ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

HUMAN COMPUTER INTERACTION GROUP

Interaction Design Principles

(outside of Cooper's book)

School of Computer and Communication Sciences

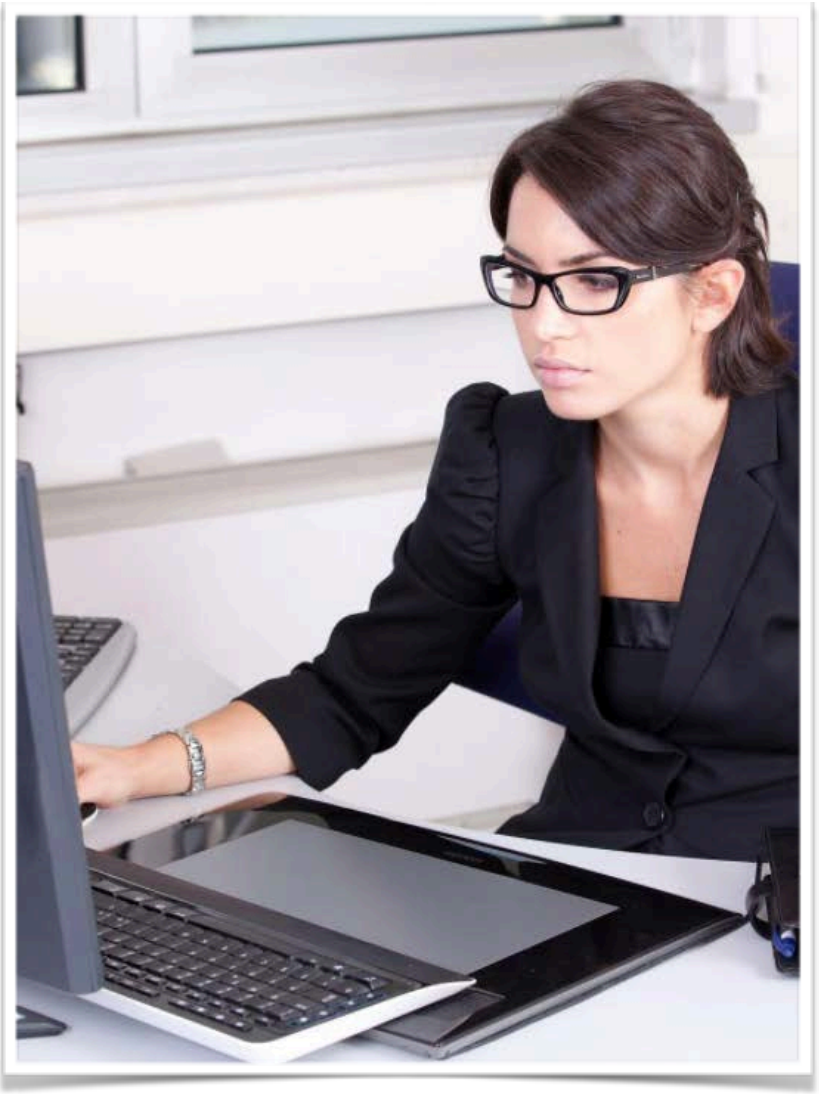
EPFL

Pearl Pu

Action cycles

Have you observed how someone interact?

