OPERATIONAL COMPLEXITY OF DIRECT MANIPULATION TASKS IN A WINDOWS ENVIRONMENT

John R. Maltby
Faculty of Business and Computing
Southern Cross University
P.O. Box 157, East Lismore
NSW 2480, Australia
jmaltby@scu.edu.au

ABSTRACT

A method to quantify operational complexity of direct manipulation tasks in a windows environment is discussed. The method utilises a formula from communication theory due to Shannon and Weaver which describes the uncertainty $H(p_1,p_2,...p_n)$ in outcome of an event which is the result of a Markov process where the individual events have probabilities of occurrence $p_1,p_2,..p_n$. A taxonomy of basic windows operations is developed for Microsoft Windows and used to show that a windows dialogue can also be described as a Markov process. The H formula is then applied to determine the total complexity of a sequence of basic windows operations and thus provide some measure of the complexity of a given task as seen by the user. An estimate of the total entropy of a Microsoft Windows language source is obtained, indicating that the redundancy of windows dialogue is about 28%.

INTRODUCTION

Guidelines for graphical user interface (GUI) and screen designs are reasonably numerous (Galitz 1989; Rivlin, Lewis and Davies-Cooper 1990; Powell 1990; Marcus 1992; Mayhew 1992; Schneiderman 1992; Hix and Hartson 1993). Most of these guidelines appear sound and are clearly constructive in helping eliminate aspects of poor interface design, but they are all qualitative and are in general directed towards the interface design *process*. Some formal methods do exist for analysing and documenting user actions and operations: these include the Keystroke-level model (Card and Moran, 1980), Command Language Grammar (CLG) (Moran, 1981), the Goals, Operators, Methods and Selection (GOMS) model (Card, Moran and Newell, 1983), Task Action Grammar (TAG) (Payne and Green, 1986) and User Action Notation (UAN) (Hartson, Siochi and Hix, 1990); however, these methods are also in general directed towards the design process. Neither the guidelines nor the formal methods listed above can provide a quantitative method of analysis of interface design which will enable different interfaces to be compared on the basis of their complexity, usability and efficiency, as seen by the user when performing specific tasks.

In this paper, we attempt to address this problem for GUIs, firstly by describing a windows dialogue as a Markov process and secondly by utilising a formula from communication theory, originally derived by Shannon and Weaver (1949), in order to describe the complexity of a GUI. Specifically, we will choose the Microsoft Windows interface for our analysis and assume that the interface is directly manipulated using the usual one or two button mouse pointing device.

THE WINDOWS USER AS A LANGUAGE SOURCE

We begin by describing the nature of the communication process between a user and a GUI. In order to complete a task in a windows environment, a user has to sequentially undertake a series of actions and operations. This sequence is similar to transmitting a series of symbols down a half-duplex communication channel and waiting for a response after each symbol has been transmitted: in the case of a GUI, the interface will respond after each operation is transmitted by changing its state.

The messages transmitted to the interface from the user in this way are the result of user actions and operations. Thus we can consider the user as the source of a sequence of actions and operations which we will refer to as basic windows actions (BWAs) and basic windows operations (BWOs). BWAs are the actions undertaken by the user on the mouse and/or keyboard, with which the user manipulates instances of windows object classes (WOCs) in a WIMP GUI. BWOs are the operations performed on the interface which are necessary in order to complete a *task*, such as copying a file, cutting and pasting text, or creating a document with a word processor. Thus *Title bar* is an example of a WOC, *click* is an example of a BWA and *open window* is an example of a BWO. In constructing a sequence of BWOs, the user is bound by the rules of the windows language in the same way that an author is the source of an English sentence and is bound by the rules of the English language. We will thus refer to the user as a *language source* for the particular GUI being utilised.

