PANIMALAR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE

IV YEAR - VII SEMESTER

CS8079 – HUMAN COMPUTER INTERACTION (R 2017)

UNIT III MODELS AND THEORIES

HCI Models: Cognitive models: Socio-Organizational issues and stakeholder requirements –Communication and collaboration models-Hypertext, Multimedia and WWW.

3.1 COGNITIVE MODELS

- One way to classify the models is in respect to how well they describe features of the competence and performance of the user. Competence models tend to be ones that can predict legal behavior sequences but generally do this without reference to whether they could actually be executed by users.
- Performance models not only describe what the necessary behavior sequences are but usually describe both what the user needs to know and how this is employed in actual task execution. The presentation of the cognitive models in this chapter follows this classification scheme, divided into the following categories:

a) hierarchical representation of the user's task and goal structure
b) linguistic and grammatical models
c) physical and device-level models

- The first category deals directly with the issue of formulation of goals and tasks. The second deals with the grammar of the articulation translation and how it is understood by the user. The third category again deals with articulation, but at the human motor level instead of at a higher level of human understanding.

3.1.1 Goal and Task Hierarchies

- Many models make use of a model of mental processing in which the user achieves goals by solving sub goals in a divide-and-conquer fashion. We will consider two models, GOMS and CCT, where this is a central feature. Imaginewe want to produce a report on sales of introductory HCI textbooks.
- To achieve this goal we divide it into several sub goals, say gathering the data together, producing the tables and histograms, and writing the descriptive material.
- Concentrating on the data gathering, we decide to split this into further sub goals: find the names of all introductory HCI textbooks and then search the book sales database for these books. Similarly, each of the other sub goals is divided up into further sub goals, until some level of detail is found at which we decide to stop. We thus end up with a hierarchy of goals and sub goals. The example can be laid out to expose this structure:

produce reportgather data
. find book names
. . do keywords search of names database
<<further subgoals>>

. . sift through names and abstracts by hand
<<further subgoals>>
. search sales database
<<further subgoals>> layout tables and histograms
<<further subgoals>>write description
<<further subgoals>>

- Different design issues demand different levels of analysis. This most abstract task is referred to as the unit task. The unit task does not require any problem- solving skills on the part of the user, though it frequently demands quite sophisticated problem-solving skills on the part of the designer to determine them.

1. GOMS

- The GOMS model of Card, Moran and Newell is an acronym for Goals, Operators, Methods and Selection [56]. A GOMS description consists of these four elements:

a. Goals: These are the user's goals, describing what the user wants to achieve. Further, in GOMS the goals are taken to represent a 'memory point' for the user, from which he can evaluate what should be done and to which he may return should any errors occur.

b. Operators: These are the lowest level of analysis. They are the basic actions that the user must perform in order to use the system. They may affect the system (for example, press the 'X' key) or only the user's mental state (for example, read the dialog box).

c. Methods: There are typically several ways in which a goal can be split into sub goals. For instance, in a certain window manager a currently selected window can be closed to an icon either by selecting the 'CLOSE' option from a pop-up menu, or by hitting the 'L7' function key. In GOMS these two goal decompositions are referred to as methods, so we have the **CLOSE-METHOD** and the **L7-METHOD**:

GOAL: ICONIZE-WINDOW
. [select GOAL: USE-CLOSE-METHOD
. . MOVE-MOUSE-TO-WINDOW-HEADER
. . POP-UP-MENU
. . CLICK-OVER-CLOSE-OPTIONGOAL: USE-L7-METHOD
. . PRESS-L7-KEY]
The dots are used to indicate the hierarchical level of goals.

d. Selection From the above snippet we see the use of the word select where the choice of methods arises. GOMS does not leave this as a random choice, but attempts to predict which methods will be used. This typically depends both on the particular user and on the state of the system and details about the goals.

- For instance, a user, Sam, never uses the L7-METHOD, except for one game, 'blocks', where the mouse needs to be used in the game until the very moment the key is pressed. GOMS captures this in a selection rule for Sam:
  User Sam:
  - ✓ Rule 1: Use the CLOSE-METHOD unless another rule applies.
  - ✓ Rule 2: If the application is 'blocks' use the L7-METHOD.

