Psychology in Human-Computer Interaction

David Kieras

University of Michigan

The Role of Psychology in HCI

Psychology contributes a scientific approach to how human abilities and limitations can be taken into account in the design of effective systems.

Considerable scientific knowledge accumulated about human psychology since late 19th century.

- Carefully controlled experiments and data collection methods.
- Elaborate statistical analysis of data to compensate for individual variation.
- Development of rigorous theory of human mental mechanisms to account for phenomena.

Differences from Human Factors and Usability:

- Emphasis on developing general scientific knowledge.
- Application to practical problems is not central.
- Based on empirical results, but goal is scientific theory.

Overview of Remainder of Presentation

Purpose: Introduce psychological topics relevant to HCI:

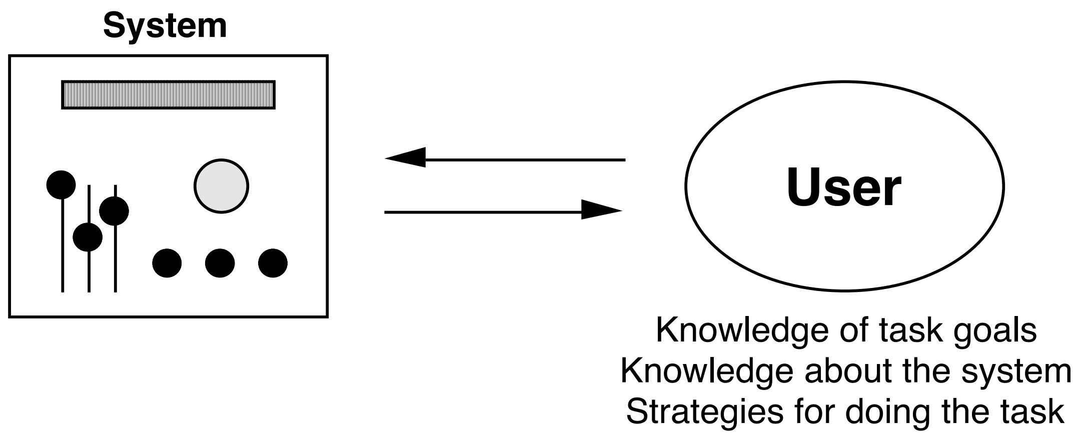
- Aspects of individual human cognition and performance.
- General approach, some useful specific results.
- Selection relevant to design, but barely scratches the surface.
- User testing, usability development already covered.

Topics:

- Background: Human Cognition and Performance
  - The Big Picture
  - The Small Picture: The Model Human Processor
- Input Basics
- Output Basics
- Procedure Basics

The Big Picture: Users and Systems

The user knows what tasks to accomplish, but also has to know how to accomplish the tasks with the system.



User must come up with a procedure by using strategies:

- Execute an already known procedure - a routine skill.
- Learn a procedure from explicit instructions.
- Infer a procedure by applying problem-solving strategies using:
  Analogy from a similar system with same function or purpose.
  Trial and error based on observation, affordances.
  Knowledge of how the system works - a "mental model."

Declarative and Procedural Knowledge

An intuitively appealing distinction:

Declarative knowledge:

- Knowledge of facts.
- List of interrelated facts, propositions.
- Stored in Long-Term Memory.
- Flexibly used, easy to report.
- Can be quickly acquired under some conditions.

Procedural knowledge:

- Knowledge of what to do.
- Represented as production rules.
  IF-THEN rules.
- Stored in production memory.
- Less flexible, can't be reported.
- Can require practice to develop.

Both are relevant to interacting with a device.

- The more knowledge required, the harder it is to learn and use the device.
- We can characterize this!

Rationality Principle

Behavior results from the person's rational attempt to meet goals given:

- Characteristics of the task.
- Operators (actions) possible in the task.
- Inputs to the person.
- Person's knowledge.
- Person's processing limitations.

Goals, task, operators do not depend on individual.

- Can often define optimum performance in these terms.
  (e.g. fastest way to complete task)

With practice people tend to optimize if possible.

- Must have the required knowledge, and work within processing limits.
- Can predict behavior based on possible optimums.

Rationality Principle and Human Factors:

- Basic goal of good system design is to alter the task, operators, and inputs so that the human can be optimal in accomplishing goals:
  In spite of processing limitations.
  With only a easily acquired amount of knowledge.

The Small Picture: The Model Human Processor (MHP)

A simple information-processing model of the overall structure and mechanisms involved in human cognition.

- Based on 100+ years of scientific theory and data recast for engineering purposes by Card, Moran, & Newell (1983).

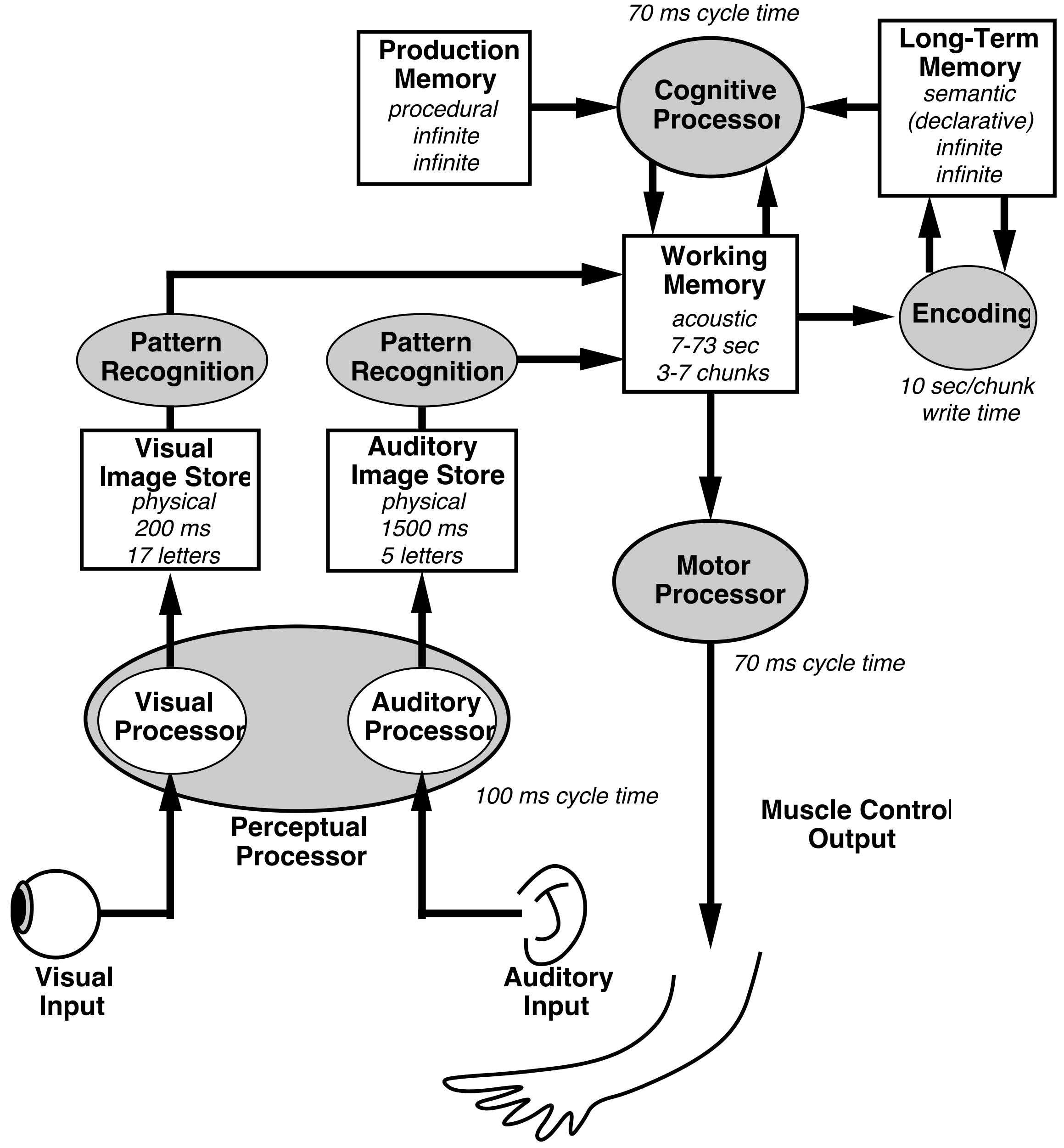
Quantitative descriptions of important phenomena.