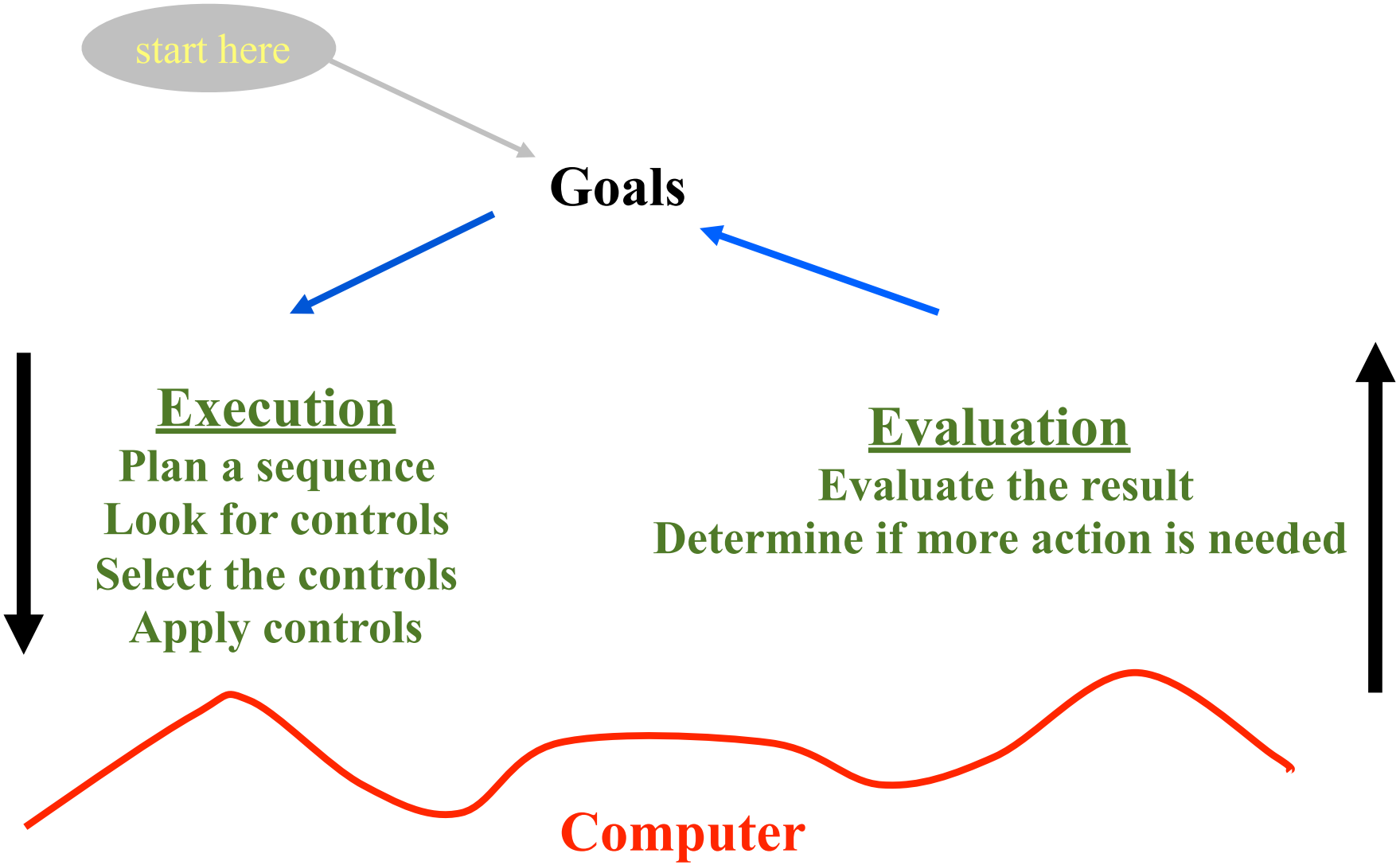
- We interact with computing machines everyday
- Have you ever observed someone else using the computers/mobile phones/bus ticket machines?
- Have you ever wondered what the cognitive and perceptive processes that are going on in their brains?



