UNIT - II

Model Based Design and Evaluation

Evaluation: tests the usability, functionality and acceptability of an interactive system.

- Occurs in laboratory, field/or in collaboration with users.
- Evaluate both design and implementation.
- Should be considered at all stage in the design life cycle.

Evaluation may be done by:

1. The designer for assessing early design and prototypes.
2. Usability expert of the working prototype or implementation

GOALS OF EVALUATION

1. To assess the extent and accessibility of the system's functionality.
2. To assess users' experience of the interaction.
3. To identify any specific problems with the system.

Evaluation through expert analysis

4 approaches to expert analysis considered:

1. Cognitive Walkthrough
2. Heuristic evaluation
3. The use of model
4. Use of previous work

In ISLC, we have learned about the process involved in interactive system design. We have learned that interactive systems are designed following the ISLC.

- Consisting of the stages for requirement identification, design, prototyping and evaluation
- Highly iterative

The iterative life cycle is time consuming and also requires money (for coding and testing). It is always good if we have an alternative method that reduces the time and effort required for the design life cycle.

Model-based design provides one such alternatives, suppose you are trying to design an interactive System. First, you should identify requirements ("know the user") following methods such as contextual inquiry:

- Time consuming and tedious process
- Instead of going through the process, it would have been better if we have a "model of the user

Idea of a Model

A 'model' in HCI refers to "a representation of the user's interaction behavior under certain assumptions" The representation is typically obtained from extensive empirical studies (collecting and analyzing data from end users).

- The model represents behavior of average users, not individuals by encompassing information about user behavior, a model helps in alleviating the need for extensive requirement identification process.
- Such requirements are already known from the model once the requirements have been identified designer 'propose' design(s) typically, more than one designs are proposed.
- The competing designs need to be evaluated

This can be done by evaluating either prototypes (in the early design phase) or the full system (at the final stages of the design) with end users. End user evaluation is a must in user centered design. Like requirement identification stage, the continuous evaluation with end users is also money and time consuming. If we have a model of end users as before, we can employ the model to evaluate design.

- Because the model already captures the end user characteristics, no need to go for actual users.
- A model is assumed to capture behavior of an average user of interactive system.
- User behavior and responses are what we are interested in knowing during ISLC Thus by using models, we can fulfill the key requirement of interactive system design (without actually going to the user).
- Saves lots of time, effort and money.

Types of Models

Two broad categorization of the models used in HCI

- **Descriptive/prescriptive models:** some models in HCI are used to explain/describe user behavior during interaction in qualitative terms. An example is the Norman's model of interaction (to be discussed in a later lecture). These models help in formulating (prescribing) guidelines for interface design
- **Predictive engineering models:** these models can "Predict" behavior of a user in quantitative terms. An example is the GOMS model (to be discussed later in this module), which can predict the task completion time of an average user for a given system.

Predictive Engineering Models

The predictive engineering models used in HCI are of three types

1. Formal (system) model
2. Cognitive (user) models
3. Syndetic (hybrid) model

1. Formal (System) Model

In these models, the interactive system (interface and interaction) is represented using 'formal specification' techniques.

For example, the interaction modeling using state transition networks.

Essentially models of the 'external aspects' of interactive system (what is seen from outside) Interaction is assumed to be a transition between states in a 'system state space'

- A 'system state' is characterized by the state of the interface (what the user sees) It is assumed that certain state transitions increase usability while the others do not.
- The models try to predict if the proposed design allows the users to make usability-enhancing transitions.
- By applying 'reasoning' (manually or using tools) on the formal specification

2. **Cognitive (User) Models**

These models capture the user's thought (cognitive) process during interaction
For example, a GOMS model tells us the series of cognitive steps involved in typing a word
Essentially models are the 'internal aspects' of interaction (what goes on inside user's mind) Usability is assumed to depend on the 'complexity' of the thought process (cognitive activities)

- Higher complexity implies less usability Cognitive activities involved in interacting with a system is assumed to be composed of a series of steps (serial or parallel).
- More the number of steps (or more the amount of parallelism involved), the more complex the cognitive activities are.
- The models try to predict the number of cognitive steps involved in executing 'representative' tasks with the proposed designs.
- Which leads to an estimation of usability of the proposed design.

3. **Syndetic (Hybrid) Model**

HCI literature mentions one more type of model, called 'Syndetic' model. In this model, both the system (external aspect) and the cognitive activities (internal aspect) are combined and represented using formal specification

Cognitive Models in HCI

- Although we said before that cognitive models are models of human thinking process, they are not exactly treated as the same in HCI.
- Since interaction is involved, cognitive models in HCI not only model human cognition (thinking) alone, but the perception and motor actions also (as interaction requires 'perceiving what is in front' and 'acting' after decision making).

Thus cognitive models in HCI should be considered as the models of human perception (perceiving the surrounding), cognition (thinking in the 'mind') and motor action (result of thinking such as hand movement, eye movement etc.

In HCI, broadly three different approaches are used to model cognition

1. Simple models of human information processing
2. Individual models of human factors
3. Integrated cognitive architectures

1. **Simple Models of Human Information Processing**

These are the earliest cognitive models used in HCI. These model complex cognition as a series of simple (primitive/atomic) cognitive steps:

- Most well-known and widely used models based on this approach is the GOMS family of models.
- Due to its nature, application of such models to identify usability issues is also known as the "cognitive task analysis (CTA)"

2. **Individual Models of Human Factors**

In this approach, individual human factors such as manual (motor) movement, eye movement, decision time in the presence of visual stimuli etc. are modeled.

