Human-Computer Interaction

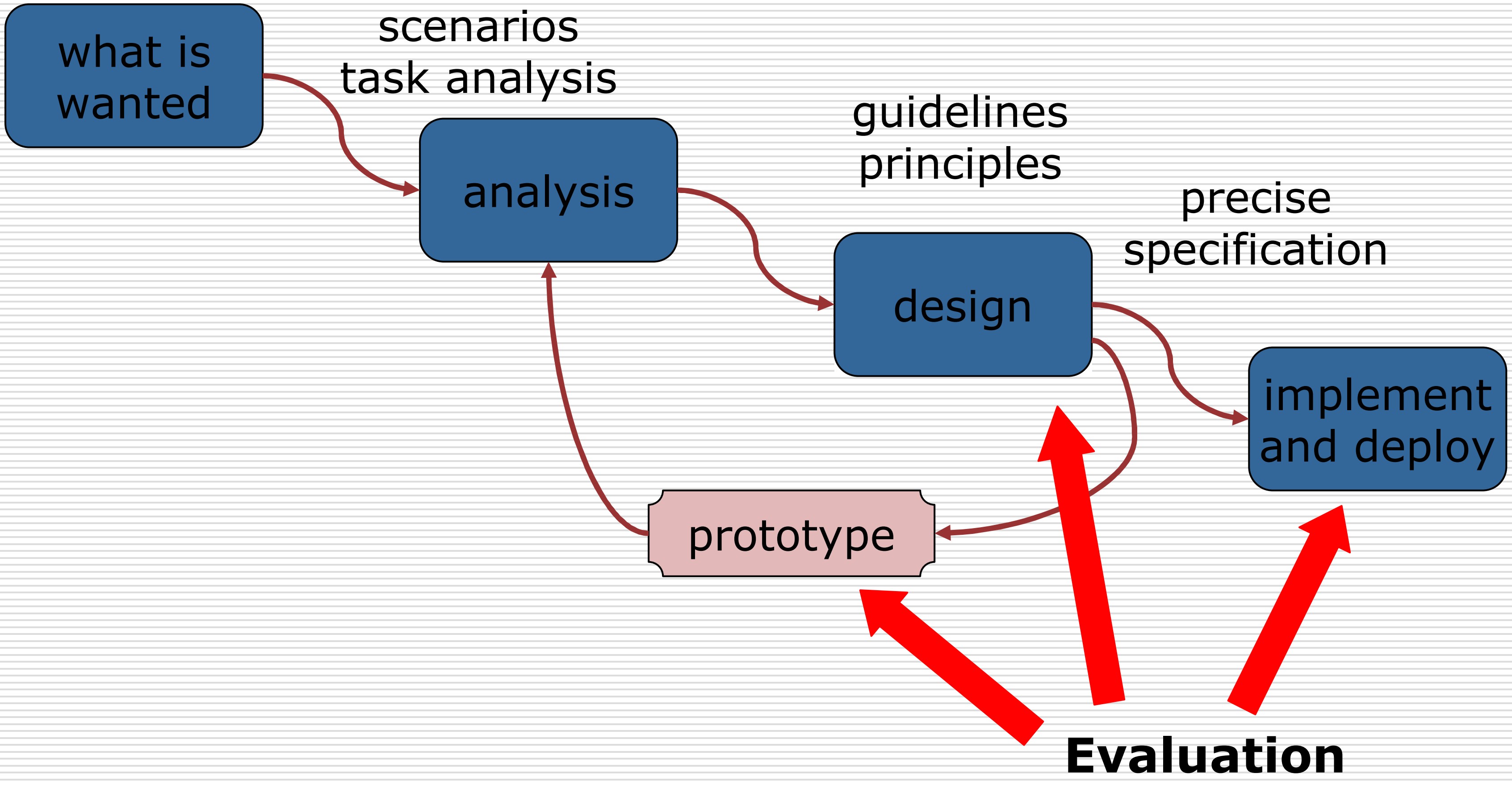
Lecture 7:

Evaluating interactive systems I

Dix et al., “Human Computer Interaction”, chapt.9

Online seminar on “Cognitive Modeling for User Interface Design”, 2005, David E. Kieras (U. of Michigan)