Any language source can be considered as a stochastic system and thus be analysed as a Markov process in which the occurrence of a future state depends upon the probability of transition from an immediately preceding state. By assigning a symbol of the English language to each state S_i of a communication system, Shannon and Weaver (1949) use a Markov process to describe the transmission of a message which consists of a series of such symbols, where a symbol can be either a letter or a word unit. The transition probability for each state (i.e. the chance of occurrence of each symbol) is determined as a series of approximations: in the zero-order approximation, each symbol in a message is independent and has an equal probability of occurrence; in the first-order approximation, the symbols are independent but occur with the frequency of English text; in the second-order approximation, the probability of occurrence of each symbol depends upon the previous symbol in the message according to the structure of English text (this is known as a digram structure); in the third-order approximation, the probability of occurrence of each symbol depends upon the previous two symbols in the message according to the structure of English text (trigram structure); etc. In order to use the same technique to describe a windows dialogue, it is necessary to define the equivalent of alphanumeric and special symbols (or word units) in a windows system. We approach this problem by first describing user actions at their most primitive level using the User Action Notation.

USER ACTION NOTATION

User Action Notation (UAN) (Hartson, Siochi and Hix, 1990) can be used to describe GUI operations in a high degree of detail. Its purpose is to enable such operations to be explicitly detailed so that they can be unambiguously implemented by an interface programmer. By way of example, we will use it here to analyse a simple task with Microsoft Windows: that of copying a file from a directory on one drive to a directory on another. We will imagine that the task is carried out by a user with a specific Microsoft Windows set up in which the File Manager icon is minimized and that the user performs the steps described in Table 1 and illustrated in Figures 1 through 6.

Steps	BWO	Operation Description
Step 1 (Figure 1)	[xi]	Execute the file manager by double clicking on file manager icon.
Step 2	[mw]	Maximise the file manager window by clicking on the maximise button.
Step 3 (Figure 2)	[of]	Open the folder *files* in the C:\ drive directory tree by double clicking on the files folder icon. This operation makes visible the icon of the file which is to be copied to A:\ drive, which is *ass4.xls*.
Step 4 (Figure 3)	[vd]	View the directories of drive A:\ by double clicking on the drive icon and opening a new window. This operation obscures the file icon *ass4.xls* which is to be copied.
Step 5 (Figure 4)	[mo]	Move the A:\ window out of the way of the source file, *ass4.xls*
Step 6 (Figure 5)	[mo]	Drag the ass4.xls icon from the C:\FILES window to the A:\ window. This operation makes the target window inactive. It also completes the file copy task. All future operations are to restore the system to its original state.
Step 7 (Figure 6)	[ma]	Make active the A:\ window again so that it can be closed.
Step 8	[cw]	Close the A:\ window.
Step 9	[nw]	Minimise the file manager window. The system is now back in its original state, as represented in Figure 1.

