Augmented Reality II

Software Tools for User Interfaces

Gudrun Klinker

June 29, 2004

Outline

1. Specification Methods
2. Interface-Building Tools
3. Evaluation and Critiquing Tools

Literature

- Ben Shneiderman, “Designing the User Interface, Strategies for Effective Human-Computer Interaction”, Addison Wesley. (Chapter 5)

1. Specification Methods

1. Specification Methods

Use tools rather than “handcraft” every system

- Grammars
- Menu-selection, dialog-box trees
- Transition diagrams
- State charts
- User-action notation (UAN)

1.1 Grammars

- BNF (context-free)

  <Telephone book entry> ::= <Name> <Telephone number>
  <Name> ::= <Last name>, <First name>
  <Last name> ::= <string>
  <string> ::= <character> | <character> <string>

  needs to be supplemented by adhoc techniques to specify semantics
- Pro:
  - concise representation
  - existing tools (parsers)
- Con:
  - difficult to follow with growing number of specifications
  - confusing to many users

1.1 Grammars

Multiparty grammars [Shneiderman]

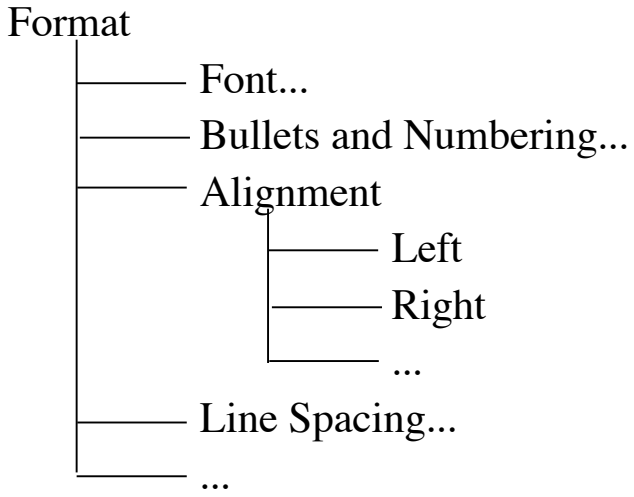
- to accomodate richness of interactive software

  <Session> ::= <U: Opening> <C: Responding>
  <U: Opening> ::= LOGIN <U: Name>
  <U: Name> ::= <U: string>
  <C: Responding> ::= HELLO [<U: Name>]

- Pro:
  - good for text-oriented command sequences
- Con:
  - not so good for 2D widgets (menus, form fillins, direct manipulation)
  - concept of (widget) tree structure and traversal is difficult to support

1.2 Menu-selection, dialog-box trees

Menu-selection tree



- needs to be laid out on a large wall
- important to see the entire structure at once for
  - consistency
  - completeness
  - lack of redundancy and ambiguity

