GOMS

Lorin Hochstein
lorin@cs.umd.edu
October 2002

Overview

GOMS is a modeling technique (more specifically, a family of modeling techniques) that analyzes the user complexity of interactive systems. It is used by software designers to model user behavior. The user's behavior is modeled in terms of Goals, Operators, Methods and Selection rules, which are described below in more detail. Briefly, a GOMS model consists of *Methods* that are used to achieve *Goals*. A *Method* is a sequential list of *Operators* that the user performs and (sub)*Goals* that must be achieved. If there is more than one *Method* which may be employed to achieve a *Goal*, a *Selection rule* is invoked to determine what *Method* to choose, depending on the context.

Several variations of GOMS have been developed. To distinguish from later variants, the original GOMS formulation is sometimes referred to as CMN-GOMS.

Scope and Application

A GOMS model provides the designer with a model of a user's behavior while performing well-known tasks. These models can be used for a variety of purposes, as follows

Functionality Coverage

If the designer has a list of likely user goals, GOMS models can be used to verify that a method exists to achieve each of these goals.

Execution time

GOMS models can predict the time it will take for the user to carry out a goal (assuming an expert user with no mistakes). This allows a designer to profile an application to locate bottlenecks, as well as compare different UI designs to determine which one allows users to execute tasks quicker.

Help systems

Since GOMS models are an explicit representation of expert user activity, they can assist in designing help systems and tutorials to assist users in achieving goals.

Principles

Goals

Goals are what the user is trying to accomplish. These can be defined at various levels of abstraction, from very high-level goals (e.g. WRITE-RESEARCH-PAPER) to low-level goals (e.g. DELETE-WORD). Higher-level goals are decomposable into subgoals, and are arranged hierarchically.

Operators

Operators are the elementary perceptual, motor or cognitive actions that are used to accomplish the goals (e.g. DOUBLE-CLICK-MOUSE, PRESS-INSERT-KEY). Operators are not decomposable: they are atomic elements in the GOMS model. Furthermore, it is generally assumed that each operator requires a fixed amount of time for the user to execute, and that this time interval is independent of context (e.g. CLICK-MOUSE button takes 0.20 seconds to execute).

Methods

Methods are the procedures that describe how to accomplish goals. A method is essentially an algorithm that the user has internalized that determines the sequence of subgoals and operators necessary to achieve the desired goal. For example, one method to accomplish the goal DELETE-WORD in the Emacs text editor would be to MOVE-MOUSE to the beginning of the word, and PRESS-ALT-D-KEY-COMBINATION (the use-mouse-delete-word method). Another method to accomplish the same goal could involve using the arrow keys to reach the beginning of the word (the use-arrows-delete-word method).

Selection rules

Selection rules specify which method should be used to satisfy a given goal, based on the context. Since there may be several different ways of achieving the same goal, selection rules represent the user's knowledge of which method must be applied to achieve the desired goal. Selection rules generally take the form of a conditional statement, such as "if the word to be deleted is less than 3 lines away from the current cursor location, then use the use-arrows-delete-word-method, else use the use-mouse-delete-word method".

Keystroke-Level Model

The Keystroke-Level Model is a simplified version of GOMS. It was proposed by Card and Moran (1980) as a method for predicting user performance. Using KLM, execution time is estimated by listing the sequence operators and then summing the times of the individual operators. KLM aggregates all perceptual and cognitive function into a single value for an entire task, using a heuristic. KLM also does not employ selection rules. The original KLM had six classes of operators: **K** for pressing a key, **P** for pointing to a location on screen with the mouse, **H** for moving hands to home position on the keyboard, **M** for mentally preparing to perform an action, and **R** for system response where the user waits for the system. For each operator, there is an estimate of execution time. Additionally, there is a set of heuristic rules to account for mental preparation time.

Examples

KLM

Consider the text editing task of searching a Microsoft Word document for all occurrences of a four-letter word, and replacing it with another four-letter word. Here we use the notation of Card et. al (1983, p264-265). **K** represents pressing a key or a button. **P** represents pointing with the mouse to a target on the display. **H** represents moving hands to the home position on the keyboard or mouse. **M** is a heuristic to incorporate mentally preparing for a task. The time intervals are taken from the same source, for an average typist (55 wpm). In the table below, operations are sometimes concantenated and repeated. For example, M4K means "**M**ental preparation, then **4 K**ey presses."

Description	**Operation**	**Time (sec)**
Reach for mouse	`H[mouse]`	0.40
Move pointer to "Replace" button	P[menu item]	1.10
Click on "Replace" command	`K[mouse]`	0.20
Home on keyboard	`H[keyboard]`	0.40
Specify word to be replaced	`M4K[word]`	2.15
Reach for mouse	`H[mouse]`	0.40
Point to correct field	`P[field]`	1.10
Click on field	`K[mouse]`	0.20
Home on keyboard	`H[keyboard]`	0.40
Type new word	`M4K[word]`	2.15
Reach for mouse	`H[mouse]`	0.40
Move pointer on Replace-all	`P[replace-all]`	1.10
Click on field	`K[mouse]`	0.20
Total		**10.2**

According to this KLM model, it takes 10.2 seconds to accomplish this task.