Designing a usable system

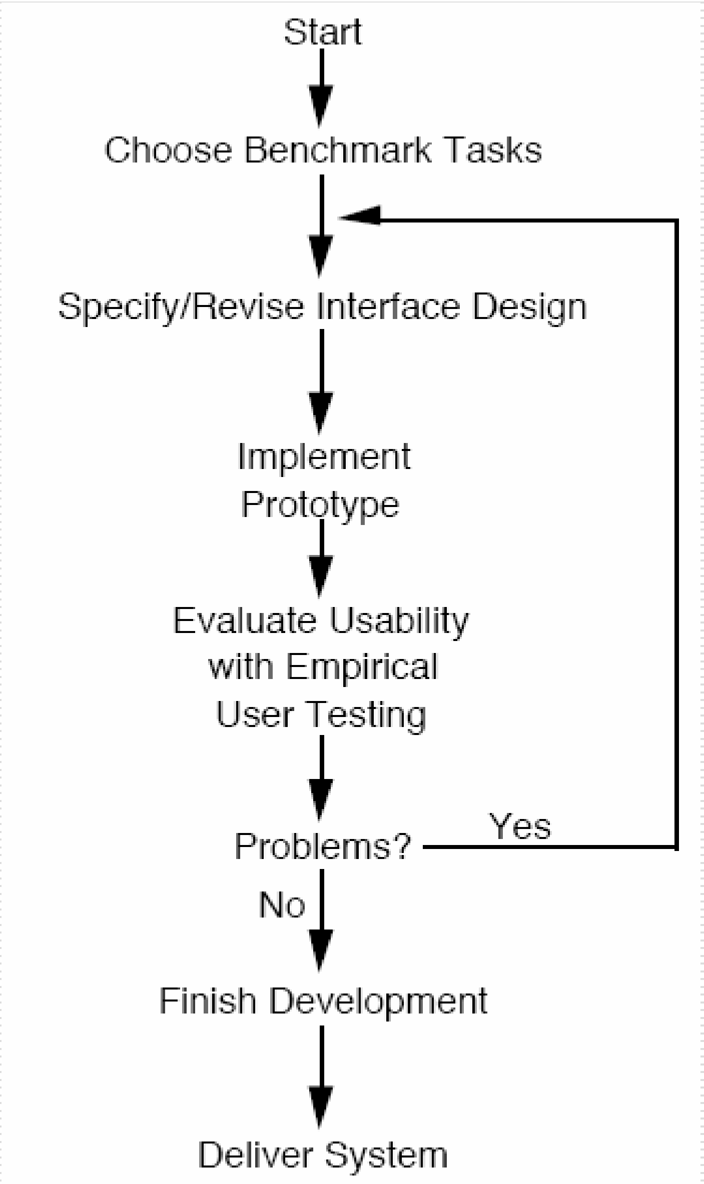


Evaluation

- occurs in laboratory, field and/or in collaboration with users
- should be considered at *all* stages in the design life cycle: design, prototype(s), implementation
- goals
  - assess extent of system functionality and usability
  - assess effect of interface on user
  - identify specific problems
  - inform improvement to the design

Human Factors Process

- Early use of guidelines, empirical user testing of prototypes
- Identify problems that impair learning or performance
- Compare user performance to a specification
- Experimental methodology
  - How to run experiments on human behavior and draw valid conclusions?



Cognitive Walkthrough

- Proposed by Polson et al. (1992)
- similar to code walkthrough in software engineering
- usually performed by expert in cognitive psychology
- expert ‘walks through’ design to identify potential problems using psychological principles
- motivation: many users prefer to learn how to use a system by exploring its functions hands on
- evaluates how well it supports user in *learning* task
- forms used to guide analysis

Cognitive Walkthrough

- 4 things needed:
  - Specification or prototype of the system
  - Description of the task, which most users will want to do
  - Complete list of actions needed to complete the task
  - Indication of who the users are, their experience and knowledge, for the evaluator to assume
- Analysis focuses on goals and knowledge: does the design lead the user to generate the correct goals?
- For *each* action, evaluator tries to answer 4 questions
  1. Is the effect of the action the same as user's goals?
  2. Will users see that the action is available?
  3. Will users know the found action is the one they need?
  4. Will users understand the feedback they get on the action?

Heuristic Evaluation

- Proposed by Nielsen & Molich
- Test dDesign (specification) if certain usability criteria are violated (see table; cf. last course)
- Carried out independently by several experts (rule of thumb: 5 experts will usually discover 75% of the usability problems)
- Heuristic evaluation `debugs' design

Visibility of system status	Recognition rather than recall
Match between system and real world	Flexibility and efficiency of use
User control and freedom	Aesthetic and minimalist design
Consistency and standards	Help users recognize, diagnose and recover from errors
Error prevention	Help and documentation

Evaluation through user participation

- Needs at least a working prototype
  - Possibly use of *Wizard of Oz* technique, where part of the system functionality is simulated by a wizard.
- Number of different ways to study how the user is doing with the system
  - Query techniques
  - Observational methods
  - Empirical/experimental methods
  - Physiological monitoring
- Lab vs field studies

Lab studies

- Experiment under controlled conditions
  - specialist equipment available
  - uninterrupted environment
- Disadvantages:
  - lack of context
  - difficult to observe user cooperation
- Prevalent paradigm in psychology

Field studies

- Experiments dominated by group formation
- Field studies more realistic
  - *distributed cognition* $\Rightarrow$ work studied in context
  - real action is *situated*
  - physical *and* social environment crucial
- sociology and anthropology – open study and rich data

Query techniques

User interview
Questionnaire

Query techniques - interviews

- analyst questions user on one-to -one basis
  usually based on prepared questions
- informal, subjective and relatively cheap

- Advantages
  - can be varied to suit context
  - issues can be explored more fully
  - can elicit user views and identify unanticipated problems
- Disadvantages
  - very subjective
  - time consuming

Query techniques - questionnaires

- Set of fixed questions given to users
  - Need careful design
- Styles of question
  - general
  - open-ended (“Can you suggest improvements to the system?”)
  - scalar (judge a statement on a numeric scale)
  - multiple-choice
  - Ranked (ordering of items in a list)

- Advantages
  - quick and reaches large user group
  - can be analyzed more rigorously
- Disadvantages
  - less flexible
  - less probing

Observational methods

Think Aloud

Cooperative evaluation

Post-task walkthroughs

Think Aloud

- user is observed while performing the task
- user asked to describe what he is doing and why, what he thinks is happening etc.

- Advantages
  - simplicity - requires little expertise
  - can provide useful insight
  - can show how system is actually use
- Disadvantages
  - subjective
  - selective
  - act of describing may alter task performance

Cooperative evaluation

- variation on ‘think aloud’
- user sees himself as collaborator in evaluation
- both user and evaluator can ask each other questions throughout the evaluation process

- Additional advantages
  - less constrained and easier to use
  - user is encouraged to criticize system
  - clarification possible

- Problems with both techniques
  - generate a large volume of information (*protocols*)
  - Protocol analysis crucial and time-consuming

Post-task walkthroughs

- transcript played back to user for comment
  - immediately $\rightarrow$ fresh in mind
  - delayed $\rightarrow$ evaluator has time to identify questions
- useful to identify reasons for actions and alternatives considered
- necessary in cases where ‘think aloud’ is not possible

Classification – observational techniques

	Think aloud & coop. evaluation	*Post-task walkthrough*
Stage	Implementation	Implementation
Style	Lab/field	Lab/field
Objective?	No	No
Measure	Qualitative	Qualitative
Information	High/low level	High/low level
Immediacy	Yes	No
Intrusive?	Yes	No
Time	High	Medium
Equipment	Low	Low
Expertice	Medium	Medium