- The models are basically analytical expressions to compute task execution times in terms of interface and cognitive parameters.
- Examples are the Hick-Hyman law, the Fitts' law

3. **Integrated Cognitive Architectures**

Here, the whole human cognition process (including perception and motor actions) is modeled.

- Models capture the complex interaction between different components of the cognitive mechanism unlike the first approach.
- Combines all human factors in a single model unlike the second approach.
- Examples are MHP, ACT-R/PM, Soar

Model-based Design Limitations

As we mentioned before, model-based design reduce the need for real users in ISLC:

- However, they cannot completely eliminate the role played by real users.
- We still need to evaluate designs with real users, albeit during the final stages.
- Model-based design can be employed in the initial design stages.
- The present models are not complete in representing average end user (they are very crude approximations only)
- The models cannot capture individual user characteristics (only models average user behavior)
  As we mentioned, the particular type of predictive engineering model that we shall be dealing with in this module are the "**simple models of human information processing**"
- GOMS family of models is the best known examples of the above type.
- GOMS stands for **G**oals, **O**perators, **M**ethods and **S**election Rules

The family consists of FOUR models:

- **K**eystroke **L**evel **M**odel or KLM
- Original GOMS proposed by **C**ard, **M**oran and **N**ewell, popularly known as CMN-GOMS.
- **N**atural GOMS **L**anguage or NGOMSL.
- **C**ognitive **P**erceptual **M**otor or (CPM) GOMS [also known as **C**ritical **P**ath **M**ethod GOMS]

Keystroke Level Model (KLM)

The model was proposed way back in 1980 by Card, Moran and Newell; retains its popularity even today This is the earliest model to be proposed in the GOMS family (and one of the first predictive models in HCI)

KLM - Purpose

The model provides a quantitative tool (like other predictive engineering models)

- The model allows a designer to 'predict' the time it takes for an average user to execute a task using an interface and interaction method.
  For example, the model can predict how long it takes to close this PPT using the "close" menu option.

How KLM Works

- In KLM, it is assumed that any decision-making task is composed of a series of 'elementary' cognitive (mental) steps that are executed in sequence. These 'elementary' steps essentially represent low-level cognitive activities, which cannot be decomposed any further.
- The method of breaking down a higher-level cognitive activity into a sequence of elementary steps is simple to understand, provides a good level of accuracy and enough flexibility to apply in practical design situations.

The Idea of Operators

To understand how the model works, we first have to understand this concept of 'elementary' cognitive steps:

- These elementary cognitive steps are known as ***operators.***
  For example, a key press, mouse button press and release etc.
- Each operator takes a pre-determined amount of time to perform.
- The operator times are determined from empirical data (i.e., data collected from several. users over a period of time under different experimental conditions).
- That means, operator times represent average user behavior (not the exact behavior of an individual).
- The empirical nature of operator values indicate that, we can predict the behavior of average user with KLM.
- The model cannot predict individual traits.

There are seven operator defined, belonging to three broad groups

1. Physical (motor) operators
2. Mental operator
3. System response operator

Physical (Motor) Operators

There are five operators that represent five elementary motor actions with respect to an interaction

Operator	Description
K	The motor operator representing a key-press
B	The motor operator representing a mouse-button press or release
P	The task of pointing (moving some pointer to a target)
H	Homing or the task of switching hand between mouse and keyboard
D	Drawing a line using mouse (not used much nowadays)

Mental Operator

Unlike physical operators, the core thinking process is represented by a single operator **M**, known as the "mental operator" Any decision-making (thinking) process is modeled by M.

System Response Operator

KLM originally defined an operator **R**, to model the system response time (e.g., the time between a key press and appearance of the corresponding character on the screen). When the model was first proposed (1980), R was significant.

However, it is no longer used since we are accustomed to almost instantaneous system response, unless we are dealing with some networked system where network delay may be an issue.

Operator Times

As we mentioned before, each operator in KLM refers to an elementary cognitive activity that takes a pre-determined amount of time to perform.

The times are shown below (excluding the times for the operators D which is rarely used and R, which is system dependent)

Physical (Motor) Operator Times

Operator	Description	Time (second)
K **(**The key press operator**)**	Time to perform K for a good (expert) typist	0.12
	Time to perform K by a poor typist	0.28
	Time to perform K by a non-typist	1.20

Operator	Description	Time (second)
B **(**The mouse- button press/release operator**)**	Time to press or release a mouse-button	0.10
	Time to perform a mouse click (involving one press followed by one release)	2×0.10 = 0.20

Operator	**Description**	**Time (second)**
P **(**The pointing operator**)**	Time to perform a pointing task with mouse	1.10
H (the homing operator)	Time to move hand from/to keyboard to/from mouse	0.40

Operator	**Description**	**Time (second)**
M **(**The mental operator**)**	Time to mentally prepare for a physical action	1.35

How KLM Works

In KLM, we build a model for task execution in terms of the operators. That is why KLM belongs to the cognitive task analysis (CTA) approach to design. For this, we need to choose one or more representative task scenarios for the proposed design.

