Márton Szakál
The Faculty of Economics,
Subotica, Serbia
tel.: +381 24 628 044
e-mail: marton@eccf.su.ac.yu

THE EVALUATION OF PERFFORMANCES OF THE USER INTERFACE WITH THE HELP OF PREDICTIVE MODELS

Summary
Predictive models give quantitative, predicative indicators of interface performances, without the real testing by users. It is of great importance in cases when users are unavailable. The non-exactness of qualitative methods and models in evaluating the user interface performances often goes in favor of the need for keeping the quantitative, experimentally acknowledged techniques with the strong theoretical basis in the design and expert armamentarium. The work points to the elementary characteristics of GOMS models and the use of the Information theory in eliminating their shortages.

Keywords
predictive models, GOMS, Key-stroke-level model, Accot-Zhai Law, Fitts law, Hick's law, Information Theory

ACM classification
H.1.1 Systems and Information Theory, *Information theory*

JEL classification
M1 – Business Administration, M15 – IT Management

INTRODUCTION

The most known predicative models of interaction between the user and the user interface is GOMS. GOMS has become known and accepted so it has become the generic name for the whole family of similar models deferring from one another only in emphasizing the aspect of the user's performances in modeling and predicting. We shall present the GOMS model, Keystroke-level model, Fitts' law, Hick's law and Accot-Zhai law. Besides, we shall also deal with the possibility of applying the Information theory in evaluating performances of the user interface.

1. THE GOMS MODEL

The GOMS model was developed in the early years of the 80s by Stu Card, Tom Moran and Allan Newell with the intention to model knowledge and cognitive processes included into interactions of the user and the user interface. GOMS is an acronym of Goal, Operators, Methods, Selection rules.

- **Goal**: It relates to the position the user wants to take (for instance, to find a web site).
- **Operators**: They relate to cognitive processes and physical actions the user should take over in order to attain a goal (for instance, to make decisions which Web machine to use, to think about it first, then to type the key words in the appropriate text box, and so on). The difference in relation to the goals is that the goals attain, and the operators execute.
- **Methods**: they are the scientific procedures for attaining goals. They consist of the exact sequence of steps (for instance, to bring the mouse pointer in the appropriate text box, left-click, type the key word on the keyboard, and click the command button "Search").
- **Selection rules**: They are used in choosing the method in case when more methods

are available. For example, after typing the key word in the appropriate box, the user can click the command button "SEARCH" or to press "ENTER".

We shall illustrate the GOMS model logic by an example of deleting the words in the MS Word text processor.

Goal: Delete a word in the sentence.

Method for deleting the word by the menu options:

Step 1: Note (remember), the word we should delete, has to be marked (selected) first.
Step 2: Note, the command for deleting is CLEAR.
Step 3: Note, command CLEAR is in the menu Edit.
Step 4: Select the word and the item CLEAR in the menu Edit.

Method for deleting the word by the button DEL:

Step 1: Note where the cursor is in relation to the word for deleting.
Step 2: Note where the key for deleting is.
Step 3: Press the button "DEL' as many as there are the letters in the word for deleting.

Operators used in the above cited methods:

- Click by the mouse
- Moving the mouse pointer across the text
- Choice of menu items
- Press the button on the keyboard

Rules of selection:

1. Use 'text deleting' by the mouse and the choice of menu items only if you delete a lot of text.
2. Use 'text deleting' by the key board if you delete a small number of letters.

2. THE KEY STROKE-LEVEL MODEL

The key stroke-level model differs from the GOMS model because it gives numerical predictions of user performances. The alternative solutions of interface designs can be compared based on the time necessary to carry out the task, respecting different strategies.

Card et al (1983), analyzing a great number of the previously done empirical studies of user performances, derived the standard set of approximate time for the main types of operation - physical and mental ones (Table one).

Further researches established that the speed of typing depends on the text contents being typed by the text processor. Most operators need 0.5 sec per character in typing senseless words with accidentally arranged letters, and for typing e-mail addresses 0.75 sec. per character.

The time needed for performing the concrete task is computed if we first describe the sequence of actions necessary to do and then sum the approximate time:

$$T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R$$

The logic on which the key stroke-level model is based can be best illustrated by the example in which we should find out the necessary time for inserting the word IS in the next sentence by using Microsoft Word, i. e.

