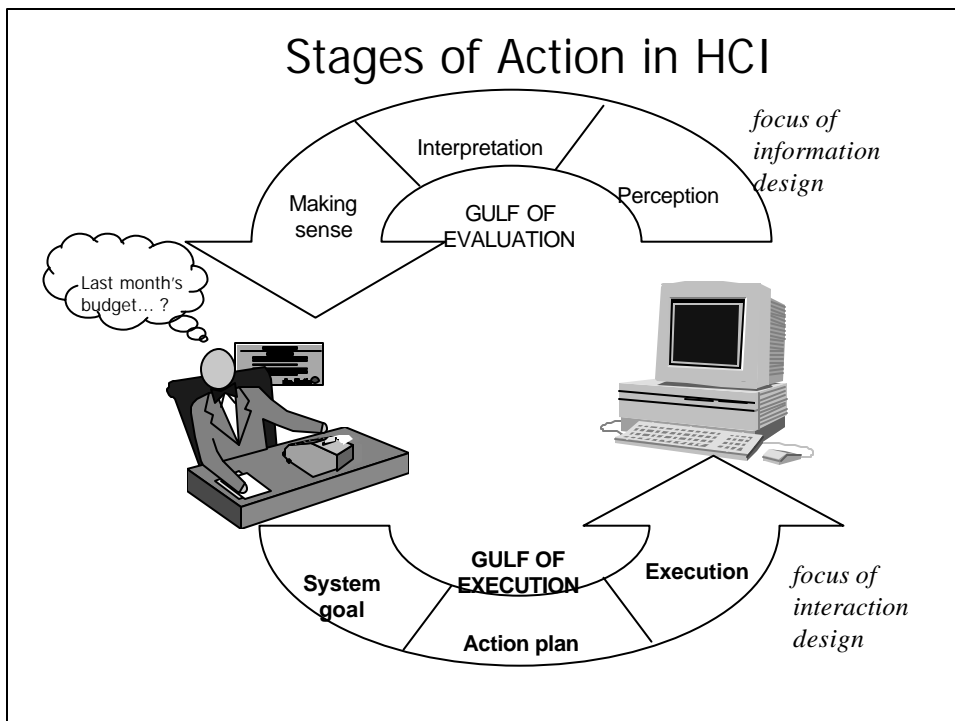


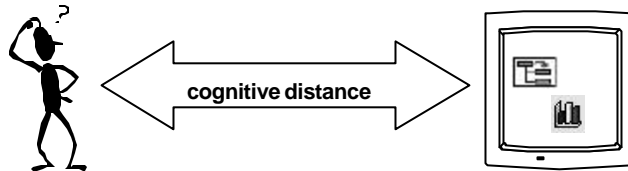
# Interaction Design

- Specifying the action sequences for planning and achieving one or more task goals
  - Conveying what *system goals* are possible, *plans* for accomplishing them, physical actions to *execute*
- Usability engineering of an interaction design
  - Ensure users can predict how to pursue goals, and that doing so is a comfortable and pleasant experience
  - Again, depends inherently on task, hence an important role of user interaction scenarios



## Selecting a System Goal

- Going from users' task concept to system concept: the *cognitive distance* between two models
  - Mental model held by users tells them what to do
  - This must make connection with designers' model that is conveyed and supported by the user interface



- The closer the match, the easier to find and pursue a relevant goal

## Suggesting Goals to the User

- Menu titles, folder names, application names, ...
- Decreasing the distance via direct manipulation:
  - UI controls appear as *physical analogs* of real objects; their affordances suggest interaction goals
  - Key ideas are visual representation, immediate and continuing feedback, and simple reversibility
- Visual or auditory UI elements sometimes lead to opportunistic selection of goals
  - Interesting object or message intrudes on a task
  - Or user is paused, choosing among things to do; especially common among novice users

## To Intrude or Not



## Recall Instead of Recognition

(Unix vs Windows)

- Commands refer to system objects and actions
  - Cognitive distance determined by *words* and *phrases* a system understands as user requests
  - Recall more demanding, but flexible & saves screen space
  - Design issues: vocabulary size and structure, familiarity and ambiguity (as discussed earlier)
  - Also the *syntax* (grammar) of the command language
- Compromise: nested menus support “hierarchy-path recall” of a large set of commands
  - e.g., Format —> alignment —> center
- What about using natural language for commands?

## Action Planning

- Plan analysis of required action sequences
  - Like HTA, goals decomposed into subgoals, steps, etc.
  - Includes choices and decision rules as relevant
  - Examine what *plan knowledge* is expected of the user
  - Look for arbitrary sequences, overall complexity, consistency, interference from one plan to another
- Ex: action plan for changing to double-space?