Next, we need to specify the design to the point where keystroke (operator)-level actions can be listed for the specific task scenarios. Then, we have to figure out the best way to do the task or the way the users will do it.

Next, we have to list the keystroke-level actions and the corresponding physical operators involved in doing the task. If necessary, we may have to include operators when the user must wait for the system to respond (as we discussed before, this step may not be ignored most of the times for modern-day computing systems).

In the listing, we have to insert mental operator M when user has to stop and think (or when the designer feels that the user has to think before taking next action)

Once we list in proper sequence all the operators involved in executing the task, we have to do the following:

- Look up the standard execution time for each operator.
- Add the execution times of the operators in the list.
- The total of the operator times obtained in the previous step is "the time estimated for an average user to complete the task with the proposed design".
- If there are more than one design, we can estimate the completion time of the same task with the alternative designs.
- The design with least estimated task completion time will be the best.

- Although there are a total of seven original operators, two (D and R) are not used much nowadays
- Any task execution with an interactive system is converted to a list of operators
- When we add the execution times of the operators in the list, we get an estimate of the task execution time (by a user) with the particular system, thereby getting some idea about the system performance from user's point of view.
- The choice of the task is important and should represent the typical usage scenario.

Example 1:

Suppose a user is writing some text using a text editor program. At some instant, the user notices a single character error (i.e., a wrong character is typed) in the text. In order to rectify it, the user moves the cursor to the location of the character (using mouse), deletes the character and retypes the correct character. Afterwards, the user returns to the original typing position (by repositioning the cursor using mouse). Calculate the time taken to perform this task (error rectification) following a KLM analysis.

Building KLM for the Task

- To compute task execution time, we first need to build KLM for the task.
- That means, listing of the operator sequence required to perform the task.

Let us try to do that step-by-step

Step 1: user brings cursor to the error location.

To carry out step 1, user moves mouse to the location and 'clicks' to place the cursor there

Operator level task sequence

Description	Operator
Move hand to mouse	H
Point mouse to the location of the erroneous character	P
Place cursor at the pointed location with mouse click	BB

Step 2: user deletes the erroneous character

–Switches to keyboard (from mouse) and presses a key (say "Del" key)

Operator level task sequence

Description	Operator
Return to keyboard	H
Press the "Del" key	K

Step 3: user types the correct character

–Presses the corresponding character key

Operator level task sequence

Description	Operator
Press the correct character key	K

Step 4: user returns to previous typing place
–Moves hand to mouse (from keyboard), brings the mouse pointer to the previous point of typing and places the cursor there with mouse click

Operator level task sequence

Description	Operator
Move hand to mouse	H
Point mouse to the previous point of typing	P
Place cursor at the pointed location with mouse click	BB

Total execution time (T) = the sum of all the operator times in the component activities

T = HPBBHKKHPBB = 6.20 seconds

Step	Activities	Operator Sequence	Execution Time (sec)
1	Point at the error	HPBB	0.40+1.10+0.20 = 1.70
2	Delete character	HK	0.40+1.20 = 1.60
3	Insert right character	K	1.20
4	Return to the previous typing point	HPBB	1.70

Something Missing!!

- **What about M (mental operator) –where to place them in the list?**

➢ It is usually difficult to identify the correct position of M
–However, we can use some guidelines and heuristics

General Guidelines

- **Place an M whenever a task is initiated**
- **M should be placed before executing a strategy decision**
  - ➢ If there is more than one way to proceed, and the decision is not obvious or well-practiced, but is important, the user has to stop and think.
- **M is required for retrieving a chunk from memory**
  - ➢ A chunk is a familiar unit such as a file name, command name or abbreviation.
  - ➢ Example -the user wants to list the contents of directory foo; it needs to retrieve two chunks - dir(command name) and foo(file name), each of which takes an M.
- **Other situations where M is required**
  - ➢ Trying to locate something on the screen (e.g., looking for an image, a word).
  - ➢ Verifying the result of an action (e.g., checking if the cursor appears after clicking at a location in a text editor).
  - ➢ Consistency –be sure to place M in alternative designs following a consistent policy.
  - ➢ Number of Ms –total number of M is more important than their exact position.
  - ➢ Explore different ways of placing M and count total M in each possibility.
  - ➢ Apples & oranges –don't use same policy to place M in two different contexts of interaction

Example: don't place M using the same policy while comparing between menu driven word processing (MS Word) vs command driven word processing (Latex)

Yellow pad heuristics

If the alternative designs raises an apples & oranges situation then consider removing the mental activities from action sequence and assume that the user has the results of such activities easily available, as if they were written on a yellow pad in front of them.

M Placement Heuristics

- **Rule 0: initial insertion of candidate Ms**
  - Insert Ms in front of all keystrokes (K).
  - Insert Ms in front of all acts of pointing (P) that select commands.
  - Do not insert Ms before any P that points to an argument.
  - Mouse-operated widgets (like buttons, check boxes, radio buttons, and links) are considered commands.
  - Text entry is considered as argument.
- **Rule 1: deletion of anticipated Ms**
  - If an operator following an M is fully anticipated in an operator immediately preceding that M, then delete the M.

    Example -if user clicks the mouse with the intention of typing at the location, then delete the M inserted as a consequence of rule 0.