Table 1: 9 steps of a file copy task

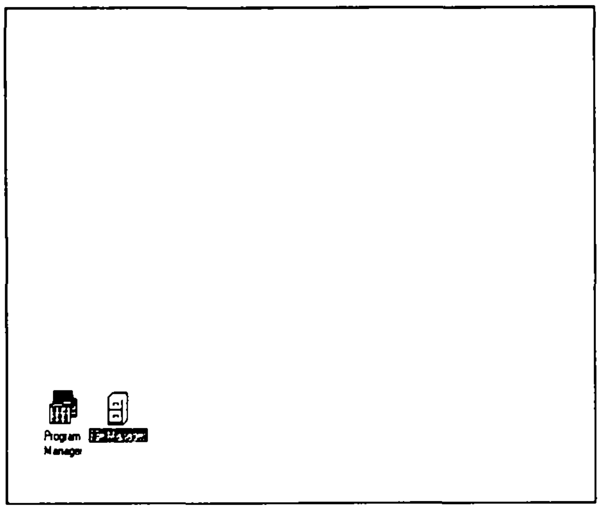
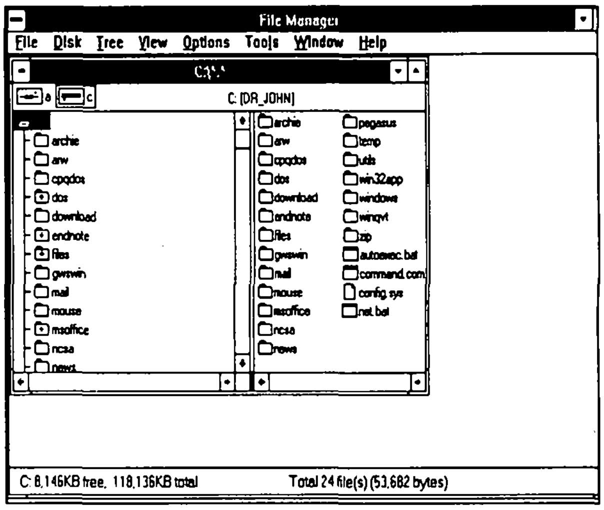
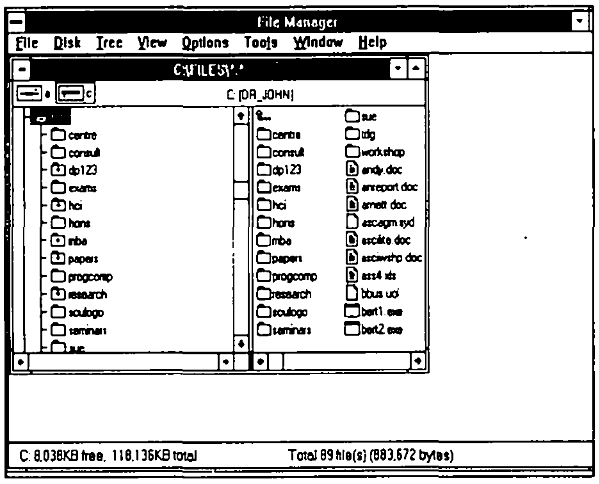


Figure 1: step 1, BWO [xi]



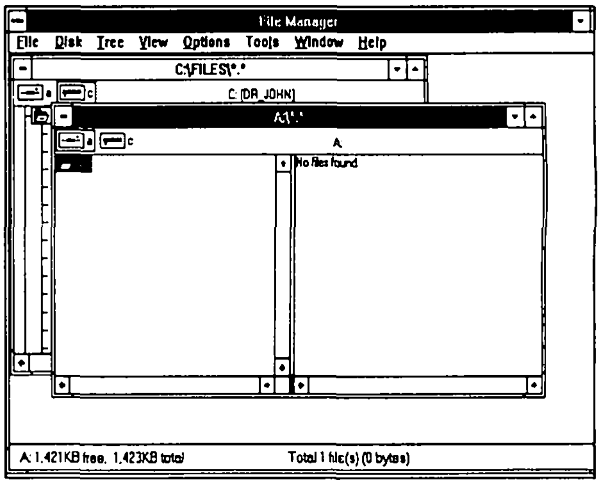
C: 8,146KB free, 118,136KB total Total 24 file(s) (53,682 bytes)

Figure 2: step 3, BWO [of]



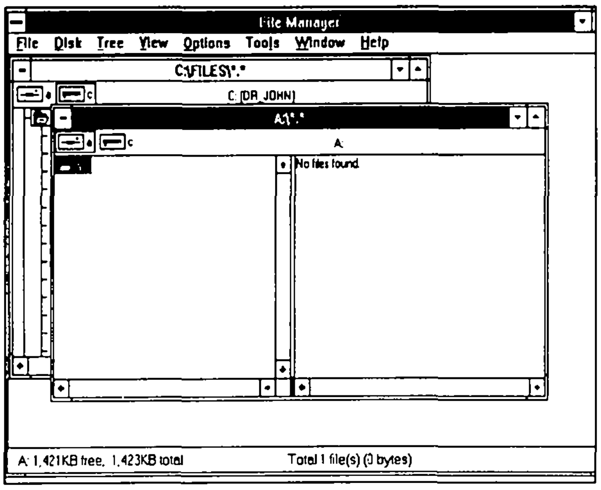
C: 8,038KB free, 118,136KB total Total 89 file(s) (883,672 bytes)