GOMS

This example is taken from John & Kieras (1996b). It models the task of moving text in a Word processor, in the context of editing a manuscript. Note the use of subgoals and selection rules, which do not exist in KLM.

```
GOAL: EDIT-MANUSCRIPT
.     GOAL: EDIT-UNIT-TASK ... repeat until no more unit tasks
.     .     GOAL: ACQUIRE UNIT-TASK
.     .     .     GOAL: GET-NEXT-PAGE ... if at end of manuscript page
.     .     .     GOAL: GET-FROM-MANUSCRIPT
.     .     GOAL: EXECUTE-UNIT-TASK ... if a unit task was found
.     .     .     GOAL: MODIFY-TEXT
.     .     .     .   [select: GOAL: MOVE-TEXT* ...if text is to be moved
.     .     .     .       GOAL: DELETE-PHRASE ...if a phrase is to be deleted
.     .     .     .       GOAL: INSERT-WORD] ... if a word is to be inserted
.     .     .     .     VERIFY-EDIT

*Expansion of MOVE-TEXT goal
GOAL: MOVE-TEXT
.     GOAL: CUT-TEXT
.     .     GOAL: HIGHLIGHT-TEXT
.     .         .       [select**: GOAL: HIGHLIGHT-WORD
.     .         .                  .   MOVE-CURSOR-TO-WORD
.     .         .                  .   DOUBLE-CLICK-MOUSE-BUTTON
.     .         .                  .   VERIFY-HIGHLIGHT
.     .         .                  GOAL: HIGHLIGHT-ARBITRARY-TEXT
.     .         .                  .     MOVE-CURSOR-TO-BEGINNING       1.10
.     .         .                  .     CLICK-MOUSE-BUTTON             0.20
.     .         .                  .     MOVE-CURSOR-TO-END             1.10
.     .         .                  .     SHIFT-CLICK-MOUSE-BUTTON       0.48
.     .         .                  .     VERIFY-HIGHLIGHT]              1.35
.     .         GOAL: ISSUE-CUT-COMMAND
.     .         .     MOVE-CURSOR-TO-EDIT-MENU                          1.10
.     .         .     PRESS-MOUSE-BUTTON                                0.10
.     .         .     MOVE-CURSOR-TO-CUT-ITEM                           1.10
.     .         .     VERIFY-HIGHLIGHT                                  1.35
.     .         .     RELEASE-MOUSE-BUTTON                              0.10
.     GOAL: PASTE-TEXT
.     .     GOAL: POSITION-CURSOR-AT-INSERTION-POINT
.     .         MOVE-CURSOR-TO-INSERTION-POIONT                         1.10
.     .         CLICK-MOUSE-BUTTON                                      0.20
.     .         VERIFY-POSITION                                         1.35
.     .         GOAL: ISSUE-PASTE-COMMAND
.     .         .     MOVE-CURSOR-TO-EDIT-MENU                          1.10
.     .         .     PRESS-MOUSE-BUTTON                                0.10
.     .         .     MOVE-MOUSE-TO-PASTE-ITEM                          1.10
.     .         .     VERIFY-HIGHLIGHT                                  1.35
.     .         .     RELEASE-MOUSE-BUTTON                              0.10
                                        TOTAL TIME PREDICTED (SEC)     14.38
```

Based on the above GOMS analysis, it should take 14.38 seconds to move text.

Variants

The GOMS model was originally proposed by Card, Moran, and Newell (1983). Since then, several variations of the GOMS model have been proposed to address issues with the original model.

Keystroke-Level Model (KLM)

KLM is the simplest variant of GOMS, as described above. KLM has been used to model applications such as mouse-driven text editors, workstations for directory-assistance telephone operators, space operations database systems, and CAD/CAM software (John & Kieras, 1996a). This model is unsuited to analyzing more abstract tasks such as EDIT-MANUSCRIPT, which involve conditionals and decomposition into subgoals.

Card, Morn, and Newell GOMS (CMN-GOMS)

CMN-GOMS is the original GOMS model proposed by Card, Morn and Newell (1983). CMN-GOMS builds on KLM by adding subgoals and selection rules. A CMN-GOMS model can predict operator sequence as well as execution time. A CMN-GOMS model can be represented in program form, making it amenable to analysis as well as execution. CMN-GOMS has been used to model word processors (Card et. al, 1983), CAD system for ergonomic design (John & Kieras, 1996a), and Sun Microsystem's web page.