    –So BBMK becomes BBK
- **Rule 2: deletion of Ms within cognitive units**
  - If a string of MKs belongs to a cognitive unit then delete all Ms except the first.
  - A cognitive unit refers to a chunk of cognitive activities which is predetermined.

    Example -if a user is typing "100", MKMKMK becomes MKKK (since the user decided to type 100 before starts typing, thus typing 100 constitutes a cognitive unit).
- **Rule 3: deletion of Ms before consecutive terminators**
  - If a K is a redundant delimiter at the end of a cognitive unit, such as the delimiter of a command immediately following the delimiter of its argument, then delete the M in front of it.

    Example: when typing code in Java, we end most lines with a semi-colon, followed by a carriage return. The semi-colon is a terminator, and the carriage return is a redundant terminator, since both serve to end the line of code.
- **Rule 4: deletion of Ms that are terminators of commands**
  - If a K is a delimiter that follows a constant string, a command name (like "print"), or something that is the same every time you use it, then delete the M in front of it.
  - If a K terminates a variable string (e.g., the name of the file to be printed, which is different each time) then leave it.

- **Rule 5: deletion of overlapped Ms**

- Do not count any portion of an M that overlaps a R —a delay, with the user waiting for a response from the computer.
  Example: user is waiting for some web page to load (R) while thinking about typing the search string in the web page (M). Then M should not come before typing since it is overlapping with R

KLM Limitations

- It can model only "expert" user behavior
- User errors cannot be modeled
- Analysis should be done for "representative" tasks; otherwise, the prediction will not be of much use in design. Finding "representative" tasks is not easy

Example 2:
Scenario 1, New User
Assumptions: *T*he new user will stop and check feedback from system at every step. The general procedure such a user will follow is to find the file icon, make sure it is selected, and drag it to the trash can, making sure the trash can has been hit (reverse videos), and then verify the whole process by checking that the trash can is bulging.

Operator sequence
1. Initiate the deletion (decide to do the task) **M**
2. find the file icon **M**
3. point to file icon **P**
4. press and hold mouse button **B**
5. verify that the icon is reverse-video **M**
6. find the trash can icon **M**
7. drag file icon to trash can icon **P**
8. verify that the trash can icon is reverse-video **M**
9. release mouse button **B**
10. verify that the trash can icon is bulging **M**
11. find the original window **M**
12. point to original window **P**

Total time = 3**P** + 2**B** + 7**M** = 11.9 sec

If this seems long, maybe it is because you are more experienced!

Example 3:
Scenario 2, Experienced User
Assumptions. We will assume the following things about the experienced user. As a result, the general procedure is simply to find the file icon to be deleted and drag it to the trash can.
Experienced user thinks of selecting and dragging an item as a single operation - a chunk. The user must find the to-be-deleted icon since it is different every time. Moving icons to the trash can is highly practiced:

The trash can does not have to be located, so finding it 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. Pointing to the original window is overlapped with finding it.

Operator sequence

1. initiate the deletion **M**
2. find the file icon **M**
3. point to file icon **P**
4. press and hold mouse button **B**
5. drag file icon to trash can icon **P**
6. release mouse button **B**
7. point to original window **P**

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

This seems to be in the right ballpark for an experienced user.

(CMN)GOMS

CMN stands for Card, Moran and Newell –the surname of the three researchers who proposed it.

KLM vs (CMN) GOMS

- In (CMN) GOMS, a hierarchical cognitive (thought) process is assumed, as opposed to the linear thought process of KLM
- Both assumes error-free and 'logical' behavior

–A logical behavior implies that we think logically, rather than driven by emotions

(CMN) GOMS –Basic Idea:

(CMN) GOMS allows us to model the task and user actions in terms of four constructs (goals, operators, methods, selection rules).

- **Goals:** represents what the user wants to achieve, at a higher cognitive level. This is a way to structure a task from cognitive point of view.
  The notion of Goal allows us to model a cognitive process hierarchically.
- **Operators:** elementary acts that change user's mental (cognitive) state or task environment. This is similar to the operators we have encountered in KLM, but here the concept is more general.
- **Methods:** these are sets of goal-operator sequences to accomplish a sub-goal.
- **Selection rules:** sometimes there can be more than one method to accomplice a goal. Selection rules provide a mechanism to decide among the methods in a particular context of interaction.

Operator in (CMN) GOMS:

- As mentioned before, operators in (CMN) GOMS are conceptually similar to operators in KLM.
- The major difference is that in KLM, only seven operators are defined. In (CMN) GOMS, the notion of operators is not restricted to those seven.
- The modeler has the freedom to define any "elementary" cognitive operation and use that as operator.

The operator can be defined

- At the keystroke level (as in KLM).
- At higher levels (for example, the entire cognitive process involved in "closing a file by selecting the close menu option" can be defined as operator).
- (CMN) GOMS gives the flexibility of defining operators at any level of cognition and different parts of the model can have operators defined at various levels.

Example

Suppose we want to find out the definition of a word from an online dictionary. How can we model this task with (CMN) GOMS?