The goal hierarchies described in a GOMS analysis are almost wholly below the level of the unit task defined earlier. A typical GOMS analysis would therefore consist of a single high- level goal, which is then decomposed into a sequence of unit tasks, all of which can be further decomposed down to the level of basic operators. The goal of decomposition between the overall task and the unit tasks would involve detailed understanding of the user's problem-solving strategies and of the application domain.

2. Cognitive complexity theory

- Cognitive complexity refer to the number of mental structures an individual uses, how abstract they are and how they interact to shape his discernment or an individual difference variable linked with a wide range of communication skills and associated abilities.
- Individuals with high cognitive complexity have the capacity to analyze a situation to discern various constituent elements and explore connections and possible relationships among the elements. These individuals think in a multidimensional way. The assumption of the complexity theory is that the more an event can be differentiated and parts considered in novel relationships, the more sophisticated the response and successful the solution. Whereas less complex individuals can be trained to understand a complicated set of detailed differentiations for a specific context, highly complex individuals are highly flexible in creating distinctions in new situations.
- Individuals with high cognitive complexity are open to new information, attracted to other individuals of high complexity, highly flexibility, socially influential, problem solvers, strategic planners, highly creative, effective communicators and generally good leaders.
- CCT has two parallel descriptions: one of the user's goals and the other of the computer system (called the device in CCT). The description of the user's goals is based on a GOMS-like goal hierarchy, but is expressed primarily using production rules. CCT uses generalized transition networks, a form of state transition network. The production rules are a sequence of rules:

  if condition then action

  where condition is a statement about the contents of working memory. If the condition is true then the production rule is said to fire.
- An action may consist of one or more elementary actions, which may be either changes to the working memory, or external actions such as keystrokes. The production rule 'program' is written in a LISP-like language.
- Rules in CCT need not represent error-free performance. They can be used to explain error phenomena, though they cannot predict them. For instance, the rules above for inserting a space are 'buggy' – they do not check the editor's mode.
- The CCT rules are closely related to GOMS-like goal hierarchies; the rules may be generated from such a hierarchy, or alternatively, we may analyze the production rules to obtain the goal tree:

  GOAL: insert space
  . GOAL: move cursor – if not at right position
  . PRESS-KEY-I
  . PRESS-SPACE
  . PRESS-ESCAPE

3.1.2 Linguistic Models

The user's interaction with a computer is often viewed in terms of a language, so it is not surprising that several modeling formalisms have developed centered around this concept. BNF grammars are frequently used to specify dialogs.

The models here, although similar in form to dialog design notations, have been proposed with the intention of understanding the user's behavior and analyzing the cognitive difficulty of the interface.

a. BNF

- Representative of the linguistic approach is Reisner's use of Backus–Naur Form (BNF) rules to describe the dialog grammar. This views the dialog at a purely syntactic level, ignoring the semantics of the language. BNF has been used widely to specify the syntax of computer programming languages, and many system dialogs can be described easily using BNF rules.
- For example, imagine a graphics system that has a line-drawing function. To select the function the user must select the 'line' menu option. The line-drawing function allows the user to draw a polyline, that is a sequence of line arcs between points. The user selects the points by clicking the mouse button in the drawing area. The user double clicks to indicate the last point of the polyline.

