CS 338: Graphical User Interfaces

Lecture 4-1: Interface Evaluation

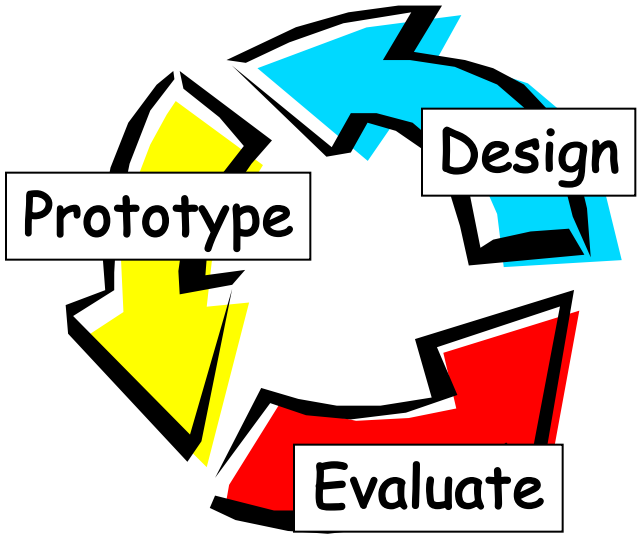
Some materials adapted from
James Landay's slides @ UC Berkeley

Reminders

- Midterm is next week. **No class!**
- It will we open from Wednesday evening to Friday evening. Once you start it you have an hour to finish it.
- Questions are concept and theories that we learned in the class. No coding/programing question!
- After the midterm I will reveal the document for the projects, and you can start thinking about it

Interface evaluation

- Why evaluate interfaces?
  - commercial? – bad GUIs can drive away customers
  - time-critical? – bad GUIs can distract users
  - safety-critical? – bad GUIs can kill… no kidding
- Evaluate early & often



Evaluation methods

- Empirical testing
- Usability walkthroughs
- Heuristic evaluation
- User modeling

Empirical testing

- The "classic" method
- Give users the interface & set of tasks
- Collect various measures from users
  - (more) subjective data
    - questionnaires
    - interviews
    - videotape
  - (more) objective data
    - time on task(s)
    - learning rate
    - low-level data: mouse movements, clicks, speech, gestures, eye movements, …

Empirical testing

- Advantages
  - user is very involved
    - can directly comment on interface
    - suggestions can be integrated… easily?
  - can yield objective measures
    - e.g., time on task, time to learn
    - but what are the right measures?

Empirical testing

- Disadvantages
  - tester expertise?
    - developing questionnaires, interviews
    - analyzing subjective videotape
  - tester's knowledge of system?
    - testers & developers often not the same people
  - expense of new study?
    - usability labs need lots of equipment --> money
  - time required?
    - with the difficulty of analysis & integration of study, testing takes more than a short time

Usability walkthrough

- Gather group of users, designers, & human factors experts
- Interface is presented to group
  - prototype, storyboard, pen & paper, …
- Freeform exploration of interface
  - ideas about usability, questionnaire, etc.
- More structured exploration:
  Cognitive walkthrough
  - "walk through" various scenarios with interface, playing the part of user performing a task

Usability walkthrough

- Advantages
  - again, users are involved
    - and... Users interact directly with experts
  - don't need a full prototype
- Disadvantages
  - time-consuming (esp. cognitive walkthrough)
  - coordination of all personnel + users

Heuristic evaluation

- Developed by Jakob Nielsen
- Small set (3-5) of evaluators examine UI
  - independently check for compliance with usability principles ("heuristics")
  - different evaluators will find different problems
  - evaluators only communicate afterwards, findings are then aggregated
- Can perform on working UI or on sketches

Heuristic evaluation process

- Evaluators go through UI several times
  - inspect various dialogue elements
  - compare with list of usability principles
  - consider other principles that come to mind
- Usability principles
  - Nielsen's "heuristics"
  - supplementary list of category-specific heuristics
    - competitive analysis & user testing of existing products
- Use violations to redesign / fix problems

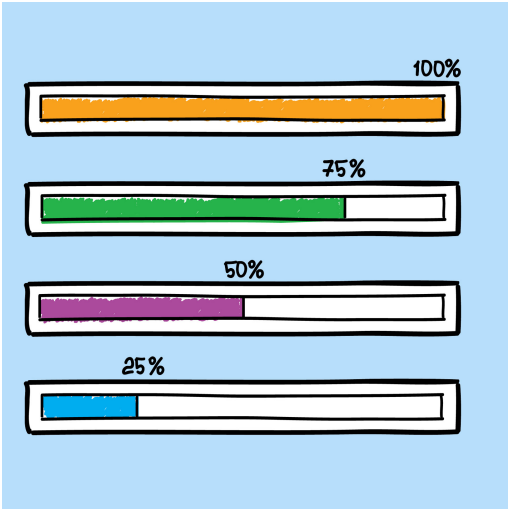
Heuristics (original)