Table 1.

Name of the operator	Description	Necessary time (sec)
K	Press on the button - average time	0.35
	Exceptionally experienced operator(135 words per Minute)	0.08
	Experienced operator (55 words per minute)	0.22
	Average operator (40 words per minute)	0.28
	Inexperienced operator	1.20
	Press on the button Shift or Control	0.08
P	Bringing the mouse pointer in the desired place on the screen	1.10
P_1	Mouse click	0.20
H	Moving the hand on the keyboard (from another device)	0.40
D	Line drawing by the mouse	Depends on the line length
M	Mental preparation for the action (decision-making)	1.35
R (t)	System response time - it is counted only if it causes waiting the user while performing the task.	t

Typing speed dependent
on the text contents

becomes

*Typing speed **is** dependent*
on the text contents.

First, the user should make decision what to do. Suppose he/she read the sentence. To start, he/she should choose the method; it is the mental operation (operator M). Then, he/she should position the cursor in the appropriate place in the sentence, moving the hand from the keyboard on the mouse (operator H). After that he/she performs, by moving the mouse, the pointer positioning (operator P), and at last, clicks in front of the word "dependent" (operator P_1). Then, he/she moves the hand from the mouse to the keyboard (operator H), thinks about the letter to press (operator M), presses the keys with the letters I and S (twice the operator K and at last, presses the space bar (operator K). Therefore,

Mental preparation (M)	1.35
Moving the hand from the keyboard on the mouse (H)	0.40
Positioning the mouse pointer (P)	1.10
Click of the mouse (P_1)	0.20
Moving the hand from mouse on the keyboard (H)	0.40
Mental preparation (M)	1.35
Typing the letter "I" (experienced operator)	0.22
Typing the letter "S" (experienced operator)	0.22
Typing the space bar (experienced operator)	0.22
Total needed time:	**5.46**

If there are too many components, their grouping is done, and it can be written as follows:

$$2\,(M) + 2\,(H) + 1\,(P) + 1\,(P_1) + 3\,(K) =$$
$$= 2.70 + 0.80 + 1.10 + 0.2 + 0.66 = 5.46$$

It seems too much for entering two letters. The problem is in the controversial characteristic of the model that giving the operator M (mental preparation) is left to the sense and prejudice of the researcher. It is also the problem that, similar to the time needed for pressing the button (the time differs depending on the operator's experience) and the time needed for the mental preparation, differs between 0.5 second and over one minute. For eliminating (partly) this shortage a heuristic for giving the operator of mental preparation is developed:

Rule 0: Initial inserting the candidate M

Inserting M in front of every K and every P means selecting the command, but not in front of every P that means the choice of the argument of a command.

Rule 1: Deleting the anticipated Ms

If the operator standing behind M is completely anticipated by the previous operator, delete M. For example, if we want to move the mouse pointer to the command button in order to click the command button, then P M K becomes P K.

Rule 2: Deleting Ms inside the cognitive units

If the string M K belongs to the broader cognitive unit, then we delete all Ms inside the cognitive unit, except the first M. The cognitive unit (entirety) is a continual sequence of the typed characters forming the name of the command or they the argument of a command. For example, the command COPY under the DOS surrounding.

Rule 3: Deleting Ms in front of consequential terminators

If the redundant delimiter M is at the end of the cognitive unit, as, for example, the command delimiter after which the argument delimiter follows, then M in front of it is deleted.

Rule 4: Deleting Ms that are the command terminators

If K is a delimiter behind the constant string, for example, the command name or any typed entity that is not changed, always the same, any time when is used, then M in front of it is deleted because adding the delimiters becomes a habitual action. Then the delimiter becomes part of the string and it does not require an additional M. But, if K is the argument delimiter or any other string that appears rarely, then M remains.

Rule 5: Deleting the folded Ms

M is not necessary to stand besides R when the user waits the system answer.

Notes: The string is a sequence of the character. The delimiter is a character denoting the start or the end of sensible entirety, series of characters, as the phone number or word in the natural language. For example, the space bars are delimiters for the majority of words, the full stop for sentences, and so on. Operators are K, P, and H.