NORMAN'S ACTION CYCLE MODELS USER INTERACTION

1. Goal (users want to achieve something)
2. Users plan a sequence of actions using mental models (their understanding of how things work)
3. Users look at the interface screen for controls (buttons, menus)
4. They select the controls that appear to be most helpful in achieving the goal
5. They apply the controls
6. They evaluate the result of this action
7. They determine if they need more action cycles

ACTION CYCLE VISUALLY

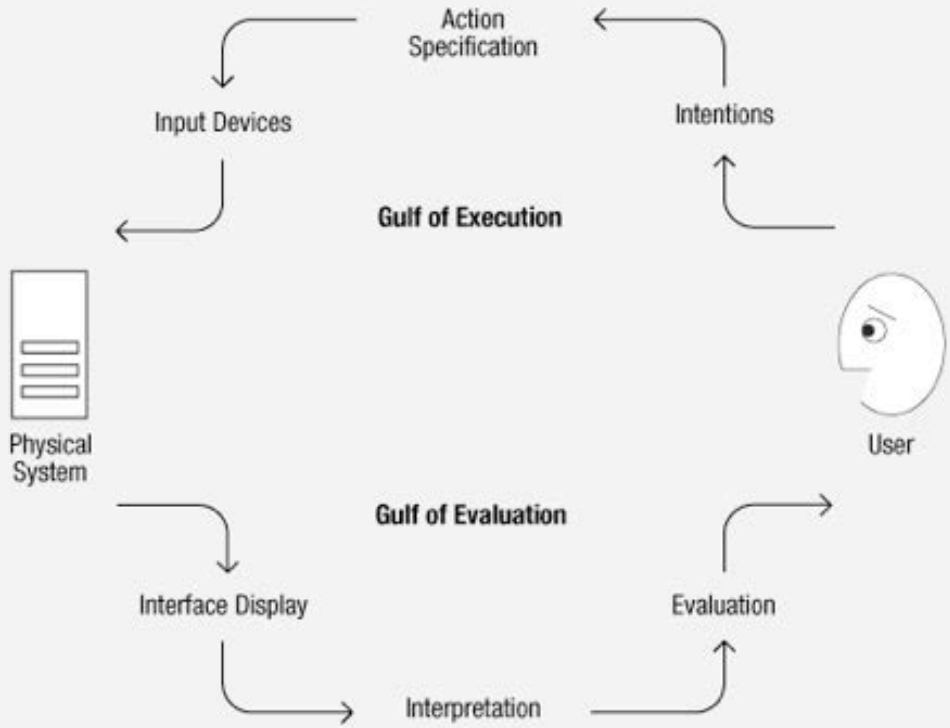


HCI design can be evaluated objectively

gulf = gap

- ***Gulf of execution*** is the ***amount of effort*** users exert to decide which interface controls can help them achieve their goals
- ***Gulf of evaluation*** is the ***amount of effort*** a person must exert to interpret ***the state of the system***.

Gulf of execution/evaluation



PROBLEMS OF POOR INTERFACE DESIGN

- A large gulf of execution - big gap between user mental model and designer's model; poor visibility of interface controls, etc.
- A large gulf of evaluation is the result of poor feedback

Two main reasons for poor interface design

CONCLUSION

- ***Action cycle*** explains how users interact with a piece of software

- ***Gulf*** of execution and evaluation summarise most interface and interaction problems