  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 + DOUBLE-CLICK-MOUSE
  position-mouse ::= empty | MOVE-MOUSE + position-mouse
- The names in the description are of two types: non-terminals, shown in lower case, and terminals, shown in upper case. Terminals represent the lowest level of user behavior, such as pressing a key, clicking a mouse button or moving the mouse.
- Non-terminals are higher-level abstractions. The non-terminals are defined in terms of other non-terminals and terminals by a definition of the form name ::= expression
- The '::=' symbol is read as 'is defined as'. Only non-terminals may appear on the left of a definition. The right-hand side is built up using two operators '+' (sequence) and '|' (choice).
- For example, the first rule says that the non-terminal draw-line is defined to be select-line followed by choose-points followed by last point. All of these are non-terminals, that is they do not tell us what the basic user actions are.
- The second rule says that select-line is defined to be position mouse (intended to be over the 'line' menu entry) followed by CLICK-MOUSE. This is our first terminal and represents the actual clicking of a mouse. To see what position- mouse is, we look at the last rule.
- This tells us that there are two possibilities for position-mouse (separated by the '|' symbol). One option is that position-mouse is empty – a special symbol representing no action. That is, one option is not to move the mouse at all.
- The other option is to do a MOVE-MOUSE action followed by position-mouse. This rule is recursive, and this second position-mouse may itself either be empty or be a MOVE-MOUSE action followed by position-mouse, and so on. That is, position-mouse may be any number of MOVE-MOUSE actions whatsoever.
- Choose-points is defined recursively, but this time it does not have the option of being empty. It may be one or more of the non-terminal choose one which is itself defined to be (like select-line) position-mouse followed by CLICK-MOUSE.
- The BNF description of an interface can be analyzed in various ways. One measure is to count the number of rules. The more rules an interface requires to use it, the more complicated it is. This measure is rather sensitive to the exact way the interface is described. For example, we could have replaced the rules for choose points and choose-one with the single definition,
  choose-points ::= position-mouse + CLICK-MOUSE| position-mouse + CLICK-MOUSE + choose-points

More robust measure also counts the number of '+' and '|' operators. This would, in effect, penalize the more complex single rule. Another problem arises with the rule for select-line. This is identical to the choose-one rule.

b. Task–action grammar

- Measures based upon BNF have been criticized as not 'cognitive' enough. They ignore the advantages of consistency both in the language's structure and in its use of command names and letters. Task–action grammar (TAG) attempts to deal with some of these problems by including elements such as parameterized grammar rules to emphasize consistency and encoding the user's world knowledge (for example, up is theopposite of down).
- To illustrate consistency, we consider the three UNIX commands: cp (for copying files), mv (for moving files) and ln (for linking files). Each of these has two possible forms. They either have two arguments, a source and destination filename, or have any number of source filenames followed by a destination directory:

  copy ::= 'cp' + filename + filename
  | 'cp' + filenames + directory
  move ::= 'mv' + filename + filename
  | 'mv' + filenames + directory link ::= 'ln' + filename + filename
  | 'ln' + filenames + directory

- Measures based upon BNF could not distinguish between these consistent commands and an inconsistent alternative – say if ln took its directory argument first. Task–action grammar was designed to reveal just this sort of consistency. Its description of the UNIX commands would be

  file-op[Op] := command[Op] + filename + filename
  | command[Op] + filenames + directory command[Op=copy] := 'cp' command[Op=move] := 'mv' command[Op=link] := 'ln'

- Hierarchical and grammar-based techniques were initially developed when most interactive systems were command line, or at most, keyboard and cursor based. There are significant worries, therefore, about how well these approaches can generalize to deal with more modern windowed and mouse-driven interfaces. Pressing a cursor key is a reasonable lexeme, but moving a mouse one pixel is less sensible. In addition, pointer-based dialogs are more display oriented. Clicking a cursor at a particular point on the screen has a meaning dependent on the current screen contents. This problem can be partially resolved by regarding operations such as 'select region of text' or 'click on quit button' as the terminals of the grammar. If this approach is taken, the detailed mouse movements and parsing of mouse events in the context of display information (menus, etc.) are abstracted away. Goal hierarchy methods have different problems, as more display-oriented systems encourage less structured methods for goal achievement. Instead of having well-defined plans, the user is seen as performing a more exploratory task, recognizing fruitful directions and backing out of others. Typically, even when this exploratory style is used at one level,