The problem relating to this model is that it can model only a small set of highly routine tasks of entering data, where the user is treated as an expert, as the model does not expect the possibility of errors. We must emphasize that the GOMS version developed later on, called Natural GOMS language - NGOMSI, takes into consideration the behavior of a non-expert, as well as the time needed for learning. It makes difficult, often impossible, a prediction in case of an average user who uses a very flexible system. Besides, many factors influence the average user as fatigue, learning effects, social and organizational factors, and so on. The model also neglects that the majority of users do not perform their tasks sequentially, but they apply the sort of multi-tasking because of continual breaks and talks with their associates.

For all these cited limitations, the key stroke-level model is valid only if it works the prediction of predictable behaviors (typical for a typical user), or the comparison of different methods of performing tasks (different interface versions) being short and clearly defined.

3. THE FITTS LAW

This model performs the prediction of time needed the pointer device to bring to the desired position on the display. The predicted time functions to determine the object size (in an inverse proportion) on which the pointer should be placed and the distance of the pointer from the object (direct proportion). One of the biggest advantages of this mode is that it helps designers to make decision where to put the concrete component (command button, option button, text box, and so on), what size the component should be and how big the space between components should be.

$$T = \mathrm{k} \log_2 (D/S + 0.5),\ \mathrm{k} \sim 100\ \mathrm{m/sec}$$

or

$$T = \mathrm{a} + b \log_2 (D/S + 1),$$

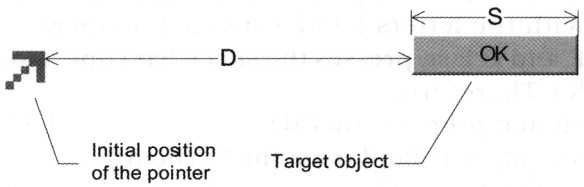
where a and b are determined experimentally

T - necessary time the pointer brings the mouse over the desired object

D - distance between the mouse pointer and the object

S - size of the object

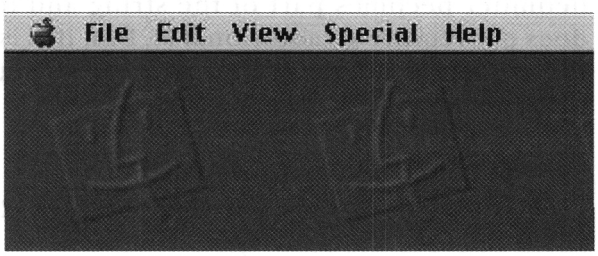
Figure 1.

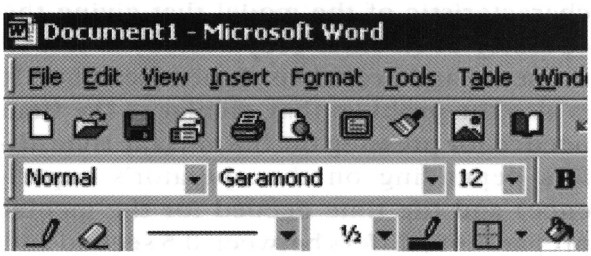


If it is a two-dimensional target, we take into consideration the fewer dimensions (width and height). The bigger object, the less time is necessary the pointer positions on it. Therefore, the interfaces with bigger command objects are easier (or even faster) for using than those with smaller components.

The majority of designers do not pay attention to the corner of the screen (or the window, if the window is maximized). The formula predicts that the mouse can be positioned the fastest on objects that are in the corners of the screen (the so-called pinning effect - the mouse pointer is impossible to put outside the edges of the window). Therefore, we can explain why the menu line on Mac OS is faster available in relation to the menu line of Windows.

Figure 2.





Suppose that the width of the menu line is 5 mm and the width of the title line another 5 mm. The fact that we do not need to stop the mouse pointer 5 mm before the edge of the screen (or the maximized window), the menu line in Mac practically takes the endless dimension. Empirical experiences show (Raskin, 2000) that, from the average distance of the mouse pointer from the menu line of 80 mm, 256 msec is necessary in Mac to bring the pointer to the menu line, and even 663 msec in Windows OS. Of course, this time is predicted only for moving the mouse. The additional 0.1 sec is necessary to click (in the keystroke-level model 0.2 sec is predicted for pressing the button, but it also means the same time for releasing the click). In the typical experimental situation, it is needed to add another 0. 25 seconds as the time of human reaction before starting the mouse move, and it amounts 0.6 seconds for bringing the mouse in the menu line in Mac, and over 1 second in Windows.

