Designing Interactive Systems I

Interaction Design Notations

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

Winter Semester ’23/’24

https://hci.rwth-aachen.de/dis
Review

• Ten Golden Rules of Interface Design
• Human Deadlines
• Latency & Responsiveness
Interaction Design Notations
Interaction Design Notations

• Alan Dix et al.: Human-Computer Interaction, 3rd ed. (2003), Chapter 16
Print "Please enter a number"
INPUT n
Print "The square of", n, "is", n*n

• What are the problems with using such a notation to specify a dialog?
Why UI Specification Languages

- In normal programming languages, UI and algorithms are mixed up
- System and user decisions are hard to distinguish
- Error checking on inputs dominates and complicates code
- First step: bundling I/O in classes/procedures
- Second step: Use a more efficient, readable language to specify the dialog
  - A priori to design the dialog
  - As part of the implementation (executable spec.)
Specifying User Interfaces

• Problem: Describe the proposed design of a user interface

• Approach: natural/semi-formal/formal languages

• Many standard computer science techniques apply

• The more modern the UI, the harder to describe textually, depending on modality and UI style
Grammars

• Mostly BNF-like
  
  \[
  \text{expr ::= empty | atom expr | '(' expr ')' expr}
  \]

• E.g., Shneiderman's multiparty grammar

  \[
  \begin{align*}
  \text{<Session> ::= <U: Opening> <C: Responding>} \\
  \text{<U: Opening} ::= \text{LOGIN <U: Name>} \\
  \text{<U: Name} ::= \text{<U: string>} \\
  \text{<C: Responding} ::= \text{HELLO [<U: Name>]}
  \end{align*}
  \]

• Great for command-line UIs, e.g., banking ATMs, Unix commands

• Less suitable for GUIs
Grammars

• Regular expressions
  • select-line click click* double-click

• E.g., Unix “copy” command synopsis:

  \texttt{cp \([-R \ [-H \ | \ -L \ | \ -P]] \ [-f \ | \ -i \ | \ -n] \ [-pv] \ source\_file \ target\_file}

  \texttt{cp \([-R \ [-H \ | \ -L \ | \ -P]] \ [-f \ | \ -i \ | \ -n] \ [-pv] \ source\_file \ ... \ target\_dir}

  \begin{itemize}
    \item recursion policies
    \item overwrite policies
  \end{itemize}

• Short and precise, but hard to read, requires additional information about semantics
Production Rules

• Unordered list of rules: if condition then action
  • Condition based on state or pending events
  • Every rule always potentially active

• Good for concurrency

• Bad for sequence
Event-based Production Rules

select-line → first
click first → rest
click rest → rest
double-click rest → <draw line>

• Note:
  • Events added to list of pending events
  • ‘first’ and ‘rest’ are internally generated events
  • Bad at state!
Graph Notations: STNs

- State Transition Networks (STNs)
  - Most common tool to specify dialogs
  - Established format (since 1960s)

- Consisting of:
  - **States** (usually the system waiting for some user action)
  - **Transitions** (which have a user action and a system response associated with them)

- Describes sequences of user actions and system responses
Graph Notations: STNs

Start ➔ Menu ➔ Circle 1 ➔ Circle 2 ➔ Finish

- Select 'circle'
- Click on center rubber circle
- Draw circle

Menu ➔ Line 1 ➔ Line 2 ➔ Finish

- Select 'line'
- Click on first point rubber band
- Double click rubber band
- Draw last line

- Click on point draw line rubber band
Example: STN for Personal Orchestra Dialog
**Legend:**

- **STATE NAME**: Description of what users do
- **input / output**: Input/output

**Description of states:**

- **FSM_INFOPAGE**: Info page selected / switch to info page
- **FSM_WAITFORCONDUCTOR**: Baton button clicked / switch to piece selection
- **FSM_CONDUCTING**: Conducting has started / start audio and video
- **FSM_COMPLAINT**: Conducted too fast or too slow / jump to complaint sequence
- **FSM_CONDUCTINGENDS**: Completed successfully / display congratulations page

**Transitions:**

- Complaint ends / display current piece selection
- Piece selected / switch to orchestra waiting
- Conducting has started / start audio and video
- Conducted too fast or too slow / jump to complaint sequence
- Completed successfully / display congratulations page
Please stand on the footprints
Bitte auf die Fußabdrücke stellen
请大家站好位置
Silhouettes: Server STN

