“Discount” Methods:

- Cognitive Walkthrough
- Heuristic Evaluation

CPSC 544 FUNDAMENTALS IN DESIGNING INTERACTIVE COMPUTATION TECHNOLOGY FOR PEOPLE

Material created by Joanna McGrenere, Karon MacLean, Leila Aflatoony, Jessica Dawson & Heather O’Brien

Where are we in the term?

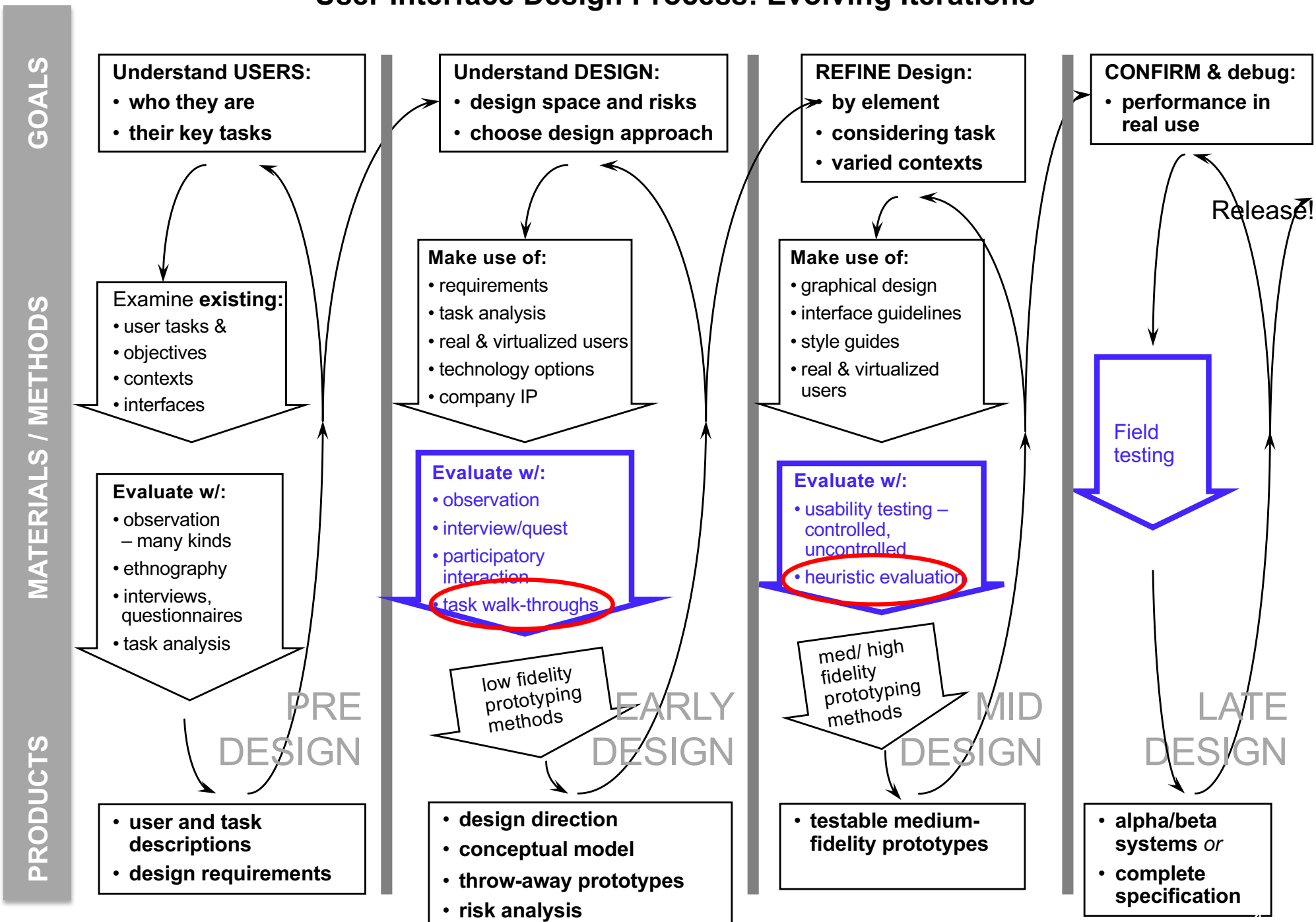
- As of today….
  - 10/13 research journals have been completed
    Remember: we'll use your 5 highest RJ marks.
  - Design project is +60% complete (50/80)

- Moving forward
  - **Design project**
    - Low-fidelity prototype presentation – **Nov 5**
    - User test report – **Nov 21**
    - Medium-fidelity prototype presentation – **Dec 6**
    - Final prototype & report – **Dec 10**
  - **Participation self-assessment – Dec 10**

Learning goals

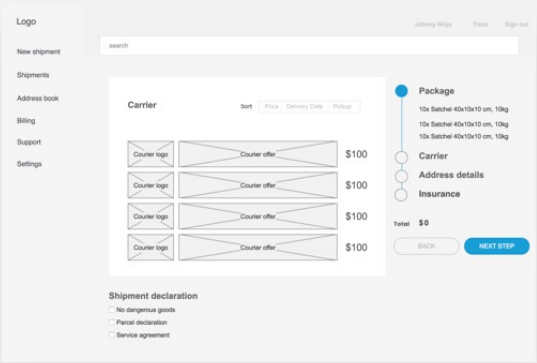
- Reflect on what we have learned in the course, and where we are going
- Explain why **cognitive walkthrough (CW)** and **heuristic evaluation (HE)** are considered **discount** usability methods
- Describe the pros/cons of cognitive walkthroughs and heuristic evaluation, and when it is an appropriate choice of evaluation method
- Outline the general procedure for conducting a cognitive walkthrough and a heuristic evaluation
- Try a mini-version of a HE