Interaction Design Principles

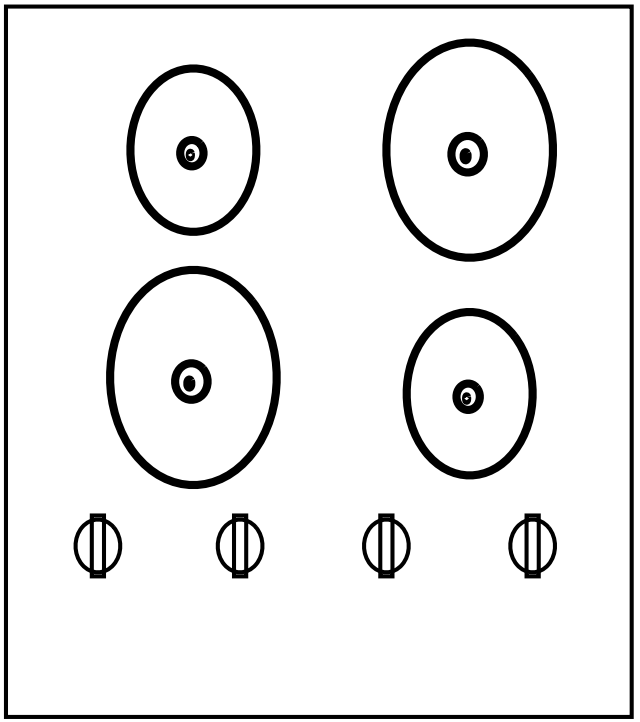
MAKE UI CONTROLS VISIBLE AND MEANINGFUL

- Visible
  - If users don’t see the controls, they will not select them
  - Use attention, color, and layout techniques to make visibility viable
- Meaningful/logical
  - If they don’t get the intended meanings, they will not select them
  - In addition, make the actions predictable

Design constraints to offer visual mapping

- Make the mapping between a goal and its actions as constrained as possible
- Avoid making the users choose between many options