- H1-1: Simple & natural dialog
- H1-2: Speak the users' language
- H1-3: Minimize users' memory load
- H1-4: Consistency
- H1-5: Feedback
- H1-6: Clearly marked exits
- H1-7: Shortcuts
- H1-8: Precise & constructive error messages
- H1-9: Prevent errors
- H1-10: Help and documentation

We'll look at the revised set...

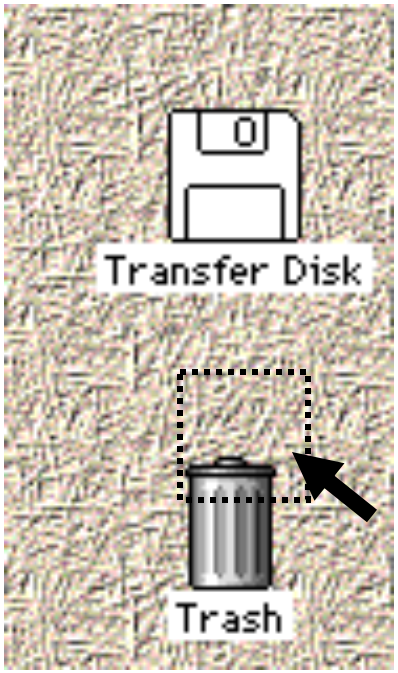
Heuristics (revised)

- H2-1: Visibility of system status
  - keep users informed about what is going on
  - example: pay attention to response time
    - 0.1 sec: no special indicators needed
    - 1.0 sec: user tends to lose track of data
    - 10 sec: max. duration for user to stay focused on action
    - for longer delays, use %-done progress bars



Heuristics (revised)

- H2-2: Match between system & real world
  - speak the users' language
  - follow real world conventions
  - example: MacOS desktop
    - dragging disk to trash

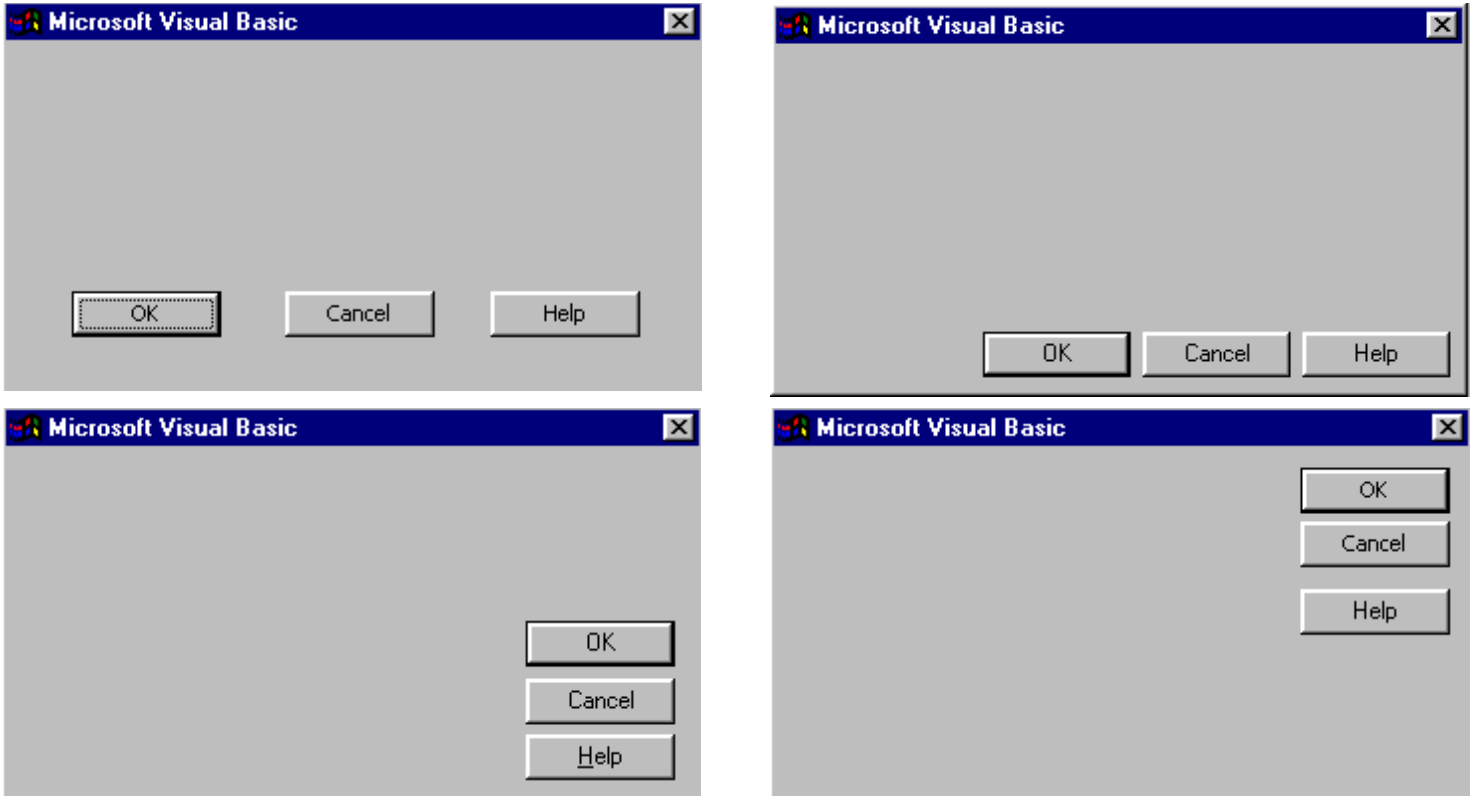


Heuristics (revised)

- H2-3: User control & freedom
  - "exits" for mistaken choices, undo, redo
  - don't force down fixed paths
  - example: "Wizards"
    - must respond to Q before going to next
    - for infrequent tasks
      - (e.g., modem config.)
    - not for common tasks
    - good for beginners
      - or, have 2 versions: one beginner, one advanced

Heuristics (revised)

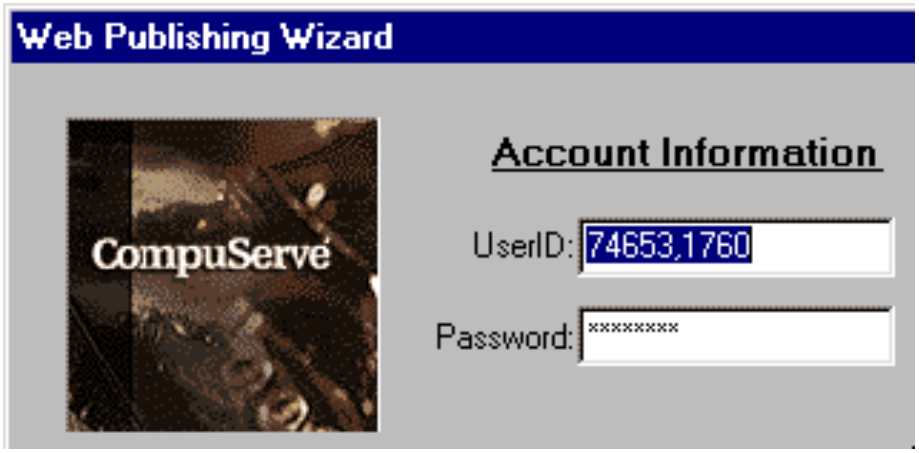
- H2-4: Consistency & standards



  - especially important for impaired users… why?

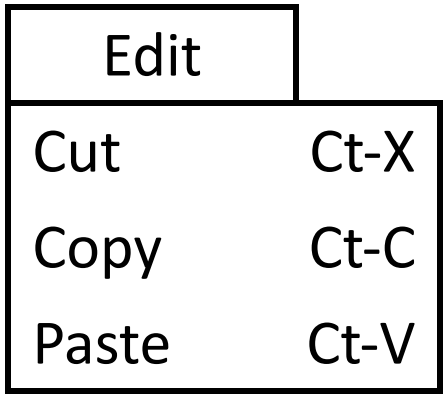
Heuristics (revised)

- H2-5: Error prevention
- H2-6: Recognition rather than recall
  - make objects, actions, options, & directions visible or easily retrievable
  - example: pre-filling input fields (autofill)
    - before dialing, asks for id & password
    - when connecting, asks again for id & password



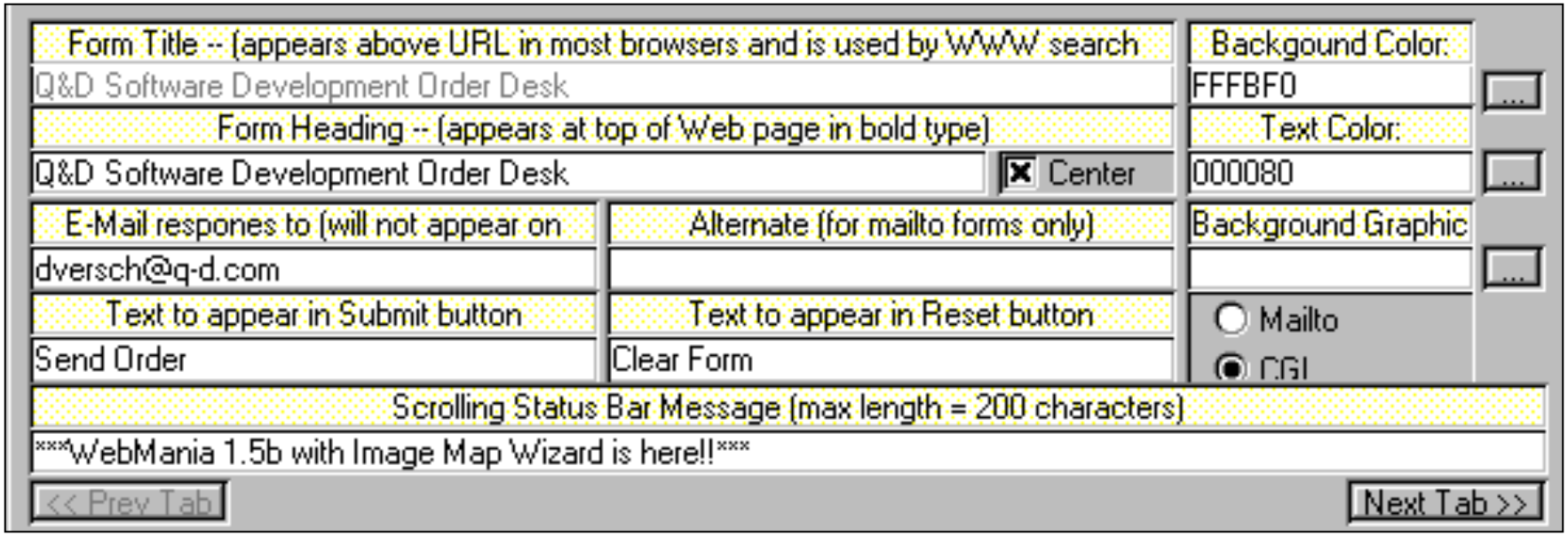
Heuristics (revised)

- H2-7: Flexibility and efficiency of use
  - accelerators for experts
    (e.g., gestures, keyboard shortcuts)
  - allow users to tailor frequent actions
    (e.g., macros)



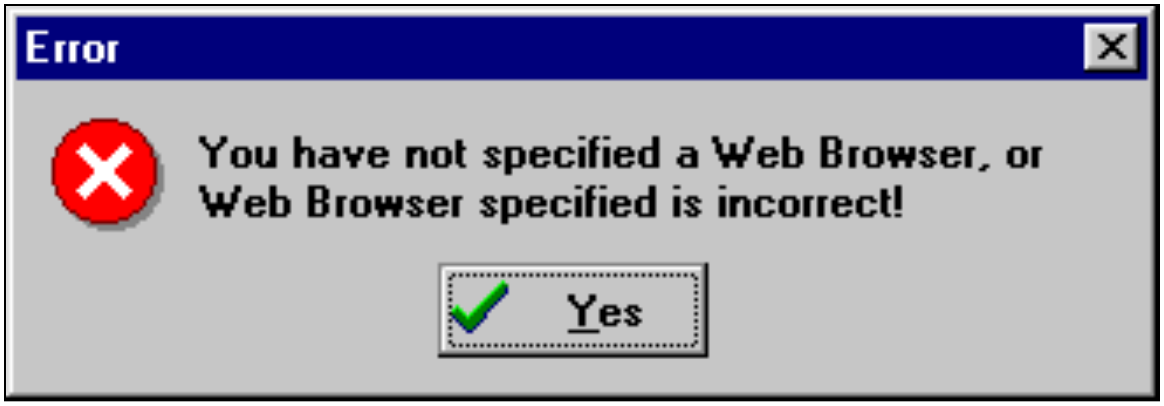
Heuristics (revised)

- H2-8: Aesthetic and minimalist design
  - no irrelevant information in dialogues



Heuristics (revised)

- H2-9: Help users recognize, diagnose, and recover from errors
  - error messages in plain language
  - precisely indicate the problem
  - constructively suggest a solution



Heuristics (revised)

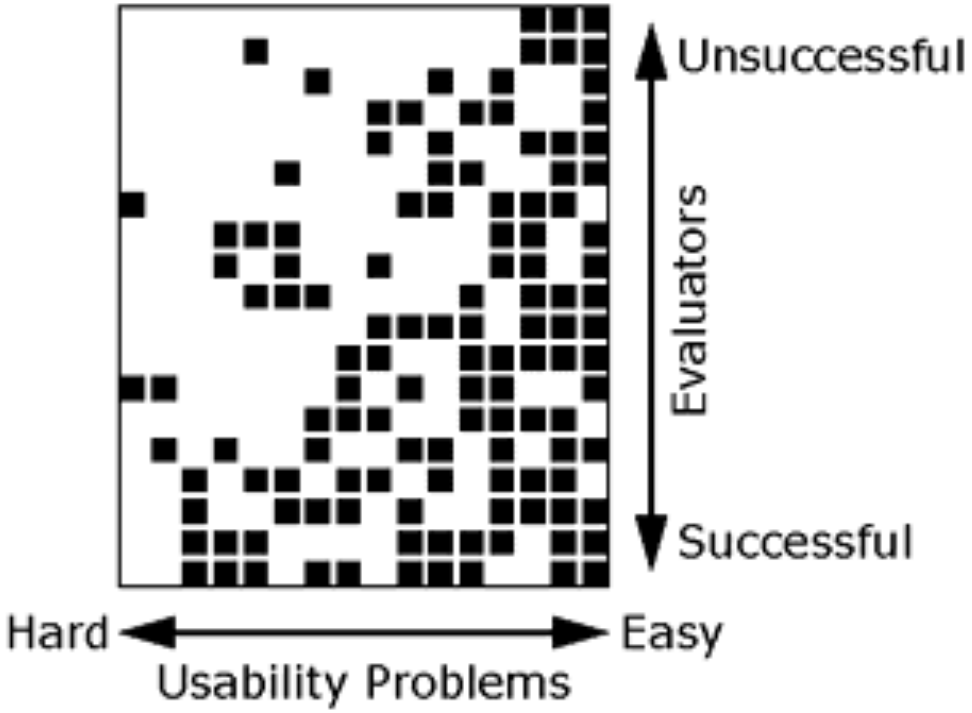
- H2-10: Help and documentation
  - easy to search
  - focused on the user's task
  - list concrete steps to carry out
  - not too large

Phases of heuristic evaluation

- 1) Pre-evaluation training
  - give evaluators needed domain knowledge and information on the scenario
- 2) Evaluation
  - individuals evaluate and then aggregate results
- 3) Severity rating
  - determine how severe each problem is (priority)
- 4) Debriefing
  - discuss the outcome with design team

Why multiple evaluators?

- Every evaluator doesn't find every problem
- Good evaluators find both easy & hard ones (but different sets of easy & hard ones!!)



Heuristic evaluation

- Advantages
  - much faster than empirical testing
    - 1-2 hours each evaluator vs. days-weeks
  - doesn't require interpreting user actions
- Disadvantages
  - user testing is far more accurate (by def.)
    - takes into account actual users and tasks
  - may miss problems & find "false positives"
- Good to alternate between HE & user testing
  - find different problems
  - don't waste participants

Heuristic evaluation

- Studies of cost benefits [Nielsen'93]
  - benefit-cost ratio = 48 (!)
    - cost was $10,500 for benefit of $500,000
    - value of each problem ~15K (Nielsen & Landauer)
  - how might we calculate this value?
- Studies of evaluators
  - 1 evaluator achieves poor results, only finds ~35% of usability problems
  - 5 evaluators find ~75% of usability problems
  - why not more evaluators? 10? 20?
    - costs more, won't find many more problems

User Modeling

- The most complex, formal method of interface evaluation (that we'll look at, anyway)
- Goal: Create a formal representation of user behavior to characterize, analyze it
- Tool: *Cognitive architectures*
- Most commonly known: GOMS

Engineering Models of Humans

- How can we model human behavior?
- For other engineering models, known capacities and limitations are built in
  - e.g., stress on bridge structures, congestion on roadways
- For humans: Cognitive architectures
  - psychological theory of thought / behavior
    - e.g., how quickly people forget information
  - computer representation of thought / behavior
    - can compute or simulate features of user behavior
  - incorporate built-in representations for human capabilities and limitations

Modeling & Levels of Abstraction

- Of course, human behavior is pretty broad.
  Can we limit this to something reasonable?
- All models (regardless of domain) operate at some level of abstraction
  - example: simulating a rocket taking off into space
    - F=ma, conservation laws, ...
    - air resistance? atmospheric heat? quantum mechanics?
  - example: simulating traffic on a new highway
    - number of people who travel, alternate routes, ...
    - individual driver behavior? accidents?
- Likewise, we have different levels of modeling for human behavior
  - our focus today: GOMS
  - but there are some more abstract, some more detailed

GOMS

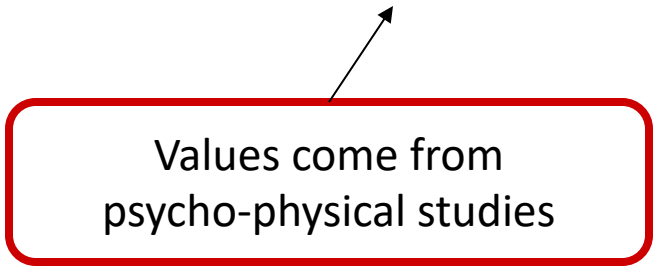
- What does GOMS stand for?
  - Goals of the user
  - Operators available to accomplish the goals
  - Methods (common sequences of operators) to accomplish the goals
  - Selection rules to choose between methods if multiple methods apply
- Several versions of GOMS
- We'll look at the "Keystroke-level model" (a.k.a., KLM-GOMS)

GOMS analysis

- Input
  - user interface to study
  - sample user task(s)
- Result
  - GOMS model of user behavior in a task
- Benefit
  - assess user performance in interface, both qualitatively & quantitatively

GOMS operators

- K: keystroke 280 ms
- P: point with mouse to a target 1100 ms
- B: press or release mouse button 100 ms
- BB: click mouse button 200 ms
- H: "home" hands to kb/mouse 400 ms
- M: "mental" act of thinking 1200 ms



GOMS analysis steps

- 1. Lay out all assumptions.
- 2. Write out a basic action sequence.
- 3. Select the operators & durations to use.
- 4. List the times for the physical operators.
- 5. Add "mental" operators & times.
- 6. Sum the times of the operators.

Operator example

- Telephone operator types in a calling card #
  - number = 14 digits
- Current interface
  - operator types entire number "blind" — no visual feedback while typing
  - error correction requires re-typing all digits
- "This isn't great. How do we improve it?"
- (This was/is a real problem!)

Operator example

- 1. Lay out all assumptions.
  - user (operator) types in 14 digits,
    but makes a mistake in the 5th digit
  - we will model the correction of this error
  - method: reset CCN field and re-type digits
- 2. Write out the basic action sequence.
  - press the CCN reset key
  - type the 14 CCN digits
  - press the Enter key

Operator example

- 3. Select the operators & durations to use.

Description	#	Type
reset CCN	1	K
type digits	14	K
press Enter	1	K

Operator example

- 4. List the times for the physical operators.

Description	#	Type	Time
reset CCN	1	K	.28
type digits	14	K	3.92
press Enter	1	K	.28
		Total Time	4.48

Operator example

- 5. Add "mental" operators & times.
- 6. Sum the times of the operators.

Description	#	Type	Time
"reset station"	1	M	1.20
reset CCN	1	K	.28
"prepare digits"	1	M	1.20
type digits	14	K	3.92
"verify done"	1	M	1.20
press Enter	1	K	.28
		Total Time	8.08

The GOMS family

- KLM-GOMS: what we've been talking about
- CMN-GOMS: the original Card-Moran-Newell formulation of goals, operators, etc.
  - bigger and deeper, but more conceptual
- NGOMSL (Natural GOMS Language): expresses GOMS as an ordinary programming language
- CPM-GOMS (Cognitive-Perceptual-Motor): expresses parallel behavior using PERT charts and analyzing critical paths

User modeling

- Advantages
  - user centered — based directly on user behavior
  - can compare interfaces & predict times
    - this is most useful for time-critical GUI use
- Disadvantages
  - requires some skill, training in modeling (though not terribly much)
  - some aspects may be intuitive without model (though some may not)
  - may not be practical for complex interfaces
  - what about learning?