We shall list the goals (high level tasks) first

–**Goal:** Access online dictionary (first, we need to access the dictionary)

–**Goal:** Lookup definition (then, we have to find out the definition)

Next, we have to determine the methods (operator or goal-operator sequence) to achieve each of these goals

–**Goal:** Access online dictionary

•**Operator:** Type URL sequence

•**Operator:** Press Enter

–**Goal:** Lookup definition

•**Operator:** Type word in entry field

-**Goal:** Submit the word

.**Operator:** Move cursor from field to Lookup button

.**Operator:** Select Lookup

•**Operator:** Read output

Thus, the complete model for the task is

–Goal: Access online dictionary

•Operator: Type URL sequence

•Operator: Press Enter

–Goal: Lookup definition

•Operator: Type word in entry field

-Goal: Submit the word

.Operator: Move cursor from field to Lookup button

.Operator: Select Lookup button

•Operator: Read output

Notice the hierarchical nature of the model

Note the use of operators

- The operator “type URL sequence” is a high-level operator defined by the modeler
- “Press Enter” is a keystroke level operator
- Note how both the low-level and high-level operators co-exist in the same model

Note the use of methods

- **For the first goal**, the method consisted of two operators
- **For the second goal**, the method consisted of two operators and a sub-goal (which has a two-operator method for itself)

Another Example

The previous example illustrates the concepts of goals and goal hierarchy, operators and methods

The other important concept in (CMN) GOMS is the selection rules. The example illustrates this concept

Example:

Suppose we have a window interface that can be closed in either of the two methods: by selecting the ‘close’ option from the file menu or by selecting the Alt key and the F4 key together. How we can model the task of “closing the window” for this system?

- Here, we have the high level goal of “close window” which can be achieved with either of the two methods: “use menu option” and “use Alt+F4 keys”
- This is unlike the previous example where we had only one method for each goal
- We use the “Select” construct to model such situations:

Goal: Close window

• [Select Goal: Use menu method

Operator: Move mouse to file menu

Operator: Pull down file menu

Operator: Click over close option

Goal: Use Alt+F4 method

Operator: Press Alt and F4 keys together]

•The **select construct implies that “selection rules**” are there to determine a method among the alternatives for a particular usage context

Example selection rules for the window closing task can be

Rule 1: Select “use menu method” unless another rule applies

Rule 2: If the application is GAME, select “use Alt+F4 method”

•The rules state that, if the window appears as an interface for a game application, it should be closed using the Alt+F4 keys. Otherwise, it should be closed using the close menu option

Example: Delete a File

```
GOAL: DELETE-FILE
.   GOAL: SELECT-FILE
.   .   [select:  GOAL: KEYBOARD-TAB-METHOD
.   .             GOAL: MOUSE-METHOD]
.   .   VERIFY-SELECTION
.   GOAL: ISSUE-DELETE-COMMAND
.   .   [select*: GOAL: KEYBOARD-DELETE-METHOD
.   .             .   PRESS-DELETE
.   .             .   GOAL: CONFIRM-DELETE
.   .             GOAL: DROP-DOWN-MENU-METHOD
.   .             .   MOVE-MOUSE-OVER-FILE-ICON
.   .             .   CLICK-RIGHT-MOUSE-BUTTON
.   .             .   LOCATE-DELETE-COMMAND
.   .             .   MOVE-MOUSE-TO-DELETE-COMMAND
.   .             .   CLICK-LEFT-MOUSE-BUTTON
.   .             .   GOAL: CONFIRM-DELETE
.   .             GOAL: DRAG-AND-DROP-METHOD
.   .             .   MOVE-MOUSE-OVER-FILE-ICON
.   .             .   PRESS-LEFT-MOUSE-BUTTON
.   .             .   LOCATE-RECYCLING-BIN
.   .             .   MOVE-MOUSE-TO-RECYCLING-BIN
.   .             .   RELEASE-LEFT-MOUSE-BUTTON]

*Selection rule for GOAL: ISSUE-DELETE-COMMAND
  If hands are on keyboard, use KEYBOARD-DELETE-METHOD,
    else if Recycle bin is visible, use DRAG-AND-DROP-METHOD,
    else use DROP-DOWN-MENU-METHOD
```

Steps for Model Construction:

- **A (CMN) GOMS model for a task is constructed according to the following steps**
  - Determine high-level user goals.
  - Write method and selection rules (if any) for accomplishing goals.
  - This may invoke sub-goals, write methods for sub-goals.
  - This is recursive, stop when operators are reached
- **Use of the Model:**
  - Like KLM, (CMN) GOMS also makes quantitative prediction about user performance.
  - By adding up the operator times, total task execution time can be computed.
  - However, if the modeler uses operators other than those in KLM, the modeler has to determine the operator times.
  - The task completion time can be used to compare competing designs.
  - In addition to the task completion times, the task hierarchy itself can be used for comparison.

- The deeper the hierarchy (keeping the operators same), the more complex the interface is (since it involves more thinking to operate the interface).

Model Limitations:

- Like KLM, (CMN) GOMS also models only skilled (expert) user behavior. That means user does not make any errors.
- Cannot capture the full complexity of human cognition such as learning effect, parallel cognitive activities and emotional behavior.

Individual Models of Human Factors –I

In the GOMS family, KLM and (CMN) GOMS, are simple models of human information processing. They are one of three cognitive modeling approaches used in HCI. A second type of cognitive models used in HCI is the individual models of human factors

Two well-known models belonging to this category:

- **The Fitts' law:** a law governing the manual (motor) movement.
- **The Hick-Hyman law:** a law governing the decision making process in the presence of choice.