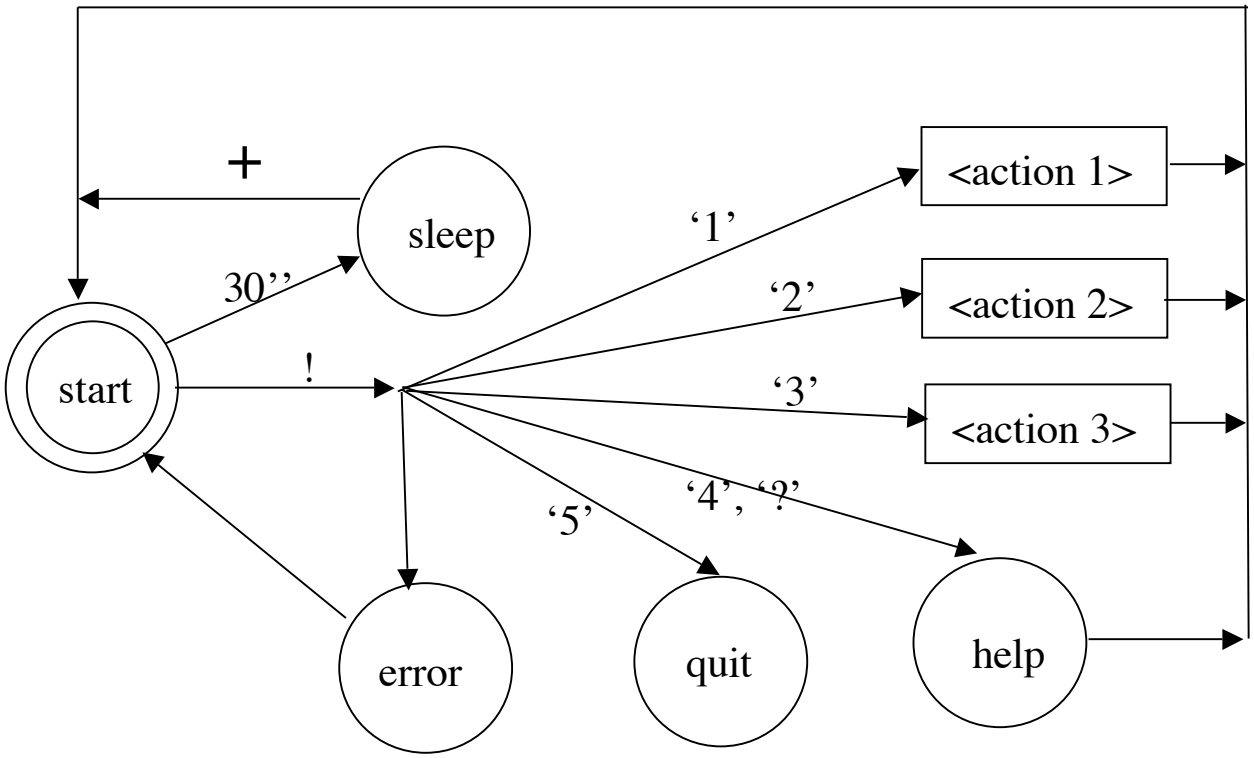
1.2 Menu-selection, dialog-box trees

- Pro:
  - simple structure to guide designer and users
  - online tools to help construct and draw the trees
  - good to get an overview
    - high-level relationships
    - low-level details

1.2 Menu-selection, dialog-box trees

- Con:
  menu trees incomplete
  - do not show entire structure of all possible users actions
    - returns to previous menu
    - jumps to startin menu
    - detours to error handling on help screens
  - would clutter the clean structure of a menu tree

1.3 Transition diagrams



- Requires large prints on walls
- e.g. used to show interaction on 350 screens of a satellite control system

1.3 Transition diagrams

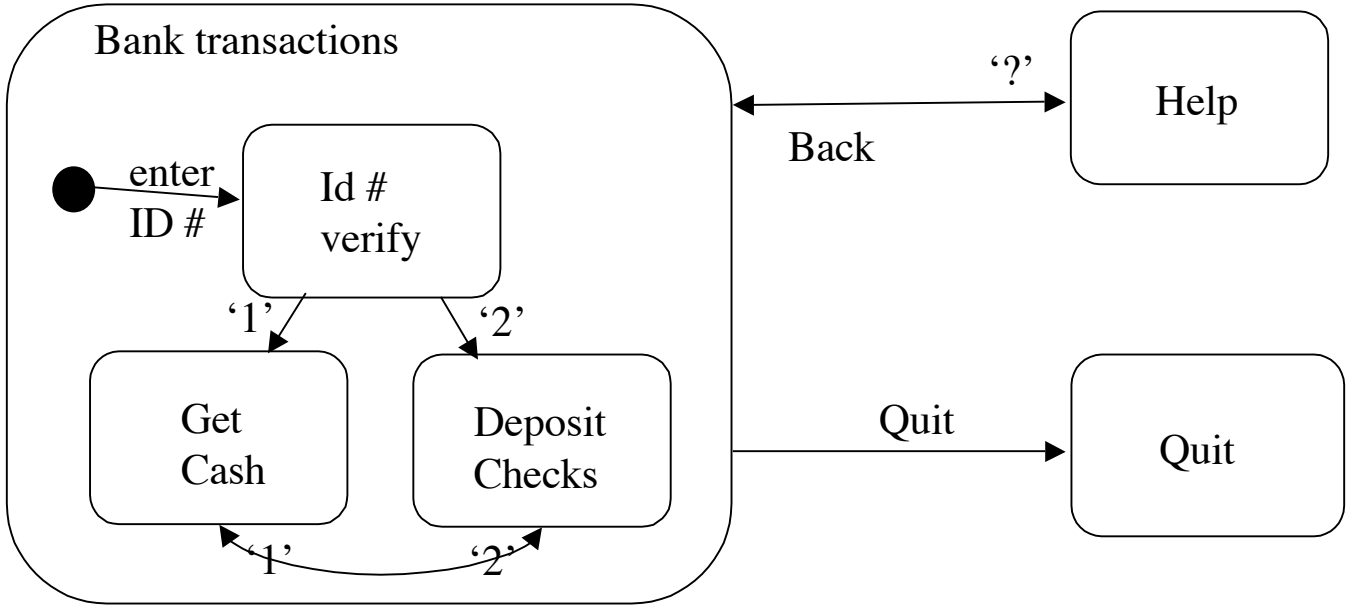
- Pro:
  - more precise specification of every possible transition
  - tools to create and maintain transition diagrams as part of CASE tools
  - helpful in design and training