Figure 3: step 4, BWO [vd]



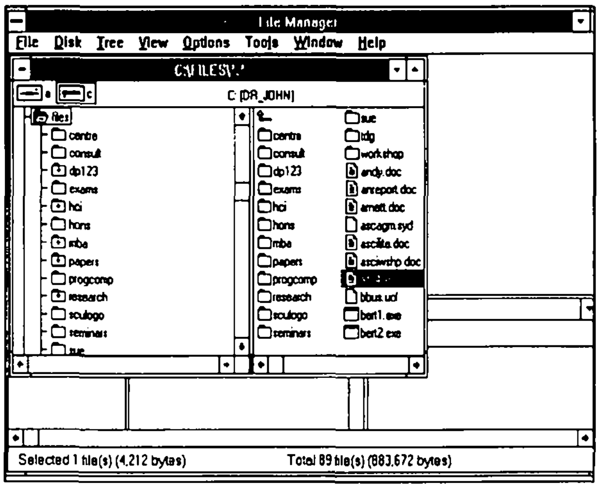
A: 1,421KB free, 1,423KB total Total 1 file(s) (0 bytes)

Figure 4: step 5, BWO [mo]



A: 1,421KB free, 1,423KB total Total 1 file(s) (0 bytes)

Figure 5: step 6, BWO [mo]



Selected 1 file(s) (4,212 bytes) Total 89 file(s) (883,672 bytes)

Figure 6: step 7, BWO [ma]

A Windows user will have noted that the above file copy task could have been completed in fewer operations: instead of opening a new window for the A drive (as in step 4, Figure 3), the user could have simply dragged the source file icon to the A drive icon. However, we will imagine in this case that either the user did not know that this was possible, or that the user wanted to check the contents of the disc in drive A before performing the copy. It is also of interest to note that the move object operation of step 6 (Figure 5) makes and leaves active the C drive window which consequently overlaps the A drive window. Not only this, but Windows immediately repositions the dragged icon to the top left of the right hand partition. In this case, the icon and accompanying text turns out to be totally obscured by the source window, as shown in step 7 (Figure 6), thus denying the user any feedback as to the result of the copy process.

A UAN description of the file copy task is given in Table 2, using the symbols recommended by Hix and Hartson (1993). As can be seen, the UAN details tasks at the user action level: the actions specified are *primitive user actions* (PUAs), so called because they cannot be broken down into sub-components. These actions include a mouse button press (Mv), a mouse button release (M^) and a mouse move (~). In UAN, the typographic symbols are chosen to symbolically represent the actual action; thus v is similar to a down arrow, ^ to an up arrow, while ~ gives the indication of movement and * is used to indicate repetition. A prime (') is used to indicate a specific instance of an object, such as a particular file. Other symbols to depict the interface feedback to the user include > to represent follow the cursor and ! to represent highlighting. The UAN symbols used in this example are summarised in Table 3.

It is seen that the analysis of even a simple task like copying a file can reveal design defects in the interface. In the task just described, at least one unnecessary action is required of the user because of step 4. If the destination window had been tiled with the source window, rather than cascaded, then the source file icon might still be visible, thus removing the need for step 5. If UAN were used by an interface designer to design an interface for copying a file, then the lines in Table 2 which depict step 5 would presumably not have been included. Such a design might not then have explicitly specified exactly how the secondary windows were to be arranged. This illustrates an important difference between using a notation for design and using it for analysis: when used for analysis, the notation can emphasise poor design because it explicitly requires the analyst to document all user actions; when used for design, however, the notation cannot compensate for any omissions of the designer. Thus certain user operations may be dictated by the nature of the interface rather than chosen by the user as direct steps towards goal accomplishment.