- Probability of correct performance, performance time.

Contains interconnected processors and memories.

- Each processor has:
  Input and output types.
  A cycle time - duration of a stage of processing.
- Each memory has:
  Information representation type.
  Half-life of information (exponential decay).
  Storage capacity.

The Model Human Processor - Diagram



Modified from Card, Moran, & Newell (1983)

Cognitive Processor

Production system model for cognitive processor:

- "Programming" for cognitive processor consists of production rules:
  Pattern-action pairs stored in production memory:
  IF (pattern) THEN (action)
  IF (pattern) THEN (action)
  ...
- Patterns tested in parallel, actions performed serially.
  A recognize-act cycle, takes 70 ms.
- Procedures or skills composed of a set of rules that fire in the proper sequence depending on the situation.
- Works pretty well in detailed theories of skill and problem-solving - especially in computational models.

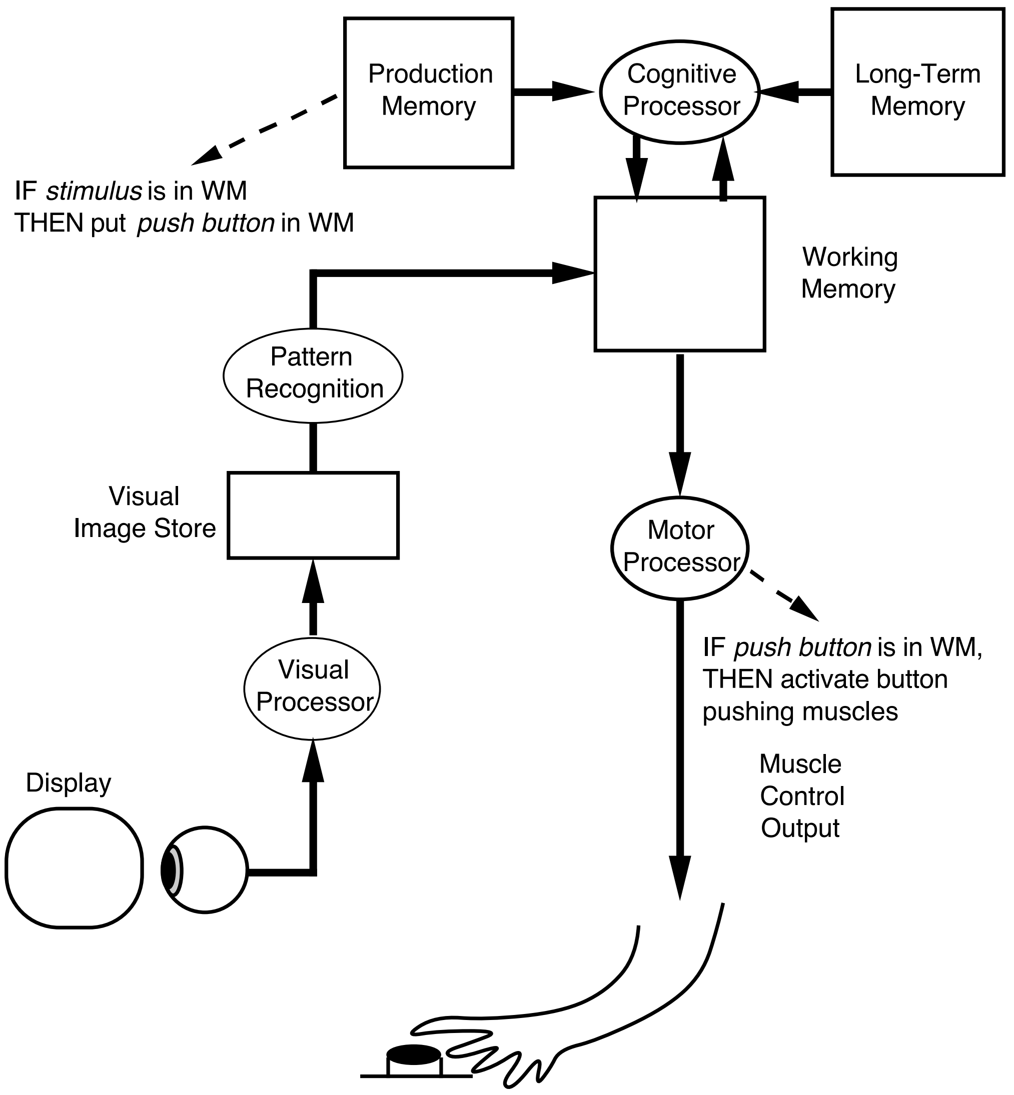
Cognitive processor "thinks" by manipulating Working Memory (WM) contents according to production rules.

- Working Memory is the scratch-pad memory.
- Holds perceptual output, results of production rule actions.
- In each cycle, contents of WM and LTM can trigger associated actions.
- Actions can:
  Alter contents of WM, controls rule triggered next.
  Eventually operate on environment, change perceptual input, change contents of WM.

Example: Simple Reaction Time

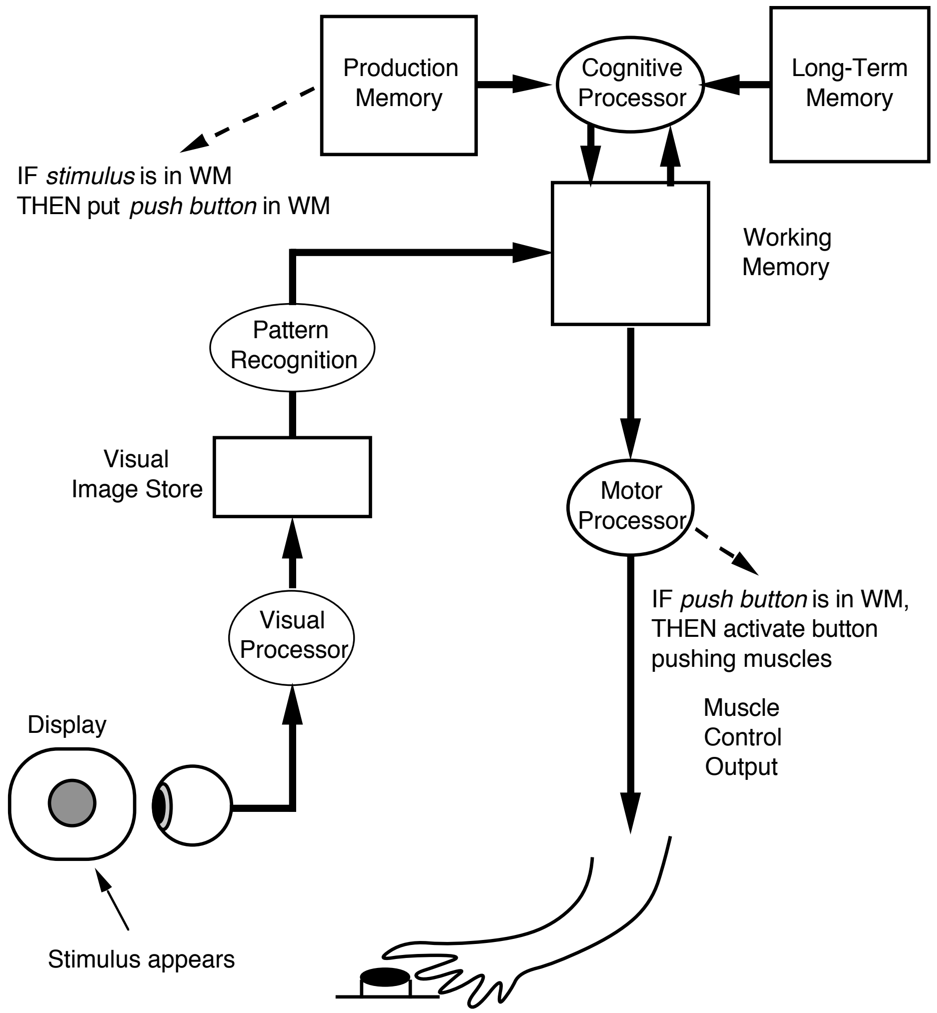
Production rule "Programming" for Simple Reaction Task.

