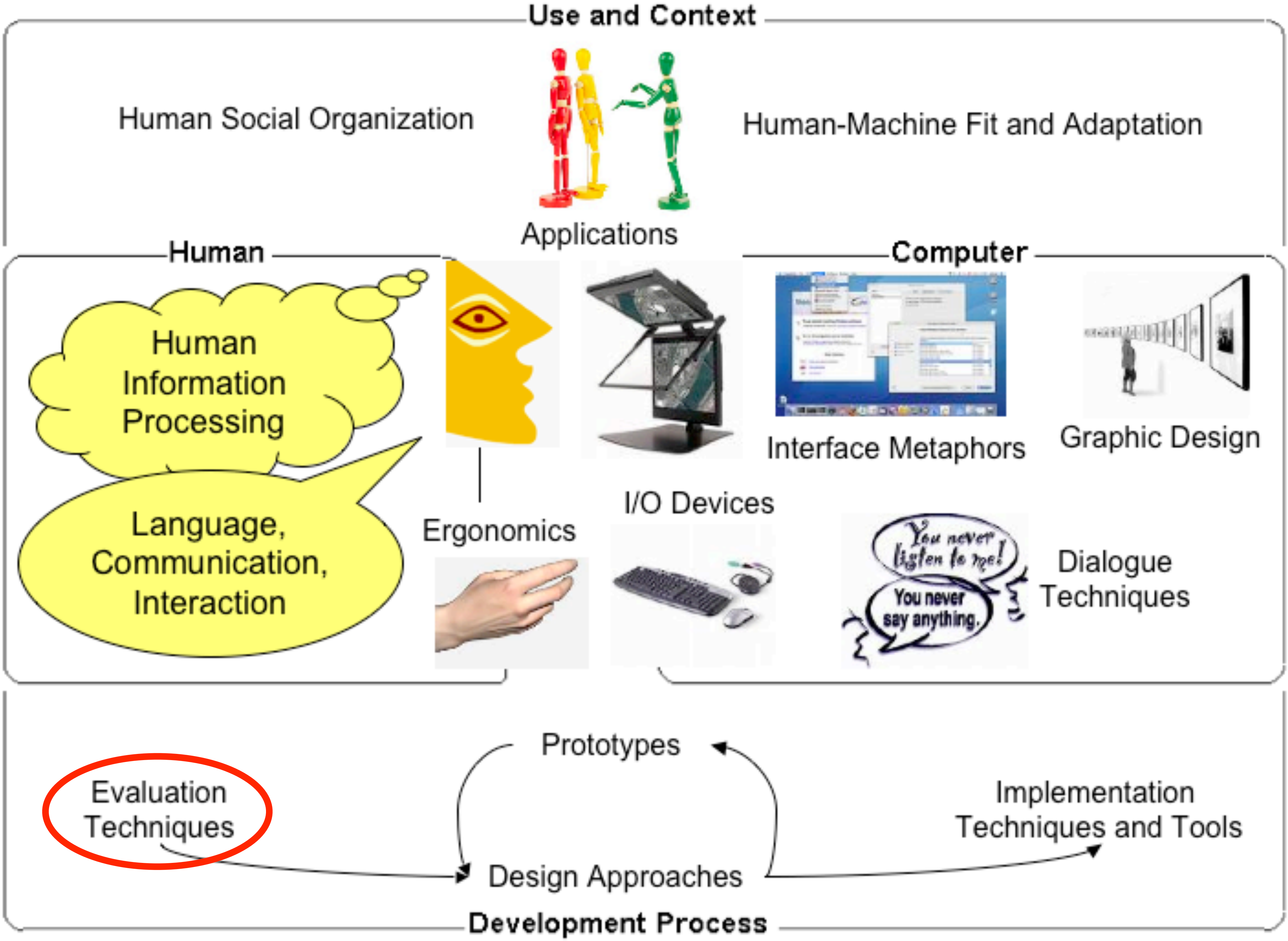
HCI Cognitive Models

Chapter 12

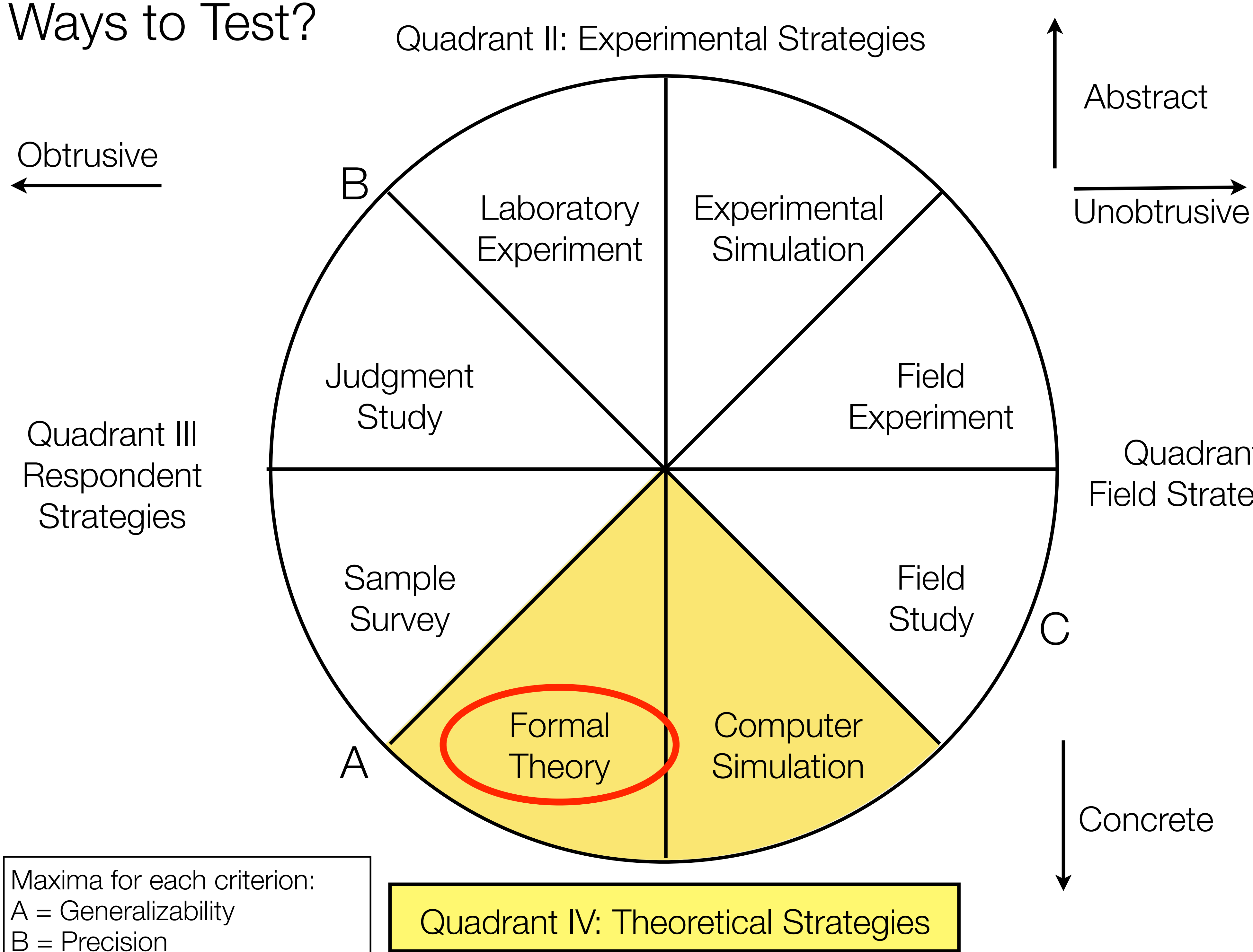
The slides in this lecture are partially based upon this from Dr. Vincent Ng and the course slides from Dix et al..

Lecture Overview

- What you will learn in this lecture
    - How do we formally model the way users interact with computers?
        - How they will complete a task
        - How long they will take


Use and Context
Human Social Organization
Human-Machine Fit and Adaptation
Applications
Human
Computer
Human Information Processing
Language, Communication, Interaction
Ergonomics
Interface Metaphors
Graphic Design
I/O Devices
You never listen to me!
You never say anything.
Dialogue Techniques
Prototypes
Evaluation Techniques
Design Approaches
Implementation Techniques and Tools
Development Process

Ways to Test?