1.3 Transition diagrams

- Con:
  easily becomes large and confusing
  - only modular if nodes are included with subgraphs
  - confusing when each node
    - has a link to a help state
    - jumps back to previous state
    - goes to start state
    - quit state
  - problems with concurrency or synchronization (except for Petri nets)

1.4 State charts [D. Harel]

- grouping feature: nested roundtangles
- repeated transitions factored out to surrounding roundtangle



1.4 State charts

Extensions:

- concurrency
- external interrupt events
- user actions
- dataflow specifications
- constraint specifications
- embedded screen prints

1.4 State charts

- Con:
  good for menus, commands, form fillins,
  but clumsy for direct manipulation
  - cannot cope conveniently with variety of permissable actions and system-provided visual feedback
  - direct manipulation interfaces depend heavily on context to determine meaning of input (e.g., current cursor position)

1.5 User-action notation (UAN)

high-level notations

- focus on users' tasks
- deal with pointing, dragging, clicking
- describe interface feedback

User Actions	Interface Feedback	Interface State
~[icon] Mv	icon!	
M^		
~[file] Mv	file!, forall(file!): file- !	selected = file
~[x,y]*	outline(file) > ~	
~[trash]	erase(file), trash!!	selected = null

1.5 User-action notation (UAN)

- Pro:
  - high-level approach to specify system behavior and user action
- Con:
  - takes time to get used to
  - no convenient specification of rich graphics (drawing programs, animations, relationships across tasks, interrupt behavior)

2. Interface-Building Tools

2. Interface-Building Tools

- General remarks
- Design tools
- Software engineering tools

2.1 General remarks (Interface building tools)

Features of user-interface-building tools

- User-interface independence (decoupling of the user-interface design from the complexities of programming)
  - Separate interface design from internals
  - Enable multiple user-interface strategies
  - Enable multiple-platform support
  - Establish role of user-interface architect
  - Enforce standards

2.1 General remarks (Interface building tools)

Features of user-interface-building tools

- Methodology and notation
  - Develop design procedures
  - Find ways to talk about design
  - Create project management

2.1 General remarks (Interface building tools)

Features of user-interface-building tools

- Rapid prototyping
  - Try out ideas very early
  - Test, revise, test, revise, ...
  - Engage end users, managers, and customers

2.1 General remarks (Interface building tools)

Features of user-interface-building tools

- Software support
  - Increase productivity
  - Offer constraint and consistency checks
  - Facilitate team approaches
  - Ease maintenance

2.2 Design tools

- Create quick sketches
  - explore multiple alternatives
  - allow communication with the design team
  - convey concepts to clients

2.2 Design tools

- UI mockups with
  - paper and pencil
  - word processor
  - slide shows

  - computer-assisted instruction software (Authorware)
  - multimedia construction tools (Macromedia Director)