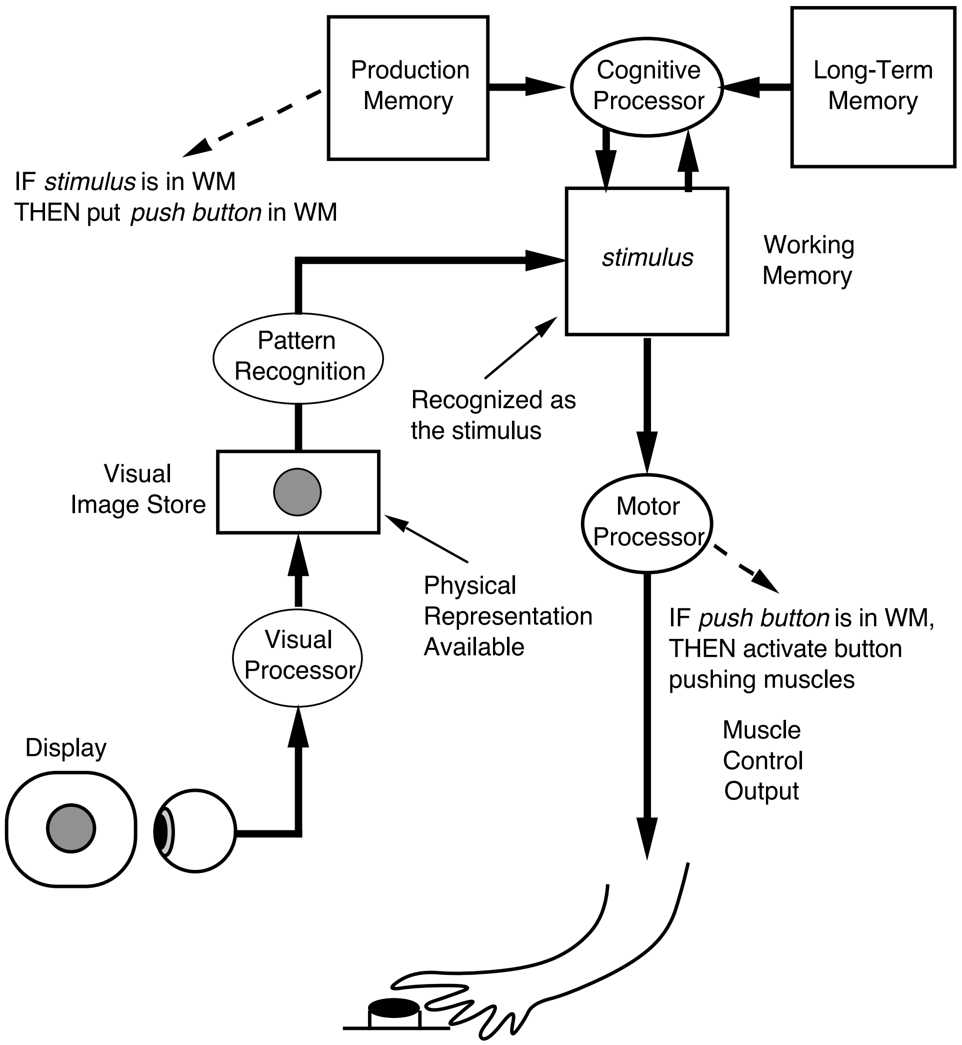
Push the button as soon as you see the light on the display.



Simple Reaction - 1

Stimulus appears, then is perceived and recognized.