Fitts' Law:

- It is one of the earliest predictive models used in HCI (and among the most well-known models in HCI also).
- First proposed by PM Fitts (hence the name) in 1954.
- The Fitts' law is a model of human motor performance.
- It mainly models the way we move our hand and fingers.
- A very important thing to note is that the law is not general; it models motor performance under certain constraints.

Fitts' Law –Characteristics:

The law models human motor performance having the following characteristics:

- The movement is related to some "target acquisition task" (i.e., the human wants to acquire some target at some distance from the current hand/finger position)
- The movement is rapid and aimed (i.e., no decision making is involved during movement)
- The movement is error-free (i.e. the target is acquired at the very first attempt)

Nature of the Fitts' Law:

Another important thing about the Fitts' law is that, it is both a descriptive and a predictive model

- **Why it is a descriptive model?**
  –Because it provides "throughput", which is a descriptive measure of human motor performance.
- **Why it is a predictive model?**
  –Because it provides a prediction equation (an analytical expression) for the time to acquire a target, given the distance and size of the target.

Task Difficulty:

- The key concern in the law is to measure "task difficulty" (i.e., how difficult it is for a person to acquire, with his hand/finger, a target at a distance D from the hand/finger's current position)
- Note that the movement is assumed to be rapid, aimed and error-free

Fitts, in his experiments, noted that the difficulty of a target acquisition task is related to two factors:

- **Distance (D):** the distance by which the person needs to move his hand/finger. This is also called amplitude (A) of the movement.
  The larger the D is, the harder the task becomes
- **Width (W):** the difficulty also depends on the width of the target to be acquired by the person
  As the width increase, the task becomes easier.

Measuring Task Difficulty:

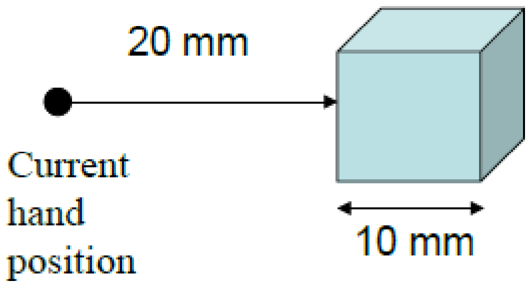
- The qualitative description of the relationships between the task difficulty and the target distance (D) and width (W) cannot help in "measuring" how difficult a task is.
- Fitts' proposed a 'concrete' measure of task difficulty, called the **"index of difficulty" (ID).**
- From the analysis of empirical data, Fitts' proposed the following **relationship between ID, D and W.**

ID = log2 (D/W+1) [unit is bits]

(**Note:** the above formulation was not what Fitts originally proposed. It is a refinement of the original formulation over time. Since this is the most common formulation of ID, we shall follow this rather than the original one).

ID –Example

Suppose a person wants to grab a small cubic block of wood (side length = 10 mm) at a distance of 20 mm. What is the difficulty for this task?



Here D = 20 mm, W = 10 mm

Thus, ID = $\log_2$ (20/10+1)

= $\log_2$ (2+1)

= $\log_2 3$ = 1.58 bits

Throughput

- Fitts' also proposed a measure called the index of performance (IP), now called throughput (TP)
- Computed as the difficulty of a task (ID, in bits) divided by the **movement time** to complete the task (MT, in seconds)
- Thus, TP = ID/MT bits/s

Throughput Example

Consider our previous example (on ID). If the person takes 2 sec to reach for the block, what is the throughput of the person for the task?

Here ID = 1.58 bits, MT = 2 sec

Thus TP = 1.58/2

= 0.785 bits/s =0.79 bits/s

Implication of Throughput:

- The concept of throughput is very important
- It actually refers to a measure of performance for rapid, aimed, error-free target acquisition task (as implied by its original name "index of performance")
- Taking the human motor behavior into account
- In other words, throughput should be relatively constant for a test condition over a wide range of task difficulties; i.e., over a wide range of target distances and target widths