  WRITE_LETTER
  .FIND_SIMILAR_LETTER
  .COPY_IT
  .EDIT_COPY

3.1.3 PHYSICAL AND DEVICE MODELS

a. Keystroke-level model

- The human motor system is well understood. KLM (Keystroke-Level Model) uses this understanding as a basis for detailed predictions about user performance. It is aimed at unit tasks within interaction – the execution of simple command sequences, typically taking no more than 20 seconds. Examples of this would be using a search and replace feature, or changing the font of a word. It does not extend to complex actions such as producing a diagram. The assumption is that these more complex tasks would be split into subtasks (as in GOMS) before the user attempts to map them into physical actions.
- It does not extend to complex actions such as producing a diagram. The assumption is that these more complex tasks would be split into subtasks (asin GOMS) before the user attempts to map them into physical actions. Thetask is split into two phases:
  - acquisition of the task, when the user builds a mental representation of the task;
  - execution of the task using the system's facilities.
- KLM only gives predictions for the latter stage of activity. During the acquisition phase, the user will have decided how to accomplish the task using the primitives of the system, and thus, during the execution phase, there is no high-level mental activity the user is effectively expert.
- KLM is related to the GOMS model, and can be thought of as a very low level GOMS model where the method is given. The model decomposes the execution phase into five different physical motor operators, a mental operator and a system response operator:
  a. K Keystroking, actually striking keys, including shifts and other modifier keys.
  b. B Pressing a mouse button.
  c. P Pointing, moving the mouse (or similar device) at a target.
  d. H Homing, switching the hand between mouse and keyboard.
  e. D Drawing lines using the mouse.
  f. M Mentally preparing for a physical action.
  g. R System response which may be ignored if the user does not have towait for it, as in copy typing.
- The execution of a task will involve interleaved occurrences of the various operators. For instance, imagine we are using a mouse-based editor. If we notice a single character error we will point at the error, delete the character and retype it, and then return to our previous typing point. This is decomposed as follows:
  1. Move hand to mouse H[mouse]
  2. Position mouse after bad character PB[LEFT]
  3. Return to keyboard H[keyboard]
  4. Delete character MK[DELETE]
  5. Type correction K[char]
  6. Reposition insertion point H[mouse]MPB[LEFT]
- 5 Notice that some operators have descriptions added to them, representing which device the hand homes to (for example, [mouse]) and what keys are hit (for example, LEFT – the left mouse button). The model predicts the total time taken during the execution phase by adding the component times for each of the above activities.

Operator	Remarks	Time (s)
K	Press key	
	good typist (90 wpm)	0.12
	poor typist (40 wpm)	0.28
	non-typist	1.20
B	Mouse button press	
	down or up	0.10
	click	0.20
P	Point with mouse	
	Fitts' law	$0.1 \log_2(D/S + 0.5)$
	average movement	1.10
H	Home hands to and from keyboard	0.40
D	Drawing – domain dependent	–
M	Mentally prepare	1.35
R	Response from system – measure	–

wpm = words per minute

Table 3.1: Times for various operators in the keystroke level model

b. Three-state model

- We saw that a range of pointing devices exists in addition to the mouse. Often these devices are considered logically equivalent, if the same inputs are available to the application. That is, so long as you can select a point on the screen, they are all the same. These different devices – mouse, trackball, light pen – feel very different.
- Buxton has developed a simple model of input devices the three-state model, which captures some of these crucial distinctions. He begins by looking at a mouse. If you move it with no buttons pushed, it normally moves the mouse cursor about. This tracking behavior is termed state 1. Depressing a button over an icon and then moving the mouse will often result in an object being dragged about. This he calls state 2.

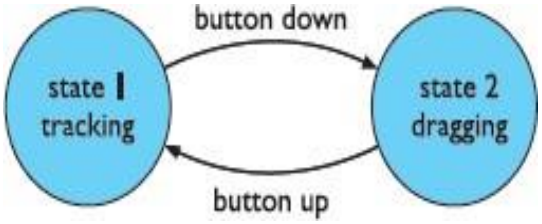

Fig. 3.1: Mouse Transitions: States 1 and 2

- Instead we consider a light pen with a button, it behaves just like a mouse when it is touching the screen. When its button is not depressed, it is in state 1, and when its button is down, state 2. However, the light pen has a third state, when the light pen is not touching the screen. In this state the system cannot track the light pen's position. This is called state 0.

  A touchscreen is like the light pen with no button. While the user is not touching the screen, the system cannot track the finger that is, state 0 again. When the user touches the screen, the system can begin to track state 1. So a touchscreen is a state 0–1 device whereas a mouse is a state 1–2 device. As there is no difference between a state 0–2 and a state 0–1 device, there are only the three possibilities we have seen.