• Unconventional notation (agreed upon in the team)

Legend:
 ◁ : operator pushes button

Note:
• server is responsible to fade sound in each state
Silhouettes: Client STN

(any state other than maintenance states) → svr. msg. "GotoStateM" → (stateM)

Init (maintenance state)
- run POST
- start Fl.
- connect to Fl.
- look for server
- look for web server

(svr. msg. "GotoStateElectricityPosing")
- send state → Fl.
- display shadow ©

ElectricityPosing
- display shadow ©
- send feature → Fl.

ElectricityCountdown
- overlay 3-2-1
- svr. msg. "GotoStateElectricityCountdown"
- send feature → Fl.

ElectricityResult
- send "illuminate building" → Fl.

Reconnecting (maintenance state)
- reconnect to server ©

WaitingForServer
- query state from server
- display error on client

(svr. msg. "DuplicateClientId")
- got server connection

Error (maintenance state)
- display error on client

(client timer)
- connection to server lost
- got server connection

Blank

(any state)
Waiting people will be asked to move up to the front and take their positions on the indicated spots.

Playing people will be presented a welcome screen.

People from the outer waiting queue will take their seats in the inside waiting booths.

DMX msg. - fade *in* spots on indicated footprints on the floor

srv. msg. "GotoStateWelcome"

DMX msg. - fade *out* light in the waiting booths

srv. msg. "GotoStateWelcome"

DMX msg. - fade *out* light in the waiting booths
A countdown will indicate that shadows will be “frozen”. People will see the representation of their shadows as buildings. Iconic illustrations will remind the player how to create buildings by their shadows while posing in front of the screens.

People will see their shadows while posing in front of the screens.
Iconic illustrations will remind the player how to pose in order to have their shadows overlap and thus create a power line.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be "frozen".

People will see illuminated buildings in 4 different versions??

ElectricityPosing
- collect features
- broadcast score

ElectricityCountdown
- overlay 3-2-1
- display shadow
- send state → Fl.
- send feature → server
- receive score ← server
- send score → Fl.
- send feature → Fl.

ElectricityResult
- send "illuminate building" → Fl.

ElectricityDescription
- send state → Fl.
- display shadow
- send feature → server
- receive score ← server
- send score → Fl.
- send feature → Fl.
- send feature → Fl.

ElectricityPosing
- send state → Fl.
- display shadow
- send feature → server
- receive score ← server
- send score → Fl.
- send feature → Fl.

ElectricityCountdown
- overlay 3-2-1
- display shadow
- send feature → server
- receive score ← server
- send score → Fl.
- send feature → Fl.
- send feature → Fl.

ElectricityResult
- send "illuminate building" → Fl.
Iconic illustrations will remind the player how to create trees by their shadows.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be “frozen”.

People will see the representation of their shadows as trees.
Iconic illustrations will remind the player how to pose in order to have their shadows overlap and thus create a water line.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be "frozen".

People will see the trees that they created with different green colours/trees with leaves?
People will see an animation of what will happen with their skyline.

People will see their city block with the block number and the URL of the DuC Website.
People will be asked to move out.

Beyond the playing area, visitors can deepen their understanding of city concepts at individual interactive stations before leaving the pavilion.
Hierarchical STNs

- **Start** and **Finish** states serve to glue an STN for a sub dialog (e.g., a certain menu selection) into a larger dialog (e.g., operating the application in general)

- Same expressive power as STNs, just more convenient

- The dialog structure of an entire system can be specified this way
Using STNs in Prototyping

• Create a simple STN for the dialogs envisioned

• Create one UI snapshot (sketch if paper prototype) per state (label it with the state name)

• Include offscreen area for annotations and to include extra buttons simulating user actions that do not correspond to simple clicks on the current screen

• When walking the user through your paper prototype, consult the STN to find out how to respond to each user action
Using STNs in Prototyping

• Alternative: Let the computer “execute” the STN to run the prototype

• Use tools such as Keynote, PowerPoint
select ‘graphics’

Graphics Submenu

select ‘text’

Text Submenu

select ‘paint’

Paint Submenu

Main Menu

select ‘graphics’

select ‘text’

select ‘paint’