2.2 Design tools

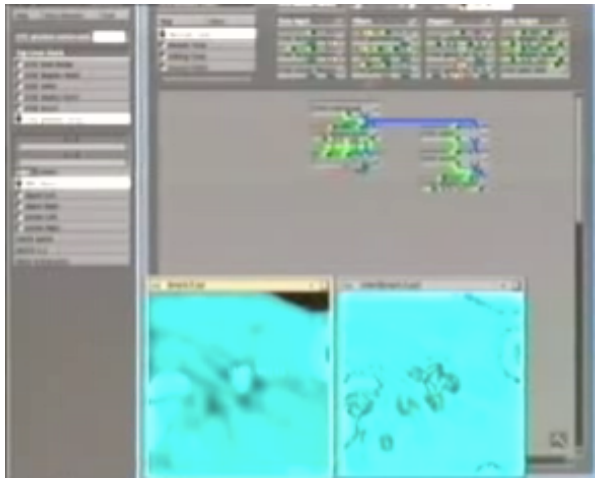
- UI mockups: e.g.:
  - slide show of still images
  - allow users to interact w. prototype (push buttons) navigation through screens

2.2 Design tools

- Visual editing tools
  - designers can interactively create layouts
- Visual programming tools
  - Prograph, LabVIEW, AVS
  - edit, execute, debug, make changes during execution
  - emphasize data flow, flowchart-like presentation
  - deeply neested modular structure
    - level 1: use existing widgets
    - level 2: program new widgets

Demos:

- AVS
- CAR



2.3 Software engineering tools

Toolkits (to be used in programs)

- common widgets (windows, scrollbars, ...)
- programming languages with accompanying widget libraries
- Pro:
  - familiar to experienced programmers
  - great flexibility
  - extensive control in creating the UI
- Con
  - requires months of learning
  - large burden of creating an application
  - maintenance is difficult
  - only partial support for consistency
  - designers/managers depend heavily on experienced programmers

2.3 Software engineering tools

Toolkits (to be used in programs)

- Tcl/TK (Ousterhout 94)
  - Tcl:
    - interpreted command language
    - cross-platform
  - Tk:
    - relatively easy/simple to set up a user interface
    - text, canvas
    - no visual editor

```
menubutton .menu1 -text “Unix commands” -menu .menu1.m -underline 0
menu .menu1.m
.menu1.m add command -label “List files” -command {ls}
.menu1.m add command -label “Get date” -command {date}
pack .menu1
```

2.3 Software engineering tools

Toolkits (to be used in programs)

- Java
  - programming language, especially for WWW
  - applets run on client machine
  - object oriented - but less complex than C++
  - automatic garbage collection
  - no pointers
- Javascript
  - embedded in HTML
  - network distribution
  - cross-platform compatiblity