- The only additional complexity is if the device has several buttons, in which case we would have one state for each button: 2left, 2middle, 2right.
- At first, the model appears to characterize the states of the device by the inputs available to the system. So, from this perspective, state 0 is clearly different from states 1 and 2. However, if we look at the state 1–2 transaction, we see that it is symmetric with respect to the two states.

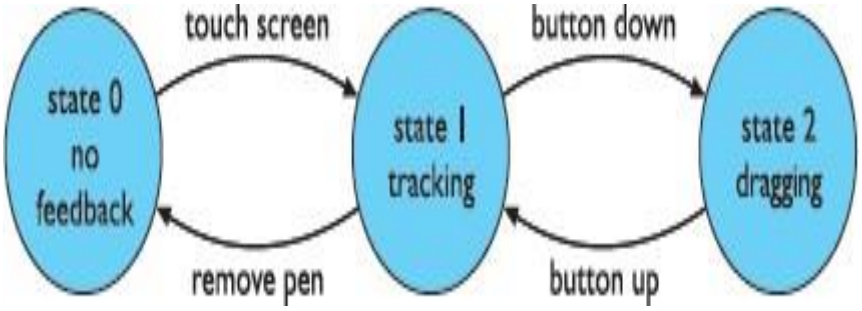

Fig.3.2: Light pen transitions: three states

- In principle, there is no reason why a program should not decide to do simple mouse tracking whilst in state 2 and drag things about in state 1 State 2 requires a button to be pressed, whereas state 1 is one of relative relaxation (whilst still requiring hand–eye coordination for mouse movement). There is a similar difference in tension between state 0 and state 1.
- It is well known that Fitts' law has different timing constants for different devices. Recall that Fitts' law says that the time taken to move to a target of size S at a distance D is:

$$a+b\log_2((D/S)+1)$$

- The constants a and b depend on the particular pointing device used and the skill of the user with that device.

	Device	*a* (ms)	*b* (ms/bit)
Pointing (state 1)			
	Mouse	–107	223
	Trackball	75	300
Dragging (state 2)			
	Mouse	135	249
	Trackball	–349	688

Table 3.2: Fitt's law coefficients

3.1.4 Cognitive Architectures

- The concept of taking a problem and solving it by divide and conquer using subgoals is central to GOMS. CCT assumes the distinction between long- and short-term memory, with production rules being stored in long-term memory and 'matched' against the contents of short-term (or working) memory to determine which 'fire'. The values for various motor and mental operators in KLM were based on the Model Human Processor (MHP) architecture of Card, Moran and Newell. Another common assumption, which we have not discussed in this chapter, is the distinction between linguistic levels – semantic, syntactic and lexical – as an architectural model of the user's understanding.

- The problem space model rational behavior is characterized as behavior that is intended to achieve a specific goal. This element of rationality is often used to distinguish between intelligent and machine-like behavior. In the field of artificial intelligence (AI), a system exhibiting rational behavior is referred to as a knowledge-level system. A knowledge- level system contains an agent behaving in an environment. The agent has knowledge about itself and its environment, including its own goals. It can perform certain actions and sense information about its changing environment. As the agent behaves in its environment, it changes the environment and its own knowledge. We can view the overall behavior of the knowledge-level system as a sequence of environment and agent states as they progress in time. The goal of the agent is characterized as a preference over all possible sequences of agent/environment states. The search proceeds by moving from one state to another possible state by means of operations or actions, the ultimate goal of which is to arrive at one of the desired states. This very general model of computation is used in the ordinary task of the programmer. Once she has identified a problem and a means of arriving at the solution to the problem (the algorithm), the programmer then represents the problem and algorithm in a programming language, which can be executed on a machine to reach the desired state. The architecture of the machine only allows the definition of the search or problem space and the actions that can occur to traverse that space. Termination is also assumed to happen once the desired state is reached.