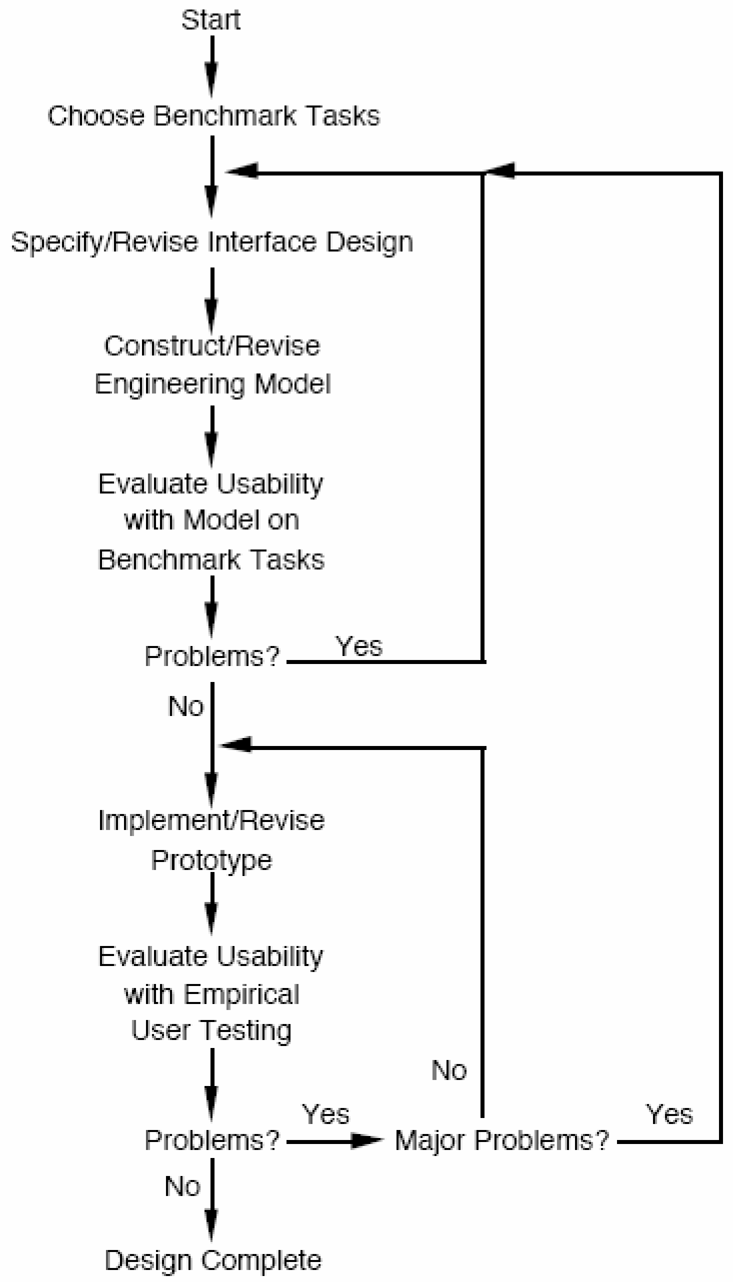
Model-based evaluation

Model-based evaluation

Four steps:

1. Describe interface design in detail
2. Build a model of user doing the task
3. Use the model to predict execution or learning time
4. Revise or choose design depending on prediction

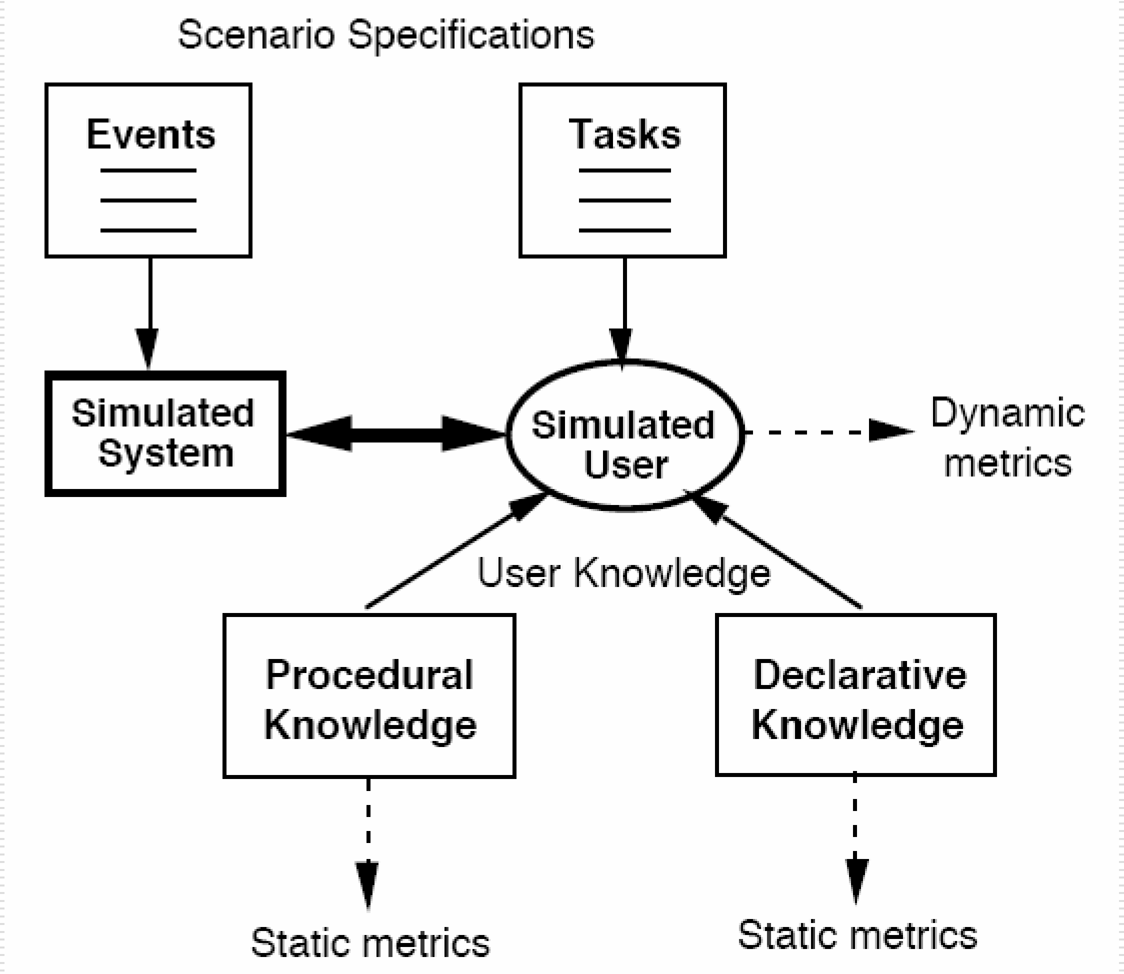
- Usability results *before* implementing prototype or user testing
- Engineering model allows more design iterations


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
Major Problems?
No
Yes
No
Design Complete

Model-based approach

- Model summarizes the interface design from the user's point of view:
  - Represents how the user gets things done with the system.
  - Components of model can be reused to represent design of related interfaces.
- *But*, current models can only predict a few aspects:
  - Time required to execute specific tasks.
  - Ease of learning of procedures, consistency effects
- User testing still required!
  - Assess aspects not dealt with by an analytic model.
  - Protection against errors, oversights, in the analysis.