User Interface Design Process: Evolving Iterations

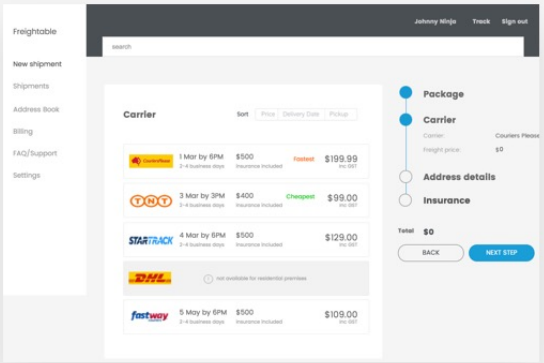


K MacLean - derived from version by Saul Greenberg (U Calgary)

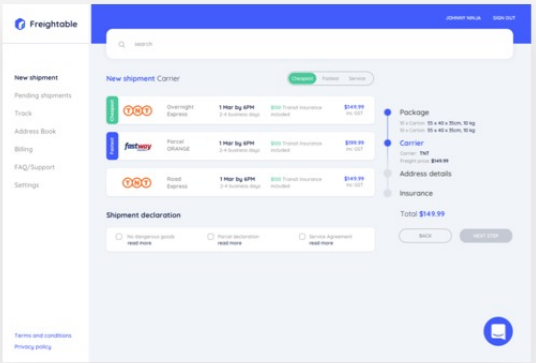
Iterative prototyping & testing



Low-fidelity



High-fidelity



Final design

Role of users

Discount methods → Usability testing → Experiments

Prototype Image: https://medium.com/7ninjas/low-fidelity-vs-high-fidelity-prototypes-903a7befaa5a

Discount usability testing

- Relative to usability testing (next class)…
- Cheap (thus ‘discount’)
  - no special labs or equipment needed
  - doesn’t need to involve users directly
- Quick
- Easy to learn

Types of discount methods

- **Cognitive Walkthrough: "mental model"**
  - Assesses "exploratory learning stage" (the novice)
  - What mental model does the system image facilitate?
  - Done by usability-experts and/or domain experts
  - How does the user think about the basic system image? Assess on low-fidelity prototypes; tests your conceptual model
- **Heuristic Evaluation: "fine tune"**
  - Targets broader use range (including expert)
  - Fine-tunes a more advanced prototype (medium to high; deployed systems)
  - HCI professionals apply a list of heuristics while simulating task execution

Today…

1. We'll briefly go over the CW process
   - You'll use it on your project in November
     (on your lo-fi prototype)
   - Describing it first … because you use it earlier in the design process.
2. Introduce the HE process,
   - Including ~10 slides on the actual heuristics
   - Then, we'll practice in an activity
     Won't have time to complete the HE, but will give you the feel of it
   - Later in this course, you'll build a medium-fidelity prototype
     But our scope excludes med-fi evaluation.
     This is one tool you'd ideally use for that step.

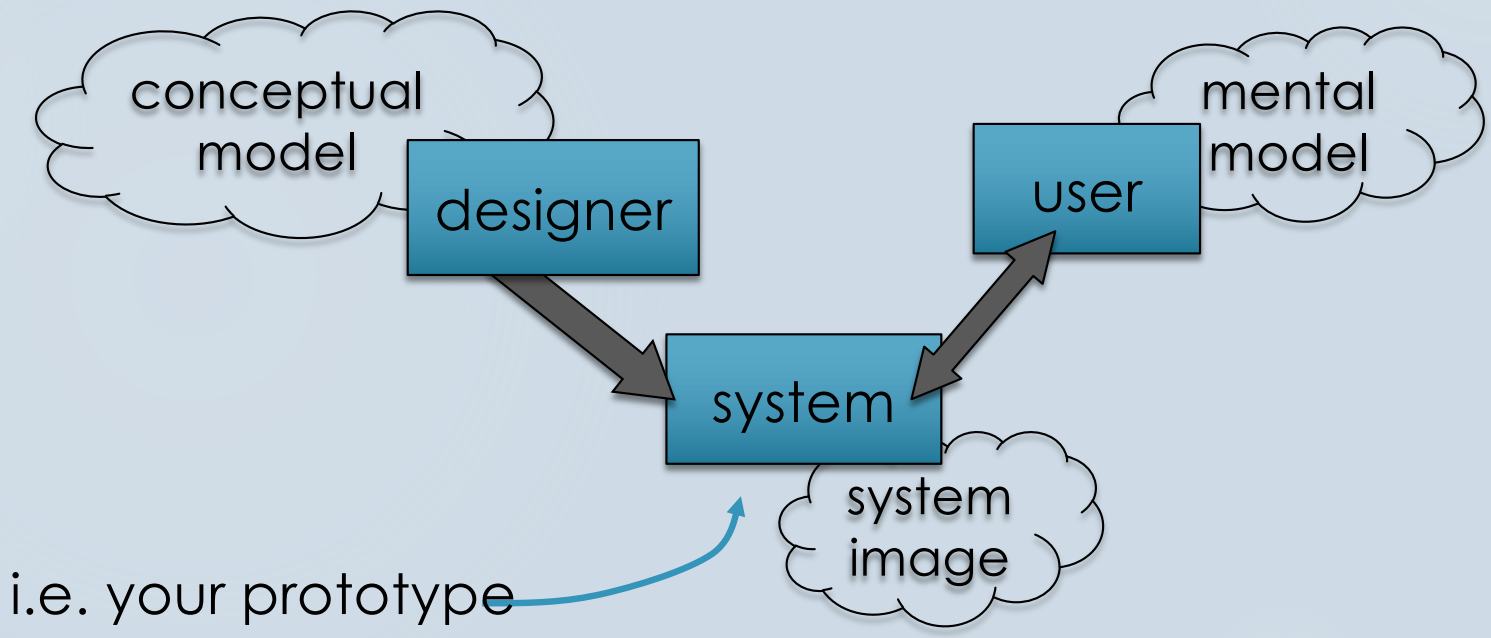
Cognitive Walkthrough

How do cognitive walkthroughs fit in?

