.csl.sony.co.jp/person/rekimoto/pickdrop/
Multimodal Device Federations

- Multi-display environments still largely built on WIMP
  - display requires large form factor of devices
  - reduced input channel
- Federate devices with different modalities
  - speech input, display for output
  - pen input on PDA, large display for output
  - assistive computing: voice output, keyboard input
- XWeb
  - Based on the architecture of the web
  - XTP Interaction Protocol, server-side data has a tree structure
  - URLs for location of objects (e.g., xweb://automate.home/lights/livingroom/)
  - CHANGE message (similar to GET in HTTP): sequence of editing operations to apply to a subtree (set/delete attribute, change/insert child object, move/copy subtree)
  - Multiple clients access the same URL, server synchronizes changes
  - Clients for voice, small screen and GUI exist
Model-based UI Development

Motivation:

- Manual UI design for applications tedious and costly

- Multitude of devices for ubiquitous computing aggravates the problem
  - more modalities
  - more device classes
  - more heterogeneous devices

- CS solution: high level language & compile
  - Model Based UI Development
  - difficult problem!
    - compare voice interfaces and GUIs
Model-based UI Development

• Concrete User Interfaces (CUI)
  - manually craft each user interface for each device class

• Abstract User Interfaces (AUI)
  - create an abstract representation of the user interface
  - will be automatically converted into a CUI

• Task Model (TM)
  - Description of the application UI in domain tasks

• Model-based UI engineering
  - create a model of the application (and context, devices, etc.)
  - at runtime, create an AUI from that, then a CUI
CUI Example: X+V

- Example for Multimodal CUI: X+V (XHTML + VoiceXML)

VoiceXML

- Audio output
- Speech synthesis
- DTMF-Input
- Speech input
- Telephony features (route, ...)
  ➔ „a language to create phone based menus“

- VoiceXML-Documents are hosted on standard web servers

X+V development

- Create independent XHTML and VoiceXML documents
- Merge documents using XML namespaces
- Simple synchronization with X+V sync statements
- More complex synchronization via XML-Events and ECMAScript
  - Hand written
  - Has to do the fusion between events
MM Online Payment

This is a multimodal application for paying Ajax or Superstore an amount of 10, 20, 30, or 40.

Recipient
- Ajax
- Superstore

Amount
- 10
- 20
- 30
- 40
CUI: X+V

```xml
[DOCTYPE html PUBLIC "-//W3C//DTD XHTML+Voice 1.0/EN" "xhtml+voice.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:vxml="http://www.w3.org/2001/vxml"
     xmlns:ev="http://www.w3.org/2001/xml-events"
     xmlns:xv="http://www.voicexml.org/2002/xhtml+voice">
  <head>
    <title>MM Online Payment</title>
    <!-- CUI: X+V -->
  </head>
  <body ev:event="load" ev:handler="#payment">
    <vxml:form id="payment">
      <vxml:block> Welcome to online payment</vxml:block>
      <vxml:field name="recipient" xv:id="recipient">
        <vxml:grammar type="application/x-jsgf">
          <![CDATA[
            grammar companies;
            public <companies> = ajax | superstore;
          ]]>
        </vxml:grammar>
        <vxml:prompt>Who do you want to pay, Ajax or Superstore?</vxml:prompt>
      </vxml:field>
      <vxml:field name="amount" xv:id="amount">
        <vxml:grammar type="application/x-jsgf">
          <![CDATA[
            grammar amount;
            public <amount> = 10 | 20 | 30 | 40;
          ]]>
        </vxml:grammar>
        <vxml:prompt>How much do you want to pay, 10, 20, 30, or 40?</vxml:prompt>
      </vxml:field>
    </vxml:form>
  </body>
</html>
```
CUI: X+V