Overview



Models = simulations of
human-computer
interaction

Procedural knowledge:
how-to procedures
→ executable

Declarative knowledge:
facts, beliefs
→ reportable

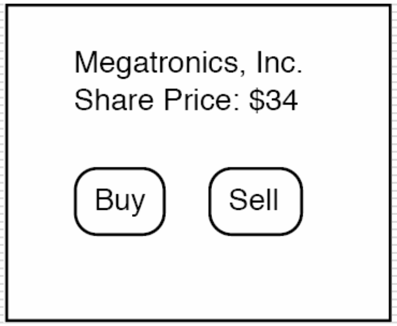
Psychological constraints

- Human abilities, limitations, time course, trajectory, …
- Evaluation of a proposed design must be a routine design activity, not a scientific research project.
- Need to be able to build models without inventing psychological theory.

- Modeling system must provide psychology *automatically*
  - Constrain what the model can do, so modeler can focus on design questions, not psychological basics
  - If model can be programmed to do any task at any speed or accuracy, something’s wrong!
  - Of course, science is never complete, but need to include as much as possible

Cognitive vs perceptual-motor constraints

- What dominates a task?
  - Heavily cognitive tasks: Human "thinks" most of the time, e.g. stock trading system
- Many HCI tasks dominated by perceptual-motor activity
  - A steady flow of physical interaction between human and computer („doing rather than thinking")
  - Time required depends on human characteristics and computer's behavior (determined by the design)
- Implication
  - Modeling perceptual-motor aspects is often practical, useful, and relatively easy.
  - Modeling purely cognitive aspects of complex tasks is often difficult, open-ended, and requires research resources.



Modeling approaches

Three current approaches:

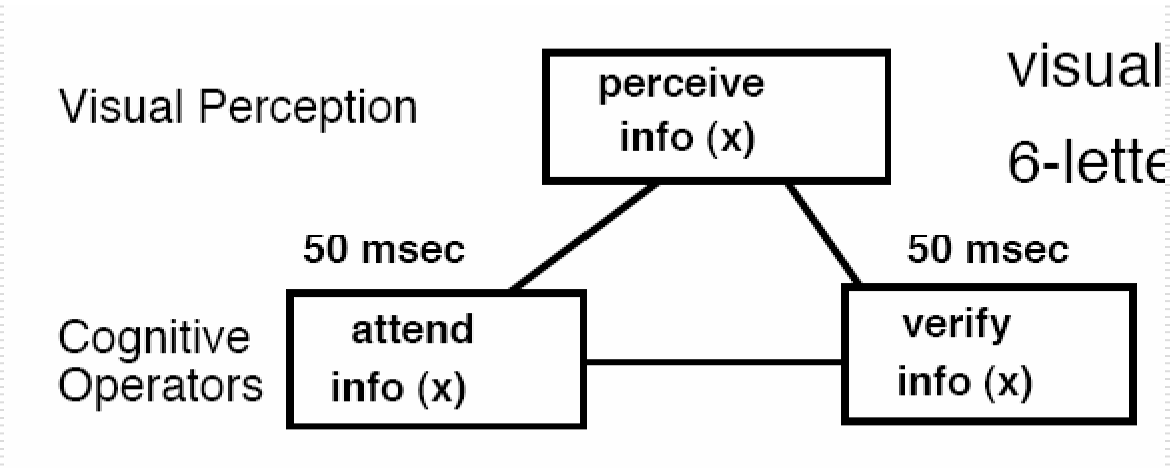
1. Task network models – before detailed design
2. Cognitive Architecture Models – packaged constraints
3. GOMS models – relatively simple & effective

Differ in constraints, detail, when to use.

Task Network Models

- Connected network of tasks:
  - Connection: one task is a prerequisite of the other
  - Both serial and parallel execution of tasks
  - Final time to a complete computed from chain of serial and parallel tasks
  - *Critical path* = chain with largest execution time
  - PERT charts (*Program Evaluation & Review Techn.*), (E)TAGs
- Tasks can be any mixture of human and machine tasks
- Each task characterized by a distribution of completion times, and arbitrary dependencies and effects

Task network - example



Cognitive architectures

Set of components and mechanisms that represent basic human abilities and limitations.

"Programmed" with a strategy to perform specific tasks.

- provides constraints on the form and content of the strategy.
- Architecture + specific strategy = a model of a specific task.

To model a specific task:

- Do a task analysis to arrive at human's strategy for the task.
- "Program" the architecture with representation of strategy.
- Run the model using task scenarios.