Variant: Natural GOMS Language (NGOMSL)

NGOMSL builds on CMN-GOMS by providing a natural-language notion for representing GOMS models, as well as a procedure for constructing the models (John, 1990 and Gray et al 1993). Under NGOMSL, methods are represented in terms of an underlying cognitive theory known as *cognitive complexity theory*, or CCT (addressing a criticism that GOMS does not have a strong basis in cognitive psychology, see the Limitations section). This cognitive theory allows NGOMSL to incorporate internal operators such as manipulating working memory information or setting up subgoals. Because of this, NGOMSL can also be used to estimate the time required to learn how to achieve tasks. Below is an example of a partial NGOMSL model, taken from John & Kieras (1996b) and modified brevity.

```
Method for goal: Cut text
  Step 1.  Accomplish goal: Highlight text.
  Step 2.  Return that the command is CUT, and
             accomplish goal: Issue a command.
  Step 3.  Return with goal accomplished.
...
Selection rule set for goal: Highlight text
  If text-is word, then accomplish goal: Highlight word.
  If text-is arbitrary, then accomplish goal: Highlight arbitrary text.
  Return with goal accomplished.
...
Method for goal: Highlight arbitrary text
  Step 1.  Determine position of beginning of text (1.20 sec)
  Step 2.  Move cursor to beginning of text        (1.10 sec)
  Step 3.  Click mouse button.                     (0.20 sec)
  Step 4.  Move cursor to end of text.             (1.10 sec)
  Step 5.  Shift-click mouse button.               (0.48 sec)
  Step 6.  Verify that correct text is highlighted (1.20 sec)
  Step 7.  Return with goal accomplished.
```

This NGOMSL model predicts that it will take 5.28 seconds to highlight arbitrary text.

Variant: Cognitive-Perceptual-Motor GOMS (CPM-GOMS)

CPM-GOMS builds on previous GOMS models by assumed that perceptual, cognitive and motor operators can be performed in parallel. It employs a schedule chart (also known as a PERT chart) to represent operators and dependencies between operators. CPM-GOMS is also known as Critical-Path-Method GOMS. The figure below is an example of a schedule chart for implementing the goal READ-SCREEN, when an eye movement is required. This figure is taken from John & Kieras (1996b.). The sequence which produces the longest path through the chart is called the critical path, and it represents an estimate of the total time required to perform the task. In this case (assuming a simple binary visual signal on the screen), the critical path is 50+50+30+100+50=280 msec. This particular model assumes that visual perception, cognitive operations, and eye movements can occur in parallel.