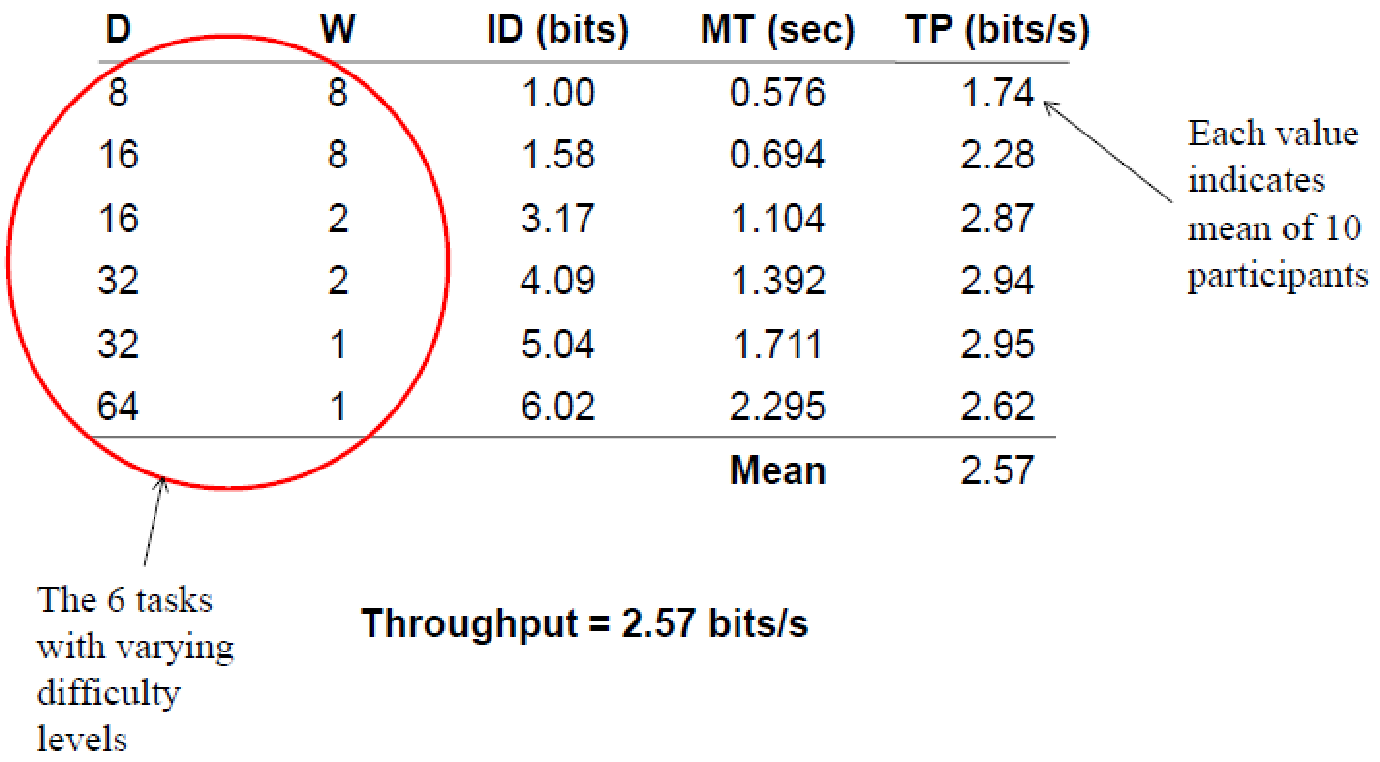
Examples of Test Condition:

- **Suppose a user is trying to point to an icon on the screen using a mouse**
  - The task can be mapped to a rapid, aimed, error-free target acquisition task
  - The mouse is the test condition here
  - If the user is trying to point with a touchpad, then touchpad is the test condition
- **Suppose we are trying to determine target acquisition performance for a group of persons (say, workers in a factory) after lunch**
  - The "taking of lunch" is the test condition here

Throughput –Design Implication:

- The central idea is -Throughput provides a means to measure user performance for a given test condition.
- We collect throughput data from a set of users for different task difficulties.
- The mean throughput for all users over all task difficulties represents the average user performance for the test condition.

Example –suppose we want to measure the performance of a mouse. We employ 10 participants in an experiment and gave them 6 different target acquisition tasks (where the task difficulties varied). From the data collected, we can measure the mouse performance by taking the mean throughput over all participants and tasks.



D	W	ID (bits)	MT (sec)	TP (bits/s)
8	8	1.00	0.576	1.74
16	8	1.58	0.694	2.28
16	2	3.17	1.104	2.87
32	2	4.09	1.392	2.94
32	1	5.04	1.711	2.95
64	1	6.02	2.295	2.62
			Mean	2.57

Each value indicates mean of 10 participants

The 6 tasks with varying difficulty levels

Throughput = 2.57 bits/s

- ➢ In the example, note that the mean throughputs for each task difficulty is relatively constant (i.e., not varying widely)
- ➢ This is one way of checking the correctness of our procedure (i.e., whether the data collection and analysis was proper or not)

We saw how TP helps in design

- ➢ We estimate the user performance under a test condition by estimating TP.
- ➢ The TP is estimated by taking mean of the TP achieved by different persons tested with varying task difficulty levels under the same test condition.

- **How TP can help in comparing designs?**
- **How the Fitts' law can be used as a predictive model?**

- **compare competing designs**

- **Suppose you have designed two input devices: a mouse and a touchpad. You want to determine which of the two is better in terms of user performance, when used to acquire targets (e.g., for point and select tasks). How can you do so?**

- You set up two experiments for two test conditions: one with the mouse and the other with the touchpad
- Determine throughput for each test condition as we have done before (i.e., collect throughput data from a group of users for a set of tasks with varying difficulty level and take the overall mean)
- Suppose we got the throughputs TP1 and TP2 for the mouse and the touchpad experiments, respectively

Compare TP1 and TP2

- If **TP1>TP2**, the mouse gives better performance
- The touchpad is better if **TP1<TP2**
- They are the **same performance-wise if TP1=TP2** (this is very unlikely as we are most likely to observe some difference)

Predictive Nature of Fitts' Law

- The throughput measure, derived from the Fitts' law, is descriptive
- We need to determine its value empirically

Fitts' law also allows us to predict performance

- That means, we can "**compute" performance** rather than determine it empirically
- Although not proposed by Fitts, it is now common to build a prediction equation in Fitts' law research
- The predictive equation is obtained by linearly regressing MT (movement time) against the ID (index of difficulty), in a MT-ID plot

The equation is of the form:

$$MT = a + b.ID$$

a and b are constants for a test condition (empirically derived)

As we can see, the equation allows us to predict the time to complete a target acquisition task (with known D and W)

How we can use the predictive equation in design?