Start

Menu

select ‘graphics’

select ‘text’

select ‘paint’

Circle 1

click on center
rubber circle

Circle 2

click on circumference
draw circle

Finish

Line 1

select ‘circle’

click on point
rubber hand

double click
draw last line

Finish

Line 2

click on point
rubber hand

draw line
rubber hand

Finish
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

<table>
<thead>
<tr>
<th>Graphics</th>
<th>Text</th>
<th>Paint</th>
</tr>
</thead>
</table>

Current state: **Main Menu**
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

Current state: Main Menu

<table>
<thead>
<tr>
<th>Graphics</th>
<th>Text</th>
<th>Paint</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>line</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

Current state: Circle 1

Click on Center

Click this button to simulate a click on the drawing area.

<table>
<thead>
<tr>
<th>Graphics</th>
<th>Text</th>
<th>Paint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Text

Paint

Circle 1

Graphics

Text

Paint
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

Current state: Circle 2

Click this button to simulate a click on the drawing area.

Click on Circumference
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

The circle is drawn now.

Current state: End of Drawing

Click the button to go back to main menu.

Back to Main Menu
Using STNs in Prototyping

Adapted from “Human–Computer Interaction” by Dix, Finlay, Abowd, and Beale, Chapter 8

Current state:
- Main Menu
- Circle 1
- Circle 2
- End of Drawing
Checking STN Properties: States

- Completeness
  - Can you get anywhere from anywhere?
  - Are all possible actions covered in every state?
  - How easily?

- Reversibility
  - Can you get to the previous state?
  - But NOT undo

- Dangerous states
  - Some states you don’t want to get to
Checking Transition Properties: Completeness

- Missing arcs indicate unspecified user input
- What happens when the user double-clicks in the Circle 2 state?
Checking Transition Properties: Reversibility

• E.g., reversing select ‘line’ requires
  \textit{Click - double click - select ‘graphics’} (3 actions)

• Note: Reverse means just getting back to a state, \textbf{not} to “undo” its effect

Prof. Dr. Jan Borchers: Designing Interactive Systems I • WS 2023/24
Dangerous States Example

• Word processor: two modes and exit
  • F1 - changes mode
  • F2 - exit (and save)
  • Esc - no mode change

• But ... Esc resets autosave
Dangerous States Example

- Exit with/without save ⇒ dangerous states
- Duplicate states - semantic distinction

- F1-F2 - exit with save
- F1-Esc-F2 - exit with no save
Dangerous States Example: Layout Matters

Old keyboard—OK

1. Old keyboard layout is intuitive and easy to use.
2. New keyboard layout is confusing and difficult to use.

Diagram:
- Old keyboard: F1, F2, F3, F4, Esc, Tab, 1, ...
- New keyboard: F1, F2, any update, Esc, ...

Diagrams showing the flow of actions:
- From edit to menu:
  - F1: go to menu
  - Esc: go back to edit
- From menu to exit:
  - F2: exit
  - Esc: go back to edit
Dangerous States Example: Layout Matters

new keyboard layout

Intend F1-F2 (save)

Finger catches Esc

F1-Esc-F2—disaster!
Checking STN Properties: Other Transition Properties

- Determinism
  - Several arcs for one action
    - Deliberate: application decides
    - Accidental: production rules
  - Nested escapes
- Consistency
  - Same action, same effect?
  - Modes and visibility
In-Class Exercise: STN

• Simple dialog to select bold, italics, and/or underline

• Draw the state diagram for:
  • Only Bold checkbox
  • Bold and italics checkboxes
  • All three checkboxes
Bold Checkbox

regular style  →  ‘bold’  →  bold

[Image of a text style dialog box with options for Bold, Italic, and Underline]
**Bold & Italic Combined**

![Diagram showing the combination of bold and italic styles]

- **Regular style**
- **Bold**
- **Italic**
- **Bold italic**

Text preview shows:
- Bold
- Italic
- Underline
All Three Options

- **Regular Style**
- **Bold**
- **Underline**
- **Italic**
- **Bold Italic**
- **Bold Underline**
- **Italic Underline**

![Text Style Preview](image)
Adding Another Option…

Normal case

CAPITALIZED
STNs: State Explosion

• Problem: Combining two concurrent STNs with N and M states leads to new STN with $N \times M$ states