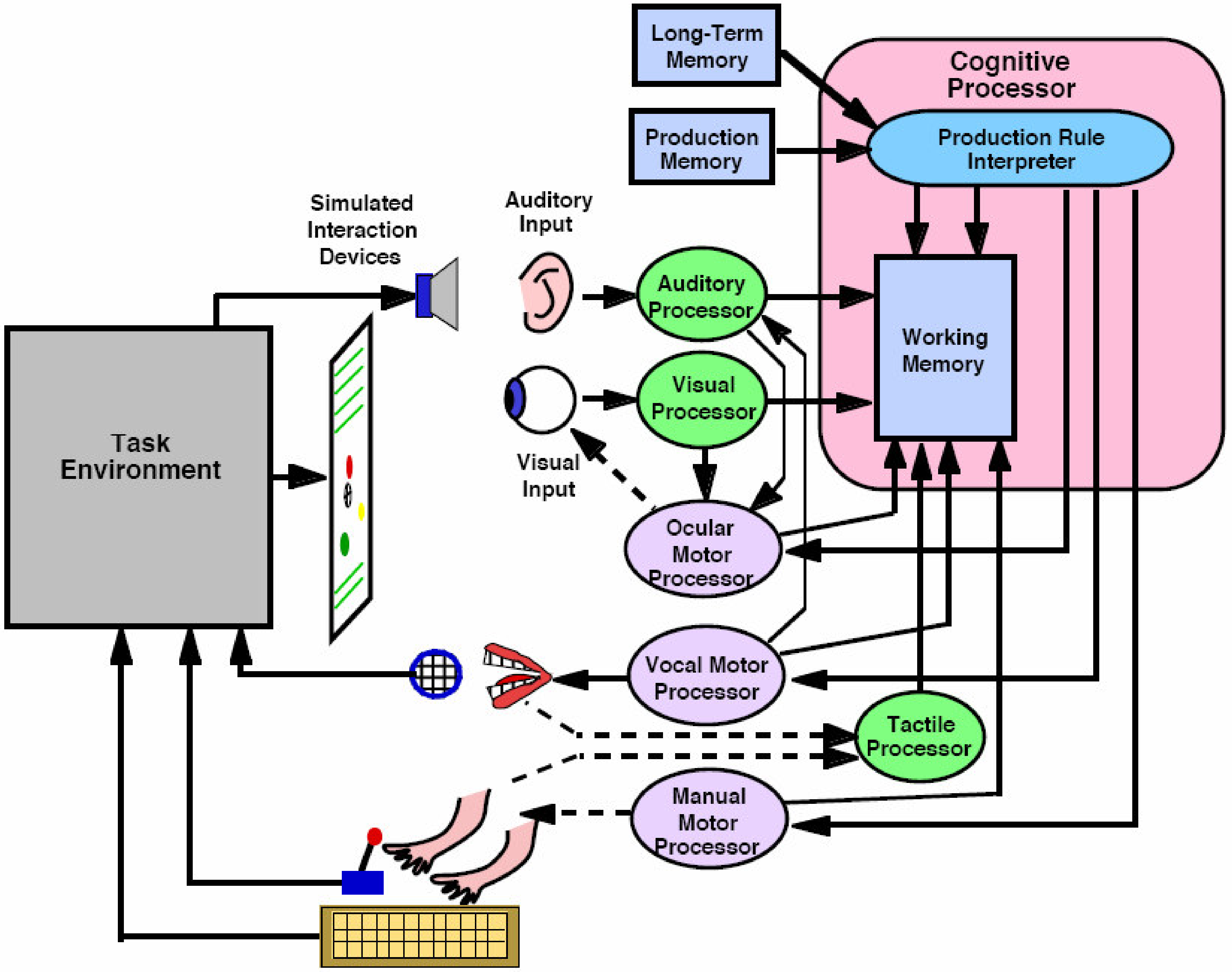
Result: predicted behavior and time course for that scenario and task strategy.

Goal is comprehensive psychological theory, so these are quite complex; used mostly in a research settings

Examples: SOAR/EPIC, ACT/R, GLEAN (see below)

Example: EPIC Architecture

- Developed to represent executive processes that control other processes during *multiple task* performance.
- **E**xecutive-**P**rocess **I**nteractive ***C***ontrol (Kieras & Meyer, mid-1990s)
- Basic assumptions
  - Production-rule cognitive processor.
  - Parallel perceptual and motor processors.
- Fixed architectural properties
  - Components, pathways, and most time parameters
- Task-dependent properties
  - Cognitive processor production rules.
  - Perceptual recoding.
  - Response requirements and styles.

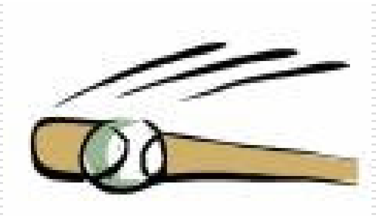


Long-Term Memory
Production Memory
Cognitive Processor
Production Rule Interpreter
Simulated Interaction Devices
Auditory Input
Auditory Processor
Working Memory
Visual Processor
Task Environment
Visual Input
Ocular Motor Processor
Vocal Motor Processor
Tactile Processor
Manual Motor Processor

GOMS models

GOMS (Card, Moran, & Newell, 1983)

- A key model-based methodology based on simplified cognitive architectures.
- An approach to describing the knowledge of *procedures* that a user must have in order to operate a system
  - ***G***oals - what goals can be accomplished with the system
  - ***O***perators - what basic actions can be performed
  - ***M***ethods - what sequences of operators can be used to accomplish each goal
  - ***S***election Rules - which method should be used to accomplish a goal
- Well worked out, quite practical, but limited due to simplifications
- Often in the "sweet spot" - lots of value for modest modeling effort



GOMS model family

- Keystroke-Level Model (KLM)
- Critical-Path Method GOMS (CPM-GOMS)
- Natural GOMS Language (NGOMSL)/Cognitive Complexity Theory (CCL)
- Executable GOMS Language (GOMSL)/GLEAN

Keystroke-level model method

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 (sum of times for tasks t_i multiplied by frequency n_i)

$$T_{execute} = \sum_i t_i * n_i$$

KLM – operators and times

K - Keystroke (.12 - 1.2 sec; use .28 sec for ordinary user).

- Pressing a key or button on the keyboard.
- Different experience levels have different times.
- Pressing SHIFT or CONTROL key is a separate keystroke.
- Use type operator T(n) for a series of n Ks done as a unit.

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

- Follows Fitts' law - use if possible: $0.1 * \log_2 (D/S + 0.5)$
- Typically ranges from .8 to 1.5 sec, average (text editing) is 1.1 sec.

B - Press/release mouse button (.1 sec; click is .2).

- Highly practiced, simple reaction.

KLM – operators and times

H - Home hands to keyboard or mouse (.4 sec).

W - Wait for system response.

- Only when user is idle because can not continue
- Have to estimate from system behavior
- Often essentially zero in modern systems

M - Mental act of thinking.

- Represents pauses for routine activity (not problem-solving).
- New users often pause to remember or verify every step.
- Experienced users pause and think only when logically necessary.
- Estimates ranges from .6 to 1.35 sec; 1.2 sec is good single value.

Example: file deletion in MacOS, original design, experienced user

General procedure

- Find the file icon to be deleted and drag it to the trash can.

