Multimodal Interaction

Lesson 3
Models for Multimodal Interaction
Maria De Marsico
demarsico@di.uniroma1.it

Model to design

- Identify system borders
- Identify possible system states
- Identify possible responses to external actions
- Anticipate possible evolutions

For multimodal interaction
  - Each channel identifies its own borders
  - Possible system states are a combination of the states for each channel → How many of them are truly significant?
  - Actions may come from different channels, and responses are not due through the same channels
  - Possible evolutions are more difficult to anticipate and address

Model is to ask questions
Design is to try responses
Why it is important to model software

• Need to identify project components
• Need to evaluate compliance with requisites
• Need to maintain software along time

• For multimodal interfaces:
  o Each channel requires a separate component, but communication and synchronization among different input modalities must also be supported
  o Requisites may be more difficult to state, and therefore to check
  o Maintainence may call for sharp separation in order to exploit technological advances at different pace

Why it is important to model interactive software

• Different levels of definition must be maintained consistent
• Identify user languages and evaluate the correspondence with interaction languages
• Evaluate the correspondence between interaction tools and application logic

• For multimodal interactive systems:
  o Consistency must also address the use of different channels
  o User languages span different modes and modalities that must be coordinated
  o Application logic can exploit the advantages of multimodality
Visual interactive systems

User

Interaction

World of media

Representation

World of application logic

Computer

Multimodal interactive systems

User

Interaction

World of media per channel

Representation

World of application logic

Computer
Problems

- Relation between representation medium (media) and interaction medium (media)
- Correspondence between application logic and interaction logic
- Need to handle representation and activation processes

UIMS

- User Interface Management System: controls the relation between presentation and functionality
- Defines separation between application semantics (business logic) and presentation (GUI); in this way it improves:
  - portability – ability to be used on different devices and systems
  - re-usability – reuse of components cuts expenses
  - multiple interfaces – to access the same functionality
  - customizability – for designer and user

- Multimodal interfaces:
  - portability - very complex
  - multiple interfaces - multiple channels
  - customizability – for special needs
Separable Interface

Seeheim Model

- Developed in early ‘80 during the Workshop on User Interface Management Systems in Seeheim by a group composed by Jan Derksen, Ernest Edmonds, Mark Green, Dan Olsen, and Robert Spence.
- Not possible at that time to fully evaluate it but ...
- ... it was a first very important step (as any first step!)
- “... because of Seeheim ...
  ... we think differently!” (Alan Dix)
- Follows the division among Lexicon, Syntax, Semantics
- Limits:
  - Dialog control is monolithic
  - Bypasser makes formal description difficult
Seeheim model

Seeheim model (continued)

- Different kinds of feedback:
  - lexical – mouse movement (the code to express that the system got user’s intent to move the focus of action)
  - syntactic – a chosen menu item is highlighted (a syntactic rule says that a selected item must be highlighted)
  - semantic – the sum of numbers on a calculator is changed (application rule)

- Semantic feedback is slower of course
- In many cases, a quicker semantic feedback is needed, e.g., in hand drawing, or in highlighting the waste basket on the desktop when a document is moved nearby, or when large volumes of output data may skip the dialog layer.
The model confuses logic architecture and implementation.

Direct communication between application and presentation for rapid semantic feedback but regulated by dialog controller.
Seeheim model (continued)

- The two main difficulties of Seeheim model:
  - when we change a presentation component, the dialog must be rewritten to adapt it to its features
  - dialog tends to be based on presentation, and presentation must be changed everytime the dialog is changed

BUT
- If we handle each block as a whole,
  - we may provide the same outer layer to different applications (but we have to change the dialog)
  - we may apply the same look and feel to a text editor, a spreadsheet, and so on, as in Microsoft products
  - in this way the user does not have to learn different dialog languages for different applications
  - conversely, we may provide a single application to be implemented behind several different outer layers, so as to allow different companies to adopt the same application with their own corporate interface style

Arch/Slinky Model

More layers

abstract

concrete

application

interface

dialogue

functional core

func. core adaptor

lexical

physical
Arch/Slinky Model (cont.)

- More layers! – lexical/physical layers are distinguished
- As in a ‘slinky' spring, different layers may be larger (more important) than others in different systems …
- ... or in different components
- Limit - is not better than Seeheim in providing indications on the precise content of components or on their design process

Separable interface

MVC (Model View Controller)
Problems in the definition of an interactive system

<table>
<thead>
<tr>
<th></th>
<th>Single components</th>
<th>Components coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>Identification of the kind of needed component</td>
<td>Layout design</td>
</tr>
<tr>
<td>Behaviour</td>
<td>Reaction to user actions and state change</td>
<td>Event propagation and view synchronization</td>
</tr>
</tbody>
</table>

- In multimodal interfaces:
  - the kind of needed component may refer to different channels
  - user actions may happen through different channels
  - layout is not 2D visual but multimodal
  - events may refer to different modalities

Identification of the kind of component

- Domain representation
  - Information (data) to represent
  - Processes to activate
- Interaction management
  - Generic interaction elements
  - Interface navigation elements
  - Support to specific behaviours
Behaviours to support:

• Process activation
• Information retrieval
• Information feeding

Coordination Policies (for behaviour)

• Policies for enable/disable
• Activation propagation
• Concurrent activation

• Multimodal interfaces:
  o Intra-channel
  o Inter-channel
Coordination Policies
(for presentation)

- Overall interface “Layout”
- Mutual constraints between elements
  - coordinated changes
  - admissible relations among elements
- Consistency with other types of interfaces (which ones???)

Readings

- Roope Raisamo. Architectures for User Interfaces - SoSE tutorial lecture course
  http://www.cs.tut.fi/~sose/phdcourse08/ArchitecturesForUserInterfaces.pdf