- Recent topics:
  - Task examples: encapsulate a task (design independent)
  - Conceptual model: how DESIGNER wants to portray system to user
  - Mental model: how the USER (actually) thinks the system works
  - System Image: what the USER sees, and uses to form their mental model. It includes the interface itself.

CW simulates mental model development

- Assessing…
  - Is the conceptual model an effective one?
  - Does the interface design communicate the conceptual model?
  - How well does it support forming a good mental model?



Cognitive Walkthrough: tests **exploratory learning**

- When to use it:
  - Develop/ debug an interface, without accessing users (which is expensive)
- Tests: how well
  - Interface design underlying conceptual model aligns with/sets up the user's mental model
- Not for:
  - Assessing performance at highly skilled, frequently performed tasks; or finding radically new approaches

Cognitive Walkthrough

- Possible outputs:
  - Loci & sources of confusion, errors, dead ends
  - Estimates of success rates, error recovery
  - Helps to figure out what activity sequences could or should be
- What's required:
  - **Task examples:** design-independent descriptions of tasks that representative users will want to perform
  - **A prototype** to provide a design
- Who does it: [theoretically] anyone – usually design team members or expert outside analysts.
  - can use real users . . . but this makes it a lot less 'discount'

Overview: A CW Evaluation

Start with: Task examples + interface design (low-fi prototype)

Process:

1. Break task down into steps:
   - user actions
   - expected system responses tot hem
2. Perform each step on the existing interface
3. If you locate a problem, mark it & pretend it has been repaired; then go on to next step.

CW, Step 1:
Generate "correct", intended steps to complete a task.

- Select a task to be performed and write down all the 'user actions', and expected "system responses".
- Two approaches:

  a) Get **very specific**: correct user action = e.g., "type '36g' into the text entry box in the middle of the screen

  OR

  a) Use **high-level directives only**: correct user action = ''enter amount of food for pet feeder to dispense"

  Think about: why would you choose one vs the other?

Return to your task examples, task definitions and requirements to prioritize the tasks you'll analyze

CW, Step II: Carry out the steps

- For each step:
  - Q1: Does the user know what to do?
    Do they have enough info? etc.
  - Q2: Explore: will the user see how to do the step?
    Look for the needed action? is it visible? it is obvious how to perform the step?
  - Q3: Interpret: will the user correctly understand the system response?
    Is the feedback understandable? Will the interpretation be correct?

- Note: even with an error, user may have progressed if error became apparent (i.e., figured out the misunderstanding).

  Distinguish this from when user **continues** with a misunderstanding.

Approaches to instructing person(s) doing CW

Corresponds to how you've defined the tasks (2 slides ago)

- Approach (a):

  LO-FI prototypes

  - Participant follows the pre-prepared steps and assesses according to expected actions/system response

- Approach (b):

  HI-FI/ released systems

  - Give the CW participant ONLY the higher level directive(s).
    - E.g., "create an event note with the following attributes. . . "
    - More exploratory
    - BUT - the steps they take might diverge from the list you made – note them down on another action-list sheet. These points should trigger further analysis

What kinds of problems should I record?

- In a CW you may note many kinds of problems, e.g.,
  - Problems with particular steps
  - Problems moving between steps
  - Larger problems that involve lots of steps
  - Larger problems that hint at deeper problems with conceptual model/design
  - Small problems that might only apply to unique users
  - Other kinds of problems that just become apparent while using interface, etc.
- Make note of these as appropriate
  - if you do a lot of CWs, you may develop your own template for noting problems

How do I become good at doing CWs?

- When you're new to CWs, it's easy to assume the user will know what to do if YOU know what to do
  - Force yourself to imagine what the user might not know
- When asking the questions at each step:
  - Really think about what the user could be thinking. . .
  - Consider the impact of misconceptions or mistakes that they could have made earlier!
- Perform lots of them!
  - You'll get better at figuring out what to focus on with practice

What do I do after the CW?

- CWs can be done in teams or individually
  - Aggregate and discuss problems
    - Possibly found over more than one CW evaluation
  - Prioritize problems based on severity, likelihood

  THEN:

  - Iterate and fix as required
    - Decide on which you can/will address
    - Iterate on conceptual model and/or interface design
  - OR write up a report/recommendations → design team
    - If you're not the one(s) doing the designing

Activity: Cognitive Walkthrough

*We will do this **Nov 8** using your low-fi prototypes*

- Note that while the walkthrough is **typically performed by the interface designer and a group of his or her peers**... We will swap team members for the walkthrough.
- Walkthrough Oral Report: Toward the end of the class, each team will briefly summarize what they learned in the walkthrough, good and bad.
  - Your assessment should focus on your task examples. If you found nothing wrong for a given task, (i.e., your interface is perfect) then outline the ways in which it worked well (e.g., "Our cognitive walkthrough showed that users can do X, Y, and Z without errors or confusion.").
- You should **leave the working class with a list of minor issues** that you will address based on the feedback. (There is not sufficient time to address major issues/start from scratch.)