Assumptions:

- user thinks of selecting+dragging icon as a single operation.
- Finding to-be-deleted icon is still required
- Moving icons to the trash can is highly practiced:
  - The trash can does not have to be located, so finding the trash can is overlapped with pointing to it.
  - Verifying that the trash can has been hit is overlapped with pointing to it.
  - Final result (bulging can) is not checked since it is redundant with verifying that the can has been hit.

Operator sequence:

initiate the deletion **M,** find the file icon **M,** point to file icon **P,** press and hold mouse button **B,** drag file icon to trash can icon **P,** release mouse button **B,** point to original window **P**

- **Total time = 3P + 2B + 2M = 5.9 sec**

Example: command key file deletion, experienced user

General procedure

- Select the file icon to be deleted and hit a command key.

Assumptions

- User operates both mouse + key with right hand.
- Right hand starts and ends on the mouse.

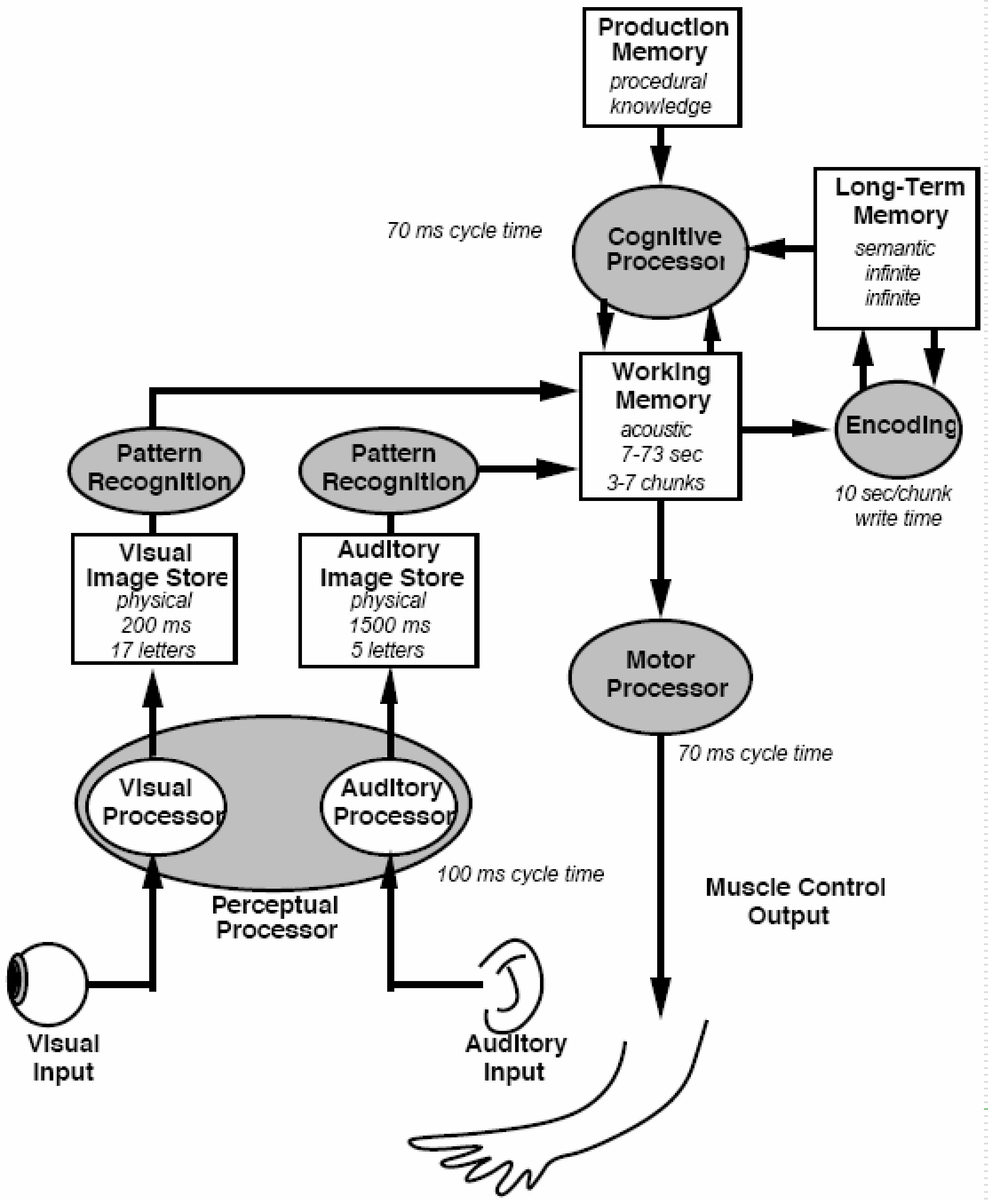
Operator sequence: initiate the deletion **M,** find the icon for the to-be-deleted file **M,** point to file icon **P,** click mouse button **BB,** move hand to keyboard **H,** hit command key **KK,** move hand back to mouse **H**

- **Total time = P + 2B + 2H + 2K + 2M = 5.06 sec**

Only slightly faster, due to need to move the hand!

CPM-GOMS

- Analyze task based on Card, Moran, & Newell (1983), „Model Human Processor"
- ***C***ognitive, ***P***erceptual, ***M***otor GOMS, or ***C***ritical ***P***ath ***M***ethod ***GOMS***
- An informal architecture of cognitive, perceptual, and motor processors with some basic characteristics and time parameters.