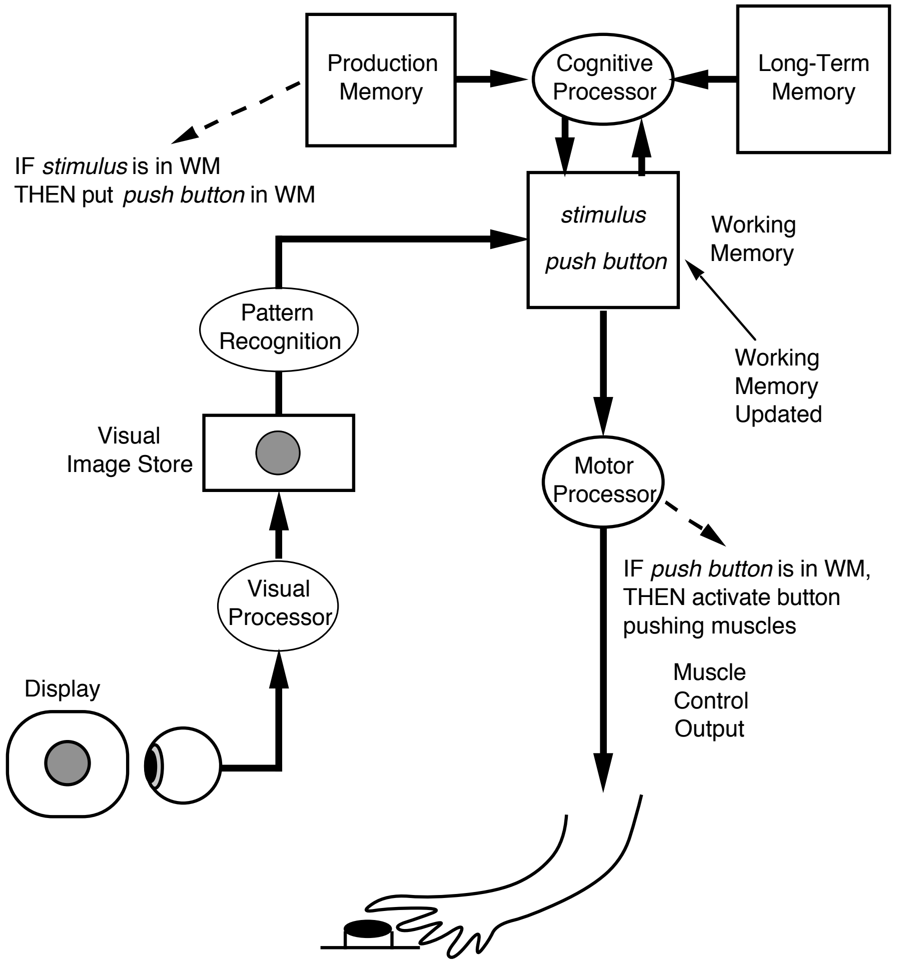
Time = 0 ms

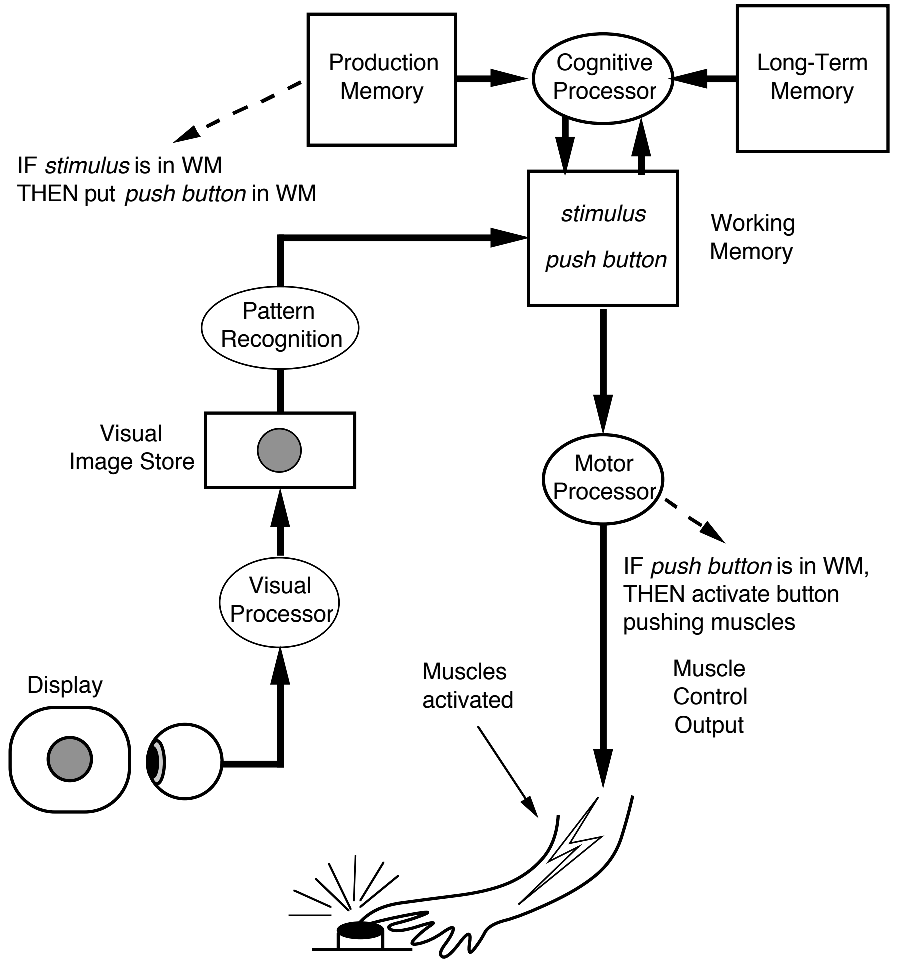
Time = 100 ms

Simple Reaction - 2

Cognitive processor uses rule to decide to make response.

Motor processor responds to command in working memory.





Time = 100 + 70 ms

Time = 100 + 70 +70 ms = 240 ms

Next: Some Details

The Big Picture provides overall framework.

- How the problem is approached.

The Small Picture (MHP) is an example of how many details can be organized into an architecture for predicting and explaining cognition and performance relevant to HCI.

- Considerable current activity on computational tools using MHP and much more sophisticated architectures to simulate what happens when a human interacts with a device - in quantitative and exact detail.
- GOMS models are a simpler version of the same approach; current work developing computational tools and applying them to analyze complex tasks.

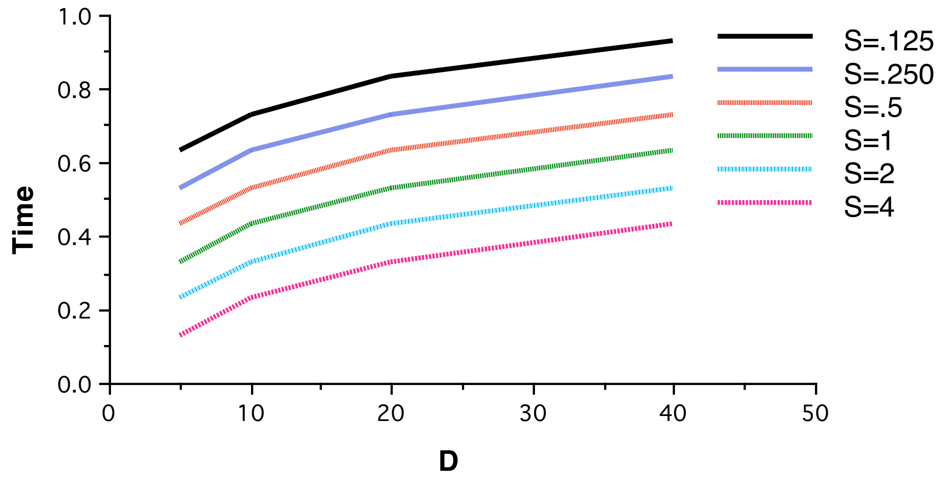
Remainder of presentation fills in some important ideas about perceptual, motor, and cognitive systems.

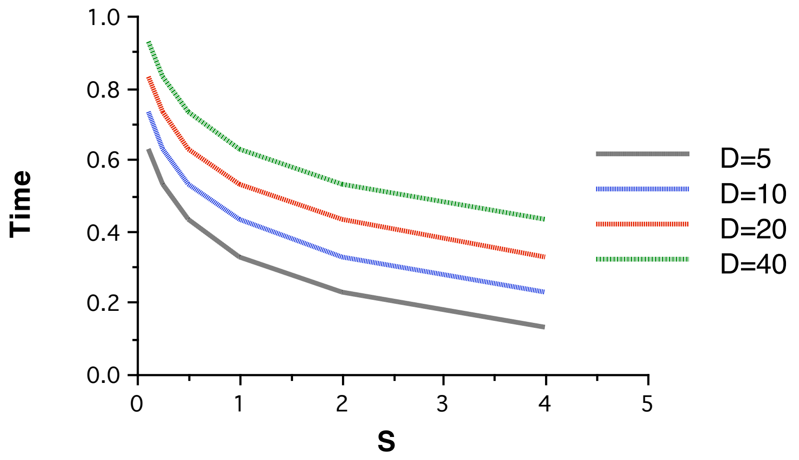
Input Basics: Aimed Movements

Aimed movements (pointing at a target) are a series of micromovements that "zero in" on the target using visual feedback.

- Movement time depends on the number of micromovements, which depends on target size (*S*) and distance (*D*).
- Follows a specific quantitative relationship, *Fitts' Law*:

$Time = I_M log_2(D/S + .5)$, where $I_M = 100\ ms/bit$





Time less than linear for distance - far targets relatively fast.

Time more than linear for size - small targets much slower.

The Mouse Follows Fitts' Law

Properly designed computer mouse follows Fitts' Law.

- Extremely efficient for long moves.
- Not much longer to reach than cursor keys.
- But small targets can be a problem.
- Time spent moving between mouse and keyboard:
  Also a Fitts' law movement.
  Relatively fast, due to big targets.

Other devices are slower because time governed differently.

- E.g. cursor keys linear with total of x & y distance.
- Trackball can be as good (if not stroked), but not better than a good mouse.
- Joysticks can be very different, in some cases quite difficult to learn and to use - depending on control regime.

Mouse is a good way to point, but the keyboard is faster if only a few keystrokes needed.

- Average mouse move = 1.1 seconds.
- Average keystroke (nonsecretarial user) = .28 sec.
- Why a well-designed keyboard-based interface can be good.

Keyboards

Why keyboards are a good idea:

- Hand entry of characters is very slow:
  Printing - 545-952 ms/char; Writing - 732 ms/char.
- Keyboards are much faster after practice:
  Completely unskilled - 1154 ms/stroke.
  Typical - 280 ms/stroke.
  Best - 60 ms/stroke.
- Pen-based interfaces are not fast, but have other advantages!

Standard QWERTY keyboard layout is OK!

- Various myths about Sholes' intentions are factually incorrect.
- Relatively fast - fairly efficient use of alternating-hand speed advantage.

Better layouts exist, but are not tremendously faster.

- Dvorak (American Simplified Keyboard) 5-20% faster.
- Ergonomic keyboards alleviate repetitive stress problems.