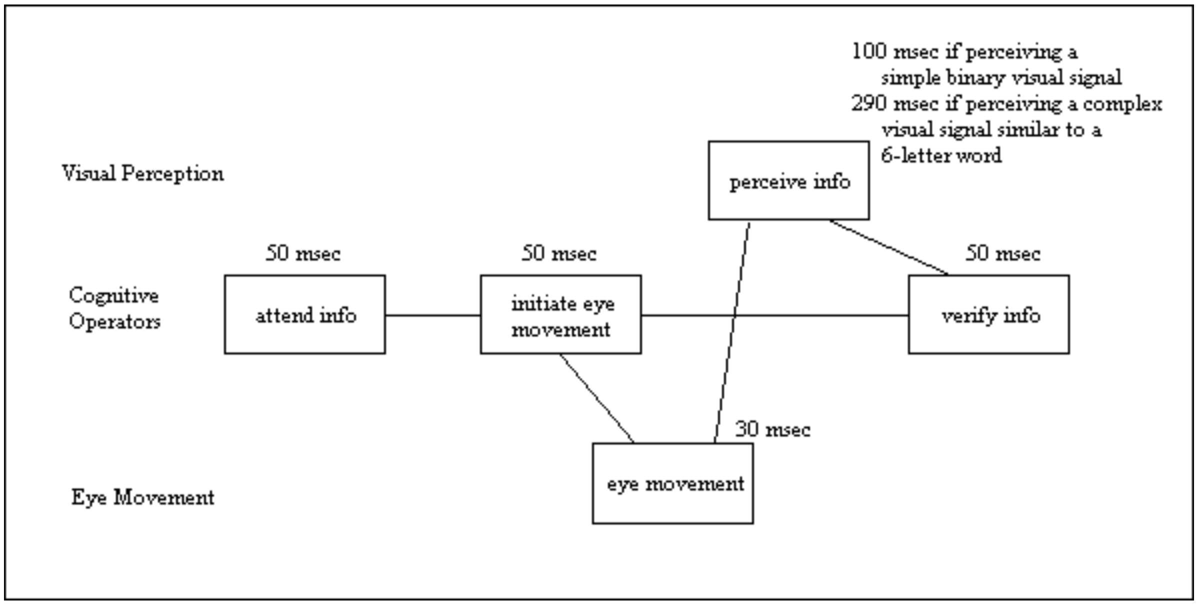

Figure 1: An example of a CPM-GOMS model

Applicability to HCI

Classification

As a theory of HCI, GOMS models can be classified as predictive, descriptive, and prescriptive. A GOMS model is predictive because it can be used to predict the time it will take a user to perform the tasks under analysis, as long as the developer can come up with time estimates for the operators involved in each model. A GOMS model is descriptive in the sense that it is a representation of the way a user performs tasks on a system. The methods, subgoals and selection rules provide the designer with a description of the process, rather than simply a time estimate. A GOMS model can also be considered prescriptive because it can serve as a guide for developing training programs and help systems. Once a GOMS model has been developed for achieving a certain goal, this model can be used to teach new users how to achieve the goal.

Because GOMS can provide quantitative estimates of task execution time, it can properly be considered a verifiable theory. The issue of falsifiability is a bit trickier: since the accuracy of a GOMS model is highly dependent on the estimates of the operator time values, a result that does not match the prediction can be explained away by variations in the operator time values for a specific user. Since the theory underlying GOMS models may not be truly falsifiable, GOMS should not be concerned a legitimate theory of human cognition (more detail on this subject is presented in the Limitations section below).

Influence

GOMS enjoys significant influence as a theory of HCI. It is one of the few widely known theoretical concepts within the field, and has been called the most mature engineering model of human performance. GOMS models continue to be applied to the evaluation of software systems, and GOMS remains an active area of scientific research.

Limitations

The GOMS approach has a number of limitations. Its most significant fault is that the predictions are only valid for expert users who does not make any errors. This is a significant shortcoming for two main reasons. First of all, even expert users will make mistakes (it could be argued that a good HCI theory would predict the number of errors expert users would make, and might use this as a basis for choosing one implementation over another). More importantly, GOMS does not take into account novices who are just learning the system, or intermediate users who make occasional errors. Since one of the goals of HCI is to aim for maximum usability for all users, especially novices, this is a serious deficiency in the model. (Note that the NGOMSL variant does attempt to model the time required to learn a task).

There are a number of other criticisms of the GOMS approach. It models all tasks as goal-directed, neglecting the problem-solving nature of some tasks. It does not take into account individual differences among users: at best, statistical averages for operator time values are used to come up with mean prediction values. In addition, the GOMS provides no insight on how useful or enjoyable the product under design will be: no other metric aside from execution speed is taken into account. Furthermore, GOMS does not address the social or organizational impact of the product under development.

GOMS has also been criticized for not being representative of current theories of human cognition. GOMS relies on a serial model of human cognition, assuming that one activity is done at a time until the task is complete. While it is possible to evaluate a GOMS model empirically by comparing predicted time to recorded time, the theory is not really based on existing theories of cognitive psychology. Therefore it can only be treated as an engineering heuristic, rather than an accurate model of cognitive processes. The NGOMSL and CPM-GOMS variants are based on more developed cognitive architectures.

References

Papers

Card, Stuart, Morn, Thomas P., and Newell, Allen, The keystroke-level model for user performance with interactive systems, *Communications of the ACM*, 23 (1980), 396-210
(This is the paper that presents KLM, the predecessor of GOMS)

Card, Stuart, Moran, Thomas P., and Newell, Allen, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ (1983).
(This book contains the original presentation of the GOMS model)

John, Bonnie and Kieras, David E., Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction 3,4* (December 1996a), 287-319.
(This paper explains which GOMS variant to use depending on the application)

John, Bonnie and Kieras, David E., The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast, *ACM Transactions on Computer-Human Interaction 3,4* (December

1996b), 320-351.
(In this companion paper to above, the different variants of GOMS are compared)

Kieras, David, Towards a practical GOMS model methodology for user interface design. In Helander, Martin (Editor), *Handbook of Human-Computer Interaction,* Elsevier Science Publishers, Amsterdam, The Netherlands (1988),135-137.
(This paper presents the NGOMSL variant)

Gray, W. D., John, B. E., & Atwood, M. E. Project Ernestine: A validation of GOMS for prediction and explanation of real-world task performance. *Human-Computer Interaction* (1993), 8, 3, 237-309.
John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. *In proceedings of CHI, 1990* (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115.
(These two papers present the CPM-GOMS variant)

Websites

- A GOMS PowerPoint presentation
- HCI (GOMS) course website at Virginia Tech
- A GOMS Tutorial
- Bonnie E. John's GOMS Page
- GOMSmodel.org

Tools that support GOMS models

- SOAR - a cognitive architecture tool that supports GOMS
- PGOMS: The Practical GOMS Tool
- QGOMS: Quick (and Dirty) GOMS - a successor of PGOMS