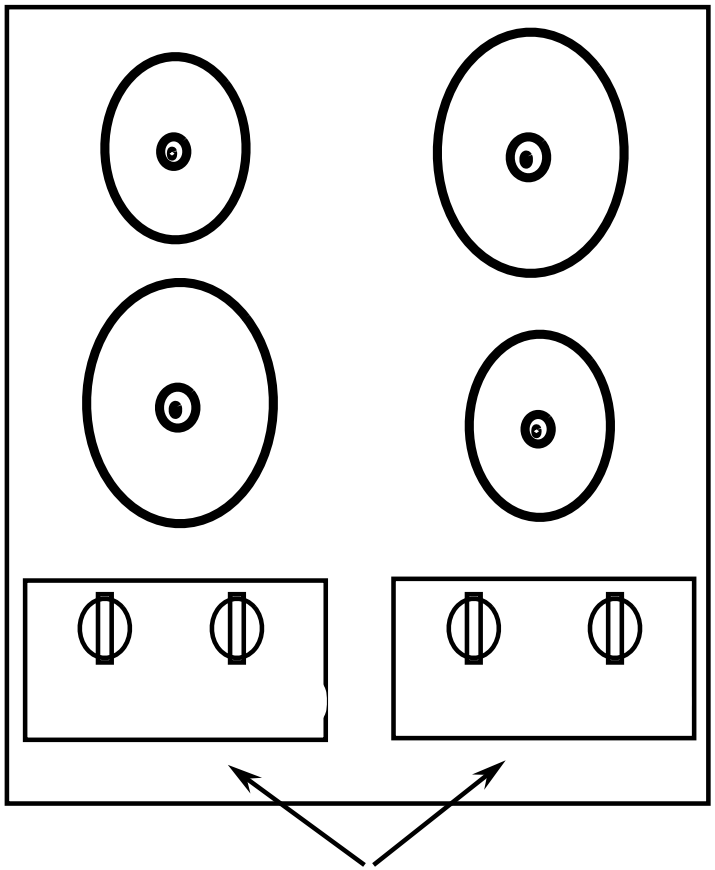
Arbitrary



24 possibilities, requires:
-visible labels
-memory

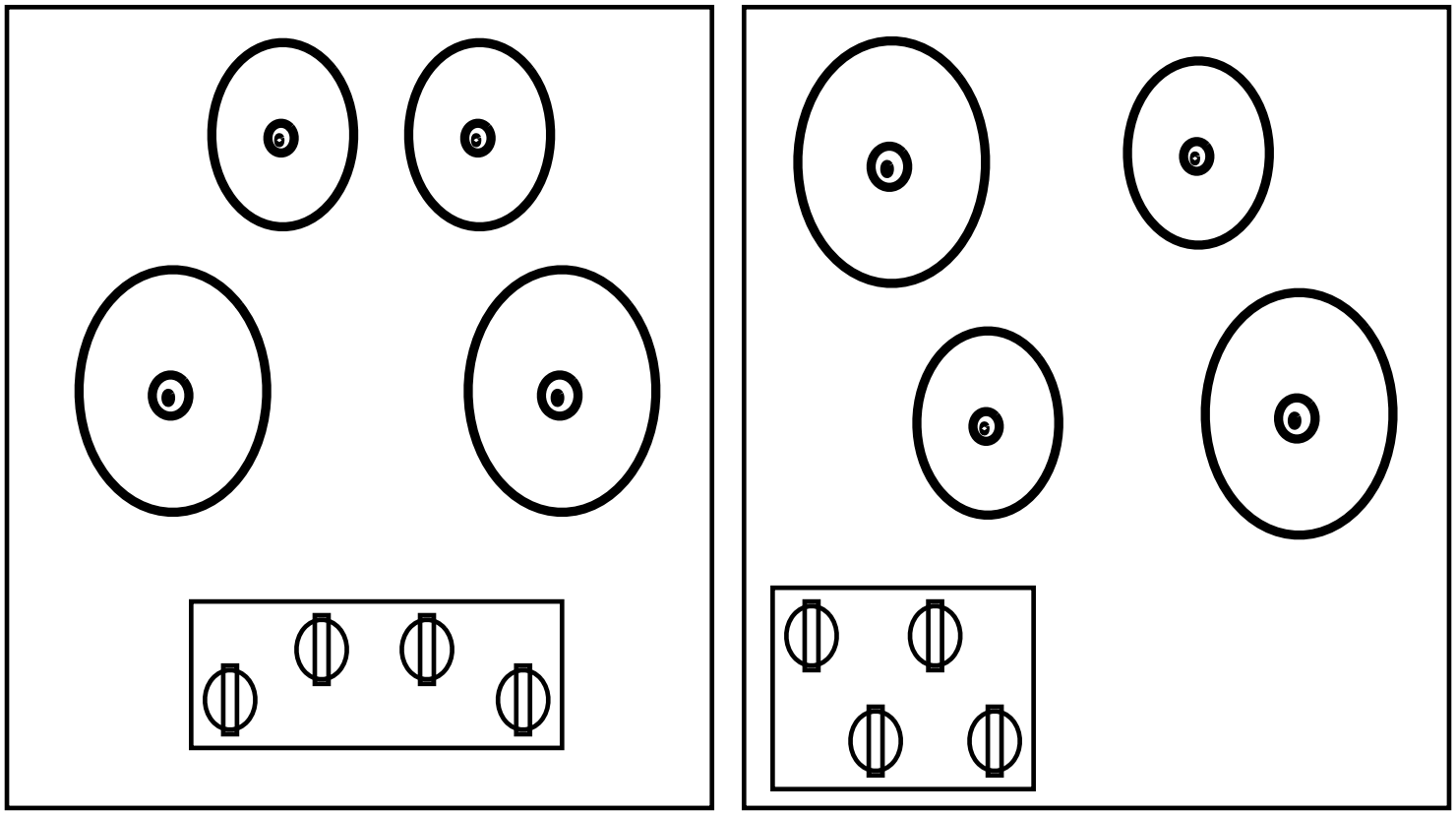
Design constraints to offer visual mapping

- Make the mapping between a goal and its actions as constrained as possible
- Avoid making the users choose between many options



2 possibilities per side
=4 total possibilities

Full mapping



One mapping!