Avoid alphabetic arrangements - generally slower:

- Almost everybody knows some QWERTY layout.
- No standard alphabetic layout - have to learn each one, slow visual search.
- Little use of alternating-hand advantages.

Output Basics: Human Vision

Human Vision uses two receptor systems:

- High resolution color system:
  - Receptors ("cones") densely packed into the fovea, fewer outside.
    - Fovea is a patch on the retina at central point of vision (central 1°).
  - Requires high light levels (e.g. normal daytime/office lighting).
  - Receptors for three color ranges: red, green, blue.
- Sensitive monochrome system:
  - Receptors ("rods") outside the fovea and in periphery.
  - Poor resolution - relatively low receptor density.
  - Extremely sensitive (e.g. starlight), but not involved in normal lighting.

Visual processing of a display requires moving foveal vision around.

- Like using a flashlight in a dimly-lit room; can see all objects, but no detail.

All visual properties can be recognized everywhere in visual field, but objects in the periphery must be larger.

- Maximum detail, resolution available in the fovea.
- Peripheral vision useful for locating objects of interest.
- Good cues: grouping by white space, relative locations, color, movement, brightness changes.

Current Computer Displays Underutilize the Visual System

Resolving power of the eye:

- Normal vision - can resolve target of about 1 minute of arc:
  A quarter at 100 yards, about 300 dpi at 12".
  Good design point is 20/40 vision - 2 minutes of arc.
- Much better than typical display devices (~ 80 dpi).
- Can put much more information on screens than usually exploited.
  Key is good visual organization and grouping to allow for fast scanning.

Large high-resolution monitors, smaller characters and icons make more information available.

- Pixels are valuable and expensive - use them efficiently!

Text on a display is normally far less legible than ordinary paper printing.

- High overhead for reader - must scroll or flip through multiple screens to get equivalent of single paper page.
- Requires best high-end current technology to match paper.

Lettering sizes:

- For 28" reading distance, .1" - .2" height is recommended minimum size.
- Displays are usually closer, so .1" is usually big enough.
- Consistent with traditional typography recommendations.

Basic Principles of Display Design

***What not to do:* Exclusive focus on appearance.**

- Be concerned only with what the screens look like: screen layout, colors, icons, attractiveness, interest, graphic quality.

***What to do:* Present the right information in the right way at the right time.**

Choose the right information:

- What decisions will the user make at each point in the task?
- What information will the user need to make those decisions?

Pick a good representation for the information:

- Different ways of presenting the same abstract information can have huge effects - the "representation effect.”

Attempt to present all relevant information simultaneously:

- Visual search of even a complex display can be extremely fast compared to bringing up additional screens.
- With good visual layout and coding, displays can deliver much more information than generally realized!

Icons vs. Words as Display Objects

Misguided GUI style: Use nifty icons, not boring words!

- Fun, decorative, but usually not helpful!

Icons are definitely better mouse targets than words.

- Square shape gives more target area.
- Words tend to be long and thin; often provide small target.

Icons are often arbitrary and meaningless compared to typical words for computer objects.

- Icons can be hard to recognize compared to words.
  Especially if item concept is abstract.
- Why make the user memorize a meaningless symbol?
  “There is a near-universal coding scheme that users know extremely well from many years of practice – words!”
  - Ruven Brooks
- Icons work best when closely resemble a concrete, familiar object commonly associated with the task.
  Can be recognized from appearance and context.
  E.g. a pencil icon in a drawing program.

Icons vs. Words - 2

Icons generally do not help visual search! Research results:

- Icons are easy to find on a screen only if they have a single simple distinctive perceptual feature known to the user.

  Color is best - produces powerful “pop out” effect.

  Simple shape is second best.
- But most icons are multicolored complex shapes!

  Almost useless for guiding visual search.
- Word labels are often more useful in search than an icon.

  E.g. can be alphabetized.

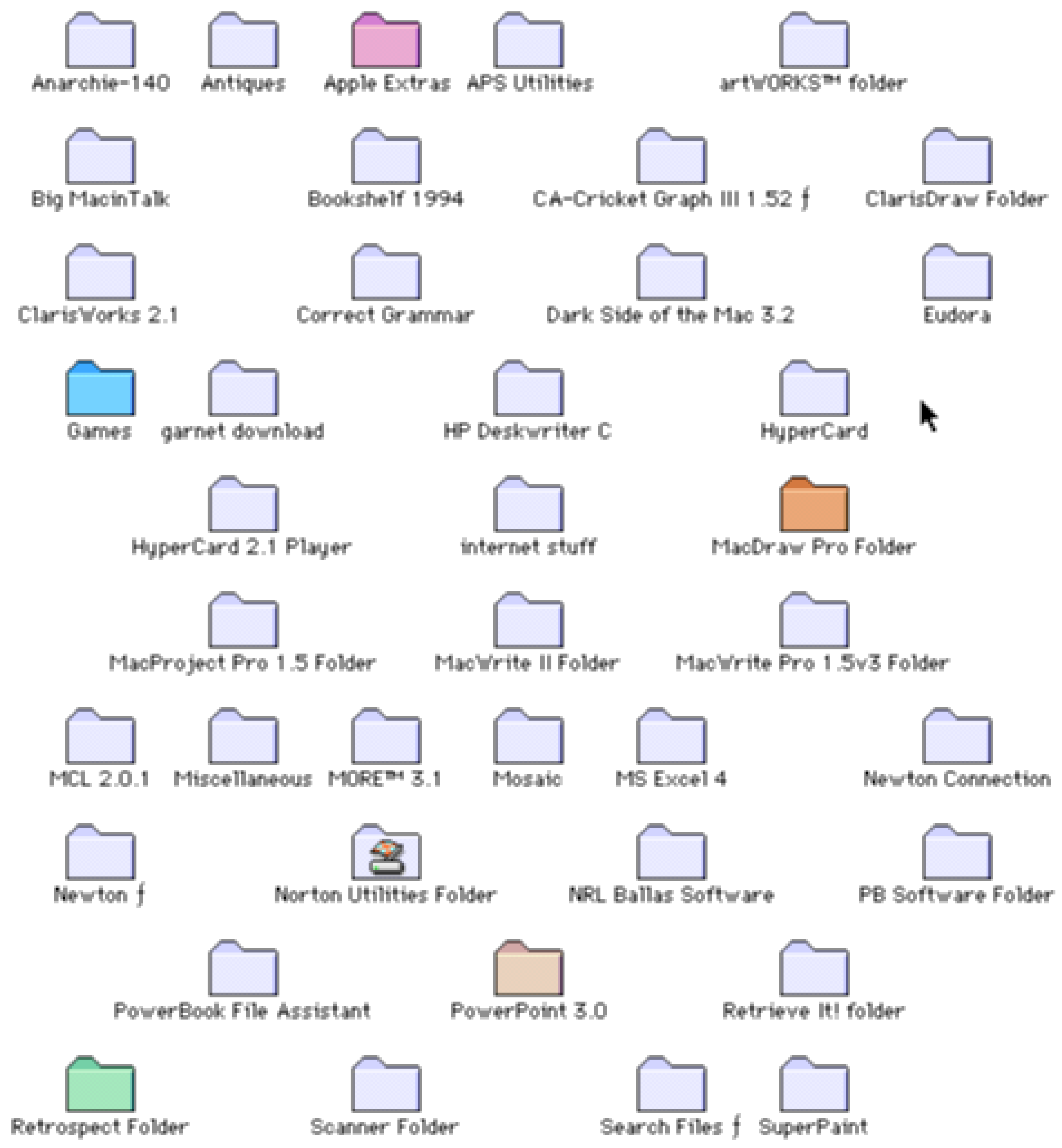
Spatial arrangement of icons on screen is not very helpful.

- Users can't remember identity of more than a few objects just by their location on the screen.