Preparation: Bring your lo-fi prototype to class;
Decide on how you will instruct the user (see slide 19);
Discuss how the team will take notes to bring together the results

Heuristic Evaluation

Heuristic Evaluation

- Identify (listing & describing) problems with existing prototypes (any kind of interface); for any kind of user, new or proficient
- Research result:
  - 4-5 evaluators usually able to identify 75% of usability problems
  - User testing and usability inspection have a large degree of non-overlap in the usability problems they find (i.e., it pays to do both)
- Cost-benefit:
  - Usability engineering activities often expensive / slow; but some can be quick / cheap, and still produce useful results
  - Inspection focuses less on what is "correct/best" than on what can be done within development constraints
  - Ultimate trade-off may be between doing no usability assessment and doing some kind

HOW TO PERFORM A HEURISTIC EVALUATION

1. Design team supplies scenarios, prototype, list of heuristics; Need 3-5 evaluators: train in method if non-expert
2. Each evaluator **independently** produces list of justified, rated problems by stepping through interface and applying heuristics at each point
   … use heuristics list & severity rating convention
3. Team meets and compiles report that organizes and categorizes problems

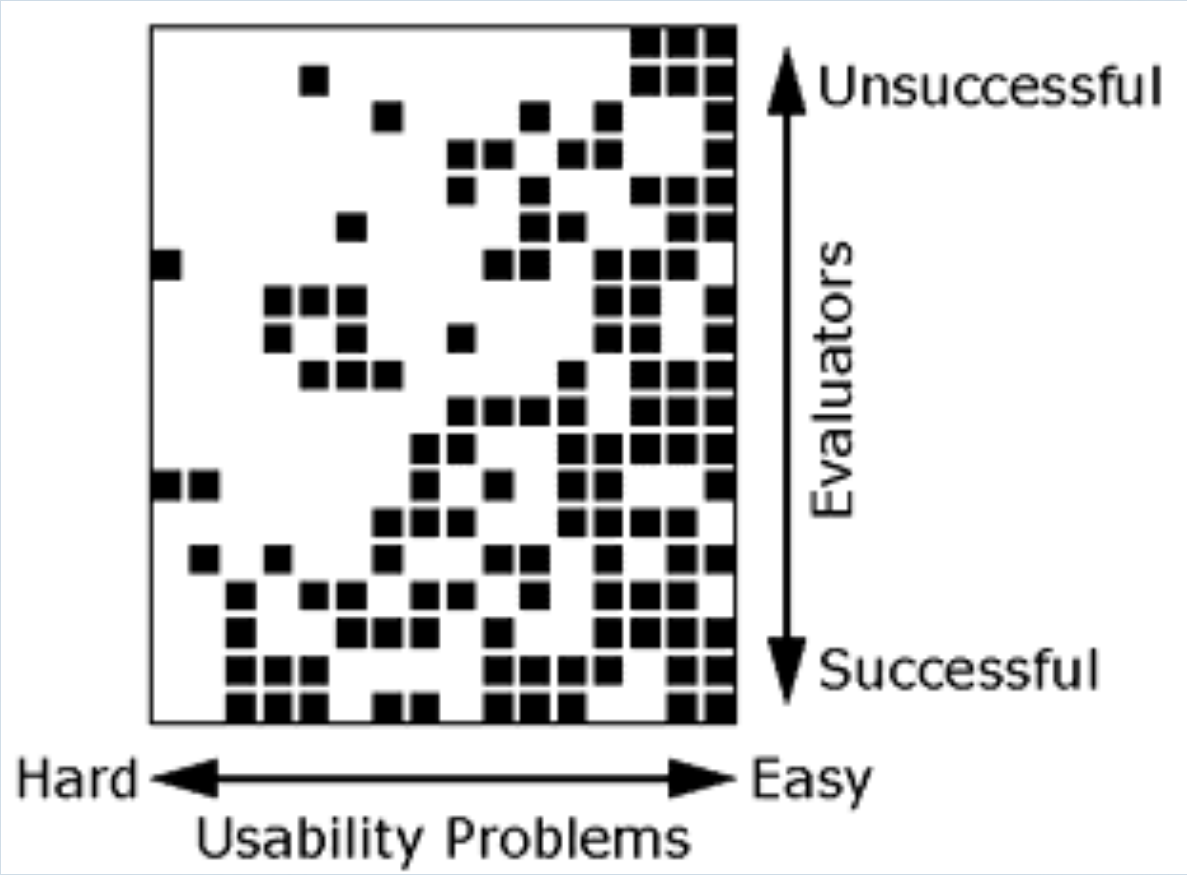
Individuals vs. Teams

- Nielsen recommends individual evaluators inspect the interface alone.

- Why?
  - Evaluation is not influenced by others
  - Independent and unbiased
  - Greater variability in the kinds of errors found
  - No overhead required to organize group meetings

why multiple evaluators?

- Every evaluator doesn't find every problem
- Proficient evaluators find both easy & hard (subtle) ones



one popular list of heuristics (Nielson, '93)

- H1: Visibility of system status
- H2: Match between system & the real world
- H3: User control & freedom
- H4: Consistency and standards
- H5: Error prevention
- H6: Recognition rather than recall
- H7: Flexibility and efficiency of use
- H8: Aesthetic and minimalist design
- H9: Help users recognize, diagnose & recover from errors
- H10: Help and documentation

https://www.nngroup.com/articles/ten-usability-heuristics/

H1: Visibility of system status

- The system should always keep users informed about what is going on, through (appropriate feedback within reasonable time)

Example 1: Visual feedback

What mode am I in now?

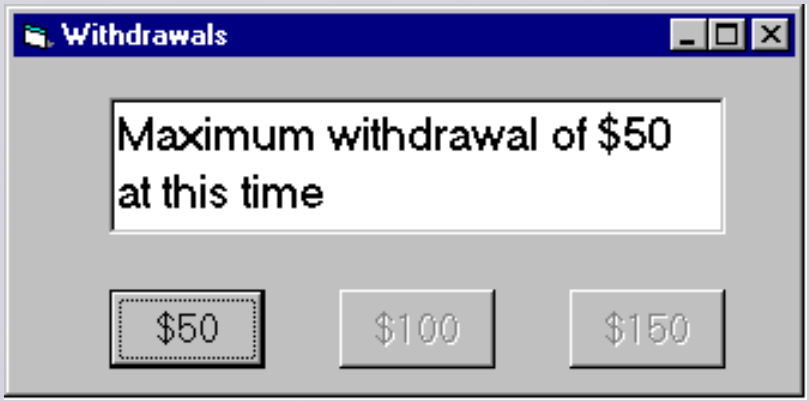
What did I select?