- The new computational model is the problem space model, based on the problem-solving work of Newell and Simon at Carnegie–Mellon University. A problem space consists of a set of states and a set of operations that can be performed on the states. Behavior in a problem space is a two-step process. First, the current operator is chosen based on the current state and then it is applied to the current state to achieve the new state. The problem space must represent rational behavior, and so it must characterize the goal of the agent. A problem space represents a goal by defining the desired states as a subset of all possible states. Once the initial state is set, the task within the problem space is to find a sequence of operations that form a path within the state space from the initial state to one of the desired states, whereupon successful termination occurs.

- We can highlight four different activities that occur within a problem space: goal formulation, operation selection, operation application and goal completion. The relationship between these problem space processes and knowledge-level activity is key. Perception that occurs at the knowledge level is performed by the goal formulation process, which creates the initial state based on observations of the external environment. Actions at the knowledge level are operations in the problem space which are selected and applied. The real knowledge about the agent and its environment and goals is derived from the state/operator information in the problem space. Because of the goal formulation process, the set of desired states indicates the knowledge-level goal within the problem space. The operation selection process selects the appropriate operation at a given point in time because it is deemed the most likely to transform the state in the problem space to one of the desired states; hence rational behavior is implied.

- Interacting cognitive subsystems (ICS) provides a model of perception, cognition and action, but unlike other cognitive architectures, it is not intended to produce a description of the user in terms of sequences of actions that he performs. ICS provides a more holistic view of the user as an information-processing machine. The emphasis is on determining how easy particular procedures of action sequences become as they are made more automatic within the user.

- ICS attempts to incorporate two separate psychological traditions within one cognitive architecture. On the one hand is the architectural and general-purpose information-processing approach of short-term memory research. On the other hand is the computational and representational approach characteristic of psycholinguistic research and AI problem-solving literature.

- The architecture of ICS is built up by the coordinated activity of nine smaller subsystems: five peripheral subsystems are in contact with the physical world and four are central, dealing with mental processes. Each subsystem has the same generic structure. A subsystem is described in terms of its typed inputs and outputs along with a memory store for holding typed information. It has transformation functions for processing the input and producing the output and permanently stored information. Each of the nine subsystems is specialized for handling some aspect of external or internal processing. For example, one peripheral subsystem is the visual system for describing what is seen in the world.

3.2 SOCIO-ORGANIZATIONAL ISSUES AND STAKE HOLDER REQUIREMENTS

There are several organizational issues that affect the acceptance of technology by users and that must therefore be considered in system design:

- systems may not take into account conflict and power relationships
- those who benefit may not do the work
- not everyone may use systems.

In addition to generic issues, designers must identify specific stakeholder requirements within their organizational context.

• Socio-technical models capture both human and technical requirements.
• Soft systems methodology takes a broader view of human and organizational issues.
• Participatory design includes the user directly in the design process.
• Ethnographic methods study users in context, attempting to take an unbiased perspective.

3.2.1 Organizational Issues

- We shall look at some of the organizational issues that affect the acceptance and relevance of information and communication systems. These factors often sit 'outside' the system as such, and may involve individuals who never use it.

➢ Cooperation or conflict?

The term 'computer-supported cooperative work' (CSCW) seems to assume that groups will be acting in a cooperative manner. This is obviously true to some extent; even opposing football teams cooperate to the extent that they keep (largely) within the rules of the game, but their cooperation only goes so far. People in organizations and groups have conflicting goals, and systems that ignore this are likely to fail spectacularly.

Imagine that an organization is already highly computerized, the different departments all have their own systems and the board decides that an integrated information system is needed. The production manager can now look directly at stocks when planning the week's work, and the marketing department can consult the sales department's contact list to send out marketing questionnaires.

The storekeeper always used to understate stock levels slightly in order to keep an emergency supply, or sometimes inflate the quoted levels when a delivery was due from a reliable supplier. Also, requests for stock information allowed the storekeeper to keep track of future demands and hence plan future orders. The storekeeper has now lost a sense of control and important sources of information. Members of the sales department are also unhappy: their contacts are their livelihood. The last thing they want is someone from marketing blundering in and spoiling a relationship with a customer built up over many years. Some of these people may resort to subverting the system, keeping 'sanitized' information online, but the real information in personal files.