HCI

SNEHA GUPTA

SEM VII

**GOMS (Goals, Operators, Methods, and Selection Rules) Model**

The GOMS model breaks down user actions into four components to understand how users interact with systems:

1. **Goals**:

* The primary objectives or tasks that users aim to accomplish while interacting with a system.
* Users can have multiple subgoals. For example, editing a document can include smaller tasks like inserting words or formatting text.

2. **Operators**:

* These are the fundamental actions that users take to achieve their goals, which can involve motor, cognitive, or perceptual activities.
* Operators may include pressing a key, selecting a file with the mouse, recalling data from memory, or verifying a result.

3. **Methods**:

* Methods are the sequences or strategies that users employ to accomplish tasks. There can be multiple ways to achieve the same goal.
* For instance, to move the cursor, a user might either use arrow keys or the mouse. Choosing a method depends on the situation and user preference.

4. **Selection Rules**:

* These are the decision-making guidelines that help users choose between different methods for achieving a goal.
* Selection rules involve considering factors like efficiency and ease. For example, to delete text, a user might decide to either use backspace repeatedly or highlight and delete a large chunk at once.

**Example**: In a text-editing task, a user might need to move a piece of text from one location to another. They would:

* Formulate a goal: Move text.
* Perform operators: Highlight the text, cut, and then paste it.
* Use methods: Select text with a double-click for a word or click and drag for a phrase.
* Selection rules: Choose how to highlight based on the length of the text (e.g., single word vs. phrase).

**Use of GOMS**: GOMS is useful for breaking down complex interaction tasks into smaller, measurable steps. It works well for tasks such as text editing, where users make decisions about how to execute commands efficiently.

**Keystroke-Level Model (KLM)**

The Keystroke-Level Model is a simplified version of GOMS, focusing on predicting the time required for expert users to complete tasks without errors. It breaks down tasks into individual actions and assigns a time estimate to each.

Key components include:

1. **Keystroking**: Time taken to press a key.
2. **Pointing**: Time taken to move the mouse or cursor to a target.
3. **Homing**: Time taken to move the hand between different input devices (e.g., from the keyboard to the mouse).
4. **Drawing**: Time taken for tasks that involve drawing or creating visual elements.
5. **Thinking**: Time taken for cognitive processes, such as decision-making or interpreting system responses.
6. **Waiting**: Time spent waiting for the system to respond.

**Example**: If a user wants to delete a file, KLM would predict the time it takes by summing:

* The time to move the cursor to the file icon (pointing).
* The time to click on the file (keystroking).
* The time to press the delete button (keystroking).
* The time spent thinking about confirming the deletion (thinking).
* Any delay while waiting for system confirmation (waiting).

**Use of KLM**: KLM is primarily used to evaluate the efficiency of user interfaces by predicting how long it takes an expert user to complete tasks. It is particularly valuable for optimizing UI design to minimize task time and improve usability.

**Widget-level Theories**

1. **Hierarchical Decomposition & Simplification**:
   * This theory suggests breaking down complex user interfaces into simpler components or widgets, which are higher-level building blocks like buttons, sliders, or scrolling lists.
   * By focusing on these widgets rather than atomic-level features, designers can manage complexity more effectively, leading to better usability and user experience.
2. **Performance Prediction through Widget Testing**:
   * Once a widget, such as a scrolling list, is tested based on parameters like the number of items or window size, designers can predict its performance in future interfaces.
   * The performance metrics can guide task frequency predictions and inform interface layout, ensuring frequently used widgets are placed conveniently for the user.
3. **Layout Appropriateness**:
   * Widget-level models also help in optimizing the layout of the interface. Frequently used widgets should be adjacent, and the layout should align with the natural task flow to minimize user effort.
   * The sequence of widgets (e.g., left-to-right) should harmonize with the task sequence, ensuring intuitive navigation and interaction.
4. **Automated Complexity Estimation**:
   * This model allows for automatic predictions of the perceptual and cognitive complexity of the interface, including motor load, guiding the designer in making adjustments that reduce user strain.
   * Such estimates help in refining the design to ensure that the interface is both functional and user-friendly.

**Context of Use Theories**

1. **Integration of Physical and Social Environments**:
   * User interaction with technology is shaped by their physical surroundings and social context, such as being at work, home, or in public.
   * Design must account for how these environments influence user behavior, as users adapt based on their situation.
2. **Dynamic and Situated User Behavior**:
   * Unlike fixed cognitive models, these theories highlight that users adjust their actions based on time, place, and circumstances.
   * Users may ask for help, take shortcuts, or behave cautiously depending on their environment.
3. **Application to Mobile and Ubiquitous Computing**:
   * Relevant to mobile devices that provide location-specific information, like city guides or museum apps.
   * These devices adapt to users’ needs based on their physical context.
4. **Guiding Mobile Device Innovations**:
   * A taxonomy of mobile apps can guide innovation, such as monitoring air quality or facilitating large group activities.
   * Mobile devices gather and share real-world data like location or health metrics.

**Object-Action Interface Model**

1. **Shift from Command Languages to GUIs**:
   * Graphical User Interfaces (GUIs) replaced complex command languages, making interaction simpler with direct manipulations of visual objects.
   * Visual representations, such as icons (e.g., trash cans for deletion), make actions more intuitive and aligned with real-world metaphors.
2. **Task Understanding and Decomposition**:
   * Designers begin by understanding the user's real-world tasks and objects (e.g., stock listings, photo libraries) and decompose them into smaller components, such as a single stock or share price.
   * Actions are similarly broken down from high-level goals into smaller steps, which can be translated into interface actions.
3. **Metaphoric Representations**:
   * After agreeing on task objects and actions, designers create metaphoric representations of these in the interface (e.g., folders, trash cans).
   * These interface objects mimic real-world actions but exist as pixels that can be manipulated through user interaction.
4. **Visibility of Actions**:
   * Designers ensure that actions are visible and understandable, allowing users to break down their tasks into smaller, manageable actions, such as opening a dialog box or performing keystrokes and clicks.