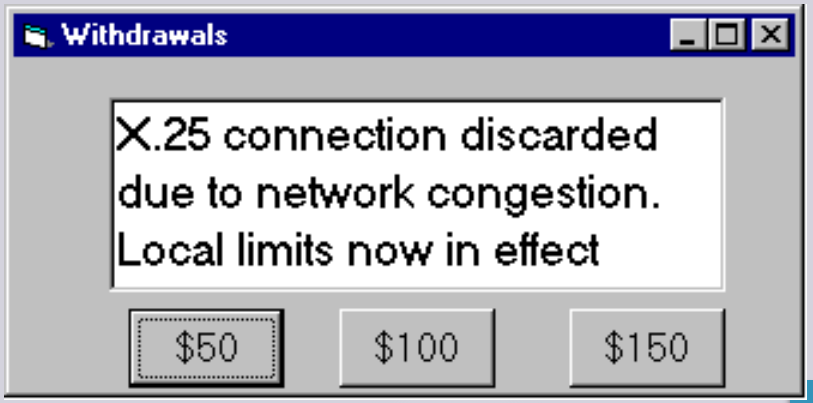
How is the system interpreting my actions?

Example 2: Wait time

H2: Match between system & real world

- Speak the users' language
- Follow real-world conventions
- Information appears in a natural and logical order
  - e.g. withdrawing money from a bank machine





H3: user control & freedom

- "Exits" for mistaken choices, undo, redo
- Don't force down fixed paths

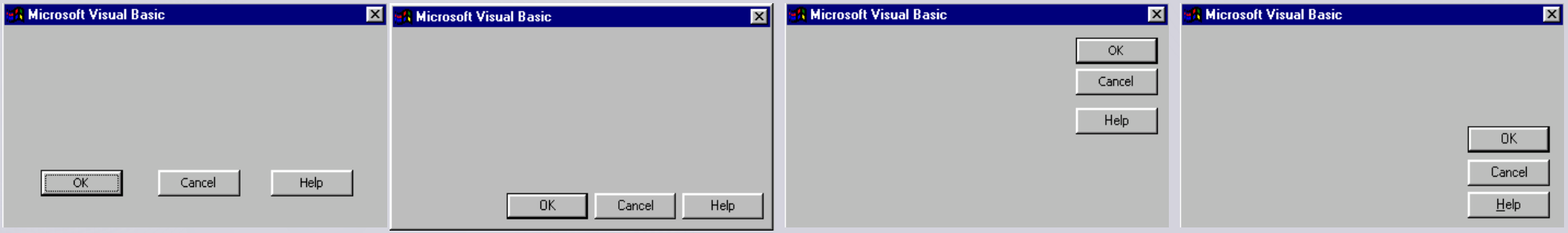
Strategies:

- Cancel button
- Universal Undo
- Interrupt
- Quit
- Defaults

How do I get out of this?

H4: Consistency & standards

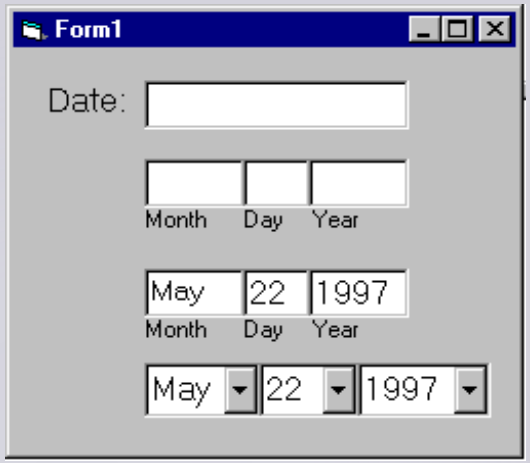
- Consistency of effects → predictability
  - Same words, commands, actions should always have the same effect in equivalent situations
- Consistency of language and graphics
  - Same info/controls in same location on all screens/dialog boxes -NOT:

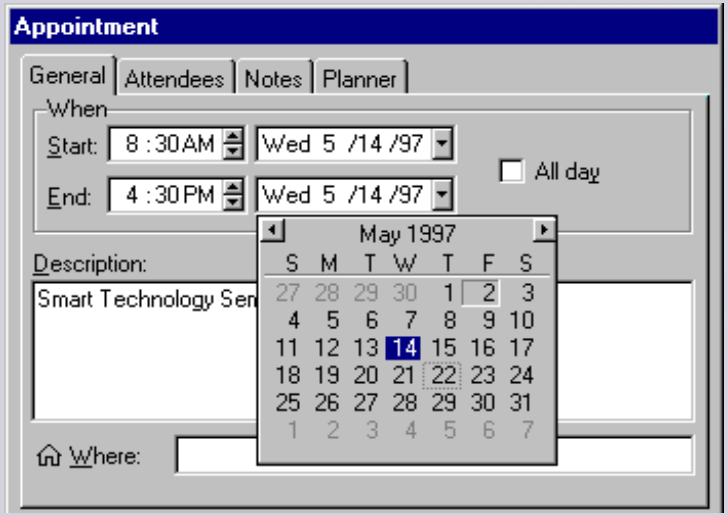


  - Same visual appearance across the system (e.g. widgets)
    - e.g. NOT different scroll bars in a single window system
- Consistency of input
  - Require consistent syntax across complete system