PROVIDE FEEDBACK

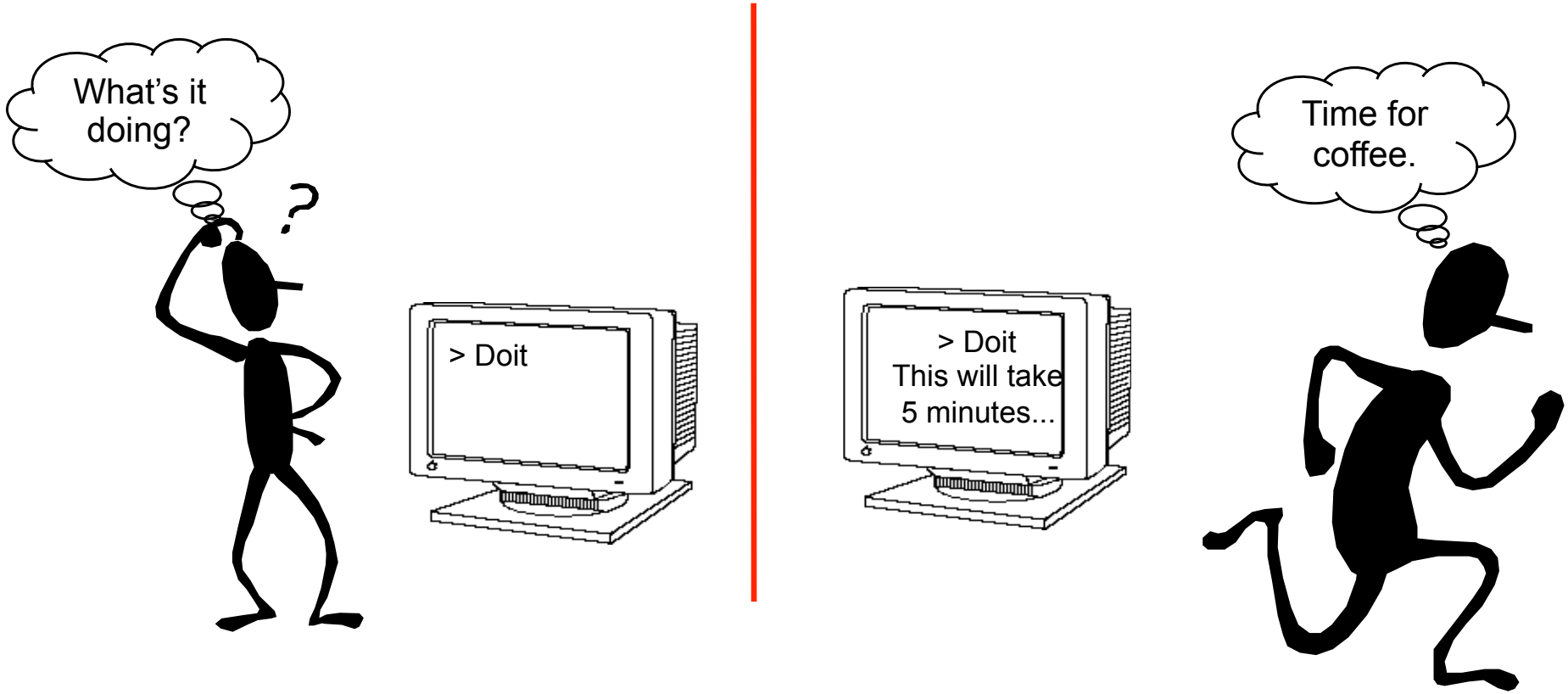
- Feedback helps users evaluate the effects of their actions
- They should be clear and informative

Read further in Sharp et al. 2006, pages 30-31

EXAMPLES

- Interface objects that give semantic feedback
  - Visual (progress bar, items in the shopping cart)
  - Text (Confirmation message of user actions)
  - Text + Visual (how much of hotel booking task is finished)