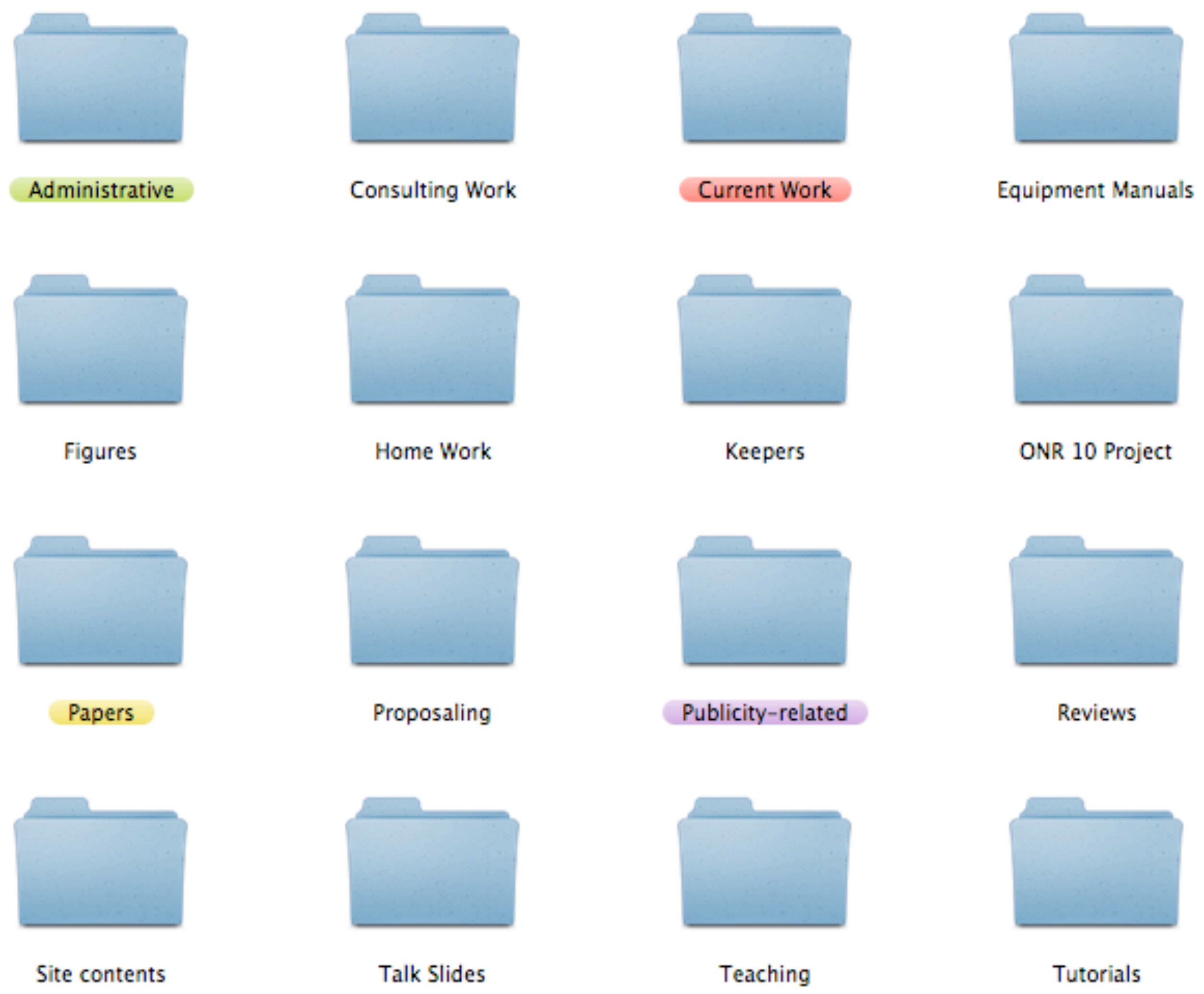
  The computer “desktop” is just like a real desktop!
- Word labels usually required for icons to be usable.

  E.g., icons for files must also have a file name.

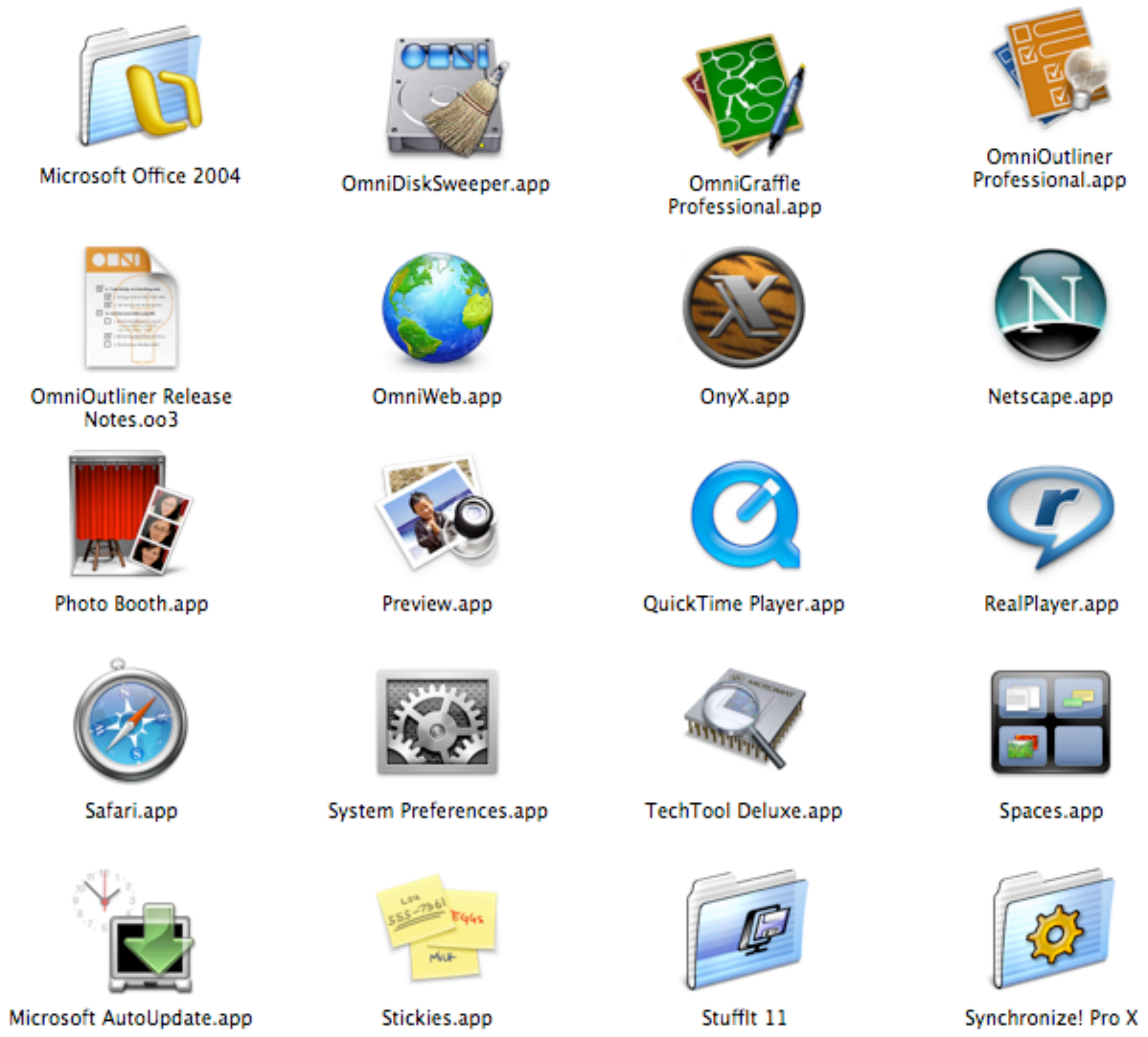
Pop-out from Color



Smaller Color Area not as Good



Icons are usually too Complex to Guide Visual Search!