Maxima for each criterion:
A = Generalizability
B = Precision
C = Realism

McGrath et al, 1994

Some terminology to start off with…

- Goal = external task, such as producing a letter
- Device = method, tool, or technique appropriate for achieving goals
- Tasks = activities necessary to achieve goals using a device
- Subtasks = components of tasks
- Actions = simple tasks w/ no control structure
- Method = plan = number of tasks or actions linked into a sequence

GOMS

- “The most mature engineering model of human performance”
- Models user behavior in terms of:
  - **Goals** that the user wishes to achieve.
  - **Operators** that the user performs.
  - **Methods** that are used to achieve **Goals** and sub**Goals**.
  - **Selection** rules that are invoked to choose between multiple **Methods**.

GOMS Approach: a Hierarchical Cognitive Model

- Hierarchical Model: Models mental processing as divide-and-conquer
- Example: Sales Report
    - produce report
    - gather data
    - .   find book names
    - .   .   do keywords search of names database
    - .   .   .   … further sub-goals
    - .   .   sift through names and abstracts by hand
    - .   .   .   … further sub-goals
    - .   search sales database - further sub-goals
    - layout tables and histograms - further sub-goals
    - write description - further sub-goals

GOMS Principles: Goals

- What the user is trying to accomplish
- Various levels of abstraction
  - High level: `DO-FINAL-PROJECT`
  - Low level: `DELETE-WORD`
- Higher level goals are decomposable into subgoals
- Hierarchical relation between goals and subgoals

GOMS Principles: Operators

- Elementary perceptual, motor or cognitive actions used to accomplish goals
  - E.g. `DOUBLE-CLICK_MOUSE` or `PRESS-INSERT-KEY`
- Atomic; not decomposable
- Assumptions:
  - Fixed amount of time is required to execute each operator
  - Execution time is independent of context
    - E.g. `CLICK-MOUSE-BUTTON` takes 0.20 seconds

GOMS Principles: Methods

- Algorithms that describe how to accomplish goals
  - Essentially, sequence of subgoals and operators.
- Example: `USE-MOUSE-DELETE-WORD` = `MOVE-MOUSE-TO-START-OF-WORD`, `PRESS-MOUSE`, `MOVE-MOUSE-TO-END-OF-WORD`, `RELEASE-MOUSE`, `PRESS-DELETE-KEY`
- Multiple methods may exist for the same goal.

GOMS Principles: Selection rules

- Specifies which method should be used to accomplish a given goal, based on the context.
- Represent user's knowledge of which method must be applied to achieve desired goal.
- Usually in form of conditional statement
- Example: "If word is longer than 10 characters, use the `USE-MOUSE-DELETE-WORD` method, otherwise, use the `USE-BACKSPACE-DELETE-WORD` method."

Simple GOMS Example: Close a window

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
            .   MOVE-MOUSE-TO-FILE-MENU
            .   PULL-DOWN-FILE-MENU
            .   CLICK-OVER-CLOSE-OPTION
            GOAL: USE-CTRL-W-METHOD
            .   PRESS-CONTROL-W-KEYS]

For a particular user:

   Rule 1: Select USE-MENU-METHOD unless
           another rule applies
   Rule 2: If the application is GAME,
           select CTRL-W-METHOD
```

Another GOMS Example: Mac Finder

- Method for goal: delete a file
  - Accomplish goal: drag file to trash.
  - Return with goal accomplished
- Method for goal: move a file
  - Accomplish goal: drag file to destination.
  - Return with goal accomplished.
- Method for goal: delete a directory
  - Accomplish goal: drag directory to trash.
  - Return with goal accomplished.
- Method for goal: move a directory
  - Accomplish goal: drag directory to destination
  - Return with goal accomplished
- Method for goal: drag item to destination
  - Locate icon for item on screen
  - Move cursor to item icon location
  - Hold mouse button down
  - Locate destination icon on screen
  - Move cursor to destination icon
  - Verify that destination icon is reverse-video
  - Release mouse button

Another GOMS Example: DOS

- Method for goal: delete a file.
  - Recall that command verb is "ERASE".
  - Think of directory name and file name and retain as first filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.
- Method for goal: move a file.
  - Accomplish goal: copy a file.
  - Accomplish goal: delete a file.
  - Return with goal accomplished.
- Method for goal: copy a file.
  - Recall that command verb is "COPY".
  - Think of source directory name and file name and retain as first filespec.
  - Think of destination directory name and file name and retain as second filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.
- Method for goal: delete a directory.
  - Accomplish goal: delete all files in the directory.
  - Accomplish goal: remove a directory.
  - Return with goal accomplished.
- Method for goal: delete all files in a directory.
  - Recall that command verb is "ERASE".
  - Think of directory name.
  - Retain directory name and "*.*" as first filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.
- Method for goal: remove a directory
  - Recall that command verb is "RMDIR".
  - Think of directory name and retain as first filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.
- Method for goal: move a directory.
  - Accomplish goal: copy a directory.
  - Accomplish goal: delete a directory.
  - Return with goal accomplished.
- Method for goal: copy a directory.
  - Accomplish goal: create a directory.
  - Accomplish goal: copy all the files in a directory.
  - Return with goal accomplished.
- Method for goal: create a directory.
  - Recall that command verb is "MKDIR".
  - Think of directory name and retain as first filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.
- Method for goal: copy all files in a directory.
  - Recall that command verb is "COPY".
  - Think of directory name.
  - Retain directory name and "*.*" as first filespec.
  - Think of destination directory name.
  - Retain destination directory name and "*.*" as second filespec.
  - Accomplish goal: enter and execute a command.
  - Return with goal accomplished.

DOS GOMS Example Continued

- Method for goal: enter and execute a command.
- *Entered with strings for a command verb and one or two filespecs.*
    1. Type command verb.
    2. Accomplish goal: enter first filespec.
    3. Decide: If no second filespec, goto 5.
    4. Accomplish goal: enter second filespec.
    5. Verify command.
    6. Type "<CR>".
    7. Return with goal accomplished.
- Method for goal: enter a filespec.
- *Entered with directory name and file name strings.*
    1. Type space.
    2. Decide: If no directory name, goto 5.
    3. Type "\".
    4. Type directory name.
    5. Decide: If no file name, return
    6. Type file name.
    7. Return with goal accomplished.

Comparison

- Mac Finder: Only 3 methods needed to accomplish all user goals.
  - Total number of steps involved: 18.
- DOS requires 12 methods with a total of 68 steps.

➡ Mac Finder is more consistent than DOS.

- Major value of GOMS: ability to characterize and quantify property of *method consistency*.

Keystroke-Level Model (KLM)

- Simplified version of GOMS
- Proposed as method for predicting user performance.
- Execution time estimated by listing sequence operators and then summing times of individual operators.
  - Targets *execution* phase of problem solving
  - Does not employ selection rules.
  - Based on knowledge of human motor system

Operations in KLM

- Motor (physical) operators:
  - K: pressing a key
  - B: pressing a mouse button
  - P: pointing to a location on screen with mouse
  - H: moving hands to home position on the keyboard.
  - D: Drawing a line with the mouse
- M: mental preparation prior to performing an action
- R: System response where user needs to wait

Time Estimates

Operator	Description	Time (s)
K	Time varies with typing skill	
	Best Typist (135 wpm)	0.08
	Good Typist (90 wpm)	0.12
	Average Typist (40 wpm)	0.20
	Non Typist	1.20
B	Down or up	0.10
	Click	0.20
P	Point with Mouse (average)	1.10
H	Move hands to “Home” on keyboard	0.40
$D(n_D,l_D)$	Draw Line Segment	$0.9n_D+0.16l_D$
M	Mentally prepare	1.35

KLM Execution Time

- Each operation takes time, therefore:
  - $T_{execute} = T_K + T_B + T_P + T_H + T_D + T_M + T_R$

KLM Example

- How long does it take to close a window?