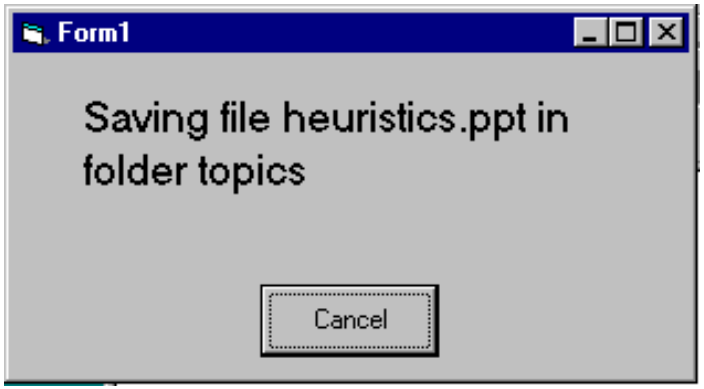
Give specific feedback for what's going on



Slide adapted from Saul Greenberg

Specific feedback continued:

Provide specific feedback within the context of the action



Slide adapted from Saul Greenberg

Feedback given along several dimensions

What mode am I in now?

What did I select?

How is the system interpreting my actions?

Slide adapted from Saul Greenberg

How quickly a feedback should be given?

- 0.1 second max → perceived as “instantaneous”
- 1 second max → user’s flow of thought stays uninterrupted, but delay noticed
- 10 seconds → limit for keeping user’s attention focused on the interaction.
- > 10 seconds → user will want to perform other tasks while waiting.

Nielson, J., Usability Engineering, AP Professional Press, Boston MA, 1994.

Slide adapted from Saul Greenberg

TASK BREAKDOWNS

- Help users accomplish their tasks by breaking down a big problem into baby steps

How do people break down tasks?

- They divide and conquer: top-down, or bottom up
- In the top-down approach, a user divides a task into subtasks, then divides subtasks into sub-subtasks. Then he solves a small unit of the problem, and backtracks to upper levels in order to proceed
- In the bottom up approach: a user constructs solutions to small chunks of problems. Then he constructs a bigger solution by combining small solutions

EXAMPLES

- Top-down: reserve a flight, buy a book, etc.
- Bottom-up: use latex to write a thesis (chapters, then thesis)