3. Evaluation and Critiquing Tools

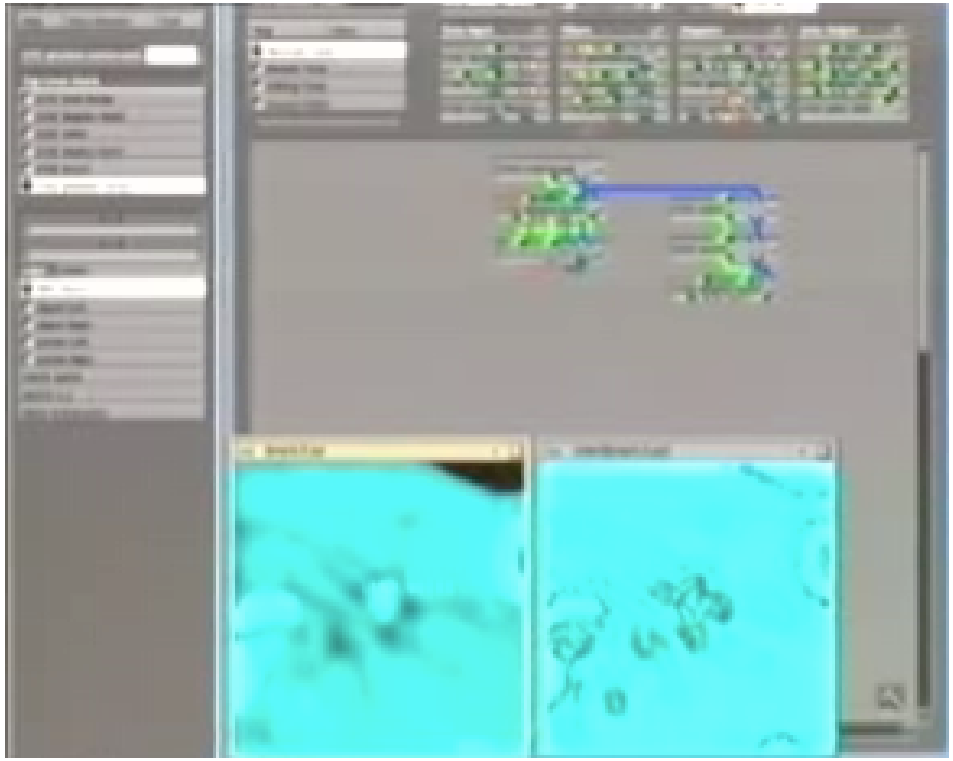
3. Evaluation and Critiquing Tools

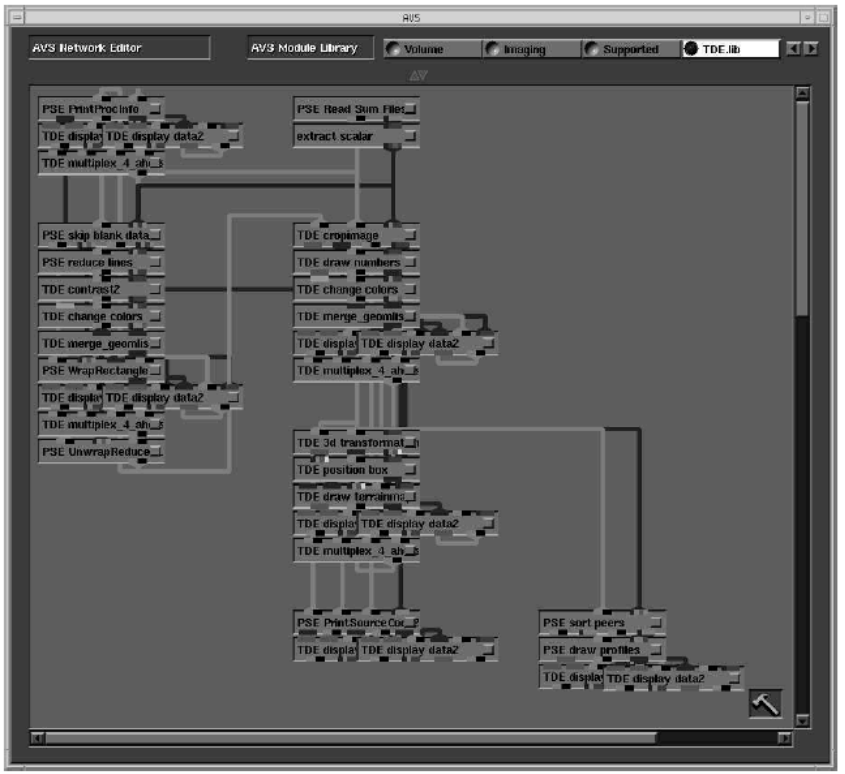
- Add procedures to software tools that evaluate/critique user interfaces
  - simple metrics (size of user interface project)
    - numbers of displays, widgets, links between displays
  - more complex
    - depth of menu tree
    - redundancies
    - consistent use of widget labels
    - proper transitions for all buttons

3. Evaluation and Critiquing Tools

- run-time logging software
  - users' activity patterns
  - reports (Display Analysis Program [Tullis 88])
    - frequency of error messages
    - menu-item selection
    - dialog-box appearance
    - help invocation
    - form-field usage
    - web-page access