Procedure Basics: How Procedures are Learned

Cognitive Processor is "programmed" with procedural knowledge acquired from learning.

At first, procedures are declarative knowledge from:

- Problem solving - infer the procedures.
  Reason from general knowledge, trial & error; difficult, slow.
- Explicit instructions - comprehend verbal material.
  Efficient unless instructions are defective - very common.

With practice, converted into procedural knowledge.

- Can be routinely executed to achieve a goal - a routine skill.

With extensive practice, a skill becomes *automated*.

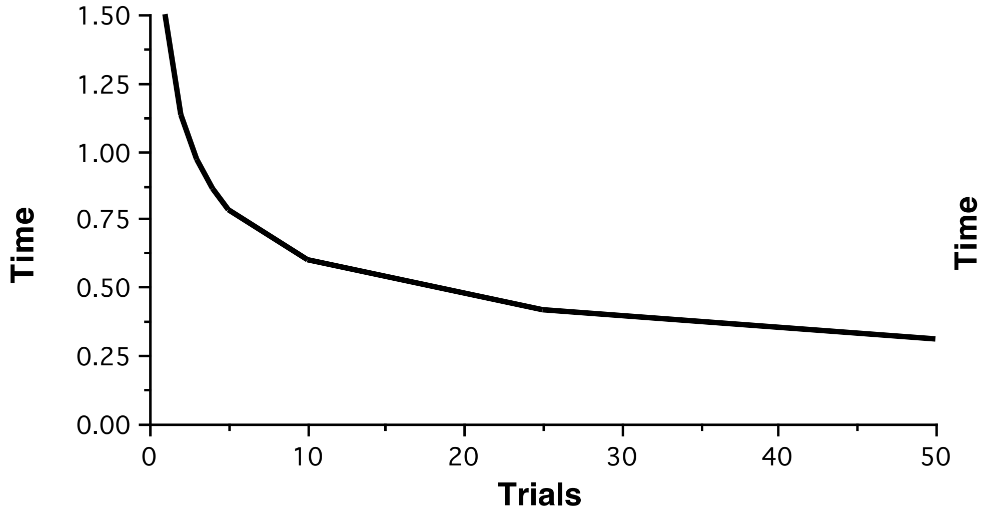
- Can perform procedure "without thinking."
- Involves minimum WM load, cognitive processor load.
- Only exact same activity can be automated.
- Thousands of trials before full automation.

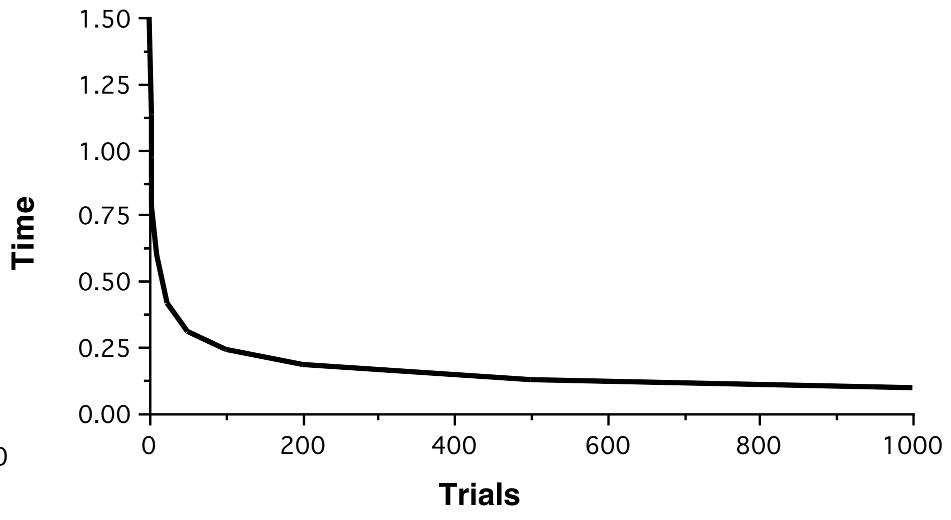
Power Law of Practice

Performance time for a skill decreases according to

$T_n = T_1 n^{-\alpha}$, where n is number of trials, $\alpha = .4$ (typical).

Rapid initial decrease, then levels off.





Many practiced skills have essentially stable performance times, enabling practical prediction.

Procedure Learning and Usability

Common view: The user interface consists of display content: the "screens," dialogs, menus, buttons, icons, etc.

More correct view: The interface also entails procedures that the user has to learn and follow to get the work done.

To defrangulate a gizmo:
Step 1. Select the gizmo.
Step 2. Point to the "Action" menu.
. . . etc. …

What role does the complexity of the procedures play?

Research results: A system with simple, consistent, and efficient procedures is both easy to learn and fast to use.

- Secret of the original Macintosh interface and its imitators.
- A few simple, consistent procedures for the most frequent tasks are shared by all applications and the operating system.
- Probably more important than any other single factor!

Model-based Evaluation of Usability from a Procedure Analysis

Use a model of the human doing the task to replace user testing results.

- Describe the interface design in detail.
- Model the user's procedures for doing the task with the design.
- Use the model to predict execution or learning time.
- Revise or choose design depending on prediction.

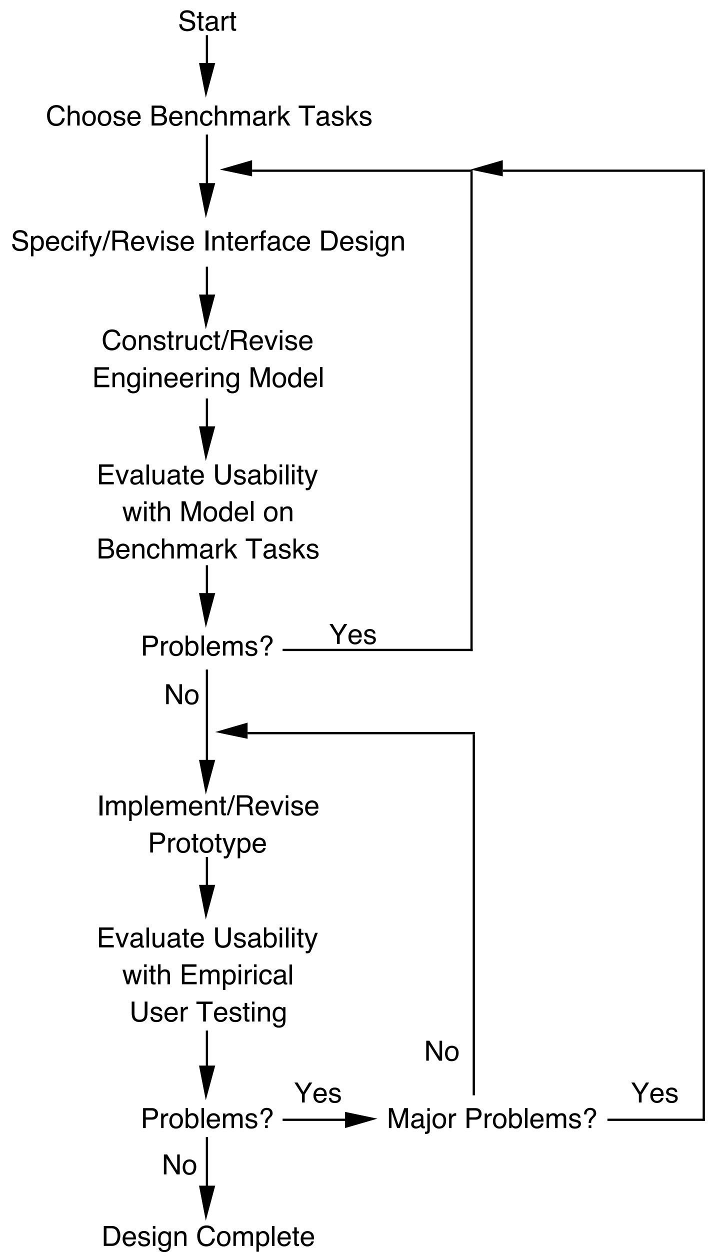
Advantages:

- Get usability results before prototyping or testing with human users.
  - Can be applied during initial design stages - saving time and money!
  - Critical in expert domains, transfer of training, complex system design.
- Model summarizes the interface design from the user's point of view.
  - Represents how the user gets things done with the system.
  - Model components can be reused in design of related interfaces.