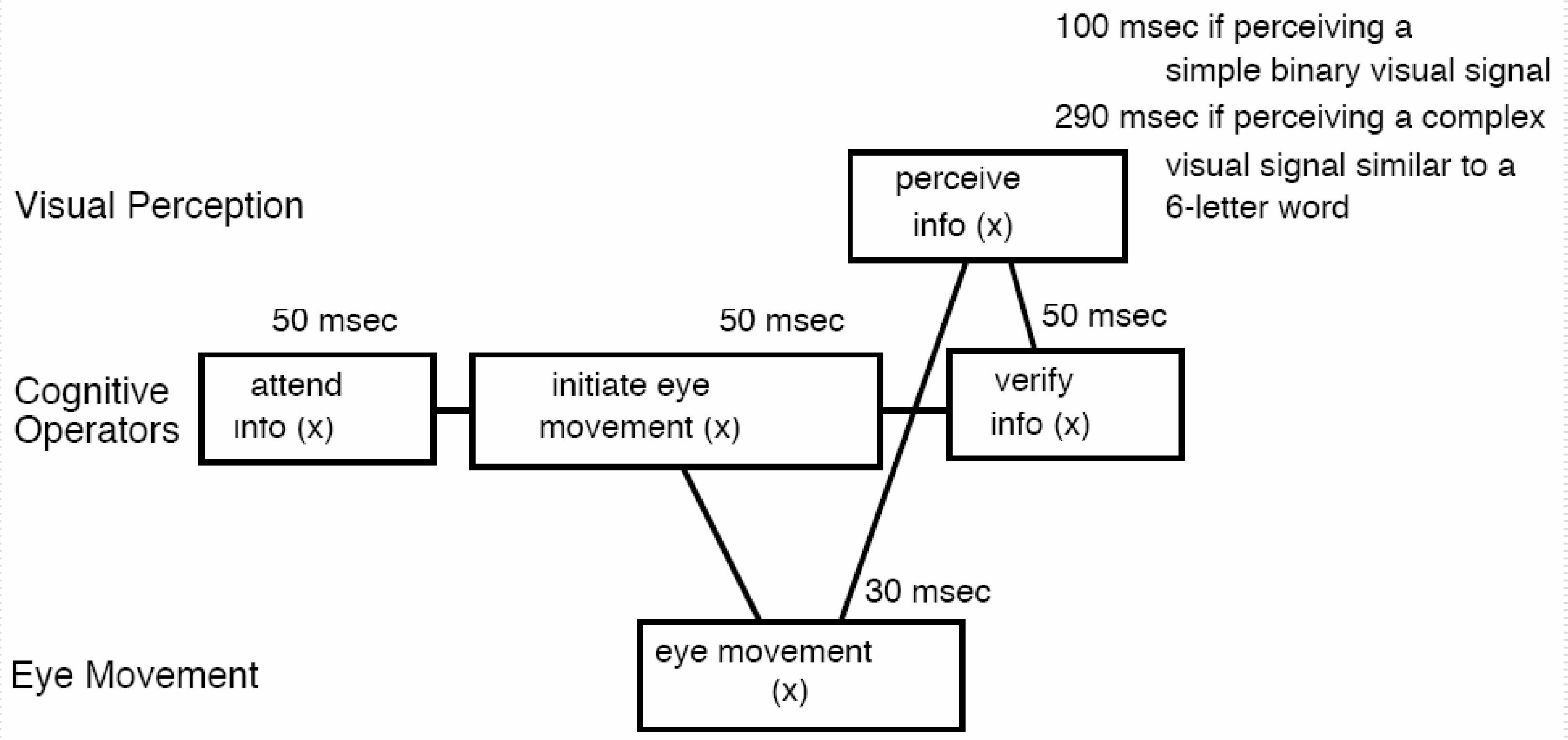

Production Memory
procedural knowledge
Long-Term Memory
semantic
infinite
infinite
70 ms cycle time
Cognitive Processor
Working Memory
acoustic
7-73 sec
3-7 chunks
Encoding
10 sec/chunk write time
Pattern Recognition
Pattern Recognition
Visual Image Store
physical
200 ms
17 letters
Auditory Image Store
physical
1500 ms
5 letters
Motor Processor
70 ms cycle time
Visual Processor
Auditory Processor
100 ms cycle time
Perceptual Processor
Muscle Control Output
Visual Input
Auditory Input

CPS-GOMS method

- Start with a task decomposition into basic activities, such as READ-SCREEN or ENTER-COMMAND.
- Express these activities in terms of Model Human Processor
- Use a task network to describe activities in MHP terms (many common activities as "templates‟, fragments of networks available)
- First assemble the activities sequentially, then attempt to interleave them where possible.
- Use chart tool to identify *critical path* and required execution time.
- Especially useful when concurrent perceptual/motor activity may be limiting execution speed.

Example template (John & Gray, 1995)

Goal: READ-SCREEN, when an eye-movement is required in the task.


100 msec if perceiving a simple binary visual signal
290 msec if perceiving a complex visual signal similar to a 6-letter word
Visual Perception
perceive info (x)
50 msec
50 msec
50 msec
Cognitive Operators
attend info (x)
initiate eye movement (x)
verify info (x)
30 msec
Eye Movement
eye movement (x)

NGOMSL (Natural GOMS Language) Models

High-level, natural language representation of a simple production-rule cognitive architecture.

Cognitive Complexity Theory (CCT), Kieras & Polson (1985)

- basic GOMS concept as *production-system*
  - Production-rule actions contain Keystroke-Level Model operators.
  - Considerable empirical validity shown: can predict both *learning* and *execution* time from the number of rules involved.
  - Assumes that GOMS methods are strictly hierarchical and sequential.
  - Production systems are too difficult for use in routine design.

NGOMSL Model

- Based on CCT research results, but suitable for practical use.
  - NGOMSL notation looks like an ordinary programming language.
  - One NGOMSL statement is basically one CCT production rule
  - Predicts relative learning time and execution time for specific design and task scenario
  - Especially sensitive to the consistency of the procedures.
  - Useful for many desktop interface design situations.

NGOMSL methodology

Top-down, breadth-first task decomposition

1. Start with the user's top-level goals.
2. Write a step-by-step procedure for accomplishing each goal in terms of subgoals and ultimately keystroke-level operators, using NGOMSL syntax
3. Write a selection rule to specify which method to use if more than one available for a goal.

Count number of statements in methods to predict *learning* time.

- ☐ Consistency reflected by presence of re-used submethods, reducing learning time.
- ☐ Similar methods also reduce learning time.

For a specific task scenario, count number of statements and operators executed to predict *execution* time.

- ☐ Not limited to specific sequences of actions.