```
GOAL: CLOSE-WINDOW
.    [select GOAL: USE-MENU-METHOD
            .    MOVE-MOUSE-TO-FILE-MENU
            .    PULL-DOWN-FILE-MENU
            .    CLICK-OVER-CLOSE-OPTION
            GOAL: USE-CTRL-W-METHOD
            .    PRESS-CONTROL-W-KEYS]
```

Example

- Assume hand starts on mouse

USE-MENU-METHOD		USE-CTRL-W-METHOD	
Operator	**Time**	**Operator**	**Time**
P [to menu]	1.1	H [to kbd]	0.40
B [LEFT down]	0.1	M	1.35
M	1.35	2K [Ctrl-W]	0.24
P [to “close”]	1.1		
B [LEFT up]	0.1		
Total	3.75	**Total**	1.99

Subtleties: The M Operator

- M: Mental preparation prior to performing an action.
  - Before each action? Every single one?
- General rule for when to apply this operator: before every chunk.

Subtleties: The M Operator

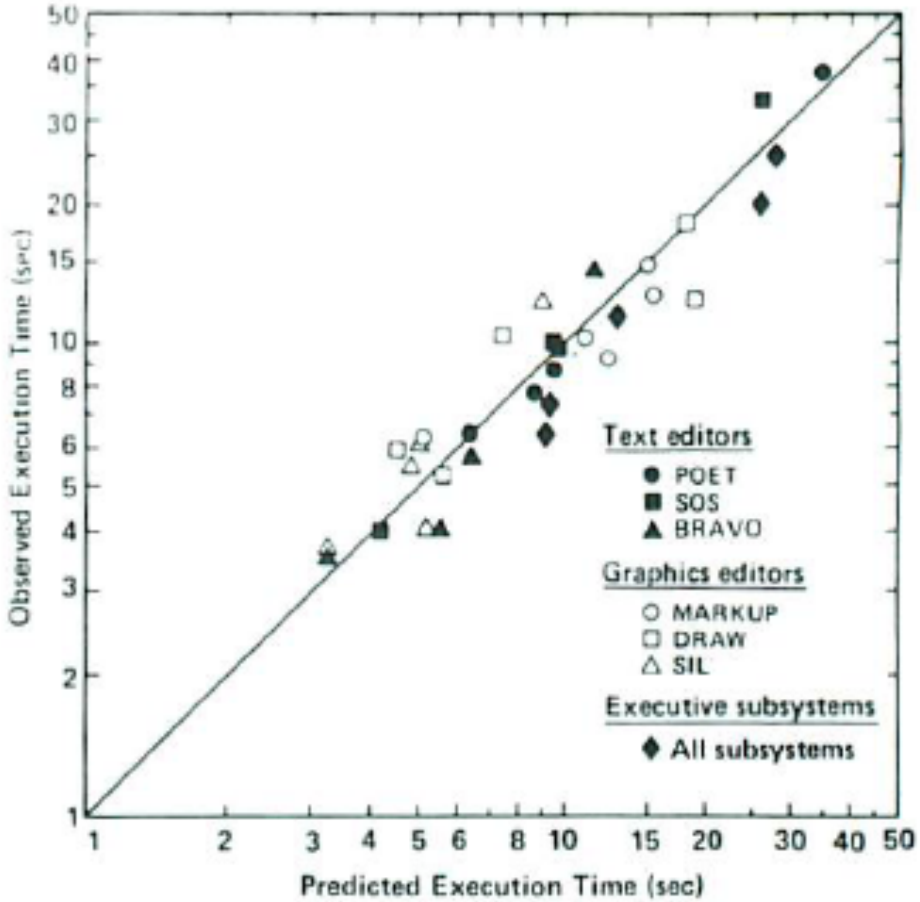
- General rule of thumb:
- If the user types a word, or a well-known command name, then this constitutes a chunk, and only one M operator is needed.
- If, however, the user is typing out an acronym which must be recalled letter by letter, we need one M operator per letter.
- Obviously will depend on operator skill
- Need to decide what sort of user we're modeling.

Some issues with GOMS goal hierarchies

- Granularity
  - Where do we start?
  - Where do we stop?
- Routine learned behavior, not problem solving
- Conflict
  - More than one way to achieve a goal
- Error

Accuracy of GOMS and KLM

- Experiments show GOMS and KLM predictions of execution time accurate to about 20%



Source: Card, Stu. Lecture on Human Information Interaction. Stanford, 2007.

Applicability of GOMS and KLM

- Predictive
  - Used to predict time needed to perform tasks
- Descriptive
  - Representation of how tasks are performed.
  - Good for checking consistency
- Prescriptive
  - Serves as a guide for developing training programs and help systems.
  - Models that are developed can be used to teach new users.

Applications of GOMS and KLM

- Given several systems, how do we decide which one is the most efficient?
- Perform a GOMS and KLM analysis on key tasks, and see which system is fastest!
- Much faster than doing experiments
- Can even be done at the prototyping level — the systems don't even need to exist!
- Can also compare methods to perform certain tasks within a system.
- Good for preparing training materials — find out faster methods, and teach only those.

Case Study: Example Application of GOMS

- NYNEX computer system for telephone operators.
- GOMS analysis was performed prior to installation of new system
  - Determined critical path to complete tasks
  - Analyzed time to traverse that path
- Result: new system would take longer to process each call
- System was abandoned before installation.

(From Dix, Finlay et al., 3rd edition, Page 424)

GOMS Limitations

- Most significant faults:
  - Predictions valid only for expert users
    - Does not take correcting for errors into account.
    - But even expert users commit errors from time to time.
  - Does not model learning curve of novices and occasional users.
- Remember: Major objective of HCI: aim for **maximum usability** for **all** users!

GOMS Limitations

- All tasks are modeled as goal-directed.
  - But some tasks are more problem-solving in nature.
- Does not take user differences into account (apart from operator time values)
- Considers only execution speed
  - Does not give insight into how useful or enjoyable the product will be.

GOMS Limitations

- Not representative of current theories of human cognition
- Serial model: activities are done one by one
  - Many models of cognitive psychology think otherwise.
- Therefore: GOMS is useful as an engineering heuristic
  - Not as a model of cognitive processes.

Linguistic Models

- Objective: understand the user's behavior and cognitive difficulty based on analysis of language between user and system.
- Similar in emphasis to dialogue models
- Example: BNF

BNF

- Stands for either Backus-Naur Form or Backus Normal Form
- Used to describe the grammar of a formal language
- Formal and precise

BNF

- Very common notation from computer science
- A purely syntactic view of the dialog
- Terminals:
  - Lowest level of user behavior
  - e.g. CLICK-MOUSE, MOVE-MOUSE
- Nonterminals:
  - Ordering of terminals
  - Higher level of abstraction
  - e.g. Select-menu, position-mouse

BNF Notation

- < > indicate *nonterminals* that needs to be further expanded
  - e.g. `<variable>`
- Symbols not enclosed in < > are *terminals*; they represent themselves
  - e.g. `if`, `while`, `(`
- The symbol `::=` means "*is defined as*"
- The symbol `|` means *or*; it separates alternatives
  - e.g. `<addop> ::= + | -`

Recursion in BNF

- Recursive elements can be represented in BNF:

- `<integer> ::= <digit> | <integer> <digit>`
  or
  `<integer> ::= <digit> | <digit> <integer>`

- “Extended BNF” or EBNF allows repetition as well as recursion

BNF Examples

- What is a number?
  - `<digit> ::=`
    `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

- What is a conditional statement?
  - ```
    <if statement> ::=
              if ( <condition> ) <statement>
            | if ( <condition> ) <statement>
              else <statement>
    ```
    ```