H5: Error prevention

- Try to minimize errors
- Modern widgets: only “legal commands” selected, or “legal data” entered

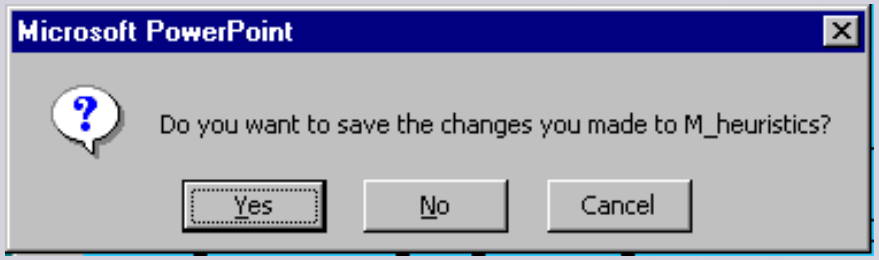




- mistakes
  - arise from conscious deliberations that lead to an error instead of the correct solution
- slips
  - unconscious behavior that gets misdirected enroute to satisfying goal

H5: Types of slips

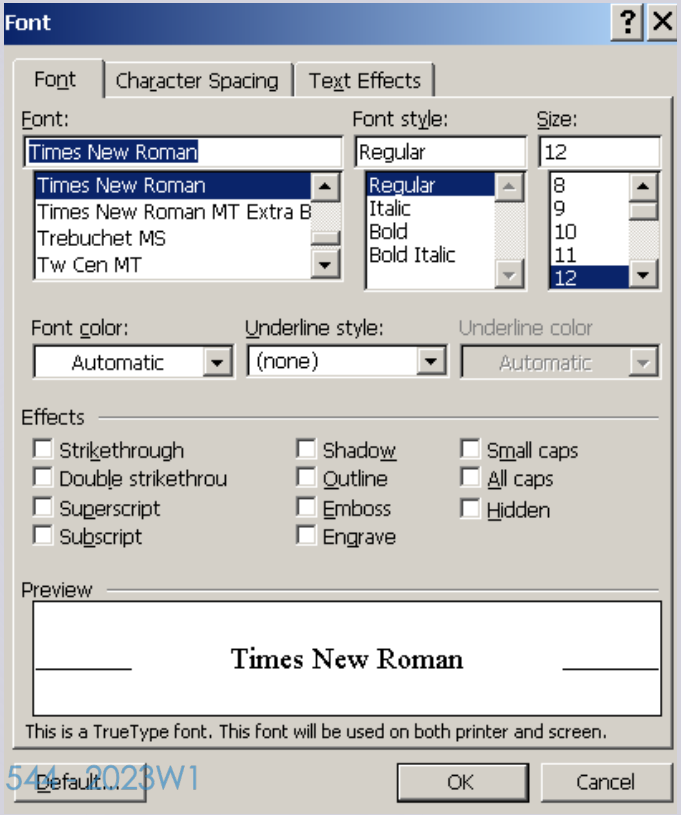
- **Capture error**
  - frequent response overrides [unusual] intended one
  - occurs when both actions have same initial sequence
    - confirm saving of a file when you don't want to delete old version

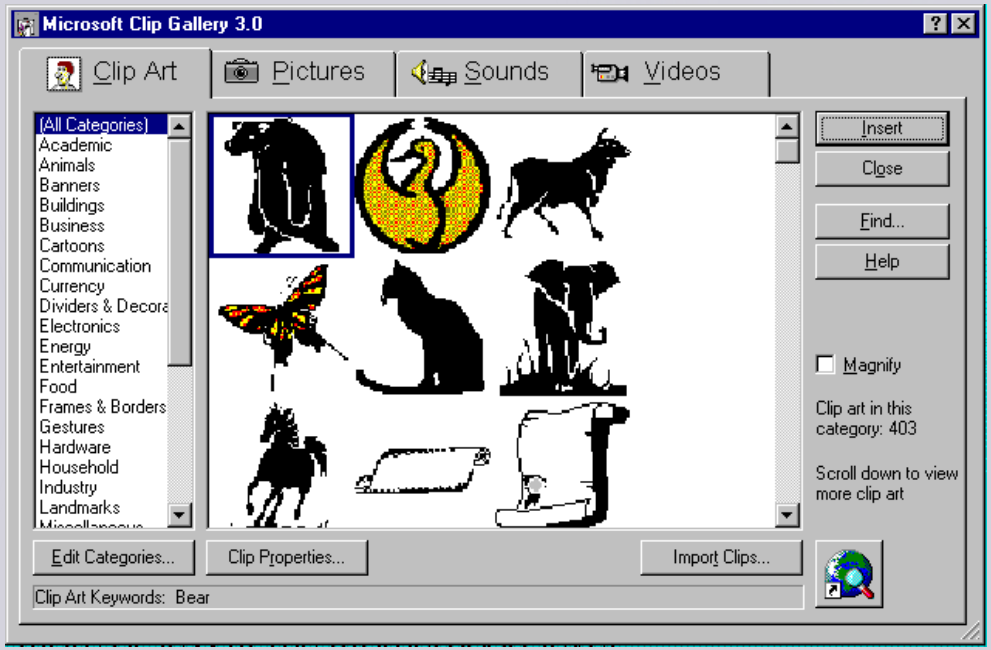


- **Description error**
  - intended action has too much in common with other actions, e.g. proximity of objects
    - move file to trash instead of to folder
- **Loss of activation**
  - forgetting the goal while carrying out the action sequence, e.g. start going to a room and forget why by the time you get there
    - navigating menus/dialogs, can't remember what you are looking for
- **Mode errors**
  - people do actions in one mode, thinking they are in another
    - look for commands / menu options that are not relevant