Example: methods for recording a program with a VCR

Selection rule set for goal: record a program

- If you are present when the program starts and you will be present when the program ends, then accomplish goal: *record a program manually*
- If you are present when the program starts and you will not be present when the program ends and you know how long the program lasts and the VCR clock is set, then accomplish goal: *record a program with One-Touch Recording*
- If you will not be present when the program starts and you know when the program will start and you know how long the program lasts and the VCR clock is set, then accomplish goal: *record a program with Timer Recording*
- Return with goal accomplished

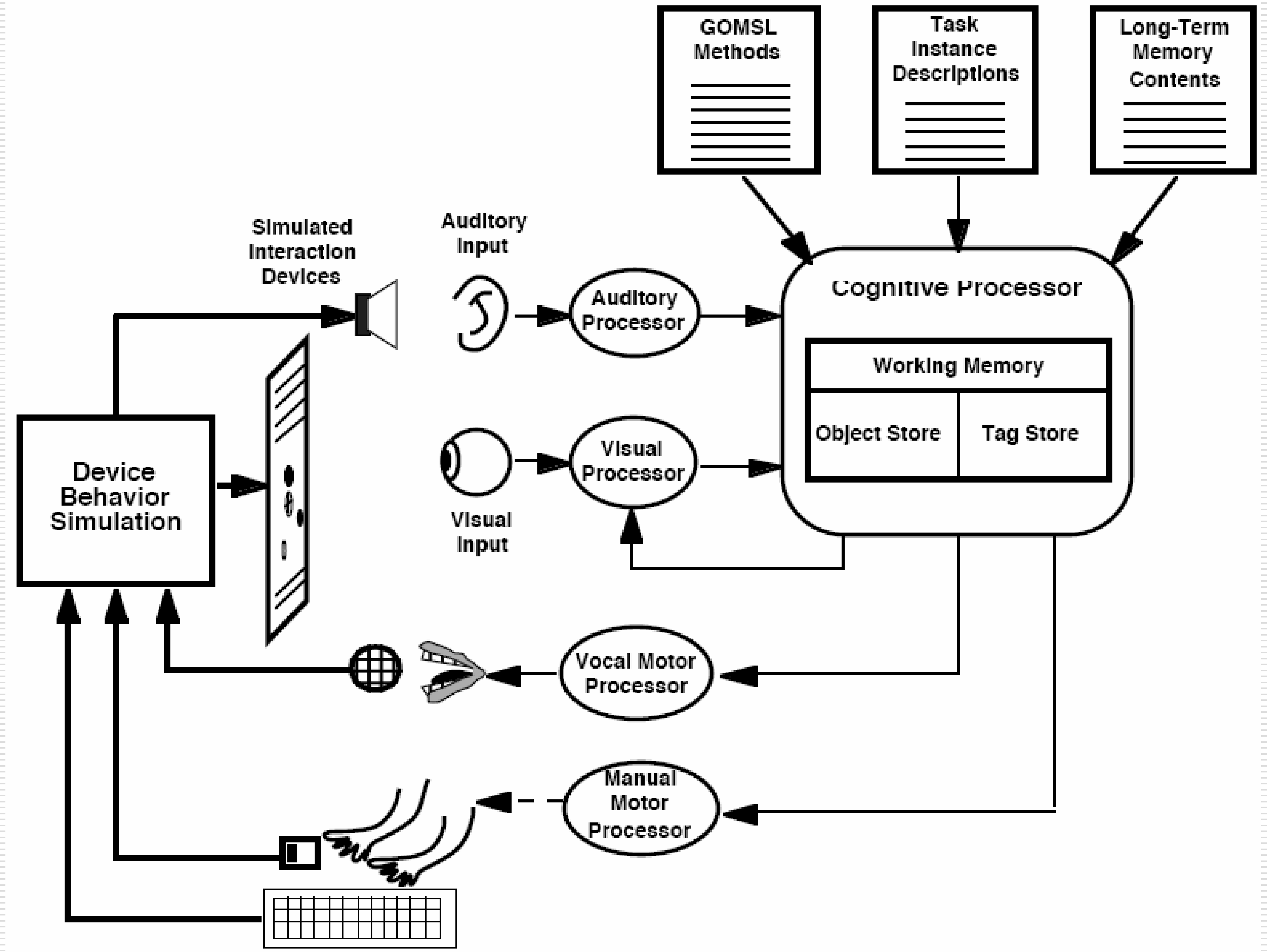
Example: methods for recording a program with a VCR (ctd)

Method for goal: *record a program manually*
Step 1. Wait for program to start.
Step 2. Hold down REC button.
Step 3. Press PLAY button.
Step 4. Release both buttons.
Step 5. Verify that display shows "REC" and arrow is moving
Step 6. Wait for program to end
Step 7. Press STOP button.
Step 8. Return with goal accomplished.

Method for goal: *record a program with One-Touch Recording*
Step 1. Wait for program to start
Step 2. Press OTR button.
Step 3. Press OTR button.
Step 4. Decide: If time shown<length of program, go to Step 3.
Step 5. Return with goal accomplished.

GOMSL and GLEAN

- *GOMS L(anguage)* **-** a formalized and executable version of NGOMSL.
- *GLEAN* **-** a simplified version of the EPIC simulation system (***G***OMS ***L***anguage ***E***valuation and ***An***alysis)
- Keystroke-Level model representation of perceptual & motor timing


GOMSL Methods
Task Instance Descriptions
Long-Term Memory Contents
Simulated Interaction Devices
Auditory Input
Auditory Processor
Cognitive Processor
Working Memory
Object Store
Tag Store
Device Behavior Simulation
Visual Processor
Visual Input
Vocal Motor Processor
Manual Motor Processor

Example of GOMSL

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

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

Steps in using GLEAN

1. Choose and represent the benchmark tasks.
2. Write the GOMS model entailed by the interface design.
3. Describe the device behavior as needed.
   - A passive "dummy" device may be adequate.
   - A scenario-driven interactive device may be required.
4. Debug the model by running it with GLEAN.
   - Look for correct and wrong task performance.
5. Obtain predictions by running final version of model.
   - Time-stamped actions on device, workload profiles.
6. Examine predictions, profiles, GOMS methods to identify problems and possible improvements.
7. Modify the design and GLEAN representations, and obtain new predictions.

Classification - analytic eval. techniques

	Cognitive walkthrough	*Heuristic evaluation*	*Model-based*
Stage	Throughout	Throughout	Design
Style	Lab	Lab	Lab
Objective?	No	No	No
Measure	Qualitative	Qualitative	Qualitative
Information	Low level	High level	Low level
Immediacy	N/A	N/A	N/A
Intrusive?	No	No	No
Time	Medium	Low	Medium
Equipment	Low	Low	Low
Expertise	High	Medium	High