Although UAN provides a method of describing primitive user actions in detail, its analysis is at too low a level for our purposes. We require to describe a GUI dialogue in terms of more coarse actions and operations, such that each "symbol" in the dialogue is the equivalent of a word in a natural language. Towards this end, we look at classifications of basic windows actions and operations.

GUI:	**Microsoft Windows**	
TASK:	**copy file**	
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
~[icon’]		
Mv^ (t<n) Mv^	display(primary window’’) display(secondary window’)	
~[folder icon’]		
Mv^ (t<n) Mv^	folder icon! display(folder tree) display(folder contents)	selected = folder’
~[disk icon’]		
Mv^ (t<n) Mv^	display(secondary window’’)	selected = title bar’’
~[title bar’]		
Mv		
~[x,y]* ~[x’,y’]	outline(title bar’) > ~	
M^	@x’,y’ redisplay(secondary window’’)	
~[secondary window’]		
Mv^		selected = title bar’
~[file icon’]		
Mv	file icon!	selected = file icon’
~(x,y)* ~(x’’,y’’)	file icon > ~	
M^		
~[secondary window’’]		
Mv^	file icon-!	selected = title bar’’
~[close icon’’]		
Mv^ (t<n) Mv^		selected = title bar’
~[minimise icon’]		
Mv^	minimise(primary window)	

Table 2: UAN sequence for a file copy operation in Microsoft Windows

~	move the cursor
[X]	the context of the object X
v	depress
^	release
M	mouse device
‘	specific instance of an object
*	iterate
>	follow cursor
(t<n)	within a specific time
!	highlight object

Table 3: some common UAN Symbols

BASIC WINDOWS ACTIONS

We will define a BWA as an action commonly associated with manipulating a pointing device (such as a mouse), or with manipulating a key on the keyboard, and use the symbols < and > to denote a specific BWA. With a mouse, there are three basic actions that can be undertaken: these are a *click*, a *double click* and a *drag*. These actions can be defined using the UAN in the following way:

< c > ← Mv^

< cc > ← Mv^ (t < n) Mv^

< d > ← Mv ~ (x, y) * ~ (x' y') M^

Thus a click is the action of both pressing *and* releasing a mouse button, a double click is the action of doing this twice within a set time period, and a drag is the action of pressing a mouse button, moving the mouse to a specific position, and then releasing the mouse button.

With a keyboard, there are two basic actions that can be undertaken: these are a single key press and holding down one key whilst pressing and releasing another (as when using the SHIFT, ALT or CTRL keys, for example). These actions can be defined in the UAN as follows:

< k > ← Kv^

< kk > ← K'v K''v^ K'^

Note that the actual *motion* of a mouse *prior to selecting a target* is not defined as a BWA, although it is normally represented in UAN. Moving a mouse prior to target selection is here considered similar to moving a hand over a keyboard before deciding which key to press. Although such actions, known as homing, are considered in some models such as the Keystroke Level Model (Card, Moran and Newall, 1980), they are not relevant to the GUI language we are trying to develop, which does not include time as an independent variable. Neither have we included the symbols for motion (x,y)* etc. in the definition of <kk>, even though, strictly speaking, a finger (perhaps on the same hand) will be moved to a different key.

Finally, an action can be performed similar to <kk> which requires a key to be pressed and held down whilst a mouse button is pressed and released. This action can be represented in UAN thus:

< kb > ← Kv Mv^ K^

BASIC WINDOWS OPERATIONS

The Windows interface consists of a number of objects each of which is an instance of a particular windows object class (WOC). These objects can be manipulated by basic windows operations (BWOs) which in turn are performed using BWAs. There are some problems in defining and identifying window operations and a decision must be made on the granularity of the dialogue analysis. For example, should *scroll up* be a single operation, or should it be a goal (or sub-task) which can be achieved by a variety of different operations? If we consider scroll up to be a BWO, then it can be achieved by using a variety of BWAs; if we consider it to be a goal, then it can be achieved by using a variety of BWOs.

We will choose to consider scroll up to be a goal *because it can be achieved by manipulating one or more of a variety of different WOCs.* