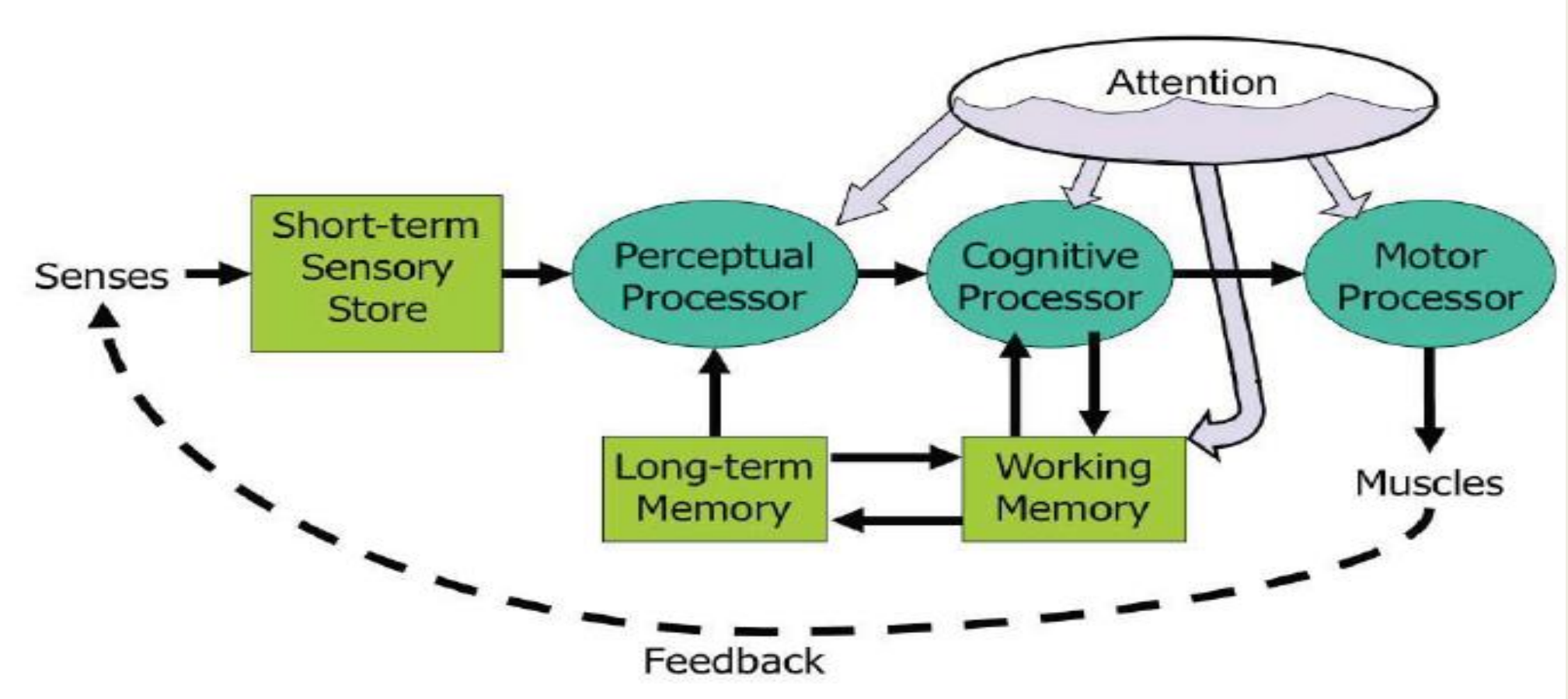
# HUMAN COMPUTER INTERACTION

Lecture 4: Efficiency

# Today's Topics

- Human information processing
- Pointing efficiency
  - *Fitts's Law and Steering Law*
- Design principles
  - *Shortcuts*
  - *Defaults, history and anticipation*
- Predicting Efficiency
  - *Keystroke-level model*

# Human Information Processing



# Processors

- Processors have a cycle time
  - $T_p$ ~ *100ms [50-200 ms]*
  - $T_c$ ~ *70ms [30-100 ms]*
  - $T_m$ ~ *70ms [25-170 ms]*


- Processor speed varies by person and conditions
  - *Fastest may be the 10x slowest*

# Perceptual Fusion

- Two stimuli within the same PP cycle (Tp ~ 100ms) appear fused
  - *Causality is strongly influenced by fusion*

# Cognitive Processing

- Cognitive Processor
  - *Compares stimuli*
  - *Selects a response*
- Types of decision making
  - *Skill-based*
  - *Rule-based*
  - *Knowledge-based*

# Motor Processing

- Open-Loop Control
  - *Motor processor runs a program by itself*
  - *Cycle time is Tm ~70 ms*
- Closed-loop control
  - *Muscle movements( or their effect on the world) are perceived and compared with desired result*
  - *Cycle time is Tp + Tc + Tm ~240 ms*

# Choice Reaction Time

- Reaction time depends on the information content of stimulus

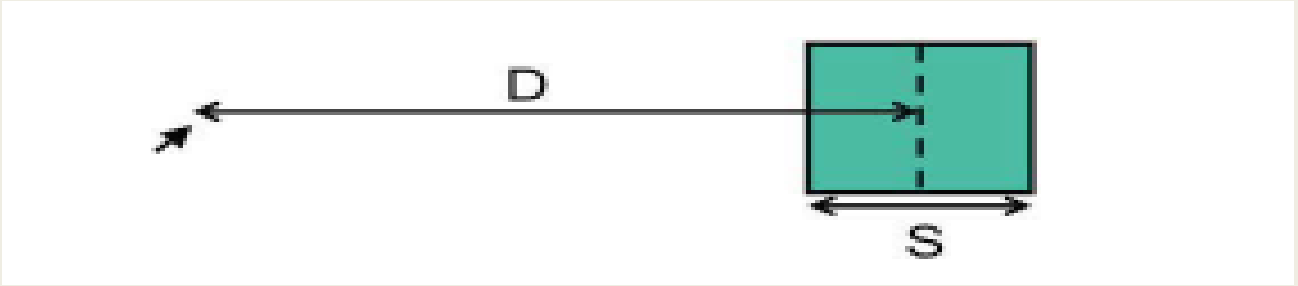
$$RT = c + d \log_2 1/Pr(stimulus)$$

  - *E.g. for N equiprobable stimuli, each requiring a different response:*

$$RT = c + d \log_2 N$$

# Fitts’s law

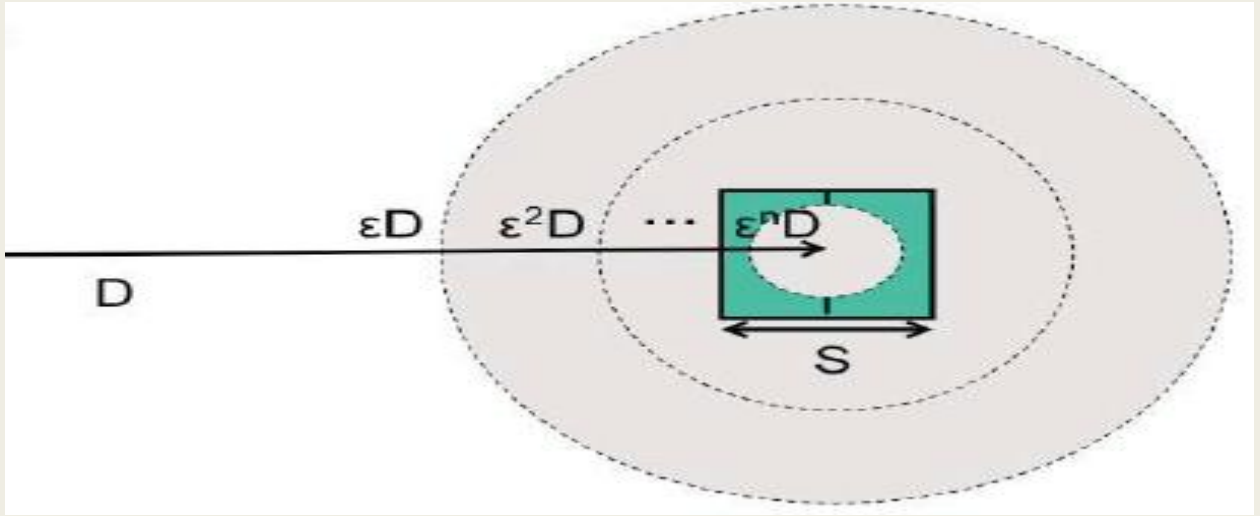
- Fitts’s Law
  - *Time T to move your hand to a target of size S at distance D away is*
    - *T = RT + MT = a+b log(D/S +1)*




    - *Depends only on index of difficulty log(D/S +1)*

# Explanation of fitts's Law

- Moving your hand to a target is closed-loop control
- Each cycle covers remaining distance D with error εD



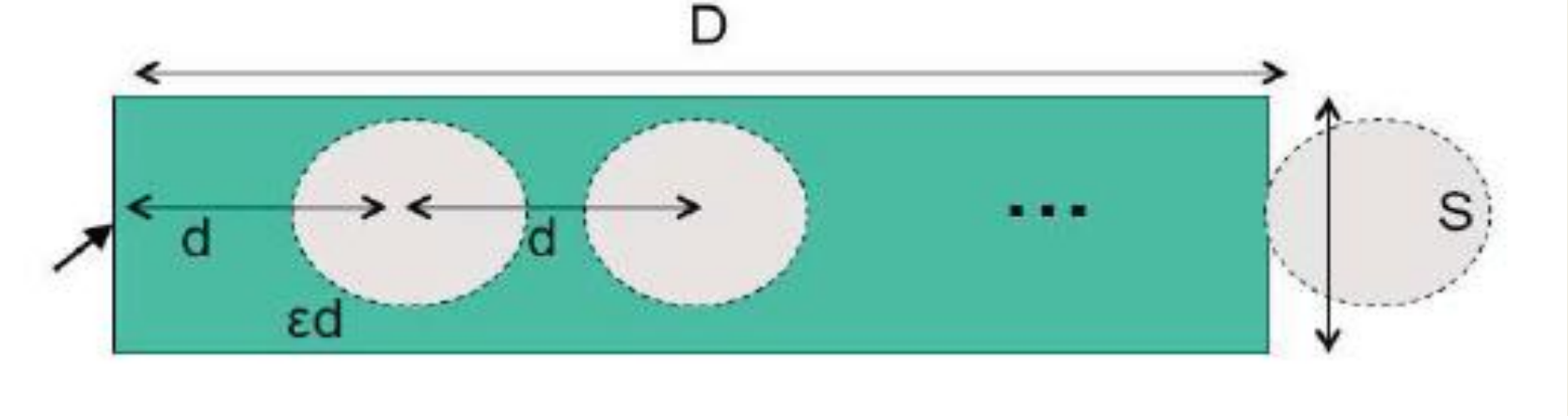
# Implication of Fitts's Law

- Target at screen edge are easy to hit
  - *Mac menubar beats Windows Menubar*
  - *Unclickable Margins are foolosh*
- Linear Popup menus vs. pie menus

# Steering Tasks

- Time T to move your hand through a tunnel of length D and width S is:

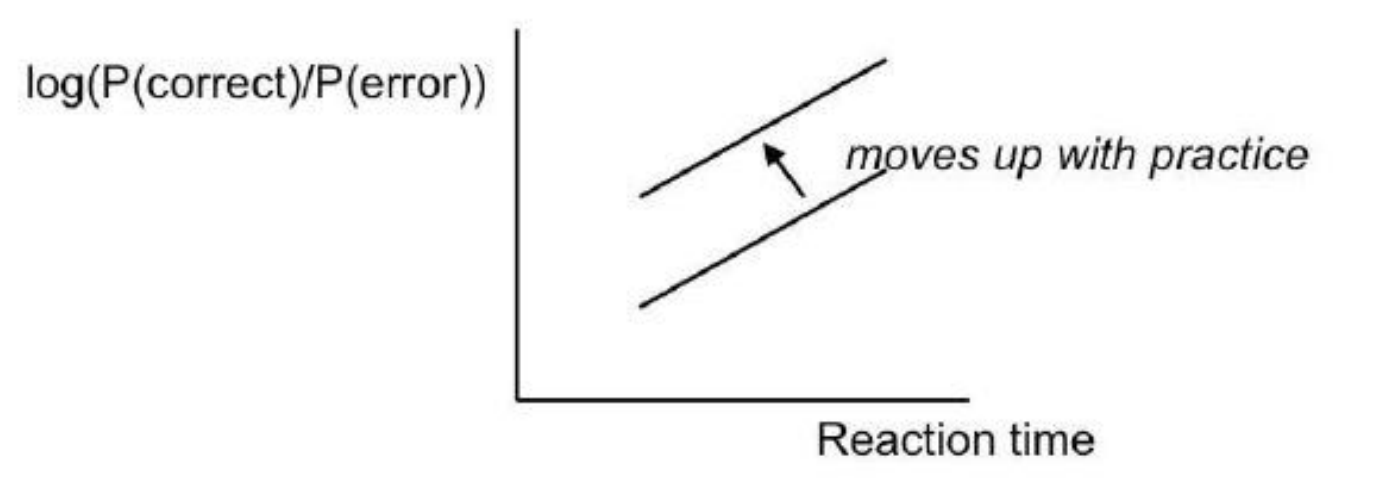
$$T = a + b\, D/S$$




- Index of difficulty is now linear, not logarithmic
  - *So steering is much harder than pointing*
- Thus cascading submenus are hard to use

# Speed-Accuracy Tradeoff

- Accuracy varies with reaction time
  - *Here, accuracy is probability of slip or lapse*
  - *Can choose any point on curve*
  - *Can move curve with practice*



# Power Law of Practice

- Time $T_n$ to do a Task the nth time is

$$T_n = T_1 \, n^{-\alpha}$$

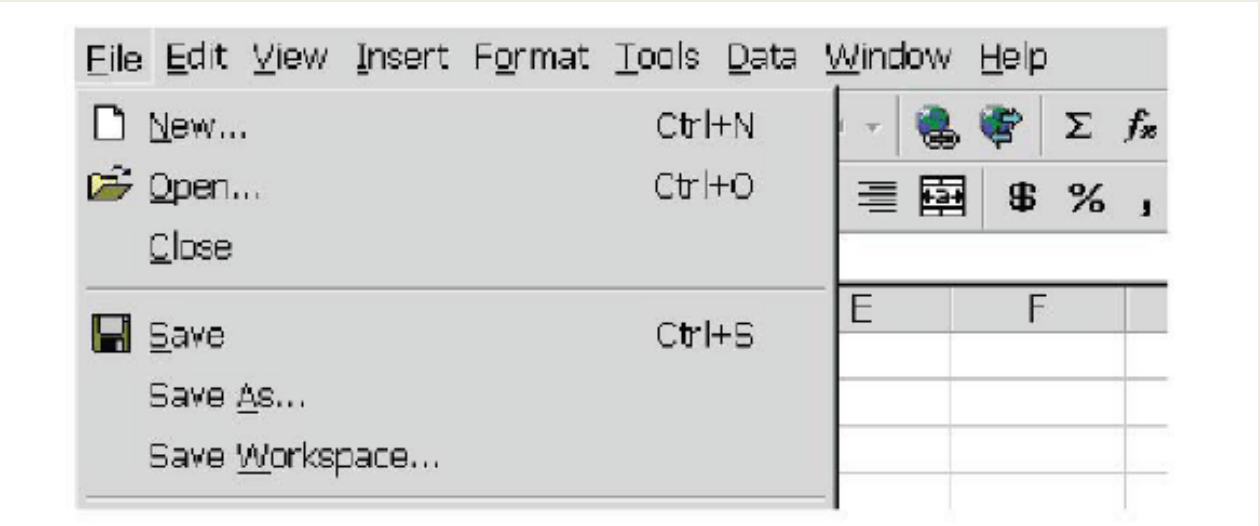
$\alpha$ is typically 0.2 – 0.6

# Improve Mouse Efficiency

- Make frequently-used targets big
  - *Use snapping in drawing editors*
- Put targets used together near each other
- Use screen corners and screen edges
- Avoid steering tasks

# Keyboard Shortcuts

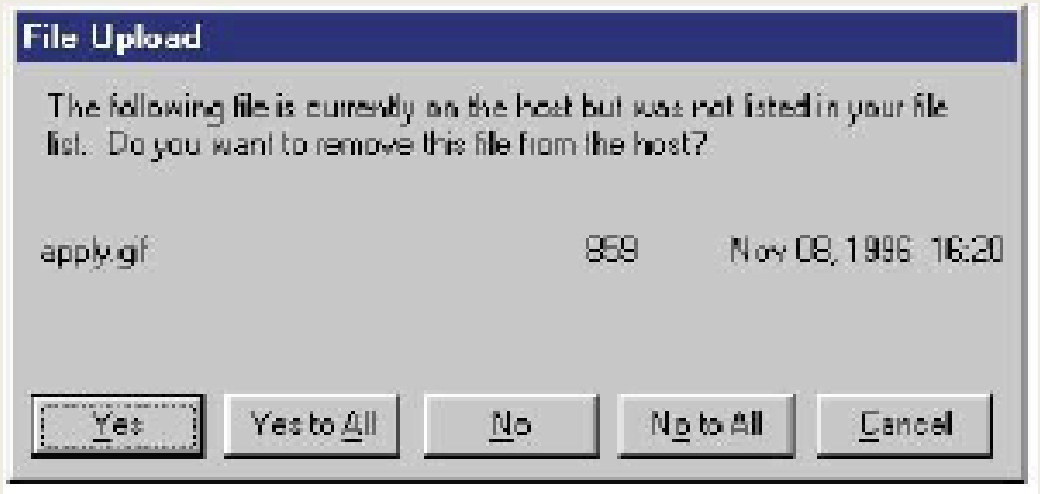
- Keyboard commands
- Menu accelerators

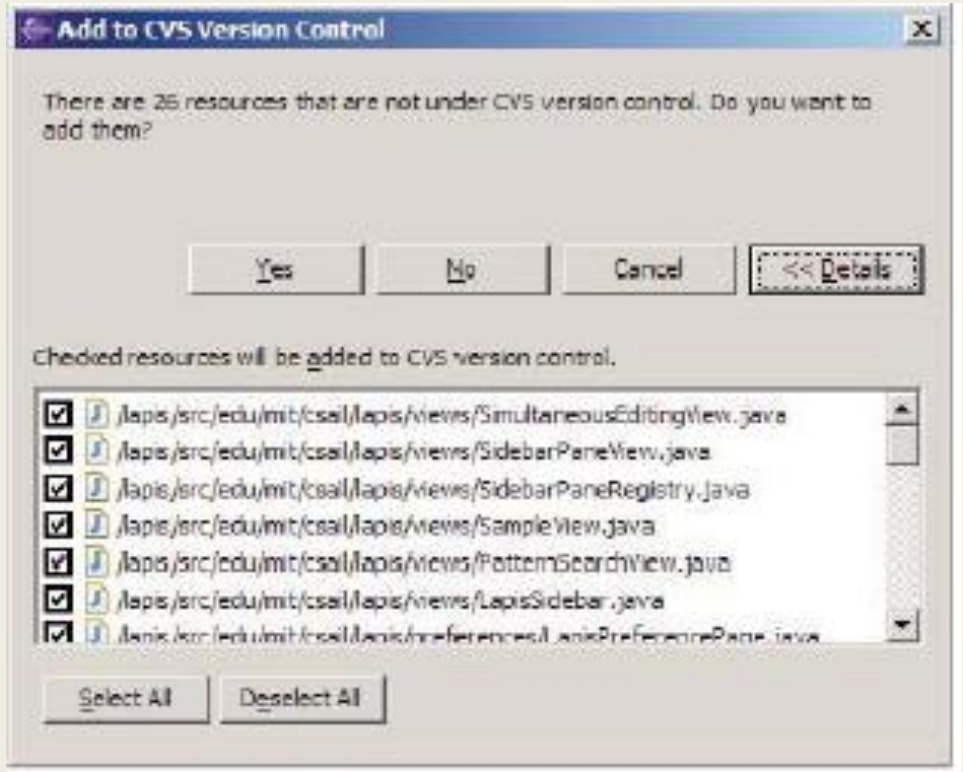


# Command Aggregates

- Styles
- Scripts
- Bookmarks

# Aggregating Questions

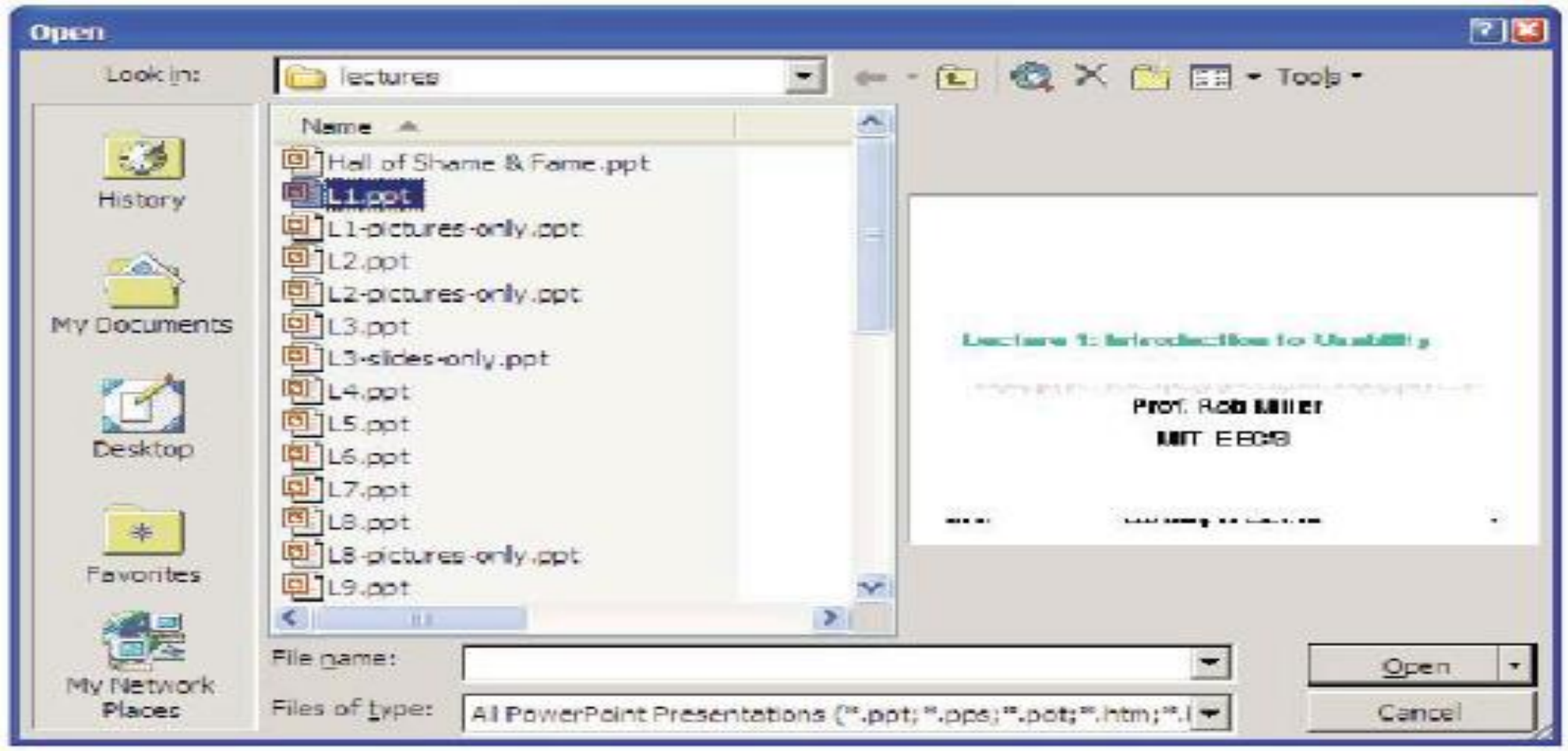


# Use Defaults and history

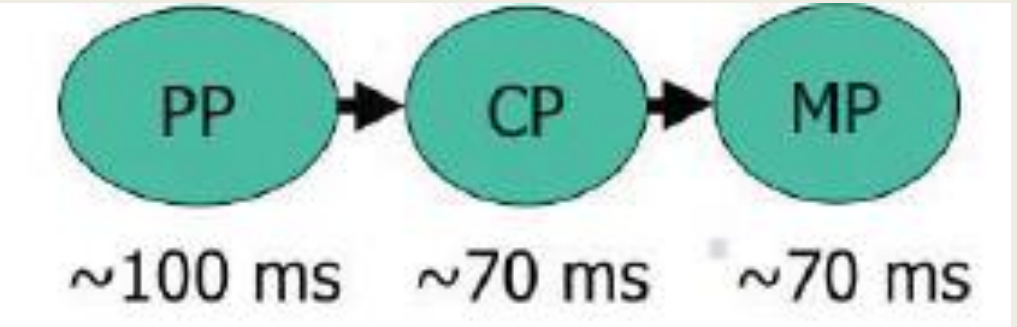
- Use default
  - *Initially, most likely entry*
  - *After use, previous entry*
- Keep histories
- Other autocompletion

# Anticipation



# Predictive Evaluation

- Predictive evaluation uses an engineering model of human cognition to predict usability
- The engineering model is
  - *Abstract*
  - *Quantitative*
  - *Approximate*
  - *Estimated from user experiments*



# Advantages of Predictive Evaluation

- Don’t have to build UI prototype
  - *Can compare design alternatives with no implementation whatsoever*
- *Don’t have to test real live Users*
- *Theory provides explanations of UI problems*
  - So it points to the areas where design can be improved
  - User testing may only reveal problems, not explain them

# Keystroke-Level model (KLM)

- Keystroke
- Button press or release with mouse
- Point with mouse
- Draw line with mouse
- Home hands between mouse and keyboard
- Mentally prepare

# KLM Analysis

- Encode a method as a sequence of physical operator (KPHD)
- Use heuristic rules to insert mental operator (M)
- Add up times for each operator to get total time for method

# Estimate operator Times

- Keystroke determined by typing speed
  - *0.28s average typist(40 wpm)*
  - *0.08s best typist (155 wpm)*
  - *1.20 s worst typist*
- Button press or release
  - *0.1s highly predicted, no need to acquire button*
- Pointing determined by fits's Law
  - *T = a + b log(d/s + 1) = a + b ID*
  - *0.8 + 0.1 ID  [Card 1978]*
  - *0.1 + 0.4 ID  [Epps 1986]*
  - *-0.1 + 0.2 ID  [MacKenzie 1990, mouse selection]*
  - *0.14 + 0.25 ID [ MacKenzie 1990, mouse dragging]*
  - *OR*
  - *T ~ 1.1 s for all pointing tasks*
- Drawing determined by steering law

# Estimated Operator Times

- Homing estimated by measurement

  *0.4s (between keyboard and mouse)*
- Mental Preparation estimated by measurement

  *1.2s*

# Heuristic Rules for adding M's

- Basic idea
  - *M before every chunk in the method that must be recalled from long-term memory or that involves a decision*
- Before each task or subtask
- Deciding which way to do a task
- Retrieving a chunk from memory
  - *Command name*
  - *File name*
  - *Parameter value*
- Finding something on screen
  - *So P is often preceded by M*
  - *Unless the location is well known form practice, in which case the visual search is overlapped with the motor action*
- Verifying entry or action result
  - *e.g. before pressing OK on the dialog*

# Example Deleting a Word

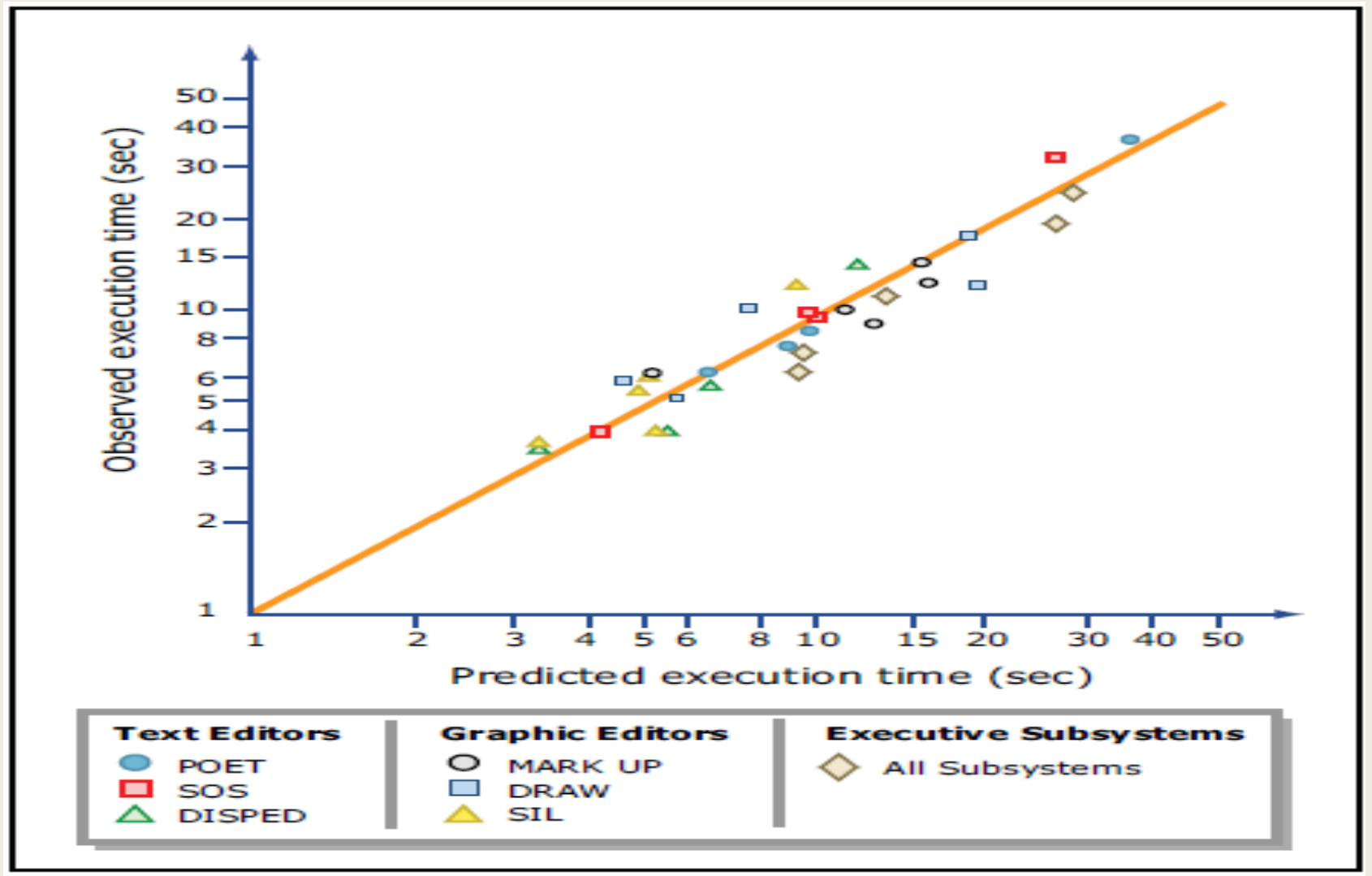
- Shift-click selection
  - *M*
  - *P [start of word]*
  - *BB [click]*
  - *M*
  - *P [end of word]*
  - *K [shift]*
  - *BB [click]*
  - *H [to keyboard]*
  - *M*
  - *K [Del]*

  *Total: 3M + 2P + 4B + 1K = 6.93 sec*

- Del key N times
  - *M*
  - *P [start of word]*
  - *BB [click]*
  - *H*
  - *M*
  - *K [Del]*
  - *x n [length of word*

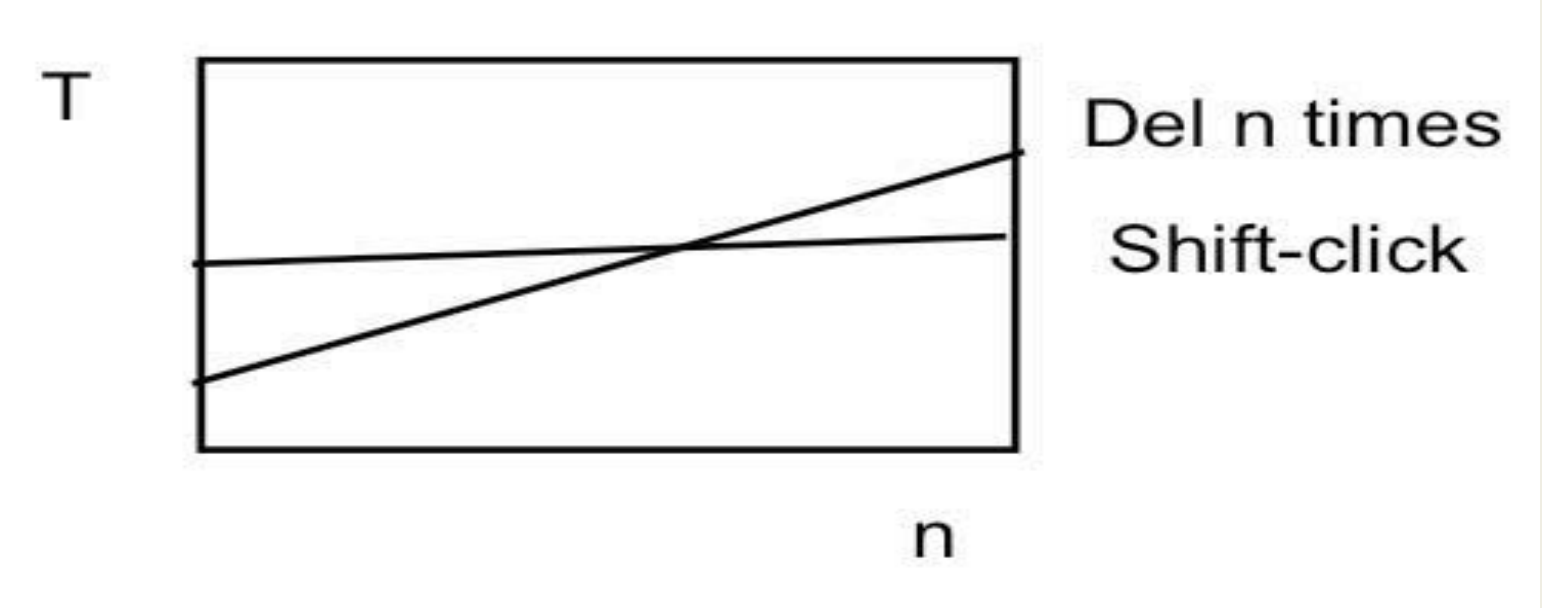
  *Total: 2M + P + 2B + H + nK = 4.36 + 0.28n sec*

# Empirical Validation of KLM



# Applications of KLM

- Comparing designs & methods
- Parametric analysis



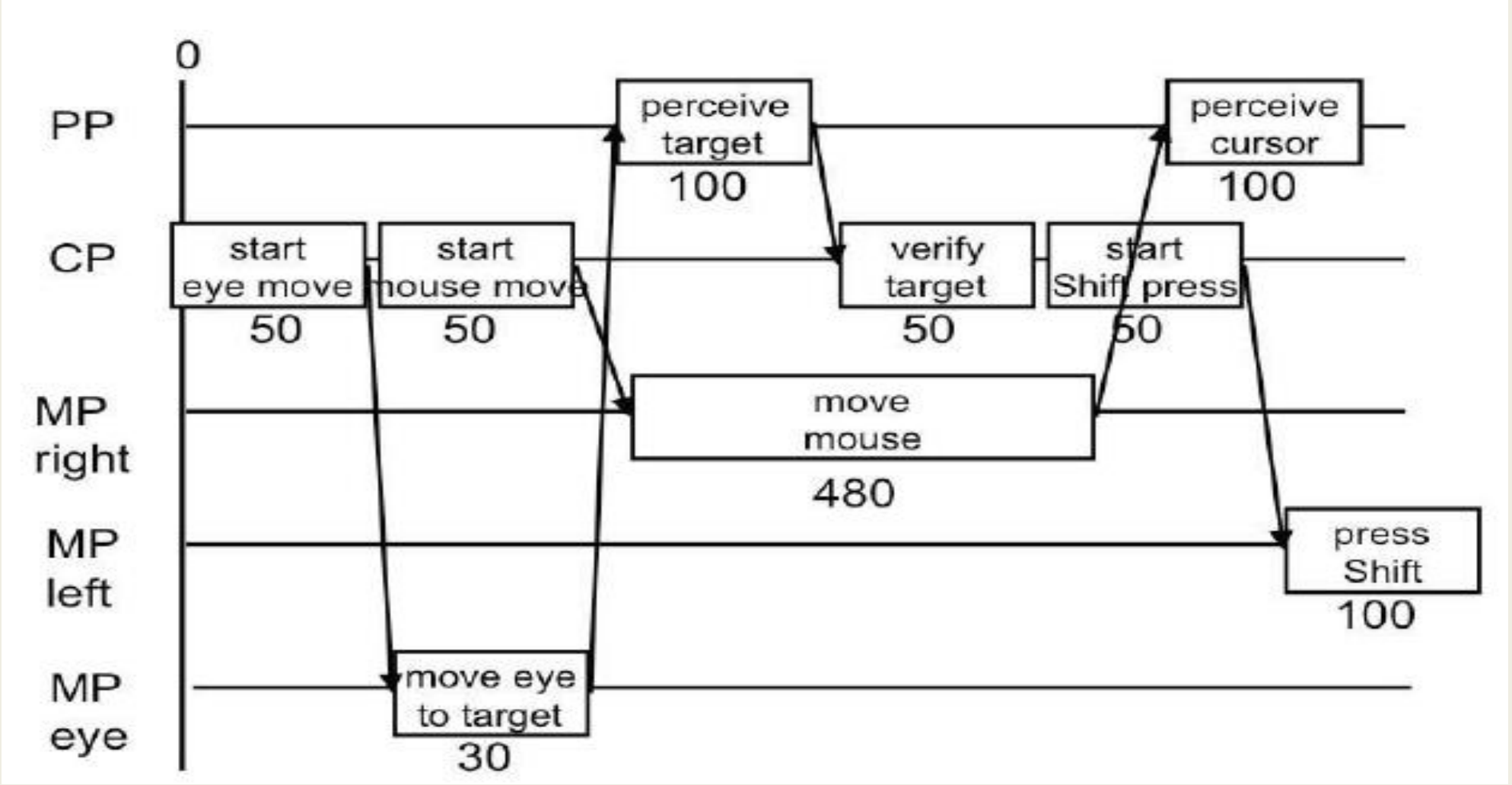
# Limitations of KLM

- Only expert users doing routine (well-learned) tasks
- Only measures efficiency
  - *Not learnability, memorability, errors, etc.*
- Ignores
  - *Errors (methods must be error-free)*
  - *Parallel action (shift-click)*
  - *Mental workload (e.g. attention & WM limits)*
  - *Planning & Problem solving (how does user select the method)*
  - *fatigue*

# CPM-GOMS

- CPM-GOMS models parallel operations
  - *E.g. point & shift-click*
- Uses parallel cognitive model
  - *Each processor is serial*
  - *Different processors run in parallel*

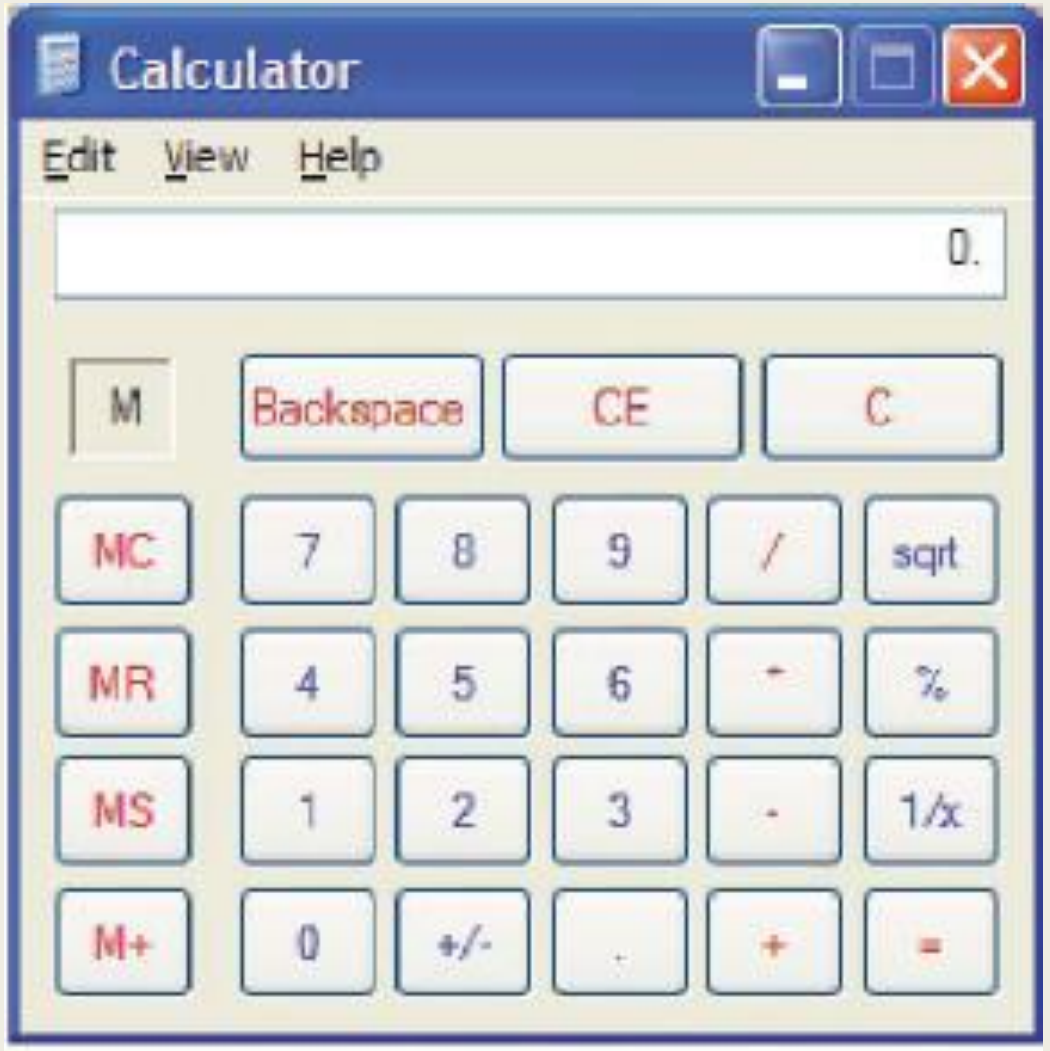
# Critical Path Determines Time



# Analysis of Phone Operator Workstation

- Phone company considering redesign of a work station ( keyboard + software) for telephone operator (411 service)
  - *Reduced keystrokes needed for common tasks*
  - *Put frequently-used  keys closer to the user's fingers*
- But new design was 4% slower than old design
  - *=1 sec/call = $3 million/year*
- Keystroke-level model has no explanation
- But CPM-GOMS explained why:
  - *Keystrokes removed were not on the critical path*
    - Used during slack time, while greeting customer
  - *A keystroke was moved from the beginning of call (during  stack time) to later ( putting it on the critical path)*

# Hall of Fame or Shame



# Today’s Topics

- Human Errors
- Design Principles
  - *Error Prevention*
  - *Error messages*
  - *User control and freedom*
- Undo

# Error Types

- Slips and Lapses
  - *Failure to correctly execute a procedure.*
  - *Slip is a failure of execution, lapse is a failure of memory*
  - *Typically found in the skilled behavior.*
- Mistakes
  - *Using wrong procedure for the goal*
  - *Typically found in rule-based behavior or problem-solving behavior*

# Slips

- Capture
  - *Leave your house and find yourself walking to school instead of where you meant to go.*
  - *Vi :w command (to save the file) vs. : wq command (to save and quit)*
  - *Excel array formulas must be entered with Ctrl-Shift-Enter, not just Enter*
- Description
  - *Putting the wrong lid on the bowl*
  - *Choosing Kendall square instead of Kenmore Square.*

# Lapses

- Loss of intention
  - *Walking to another room and forgetting why you went there*
- Omissions due to  interruption
  - *Getting coat to go out, then interrupted by the phone call then go oput without your coat.*
- Omission due to alredy-satisfied goal
  - *Walking away from an ATM without your card*
  - *Walking away form copier without your originals*

# Mode Error

- Modes: states in which actions have different meanings
  - *Vi's insert mode vs command mode*
  - *Caps Lock*
  - *Drawing palette*

# Common Features of Human Error

- Inattention or inappropriate attention
  - *Causes slips and lapses, but not mistakes*
- "Strong-but-wrong" effect
  - *Similarity*
  - *High frequency*

# Avoiding Capture and Description Slips

- Avoid habitual action sequences with identical prefixes
- Avoid Action with very similar descriptions
- Keep dangerous commands away from common ones

# Avoiding Mode Errors

- Eliminate modes
- Increase visibility of mode
- Spring-loaded or temporary modes
- Disjoint action sets in different modes

# Avoiding Lapses

- Keep procedures short
  - *Provide dialog closure*
- Minimize interruptions
- Use forcing functions
  - *In automatic transmission, you must hold down the brake in order to shift out of Park*
  - *Must take your ATM card out of the machine before you get your money*

# Other Rules for Error Prevention

- Disable illegal commands
- Use menus and forms, not command languages
- All needed information should be visible
- Use combo boxes, not textboxes
  - *But don't go overboard*
- *Protect User work*

# Confirmation Dialogs

# Writing Error Message Dialogs

- Best error message is **none at all**
  - *Errors should be prevented*
  - *Be more flexible and tolerant*
  - *Nonsense entries can often be ignored without harm*

# Be Precise and Comprehensible

- Be precise
  - *File missing or wrong format*
  - *File cant be parsed*
  - *Line too long*
  - *Name contains bad characters*
- Restart user's input
  - *Not " Cannot open file", nut "Cannot open file named lecture.pptx"*
- Speak the User language
- Not " FileNotFoundException"
- Hide technical details (like a stack trace) until requested

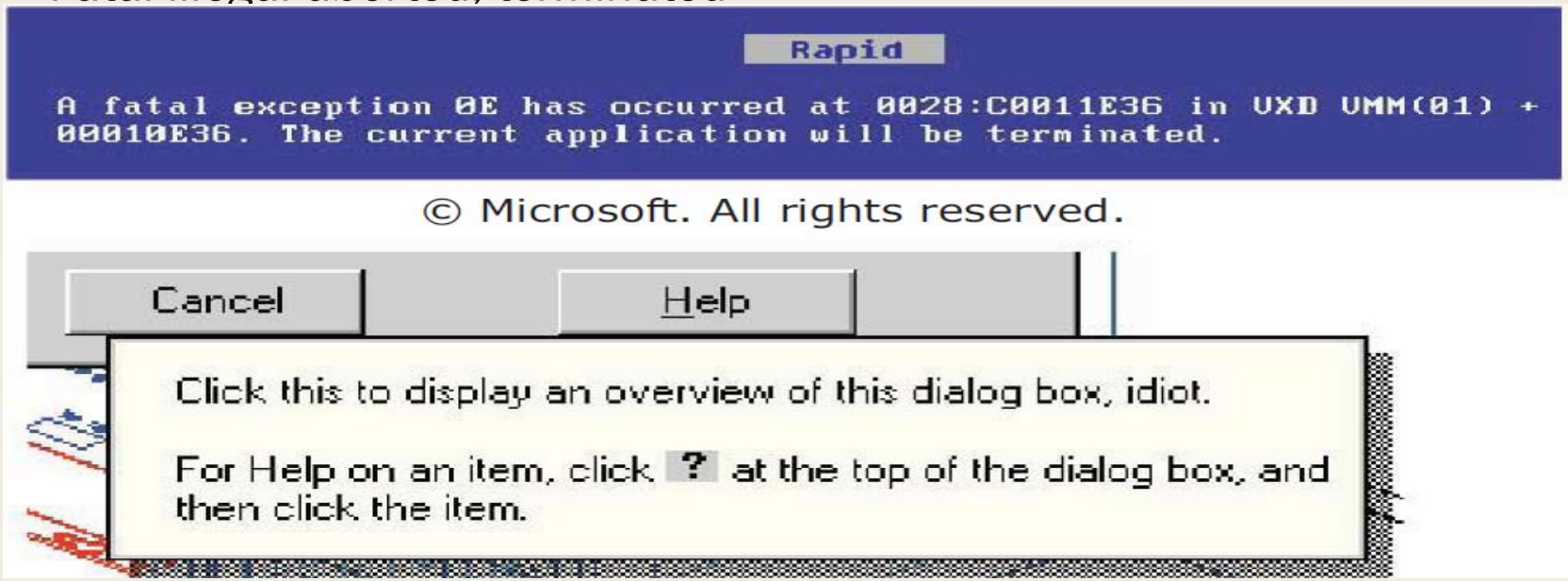
# Suggest Reasons and Solutions

- Give constructive help
  - *Why error occurred and how to fix it*

# Be Polite

- Be polite and nonblaming


- Avoid loaded words
  - *Fatal illegal aborted, terminated*

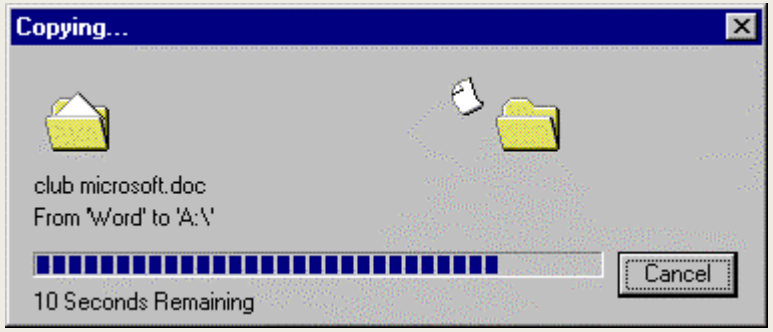


# User Control & Freedom

- Learning by Exploring
- Dealing with Errors
- User is sentient, computer is not

# Clearly Marked Exits

- Long Operations Should be cancelable

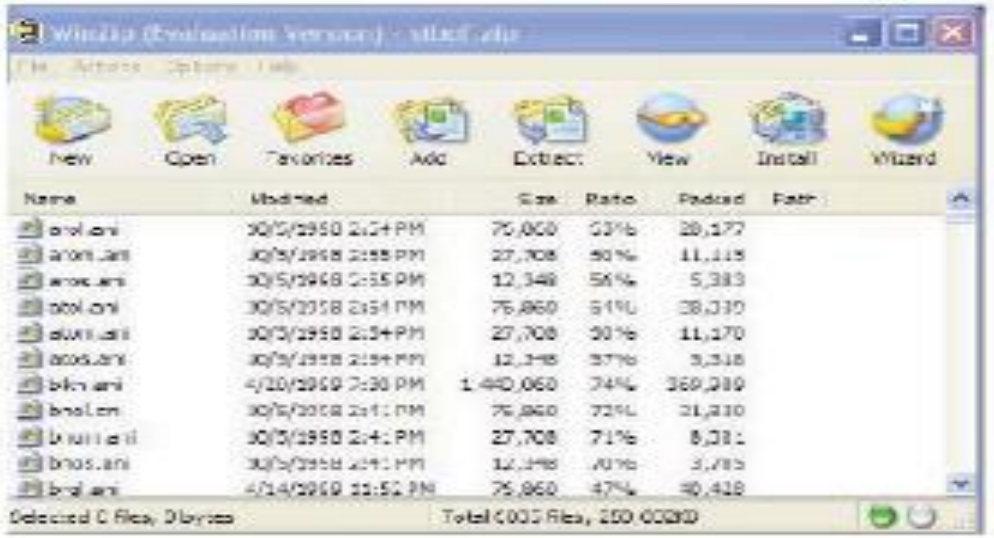



- All Dialogs should have a cancel button

# Wizard vs. Center Stage: Who's in Control?

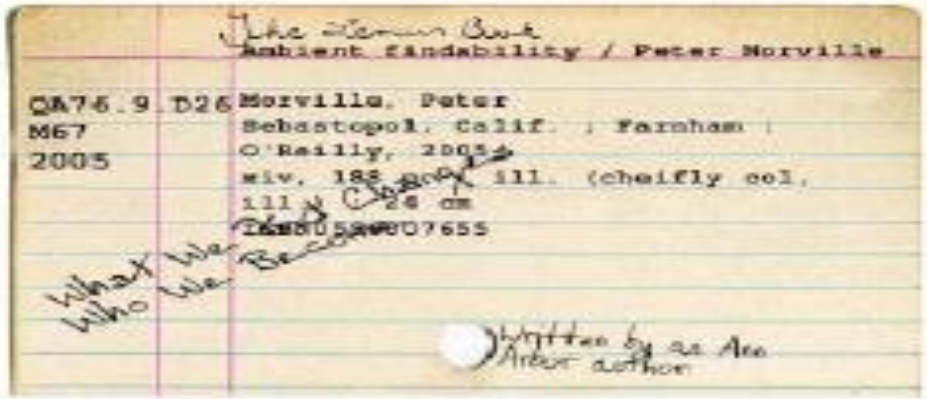

Wizard

Center Stage



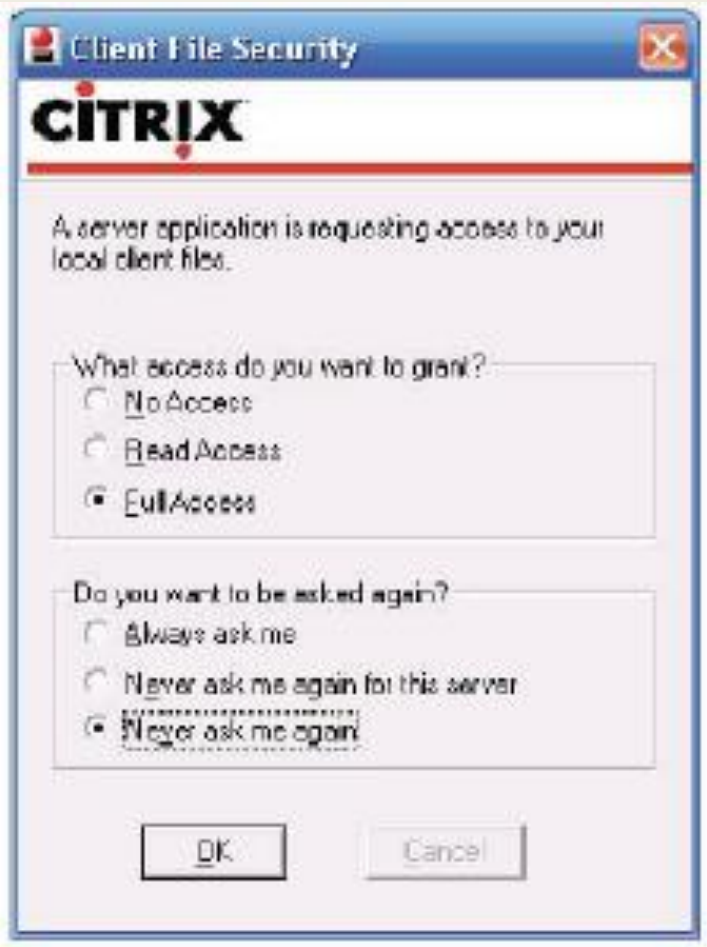
# Manual Overrides for the automatic systems






# Never Ask Me Again



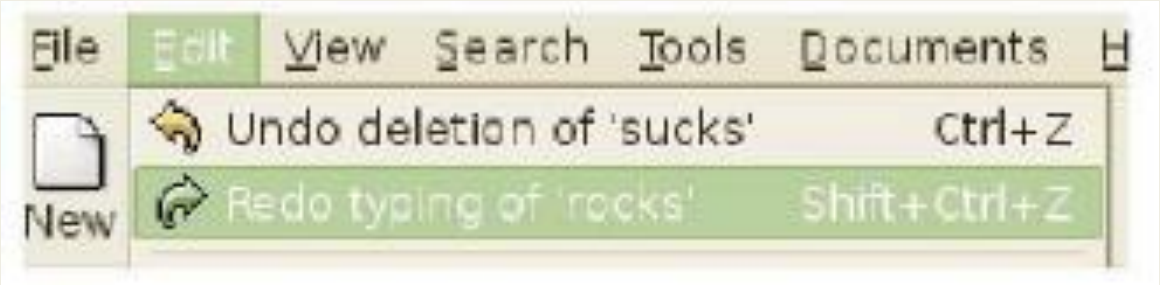

# User Control Over Data

- Data entered by the user should be editable by the user
- UI should give the power to:
  - *Create a data item*
  - *Read it*
  - *Update it*
  - *Delete it*

# No Arbitrary Limits on User-Defined Names

# Support Undo

- Desktop




- Web


- Revision History

# Forming a Mental Model of Undo

- Undo reverses the effect of an action
- But that leaves many questions:
  - *What stream of action will be undone?*
  - *How is the stream divided into undoable units?*
  - *Which actions are undoable, and which are skipped?*
  - *How much of the previous state is actually recovered by undo?*
  - *How far back in the stream can you undo?*

# What stream of actions will be undone?

- Action in this window (MS Office)
- Action in the text widget(web browser)
- Just my actions, or everybody's (multiuser apps)
- Action made by the computer
  - *Ms Office AutoCorrect and AutoFormat were the undoable, even user didn't do them*

# How is the Stream divided into units

- Lexical Level
  - *Mouse clicks, key presses, mouse moves*
- Syntactic Level
  - *Commands and button presses*
- Semantic Level
  - *Change to the application data structures (Normal Level)*
- Text entry is aggregated into a single action
  - *But other editing commands (like Backspace) and newlines interrupt the aggregation*
- What about user-defined macros?
  - *Undo macro actions individually, or as a unit?*

# Which actions are undoable?

- User action stream may include many action that are ignored by undo
  - *Selection*
  - *Keyboard focus*
  - *Changing viewpoint*
  - *Changing layout*
  - *UI customization*
- So which actions does Undo actually undo?
  - *Some applications have Undo/Redo for the editing stream back/forward for the viewpoint stream*

# How much state is recovered?

- Select text, delete it and then undo
  - *Text is restored*
  - *But is the selection restored? Cursor position?*

# How Far back can you undo?

- Often a limit on the history size
  - *Used to be on action now usually hundreds or infinite*
- Does action n stream persist across application session?
  - *If so stream must be saved to file*
- Does it persist across File/Save?

# Design Principles for Undo

- Visibility
  - *Make Sure undone actions are visible*
- Aggregation
  - *Units should be “chunks” of the action stream: typed strings dialogs, macros*
- Reversibility of the undone itself
  - *Support Redo as well as Undo*
  - *Undo to a state where user can immediately reissues the undone command*

# Design Principles for Undo (Contd…)

- Reserve it for the model changes not view changes
  - *For consistency with other applications reserve undo for changes to backend data*
- “Undo” is not only the way to support reversibility
  - *Backspace undoes typing, Back undoes browsing, Recent Files undoes the file closing, scrolling back undoes the scrolling*
  - *Forward error recovery: Using new actions to fix errors*

# Case Study: Outlook 2007