H6: Recognition rather than recall

- Minimize the user's memory load
- Make objects, actions, and options visible
- Instructions for use of the system should be visible or easily retrievable





H7: Flexibility and efficiency of use

- **Experienced users should be able to perform frequently used operations quickly**
- Strategies:
  - Keyboard and mouse accelerators
    - abbreviations
    - command completion
    - menu shortcuts & function keys
    - double clicking vs. menu selection
  - Type-ahead (entering input before the system is ready for it)
  - Navigation jumps
    - go to desired location directly, avoiding intermediate nodes
  - History systems
    - WWW page revisits

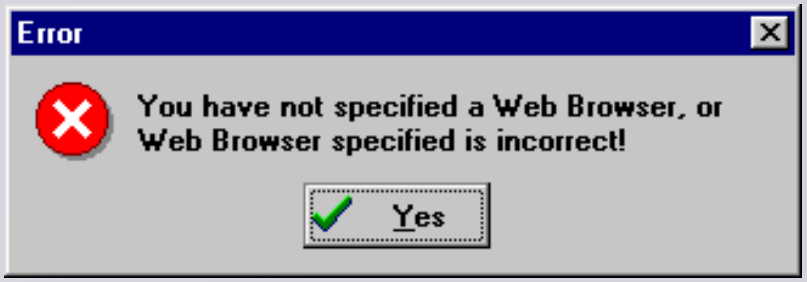
H8: Aesthetic and minimalist design

- Dialogues should not contain information which is irrelevant or rarely needed.

Image: https://www.justinmind.com/blog/10-90s-websites-designs-you-wont-believe-existed/

H9: Help users recognize, diagnose, and recover from errors

- Error messages should be expressed in plain language
- indicate the problem, and constructively suggest a solution

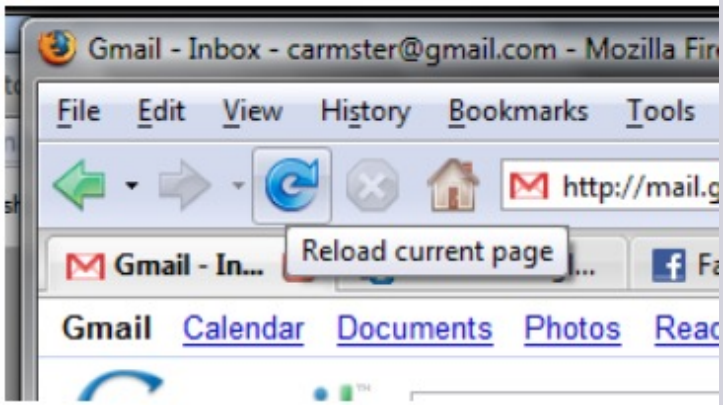


H10: Help & documentation

- Help is not a replacement for bad design!
- Simple systems: walk up and use; minimal instructions
- Most other systems:
  - Feature-rich
  - Some users want to become “expert” rather than “casual” users
  - Intermediate users need reminding, plus a learning path
- Many users do not read manuals
- Usually used when users are panicked & need help NOW
  - Need online documentation, good search/lookup tools
  - Online help can be specific to current context
- Sometimes used for quick reference
  - Syntax of actions, possibilities
  - List of shortcut