More BNF Examples

```
• <unsigned integer> ::=
        <digit> | <unsigned integer> <digit>

• <integer> ::=
         <unsigned integer>
       | + <unsigned integer>
       | - <unsigned integer>
```

More BNF Examples

- From programming:

- <identifier> ::=

```
        <letter>
      | <identifier> <letter>
      | <identifier> <digit>
```

- <block> ::= { <statement list> }

- <statement list> ::=

```
        <statement>
      | <statement list> <statement>
```

More BNF Examples

-
```
<statement> ::=
          <block>
        | <assignment statement>
        | <break statement>
        | <continue statement>
        | <do statement>
        | <for loop>
        | <goto statement>
        | <if statement>
        | . . .
```

BNF Example

- For Computer Interfaces:
- Examples:

draw line ::= select line + choose points + last point

select line ::= position mouse + CLICK MOUSE

choose points ::= choose one | choose one + choose points

choose one ::= position mouse + CLICK MOUSE

last point ::= position mouse + DBL CLICK MOUSE

position mouse ::= NULL | MOVE MOUSE+ position mouse

Use specification to work out how many basic actions are required for a task

Measurements with BNF

- Number of rules
  - The more rules, the more complex the system
  - But depends on how we specify the rules (can cheat by using lots of “|” operators)
- Number of + and | operators
- Number of basic actions to complete a task.

Constraints of BNF

- BNF is one of the most commonly used ways of defining grammars and dialogs
- Also: nothing clearly better is available at the moment.
- But: has some constraints:
  - No easy way to do counting (what is the maximum length of a command?)
  - No way to impose context-dependent constraints, such as a variable must be declared before it is used.
  - Same syntax for different semantics
    - Compare select-point and choose-one
    - No reflection of user's perception, only user's actions are represented

Constraints of BNF (cont'd)

- Describes only syntax, not semantics (how to do a command, rather than what it means)
    - Does not check for consistency
        - E.g. Take the three UNIX commands: copy (cp), move (mv), link (ln)
        - copy ::= cp + filename + filename | cp + filenames + directory
        - move ::= mv + filename + filename | mv + filenames + directory
        - link ::= ln + filename + filename | ln + filenames + directory
        - Nothing to stop us from declaring an inconsistent link definition:
            - link ::= ln + filename + filename | ln + directory + filenames