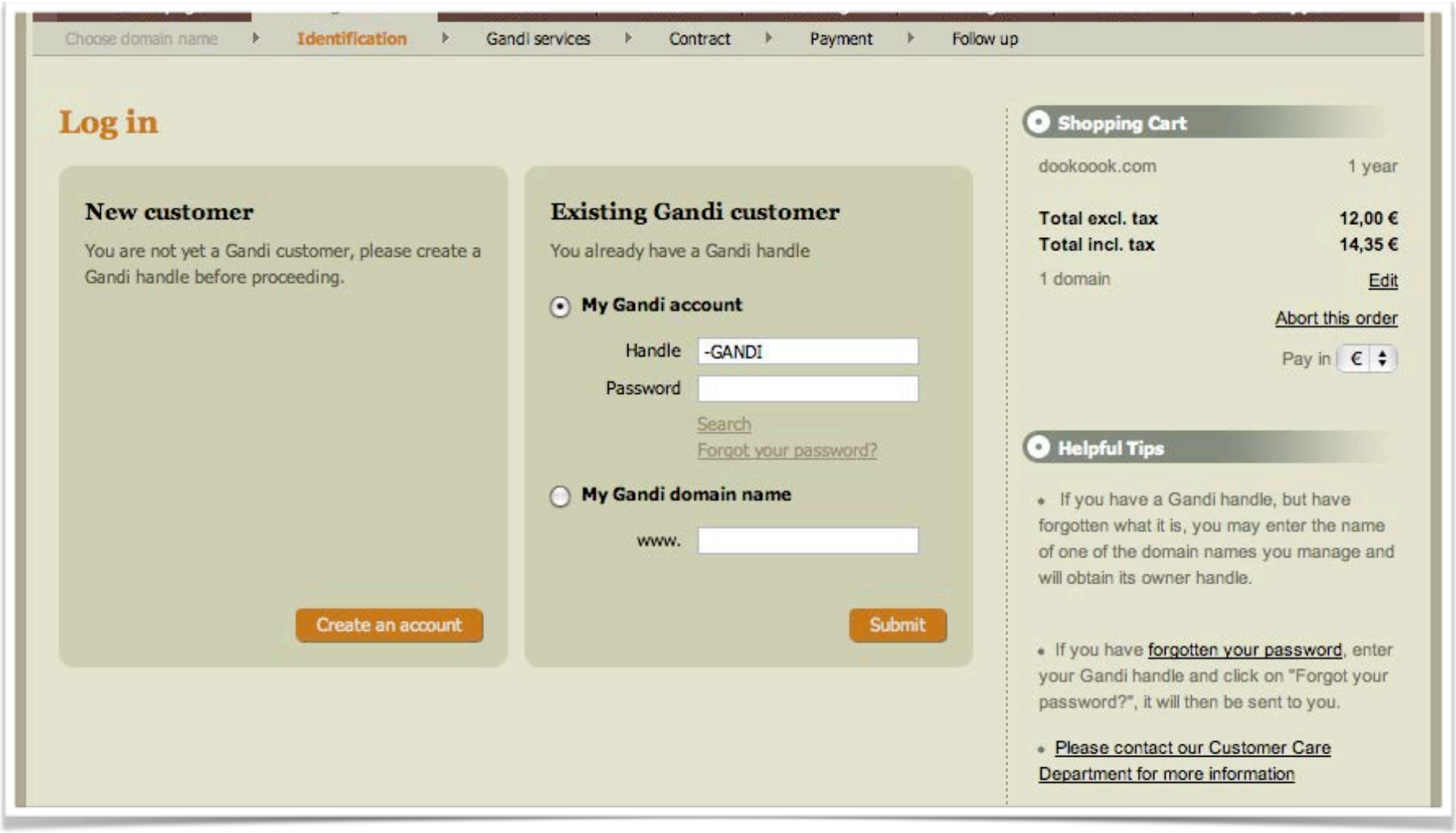
SAVING INTERMEDIATE RESULTS

- In either case, we need to help users save intermediate results
  - Use baskets to store temporary or intermediate items of interest
  - Use SAVE function to save temporary or intermediate results

TASK CLOSURE

- Help and remind users to finish the task
  - You have 4 items saved in basket. Would you like to check out?

Bad example



Can’t easily find the button to pay

TASK CLOSURE

- Designers should not distract users from their task closure by forcing them to work in different windows, or traversing a long list of choice items.
- Designers should help users focus by providing clear cues for context switching if side paths are necessary

CONGRATULATE THE USERS

- Designers often forget the best feedback users want from them
  - When the task is finished, congratulate him/her

PROVIDE CLEAR EXIT MARKS

- Provide clear exit marks especially in non-trivial places

- Example:
  - Even though it is possible to terminate a user session by closing the navigator's window, users are looking for the logout button when they finish

PROVIDE PREEMPTIVE EXISTS

- Users make many errors
  - Partly due to poor visibility and poor predicability
- Users change minds
- Users don’t like to feel trapped by the computer!
- Provide pre-emptive exits

EXAMPLES

- Strategies:
  - Cancel button (for dialogs waiting for user input)
  - Universal Undo (can get back to previous state)
  - Interrupt (especially for lengthy operations)
  - Quit (for leaving the program at any time)

Slide adapted from Saul Greenberg

Design principles for problem-solving

1. User research and mental model minimize gulf of execution
2. Design the interaction sequence as close as possible to the task sequences depicted in the task tree
3. Provide clear visibility of user actions in the UI
4. Design constraints to offer visual mapping between tasks and controls (the stove example)
5. Provide meaningful feedback to ease evaluation
6. Help users accomplish their tasks by breaking down a big problem into smaller pieces (piece-wise problem solving)
7. Store temporary results
8. Help users finish their tasks by providing task closure
9. Congratulate the user
10. Provide clear exit marks
11. Allow errors. Provide preemptive exists

THE END