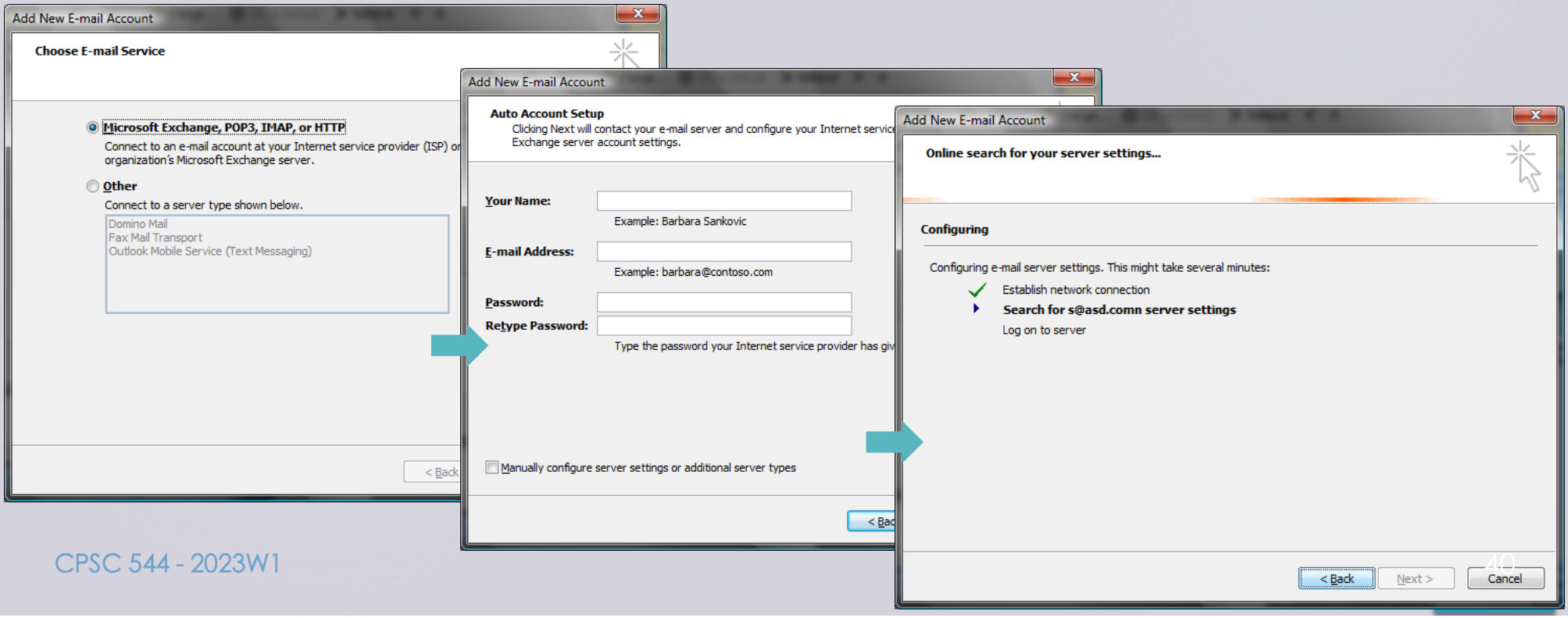
H10: types of help

- Tutorial and/or getting started manuals
  - Short guides that people usually read when first encounter system
  - On-line "tours", exercises, and demos
- Reference manuals
  - Used mostly for detailed lookup
  - On-line hypertext
    - Search / find; table of contents; index
- Reminders
- Keyboard templates
  - shortcuts/syntactic meanings of keys
- Tooltips
  - Text over graphical items indicates their meaning or purpose



H10: Types of help (cont'd)

- Context-sensitive help
  - System provides help on the interface component the user is currently working with
- Wizards
  - Walks user through typical tasks
  - Reduces user autonomy



Step 1: Briefing session

- Get your experts together
  - Brief them on what to do, goals of system, etc.
  - Discuss heuristics to be applied

- May also want to provide experts with:
  - Some examples of tasks
  - Descriptions of user personas
  - Simple instructions/guidance
    - especially if NOT a fully functioning system

Step 2: individual evaluation

- At least two passes for each evaluator
  - First to get feel for flow and scope of system
  - Second to focus on specific elements
- Each evaluator produces list of problems
  - Explain problem with reference to heuristic or other info
  - Be specific and list each problem separately
  - Assign rating of severity to each violation

Severity ratings

- Each violation is assigned a severity rating
  - Many other methods of doing this
- Usually some combination of:
  - Frequency
  - Impact
  - Persistence (one time or repeating)
- Used to:
  - Help prioritize problems
  - Allocate resources to fix problems
  - Estimate need for more usability efforts
- Can be done independently by all evaluators or later as group prioritizes

Example severity & extent scales

- One severity scale:
  - 0 - don't agree that this is a usability problem
  - 1 - cosmetic problem
  - 2 - minor usability problem
  - 3 - major usability problem; important to fix
  - 4 - usability catastrophe; imperative to fix
- One extent scale:
  - 1 = single case
  - 2 = several places
  - 3 = widespread

Why might we want to know about frequency?

Evaluation form

Example Heuristic Evaluation Form

Evaluator: _____ Prototype: _____ Date/Time: _____ Pg: ___ / ___

Heuristic violated	Description / Comment	Severity

Step 3: Aggregating results & making recommendations

- Evaluation team meets and compares results
- Through discussion and consensus, each violation is documented and categorized in terms of severity, extent
- Violations are ordered in terms of severity
  - e.g., use an excel spreadsheet (which can be sorted)
- Combined report goes back to design team.

Heuristic evaluation

Advantages

- Contributes valuable insights from objective observers
- "Minimalist" approach
  - general guidelines can correct for majority of usability problems
  - easily remembered, easily applied with modest effort
  - systematic technique that is reproducible with care.
- Discount usability
  - cheap and fast way to inspect a system
  - can be done by usability experts and rapidly-trained end users

Problems

- Principles must be applied intuitively and carefully
  - can't be treated as a simple checklist
- Heuristics can narrow focus on some problems at cost of others
- Can reinforce existing design (not for coming up with radical ideas)
- Doesn't necessarily predict users/customers' overall satisfaction
- May not have same "credibility" as user test data

Designer bias? Yes

Expertise?

Activity: HE Briefing session

- You are the experts!
- What system are we using? G&PS website (https://www.grad.ubc.ca/)
- System goals:
  - Be an authoritative information source for prospective and current graduate students and their supervisors about awards, deadlines, program milestones and completion, and so on.
- Decide on a set of heuristics
  - Today, use list supplied on sheet; review slides for description
- Work on sample tasks independently – briefly describe any issues encountered

Sample tasks

- Figure out what steps you need to take to submit your thesis (and graduate!)
- What awards do you qualify for?
  What are the deadlines?
- How do you apply for a leave of absence from your program?
- What are tuition fees for international students?

Won't get through all of these...10 minutes to search: remember this is about finding and documenting *problems* not answers

Come together

- What issues have been noted? What is their severity?
  - Frequency
  - Impact
  - Persistence

Severity scale:

0 - don't agree that this is a usability problem

1 - cosmetic problem

2 - minor usability problem

3 - major usability problem; important to fix

4 - usability catastrophe; imperative to fix

Extent scale:

1 = single case

2 = several places

3 = widespread

Was there value in the HE? Why or why not

References

Heuristics for accessibility: https://www.deque.com/blog/supporting-the-design-phase-with-accessibility-heuristics-evaluations/

- Nielsen, J. (1994, updated 2020). 10 usability heuristics for user interface design. Neilsen Norman Group. Retrieved, https://www.nngroup.com/articles/ten-usability-heuristics/
- Travis, D. (2014 October 22). *247 web usability guidelines.* Updated 2016 April 12. Retrieved, https://www.userfocus.co.uk/resources/guidelines.html
- Wilson, C. (2013). Cognitive walkthrough. *User interface inspection methods: a user-centered design method*. (pp. (pp. pp. 65-79). Newnes.
- Wilson, C. (2013). Heuristic evaluation. *User interface inspection methods: a user-centered design method*. (pp. (pp. 1-31). Newnes.