Extended BNF

- EBNF inserts elements of regular expressions into BNF
  - `[ ]` enclose an optional part of the rule
    - Example:
```
<if statement> ::=
   if ( <condition> ) <statement> [ else <statement> ]
```
  - `{ }` mean the enclosed can be repeated any number of times (including zero)
    - Example:
```
<parameter list> ::= ( )
                  |   ( { <parameter> , } <parameter> )
```

Other Models: TAG

- Task Action Grammar (TAG)
- Linguistic Model
- Makes consistency more explicit
- Encodes user’s world knowledge
- Parameterized grammar rules
- Nonterminals are modified to include additional semantic features.

TAG

- Tries to resolve the consistency problem in BNF
- Recall the three UNIX commands consistency problem.
- General command:
  command-name + filename + filename |
  command-name + filenames + directory
- BNF cannot check for one rogue command definition that doesn't follow this rule.

Consistency in TAG

- TAG makes this argument order explicit by using a parameter, or a semantic feature:
- e.g. For the file operations, possible values for the feature would be:

```
Op = copy; move; link
```

- Rules would then be:

```
file-op[Op] ::= command[Op] + filename + filename
              | command[Op] + filenames + directory

command[Op = copy] ::= cp

command[Op = move] ::= mv

command[Op = link] ::= ln
```

Other uses of TAG

- Can incorporate user's existing knowledge
- Can model congruence between features and commands
- All these are modeled as derived rules.

CCT Example: Editing with vi

- Background:
- vi is an old, but very well known, text editor.
  - Completely text-based
  - No mouse input, all keyboard.

CCT Example: Editing with vi

- Scenario: In a document, the user typed in “computerscience” instead of “computer science”.
- Mistake is in Line 5, the space should be at character 23.
- We need to get to that location, then enter in a space.

CCT Example: Editing with vi

- CCT Model:
  - Production rules are in long-term memory
  - Model contents of working memory as attribute-value mapping

```
(GOAL perform unit task)

(TEXT task is insert space)

(TEXT task is at 5 23)

(CURSOR 8 7)
```

Four possible ways (rules) to inserting a space

```
(SELECT-INSERT-SPACE
IF (AND (TEST-GOAL perform unit task)
        (TEST-TEXT task is insert space)
        (NOT (TEST-GOAL insert space))
        (NOT (TEST-NOTE executing insert space)))
THEN (  (ADD-GOAL insert space)
        (ADD-NOTE executing insert space)
        (LOOK-TEXT task is at %LINE %COLUMN)))
```

SELECT-INSERT-SPACE
INSERT-SPACE-MOVE-FIRST
INSERT-SPACE-DOIT
INSERT-SPACE-DONE

CCT Example: Editing with vi

- Production rules are pattern-matched with mapping in working memory:

e.g., LOOK-TEXT task is at %LINE %COLUMN
is true, with LINE = 5 COLUMN = 23.

Four rules to model inserting a space

Active rules:

SELECT-INSERT-SPACE
INSERT-SPACE-MOVE-FIRST
INSERT-SPACE-DOIT
INSERT-SPACE-DONE

New working memory

(GOAL insert space)
(NOTE executing insert space)
(LINE 5) (COLUMN 23)

SELECT-INSERT-SPACE
matches current working memory

```
(SELECT-INSERT-SPACE
IF (AND (TEST-GOAL perform unit task)
        (TEST-TEXT task is insert space)
        (NOT (TEST-GOAL insert space))
        (NOT (TEST-NOTE executing insert space)))
THEN (  (ADD-GOAL insert space)
        (ADD-NOTE executing insert space)
        (LOOK-TEXT task is at %LINE %COLUMN)))
```

Notes on CCT

- Parallel Model
- Proceduralization of actions
- Novice vs. Expert style rules
- Error behavior can be represented
- Measures
  - Depth of goal structure
  - Number of rules
  - Comparison with device description

Models (In Summary)

- All of these cognitive models make assumptions about the architecture of the human mind.
    - As such, they are called *architectural models*.
- Long-term/short-term memory
- Problem spaces
- Interacting cognitive subsystems

Models (in Summary)

- Most cognitive models do not take into account user observation and perception
- Some techniques have been extended to handle system output
  - BNF with sensing terminals, Display-TAG