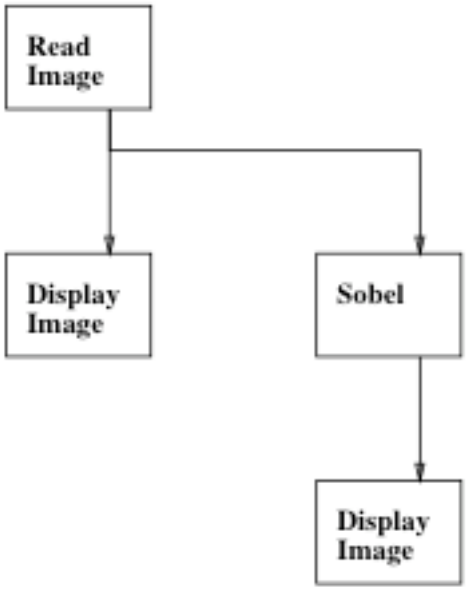
AVS




AVS
AVS Network Editor
AVS Module Library
Volume
Imaging
Supported
TDE.lib
PSE PrintProcInfo
TDE display
TDE display data2
TDE multiplex_4_ah
PSE Read Sim Filet
extract scalar
PSE skip blank data
PSE reduce lines
TDE contrast2
TDE change colors
TDE merge_geomlis
PSE WrapRectangle
PSE UnwrapReduce
TDE croplmage
TDE draw numbers
TDE position box
PSE sort peers
PSE draw profiles

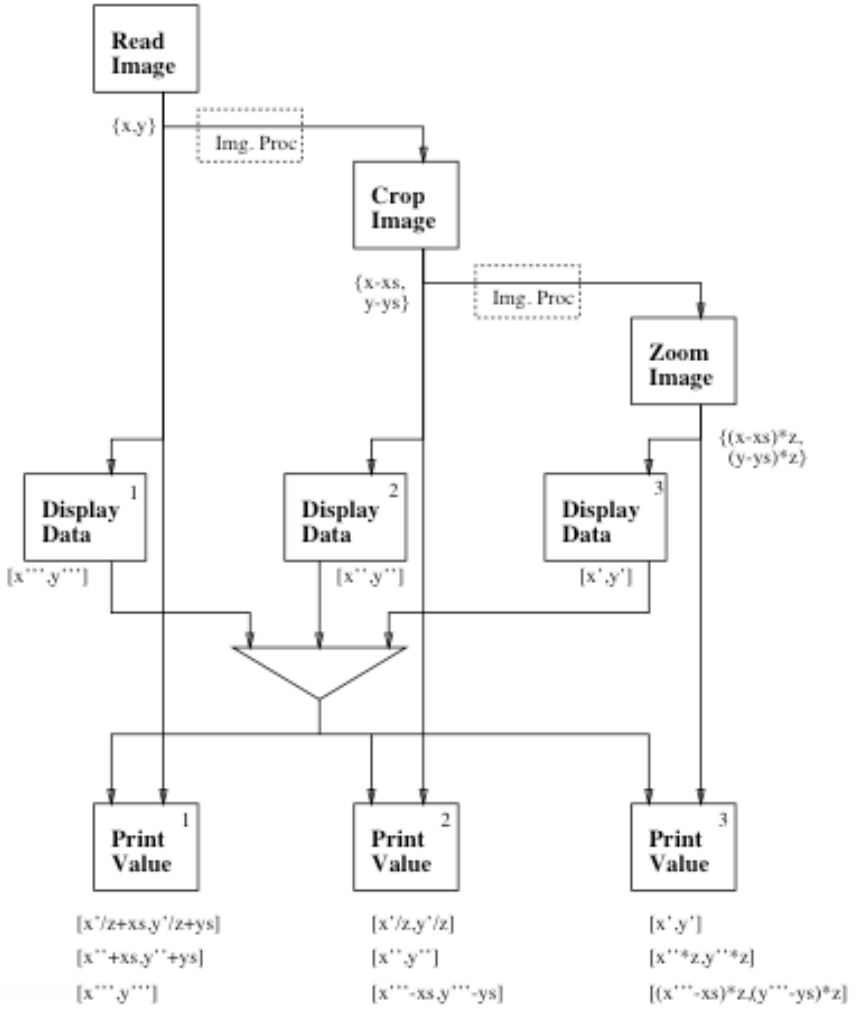
TDE: Telecollaborative Data Exploration

Basic example



TDE: Telecollaborative Data Exploration

Transformation logging



TDE: Telecollaborative Data Exploration

Tele-collaboration