<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML+Voice 1.0/EN" "xhtml+voice.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:vxml="http://www.w3.org/2001/vxml"
     xmlns:ev="http://www.w3.org/2001/xml-events"
     xmlns:xv="http://www.voicexml.org/2002/xhtml+voice">
    <head>
        <title>MM Online Payment</title>
        <vxml:form id="payment">
            <vxml:block> Welcome to online payment</vxml:block>
            <vxml:field name="recipient" xv:id="recipient">
                <vxml:grammar type="application/x-jsgf">
                    <![CDATA[
                        grammar companies;
                        public <companies> = ajax | superstore;
                    ]]>}
                </vxml:grammar>
                <vxml:prompt>Who do you want to pay, Ajax or Superstore?</vxml:prompt>
            </vxml:field>
            <vxml:field name="amount" xv:id="amount">
                <vxml:grammar type="application/x-jsgf">
                    <![CDATA[
                        grammar amount;
                        public <amount> = 10 | 20 | 30 | 40;
                    ]]>}
                </vxml:grammar>
                <vxml:prompt>How much do you want to pay, 10, 20, 30, or 40? </vxml:prompt>
            </vxml:field>
        </vxml:form>
    </head>
    <body ev:event="load" ev:handler="#payment">
        <h1>Online Payment</h1>
        <p>This is a multimodal application for paying Ajax or Superstore an amount of 10, 20, 30, or 40.</p>
        <form>
            <label>Recipient</label>
            <select name="recipientGUI">
                <option value="Ajax">Ajax</option>
                <option value="Superstore">Superstore</option>
            </select>
            <label>Amount</label>
            <select name="amountGUI">
                <option value="10">10</option>
                <option value="20">20</option>
                <option value="30">30</option>
                <option value="40">40</option>
            </select>
        </form>
    </body>
</html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:vxml="http://www.w3.org/2001/vxml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xv="http://www.voicexml.org/2002/xhtml+voice">
<head>
<title>MM Online Payment</title>
<vxml:form id="payment">
<vxml:block>Welcome to online payment</vxml:block>
<vxml:field name="recipient" xv:id="recipient">
<vxml:grammar type="application/x-jsgf">
<![CDATA[
grammar companies;
public <companies> = ajax | superstore;
]]>
</vxml:grammar>
<vxml:prompt>Who do you want to pay, Ajax or Superstore?</vxml:prompt>
</vxml:field>
<vxml:field name="amount" xv:id="amount">
<vxml:grammar type="application/x-jsgf">
<![CDATA[
grammar amount;
public <amount> = 10 | 20 | 30 | 40;
]]>
</vxml:grammar>
<vxml:prompt>How much do you want to pay, 10, 20, 30, or 40?</vxml:prompt>
</vxml:field>
</vxml:form>
<body ev:event="load" ev:handler="#payment">
<h1>Online Payment</h1>
<p>This is a multimodal application for paying Ajax or Superstore an amount of 10, 20, 30,
or 40.</p>
<form>
Recipient
<select name="recipientGUI">
<option value="Ajax">Ajax</option>
<option value="Superstore">Superstore</option>
</select>
Amount
<select name="amountGUI">
<option value="10">10</option>
<option value="20">20</option>
<option value="30">30</option>
<option value="40">40</option>
</select>
</form>
</body>
</html>
AUI

**Abstract UI**
- Applications use abstract UI widgets
  - e.g., binary input, select one from many, date input, ...
- AUI transformed into CUI, typically at runtime
- Devices render CUIs
- One transformation program/ruleset required per device
- Advantage: Transformations can be reused and evolved independent of applications

**Implementation example: XML+XSLT**
- AUI described in some XML representation
- XSL stylesheets define translations onto target device
- However, implementation of synchronization between modalities is complex
  - Can already be complex within same browser (cf. X+V)
  - Multiple devices -> AJAX?
• **Modgets** (moldable gadgets) are the AUI elements
  - contain the data of the model
  - Higher-level than CUI
    • Binary input $\equiv$ checkboxes, checkable menu items, Yes/No
    • Select 1 from many $\equiv$ list-/comboboxes, radio buttons, menus
    • Tree selection $\equiv$ trees, hierarchical menus
    • Freeform text entry
  - More specific than CUI
    • Grammar for freeform text
    • Interdependencies (if Foo then not Bar)

• **Embodiments** are wrappers for corresponding CUI elements
  - Know of platform’s possibilities & limitations
  - Several embodiments for one widget possible
    • Select -> List, Radio Button, etc.
    • Best fitting embodiment is used
    • Multiple embodiments for a widget -> multimodality
Mundo Partes Architecture

Location

- anywhere
- near to user
- on each device

Mundo-based Application
- Modgets + Hints
- Modgets + generic Evaluation

Adaptation Service
- Plug-In Interface

Embodiments

Presentation Service
- Events

MundoCore

Modgets +
generic Evaluation

Plug-In Interface

28
Concur Task Tree (CTT)

- Task Model: captures application tasks in domain terminology
- UML Use Cases → lack formality
- UML State Chart / Activity Diagram → too detailed, not hierarchical
- State of the Art: Concur Task Trees (CTT) (developed by F. Paterno)
CTT: Task Types