Fitts Law explains too, that icons, under which the title is written, are easier to use because they are bigger in relation to the icons under which there is no title (if the pictures are of the same size). The space between the icons is a good practice because an accidental click outside the frame will not cause unwanted actions. However, two pixels of distance from the lower edge of the button START to the lower edge of the window, according to Fitts Law, do not make sense.

Fitts Law is valid for every kind of moves done with the help of pointer devices if these moves are small in relation to the size of the human body and if they are continuous.

Although Fitts published only two works about his law (1954-1964), there are hundreds of related researches initiated by his works in HCI (Human-Computer Interaction) literature, and probably, thousands of articles published in the broader context (psychological, before all). The Law, according to some authors, has caused the commercial use of the mouse in Xerox. Fitts Law was tested in different entering devices, in different circumstances, using hands, legs, moves of the eyes, and so on. It was even tested in water. It was tested on young, old, retarded people, and the people who took drugs.

4. THE ACCOT-ZHAI LAW

This Law predicts the average time needed for the navigation or pointer devices management through 2D path (tunnel, trajectory), with a view that the user brings the pointer from one end to another one as fast as possible, taking care that the pointer stays in the limited trajectory. This Law has an expressive application with the hierarchical cascade menus.

This Law is, in fact, the Fitts Law modified with the help of integral calculus and it is experimentally confirmed with the congruence anticipated with measured variables that even surprised the author of the model, Johnny Accot.

$$T = a + b\int_C \frac{ds}{W(s)}$$

where

T – average time needed for navigation through the area

C – path with the parametric s

W(s) – with of the path with s

a and b – experimentally established constants

By differentiating the both sides, we get:

$$\frac{ds}{dT} = \frac{W(s)}{b}$$

It points that the speed of the user is proportionate to the with of the tunnel. [1]

5. THE HICK'S LAW

Hick's Law builds onto the Fitts Law, taking into consideration the suppositions that the user, before performing an action, chooses first between n available alternatives. When we should choose between n alternative actions and when the probability of choosing every action is equal, the time needed for the choice of an alternative is proportionate to the logarithm with the basis 2 of the number of alternatives incremented with 1.

$$T = a + b \log_2 (n+1)$$

If the probability of choosing the alternative i is $p(i)$, then we can write:

[1] As it is easier to keep the car going on the winding road if the road is wider.

$$\sum_i p(i)\log_2(1/p(i)+1)$$

The coefficients a and *b* in the Hick Law can be dependent on many conditions (for example, how alternatives are presented, how much the user is familiar with the system, and so on). If the alternatives are presented so confusing, the value of coefficients a and *b* increases; the more routine of the user decreases *b*. According to the Hick Law, giving the user more choices simultaneously leads to a faster choice than the hierarchically organized options.

6. THE APPLICATION OF INFORMATION THEORY IN EVALUATING THE EFFICIENCY OF THE USER INTERFACE

In the previous chapters, we demonstrated that GOMS-based models (to some extent) are available to perform the prediction of time needed to the user to perform clearly defined task with the given interface. However, the explained models do not answer the question how fast the interface should be. A possible answer can be found in Information Theory. Therefore, the notion of information is taken in a technical sense, as data quantification that should be exchanged in communications.

Minimal information quantity the user should inform the interface to perform the task is independent on the interface design. If more information than minimal is needed to use the interface, then it is an indicator that the user spends his time unnecessary and puts great efforts, and the interface can be improved. On the other side, if the interface requires information quantity equal to minimal quantity, then a maximal information efficiency is attained (it does not means that the interface should not be improved according to criteria).

Information efficiency (E) is a relation of minimally needed information quantity for task performing and information quantity the user should give to the interface. Possible values of information efficiency are in the scope of 0 and 1. Information efficiency

Figure 3.

has the value of 0 when the user reports information to the interface that are quite unnecessary. Surprisingly, but there is a big percent of these segments of the interface, but a typical example is Message Box with the message and possibility of a click just on one command button (Figure 2).