Limitations:

- Current of-the-shelf models can only predict a few aspects of usability:
  - Execution time, procedure learning time, effects of consistency.
- Some user testing still required, but less if modeling is used.
  - Assess usability aspects not covered by available models.
  - Protection against errors, oversights, in the model analysis.

Model-based Evaluation Process


Start
Choose Benchmark Tasks
Specify/Revise Interface Design
Construct/Revise Engineering Model
Evaluate Usability with Model on Benchmark Tasks
Problems?
Yes
No
Implement/Revise Prototype
Evaluate Usability with Empirical User Testing
Problems?
Yes
No
Major Problems?
No
Yes
Design Complete

Model first, user testing at end; loop back if needed.

GOMS Models: A Family of Model-based Evaluation Methods

GOMS: An approach to describing the procedural knowledge that a user must have in order to operate a system.

- Proposed by Card, Moran, & Newell (1983).
- Several types of GOMS models (see John & Kieras, 1996).
- The Keystroke-Level Model is simplest, most heavily used.
- Procedural GOMS Models are most powerful, general.

Components of a GOMS model:

- Goals - what can be accomplished with the system.
- Operators - basic actions that can be performed.
- Methods - sequences of operators that accomplish each goal.
- Selection Rules - choose which method should be used for.

Use characteristics of methods to predict usability

- Length, overlap predicts time to learn.
- Execution path, operator sequence predicts time to execute.

Keystroke-Level Model: The Simplest GOMS Method

Together with precursors and variants, very heavily used.

Estimate the time to do a task by defining the required method and totaling the estimated times for each "keystroke-level" operator in the method.

Recipe adapted from Card, Moran, & Newell (1983):

1. Choose one or more representative task scenarios.
2. Have design specified to the point that keystroke-level actions can be listed.
3. List the keystroke-level actions (operators) involved in doing the task.
4. Insert mental operators for when user has to stop and think.
5. Look up the standard execution time to each operator.
6. Add up the execution times for the operators.
7. The total is the estimated time to complete the task.

Typical Operators and Times for the Keystroke-Level Model

K - Keystroke.

- Pressing a key or button on the keyboard.
- Depends on skill levels - about 0.28 s for typical non-secretarial user.
- Pressing Shift or Control key is a separate keystroke.

P - Point with mouse to a target on the display.

- Typically ranges from .8 to 1.5 sec for text-editing.
- Suggested average is 1.1 sec, use Fitts' law for more accuracy.

B - Press mouse button.

- Use .1 sec, mouse button click (BB) takes .2 sec.

H - Home hands to keyboard or mouse.

- Takes .4 sec.

W - Wait for system response.

- Only when user is idle because can not continue; measure from system.

M - Mental act of thinking - routine thought, not problem-solving.

- Estimates ranges from .6 to 1.35 sec.
- Use 1.2 sec if no better information available.

Additional operators can be defined and measured - and have been!

Quick Example Keystroke-Level Model

How long does it take to delete a file?

- Method and Operators (no Mental operators - for simplicity):
  1. Point to the file icon (P).
  2. Press and hold mouse button (B).
  3. Drag icon to trash/recycle bin icon (P).
  4. Release mouse button (B).
- Total time = 2P+2B = 2(1.1)+2(.1) = 2.4 s

Would a mouse + keyboard sequence be faster?

- Method and Operators (assume left hand stays on keyboard):
  1. Point to the file icon (P).
  2. Click mouse button (2B).
  3. Type ctrl-D (2K).
- Total time: 1P+2B+2K = 1.1 + .2 + 2(.28) = 1.86 s

Answered a design question with a model!

- No need to collect human user data!

Example of Procedural GOMS Model

Concept: Explicit, executable, general procedures for a class of tasks.

Excerpt from a model for a radar workstation - GOMSL notation.

- During task, user needs to close the track data window.
- Goal triggers first method, which asserts a subgoal to click on a button.
- Sub-goal triggers second method, which finds the button object with the label, and then points to it and clicks.

```
Method_for_goal: Close Track_data_window
  Step 1. Accomplish_goal: Click_on Button using "Close CRO".
  Step 2. Return_with_goal_accomplished.

Method_for_goal: Click_on Button using <button_label>
  Step 1. Look_for_object_whose Label is <button_label>,
   and Type is Button and_store_under <button>.
  Step 2. Point_to <button>.
  Step 3. Click Left_mouse_button.
  Step 4. Delete <button>; Return_with_goal_accomplished.
```

Predicts human performance fairly accurately and generally.

- Total method length predicts procedure learning time.
- Execution trace contains sequence of operators for the entire task.
- Can handle complex tasks and scenarios easily.
- Simulate by hand, or use simulation modeling tool.

Newer Developments in HCI Modeling

High-Level GOMS Models for *design of functionality*.

- Methods invoke assumed system functionality to accomplish goals, without any interface specifics.
- Iterate on choice of functionality until high level methods are simple, fast, and efficient.
- Expand into a specific interface design and the corresponding GOMS model to evaluate usability of the complete system.

Cognitive Architecture Models - detailed simulated humans.

- Architecture contains detailed models of perceptual, cognitive, and motor systems, programmed with production rules.
- Account for detailed effects and strategies in HCI tasks.
  E.g. role of visual acuity, eye movements, and strategies in visual search and selection in complex displays.
- Directly connects modern cognitive psychology theory and results to problems in HCI design.

Further Readings

Anderson, J.A. (2004). *Cognitive Psychology and its Implications*. (6th Ed). New York: Freeman, 1995.

Boff, K.R., Kaufmann, L. & Thomas, J. P. (Eds). (1986). *Handbook of perception and human performance*. New York: Wiley.

Byrne, M. D. (2003). Cognitive architecture. In J. Jacko & A. Sears (Eds), *Human-Computer Interaction Handbook*. Mahwah, N.J.: Lawrence Erlbaum Associates.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Hollands, J., & Wickens, C. D. (1999). *Engineering Psychology and Human Performance* (3rd Ed), New York: Prentice-Hall.

John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction*, 3, 287-319.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: *Comparison and contrast. ACM Transactions on Computer-Human Interaction*, 3, 320-351.

Kieras, D.E. (2003). Model-based evaluation. In Jacko, J.A. & Sears, A. (Eds) *The human-computer interaction handbook*. Mahwah, New Jersey: pp. 1139-1151.

Kieras, D. E. (2004). Task analysis and the design of functionality. In A. Tucker (Ed.) *The Computer Science and Engineering Handbook* (2nd Ed). Boca Raton, CRC Inc. pp. 46-1 - 46-25.

Mayhew, D.J. (1992). *Principles and guidelines in software user interface design*. New York: Prentice-Hall.

Sanders, M.S. & McCormick, E.J. (1993). *Human factors in engineering and design*. (7th Ed). New York: McGraw-Hill.

Shneiderman, B., & Plaisant, C. (2005). *Designing the User Interface: Strategies for effective human-computer interaction*. (4th Ed.) Reading, Massachusetts: Addison-Wesley.

Zhang J. (1996). A representational analysis of relational information displays. *International Journal of Human-Computer Studies* 45, 59-74.