- We determine the constant values (a and b) empirically, for a test condition
- Use the values in the predictive equation to determine MT for a representative target acquisition task under the test condition
- Compare MTs for different test conditions to decide (as with throughput)

A Note on Speed-Accuracy Trade-off

Suppose, we are trying to select an icon by clicking on it. The icon width is D

- Suppose each click is called a "hit". In a trial involving several hits, we are most likely to observe that not all hits lie within D (some may be just outside)
- If we plot the hit distributions (i.e., the coordinates of the hits), we shall see that about 4% of the hits are outside the target boundary

- This is called the speed-accuracy trade-off
- When we are trying to make rapid movements, we cannot avoid errors
- However, in the measures (ID, TP and MT), we have used D only, without taking into account the trade-off
- We assumed all hits will be inside the target boundary

We can resolve this in two-ways

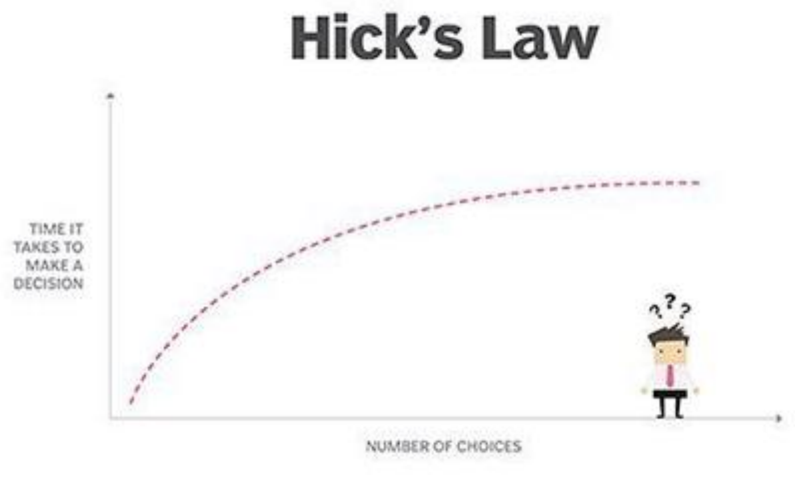
- Either we proceed with our **current approach**, with the knowledge that the measures will have 4% error rates
- Or we take the effective width De (the width of the region enclosing all the hits) instead of D. The **second approach** requires us to empirically determine De for each test condition

The Hick-Hyman Law

While Fitts' law relates task performance to motor behavior, there is another law popularly used in HCI, which tell us the **"reaction time"** (i.e., the time to react to a stimulus) of a person in the presence of "choices". The law is called the Hick-Hyman law, named after its inventors.

What Is Hick's Law?

Hick's Law (or the Hick-Hyman Law) is named after a British and an American psychologist team of William Edmund Hick and Ray Hyman. In 1952, this pair set out to examine the relationship between the number of stimuli present and an individual's reaction time to any given stimulus. As you would expect, the more stimuli to choose from, the longer it takes the user to make a decision on which one to interact with. Users bombarded with choices have to take time to interpret and decide, giving them work they don't want.



The formula for Hick's Law is defined as follows:

$$RT = a + b \log_2 (n)$$

Where "RT" is the reaction time, "(n)" is the number of stimuli present, and "a" and "b" are arbitrary measurable constants that depend on the task that is to be carried out and the conditions under which it will be carried out.

Example

A telephone call operator has 10 buttons. When the light behind one of the buttons comes on, the operator must push the button and answer the call. When a light comes on, how long does the operator takes to decide which button to press?

In the example,

- The "light on" is the stimulus
- We are interested to know the operator's "reaction time" in the presence of the stimulus
- The operator has to decide among the 10 buttons (these buttons represent the set of choices)
- The Hick-Hyman law can be used to predict the reaction times in such situations

The Law

- As we discussed before, the law models human reaction time (also called choice-reaction time) under uncertainty (the presence of choices).
- The law states that the reaction (decision) time T increases with uncertainty about the judgment or decision to be made.

We know that a measure of uncertainty is entropy (H)

Thus, T α H or **equivalently, T = kH**, where **k is the proportionality constant** (empirically determined)

We can calculate H in terms of the choices in the following way. Let, pi be the probability of making the i-th choice

$$\text{Then, } H = \sum_i p_i \log_2(1/p_i)$$

Therefore,

$$T = k\sum_{i=1} p_i \log_2(1/p_i)$$

When all the probabilities of making choices becomes equal, we have H = log$_2$N (N = no of choices)

$$\text{– In such cases, } T = k \log_2 N$$

Example Revisited

Then, what will be the operator's reaction time in our example?

–Here N = 10

–A button can be selected with a probability 1/10 and all probabilities are equal

–Here K=a+b

$$\begin{aligned} \text{– Thus, } T &= k \log_2 10 \\ &= 0.66 \text{ ms (assuming } a = 0,\ b = 0.2) \end{aligned}$$