## Modeling Plan Knowledge with GOMS

GOAL: CLOSE-ACTIVE-WINDOW

GOAL: USE-MENU-METHOD

MOVE-MOUSE-TO-MENU-BAR

DRAG-DOWN-FILEMENU

RELEASE-ON-CLOSE-OPTION

GOAL: USE-HANDLE-METHOD

MOVE-MOUSE-TO-CORNER

CLICK-ON-CLOSE-BOX

GOAL: USE-CONTROL-KEY

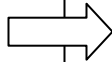
PRESS CONTROL+W]

Goals, operators, methods, and selection rules

# Designing Learnable Action Plans

- Try to make the sequence of actions match how user thinks about the real world task
  - design (or select from toolkit) interaction widgets that have good affordances
- Design a sequence, then analyze and refine
  - limited storage capacity of short-term memory (7 +/- 2)
  - look for ways to *chunk* long sequences of steps
  - use intermediate feedback or physical action to “mark” boundaries of subplans
  - ensure that subplan chunks match task subgoals

1. Specify text selection start
2. Specify text selection end
3. Select Format menu
4. Select Paragraph option
5. Set Special to FirstLine
6. Type value for FirstLine
7. Accept new settings



## 1. Select text

The example in the figure continues through the cycle to emphasize the important role of system feedback. While the execution takes place, some visual changes appear, when the file is opened, a new figure (the window) is seen. These changes are accompanied with respect to the spreadsheet content, and ultimately with respect to the budget equation.

## 2. Open Paragraph settings



## 3. Set indentation



## Guiding Action Planning with Forms

- Users are already familiar with form fill-out
- Procedure (action plan) is implicit in the layout
  - numbering or instructions may emphasize
  - tabs, or auto-advance for convenience
- Design issues similar to those with menus
  - size and complexity can be intimidating
  - decomposing into simpler forms that match task
  - flexibility, not everything is text-based
  - consistency in look and feel

## Giving Control to the User

- Humans are good at—and expect support for—doing multiple things at once
  - Working in parallel or “stacking” then reinitiating a task
  - Internal rather than external *locus of control*
  - When computer drags them along, it’s unpleasant!
- BUT, what does this mean for interaction design?

## Multi-threaded Interaction

- Multiple windows, each holds one thread
  - Tradeoffs between tiled versus overlapping?
  - Implies good support for *window management*
- Avoid modal dialogs unless they have task purpose
  - e.g., a preemptive dialog box that must be dismissed
  - When should you *deliberately include* modal dialog?
- Crucial role of status information
  - Users must be able to tell when they return to a window what they have done so far, what is possible now, etc.
  - Scenario-based design of “picking up the pieces”

## Action Execution

- Physical actions of dragging, clicking, typing, etc.
  - Design goal is to minimize, make things “hang together” as movements, not just plans
- Crucial role of *input device*—just like icons, etc., different devices have different affordances:
  - keyboard?
  - mouse?
  - joystick?
  - trackball?
  - data glove?

# Common User Input Devices

Device	Input Characteristics	Sample Applications
--------	-----------------------	---------------------

## Speech Input and Output

- Speech I/O inherently linear, relatively slow
  - Trades off with familiarity, naturalness
  - May address with restricted vocabularies—commands
- Speech recognition accuracy still limited
  - Depends on speaker, amount of training up front
- Synthetic speech output quality also limited
  - Biggest challenge is *prosody* (intonation contours)
  - Many systems use digitized natural speech snippets
  - BUT useful for alerts, warnings (why?)
- Biggest benefit: parallel processing, multi-modal
  - Also critical for hands-busy, heads-up tasks

# Designing for Errors

- Carefully analyze physically challenging actions
  - *Fitts Law*: time to select target is a regular function of distance and size of the target
  - But of course not all targets can be big and large :-)
- Overlearned procedures (e.g., From other systems) lead to intrusions (slips, not mistakes)
  - Most common form is typos, transposition of letters
  - e.g., Hitting delete before I get the text selected
  - Making a menu selection before menu pops-up

## Slips versus Mistakes

Type of Error	Example Situation	Design Approach
<u>Mistake</u> : asking for non-existent function or object	Mistyping the name of a command so that its function can not be executed	Represent (e.g., in lists, icons) what is available
<u>Mistake</u> : over-generalizing an earlier experience	In a listserv, using “reply” when intending to reply only to the sender of a message	Present through training or documentation a more complete set of examples
<u>Slip</u> : doing something that is appropriate, but not in current mode	Trying to input text into a document while the Font dialog box is open	Minimize modes and when necessary mark well with status and feedback cues
<u>Slip</u> : making a request that is interpreted as something else	Using keyboard short-cut to turn off underline before adding space (in PowerPoint reverses existing underline)	Improve consistency of low-level controls within and across applications
<u>Slip</u> : completing an automated (but inappropriate) action	Deleting a text selection before the selection has not been correctly specified	Predict locus of such errors and increase the amount of feedback (or alerts) provided

## Supporting Error Correction

- Forward/backward delete, click to de-select
  - usually not a question of design, built into UI platform
  - but what does “Back” on a browser get you?
- Common design problem is providing for Undo
  - predicting, supporting right level of reversibility
  - what are the issues here?

## Optimizing Execution Sequences

- Feedback and good defaults are essential!
  - especially in long, costly, or tedious transactions
- Consider implications of long term use
  - focus on actions for frequent choices, fast-paths
  - BUT, be careful to note when you are
    - violating overall consistency, or favoring one task at expense of other important or common tasks
- Customization: users define their own sequences
  - e.g., mapping commands to key combinations
  - can be critical when supporting users with special needs