The fact that this method takes into consideration the relationship of information quantity needed to perform the task and information quantity required from the user makes possible that different design alternatives of the interface have the same value of information efficiency, but different time for task performing. It is possible that the method with the better E indicator has the worse time indicator. For example, suppose there are two alternative design interfaces, by which an identical task is solved, but in two different ways: in the first case M K M K, in the second case M K K K. In the first case, it is necessary to enter only two characters, in the second case three. However, the first method, although slower[2], has the bigger information efficiency.

Information quantity is measured in bits. One bit is the command between two available alternatives. To choose between four alternatives, four bits are necessary. Three bits are enough for eight alternatives; i. e. information quantity is determined by the logarithm of the base two of the number of alternatives (Balaban, Ristic, Djurkovic, Trninic, 2005).

$$\log_2 n$$

where n is the number of alternatives. Hence, information quantity needed for one alternative:

$$(1/n)\log_2 n$$

Therefore, E=0 in case illustrated in Figure 3 ($1/1\ \log_2(1)=0$).

If the possibility of choosing alternatives is not equal for every alternative and if

[2] Based on the values in Table 1, it follows: $T_I = T_K + T_M + T_K = 0.35+1.35+0.35 = 2.05$ sec

alternative i has the possibility of choosing *p(i)*, then information quantity is associated on with that alternative.

$$p(i) \log_2 (1/p(i))$$

Suppose that we modified the code that generates the situation illustrated in Figure 3, so that the software shall check hypothetically in the next two minutes if the event of the click happened on the command button "OK". If the expected event happens in the anticipated period, the operation X will be performed. If the event of the click on the command button "OK" does not happen in the next two minutes in relation to the generation of the illustrated message box, the operation Y will be performed. If the possibility that we click on the command button "OK" having the mark **p** in the anticipated period, then it is possible that the expected event will not happen is **1-p.**

$$p \log_2 (1/p) + (1 - p) \log_2 (1/(1 - p))$$

if p=1/2, then we have

$$0.5 \times 1 + 0.5 \times 1 = 0.5 + 0.5 = \mathbf{1}.$$

If *p* is different from 1/2, E will be less than 1, and if p=0 or p=1, E will be 0.

This has an important message: information quantity contained in the message we can measure only in the context of the set of possible messages that are *perhaps* received (namely, the writing in the message box is not changed so a hypothetical user cannot know that he/she has only two minutes to click the command button "OK"). To calculate information quantity reported by the reception of a message, we have to know the probability that the message is sent. Information quantity in any message is independent on the other messages sent in the past or that will be sent in the future. It does not depend on the time or any other events.

We can calculate information quantity delivered by devices with no keyboards, too. For example, the screen is divided into two parts, the left click is YES, the right click is NO, and it means that there are two alternatives and we need 1 bit of information. If there is *n* area on the screen as a characteristic of the click target, $\log_2 n$ information is needed. If targets are not equal, information quantity does not change, only the necessary time to click the target because of their different size changes. If the possibility of the click on some targets is not the same, the same formula as with the keyboard is applied. The difference is that the button on the keyboard is pressed for 0. 35 seconds, for the operation with the on-screen object is necessary 1. 3 seconds, not taking into consideration the time needed to move the hand from the keyboard to the mouse.

REFERENCES

Cooper, A. i Reimann, R. (2003): About Face 2.0 (The Essentials of Interaction Design), John Wiley & Sons, New York, USA

Raskin, J. (2000): *Human Interface, The: New Directions for Designing Interactive Systems*, Addison Wesley Longman

Card S. et al (1983): The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, USA

Balaban, N., Ristić, Ž., Đurković, J., Trninić, J. (2005): *Informacioni sistemi u menadžmentu (Management Information systems)*, Savremena administracija, Beograd

Cooper, A. (1999): The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity, SAMS, Indianapolis, USA

Card S. et al (1983): The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, USA

Raskin, J. (2000): *Human Interface, The: New Directions for Designing Interactive Systems*, Addison Wesley Longman

Biography:

Marton Szakal is Assistant Professor at the Faculty of Economics, Novi Sad University, in the field of Information Systems and Engineering. He is primarily interested in user centric interterface development, human factors in technical communications, web usabiltiy, software development. He is coauthor of several works in the cited field.