• STN hides clear structure of the dialog

• Especially problematic with modern GUIs

• Similar problems with “Escape” and “Help” options
  • ESC can be modeled as special second “Finish” exit active throughout subdialog
  • Help can be modeled as little subdialog hanging off every single state in the STN

• Gets messy
Example: ESC & Help in STNs

- Click on center
- Rubber band
- Press HELP button
- Draw circle

- Click on circumference
- Press HELP button
- Help Subsystem

- Select ‘graphics’
- ESC
- Normal finish

- Select ‘text’
- ESC
- Normal finish

- Select ‘paint’
- ESC
- Normal finish

- Menu
- Finish
Petri Nets

• Better approach to dialogs that have several states at once

• But not better for sequential dialogs and mutually exclusive UI elements (radio buttons)

• Relatively old formalism to model concurrency
Petri Nets

- Transition fires when all input places have one or more token
  - A token is produced in each output place
- Positions of all tokens represent the current state
  - NOTE: This is different from state machines
In-Class Exercise

Draw the Petri net for our dialog box with concurrent “Bold” and “Italic” options (ignore “Underline” for now)
Petri Net For “Bold & Italic” Dialog

Tokens are consumed from all input places

User presses ‘Bold’

User action represented as a new token

Transition ‘fires’ when all input places have tokens

A token is produced in each output place
Petri Net For “Bold & Italic & Underlined” Dialog

User presses ‘Bold’

Bold On

User presses ‘Italic’

Italic On

User presses ‘Underlined’

Underlined On
State Charts

• By Harel; used in UML
• Example: TV Control Panel
• State Charts extend STNs
  • Hierarchy
  • Concurrent sub-nets
    • ON resumes both state machines
  • Escapes
    • OFF always active
  • History
    • Link marked “H” goes back to last state on re-entering subdialog
Diagrams For User Documentation

• Some dialog descriptions are clear enough to serve as user documentation (similar to GOMS)

• Especially if description uses screen shots and is semi-formal
Digital Watch – User Instructions

- Two main modes
- Limited interface
  - 3 buttons
- Button A changes mode

![Diagram of watch modes]

- Time display
- Stop watch
- Time setting
- Alarm setting

Depress button A for 2 seconds.
Digital Watch – User Instructions

• Dangerous states

• Completeness
  • Distinguish depress A and release A

• What do they do in all modes?
Digital Watch – User Instructions

and… that’s just one button
Semantics - Raw Code

- Event loop for word processor
- Dialogue description: very distributed
- Syntactic/semantic trade-off: terrible!

```java
switch (ev.type) {
    case button_down:
        if (in_text(ev.pos)) {
            mode = selecting;
            mark_selection_start(ev.pos);
        }
        ...
    case button_up:
        if (in_text(ev.pos) && mode == selecting) {
            mode = normal;
            mark_selection_end(ev.pos);
        }
        ...
    case mouse_move:
        if (mode == selecting) {
            extend_selection(ev.pos);
        }
        ...
} /* end of switch */
```
Further Reading

Alan Dix et al.: Human–Computer Interaction, 3rd ed. (2003), Chapter 16

Roadmap

Human

- Performance
- Models of interaction
  - Affordances
  - Mappings
  - Constraints
  - Types of knowledge
- Errors
- Visual Design

Case Studies

- History of HCI
- Visions
- Technology Phases

Development Process

- Iterative design
- User observation
- Ideation
- Prototyping
- User studies and evaluation
- Interaction design notation

Diagram:
- Goal
- Plan
- Compare
- Specify
- Interpret
- Perform
- Perceive
- World

Prof. Dr. Jan Borchers: Designing Interactive Systems I • WS 2023/24
What’s Next?

• **Designing Interactive Systems 2** (6 ECTS)
  https://hci.rwth-aachen.de/dis2
  - What makes a UI tick?
  - Technical concepts, software paradigms and technologies behind HCI and user interface development

• **Current Topics in HCI** (6 ECTS)
  https://hci.rwth-aachen.de/cthci
  - Understand & practice ways to do research in HCI
  - Learn about the latest research in HCI from recent conference and journal articles (and meet our Ph.D. students!)

Interested in a HiWi position or B.Sc./M.Sc. thesis?
https://hci.rwth-aachen.de/jobs