- abstract task
  - for structuring

- system task
  - background work

- interaction task
  - user interacts with the system

- user task
  - user does other things
CTT: Hierarchical Decomposition

- In order to perform T
  - you have to perform T1 and T2
  - you have to perform T1 or T2
  - you have to perform T1, and then T2
  - ... to decide use temporal relations (next slide)
CTT: Temporal Relationships

- when super task is enabled...
  - \( T_1 \) [] \( T_2 \) - **Choice**: ... \( T_1 \) and \( T_2 \) are enabled. When either \( T_1 \) or \( T_2 \) are performed, the other one is disabled.
  - \( T_1 \) || \( T_2 \) - **Concurrency**: ... \( T_1 \) and \( T_2 \) are enabled and remain enabled until finished when the other is performed.
  - \( T_1 >> T_2 \) - **Enabling**: ... \( T_1 \) is enabled. Only when \( T_1 \) is performed, \( T_2 \) becomes enabled.
  - \( T_1 \) \( > \) \( T_2 \) - **Deactivation**: ... \( T_1 \) and \( T_2 \) are enabled. When \( T_2 \) is performed, \( T_1 \) becomes disabled.
  - (there are 4 more; we omit here: enabling with information passing, concurrent communicating tasks, task independence, suspend-resume)

- task is disabled when it is done (e.g., user entered data)
- a super task is disabled, when there are no enabled subtasks
- initially the root task is enabled
CTT: Deactivation

- In order to perform T, you have to perform
  - T1 and then T2 or
  - only T2

![Diagram showing the relationship between T, T1, and T2 with an arrow pointing from T to T1 and another from T to T2.](image)
CTT Disambiguation

- Introduce “abstract tasks” to disambiguate
- only one operator type on each level
From CTT to UI

- Annotate Concur Task Tree (CTT) with Abstract UI (AUI)
- Calculate **Enabled Task Sets** (ETS)
  - ETS = a set of tasks that are logically enabled to start their performance during the same period of time
  - Interactive Tasks in ETS correspond to UI elements

![Diagram showing CTT to UI mapping](image)
Annotating CTT with AUI

- select read SMS
- select SMS
- show SMS
- read SMS
- select adjust
- adjust volume

- menu item
- choose 1:n
- text
- menu item
- slider

- shutdown
Calculating ETS

ETS1 = {select read SMS, select adjust, shutdown}
Calculating ETS

ETS2 = {select SMS, shutdown}
Calculating ETS

ETS3 = \{\text{show SMS, shutdown}\}
Calculating ETS

ETS4 = \{read SMS, shutdown\}
Calculating ETS

ETS5 = \{\text{adjust volume}, \text{shutdown}\}
Turning ETS into UI

- After completion of each task, recalculate ETS
- If ETS changed, then create new dialog
  - take AUI snippet for every interactive task
  - layout according to rules, constraints, patterns
  - map to underlying toolkit / CUI
Intelligent UIs

- Remember Clippy?
- Needs to be revisited for ubiquitous computing!
  - interaction costs are high → use prediction „Proactive UI“
  - natural interaction → use recognition algorithms (machine learning) „Perceptive UI“
- Example: AUGUR – Predicting Form Entries from Context Data
Ambient UIs

- Don’t distract users, only show subtle information
  - Embed display-capabilities into ubiquitous devices
  - Non-WIMP - but: no real interface, just a face
- Ambient Orb
  - Changes color according to traffic information/weather forecast/stock market trends
- Ambient Umbrella
  - Handle glows on rain-forecast
- Java Toaster
  - Produces toasts with weather forecast
- Ambient Sound
  - Ambient displays need to be looked at
  - Sound as background information
  - Intuition automatically focuses on unusual patterns
  - Annoyance is a problem - sounds must be well-picked
  - Examples:
    - Real-time audible Web Server log
    - rcbirds: bird sounds illustrate Wikipedia edits
Summary

- Multimodality
- Input Fusion
  - Early Fusion
  - Late fusion
- Speech is different!
- More than one device, no closed world!
  - dynamic device federations
  - user driven / context aware
- Model-based development of user interfaces
  - Concrete User Interfaces (CUI)
  - Abstract User Interfaces (AUI)
  - Model-based UI Engineering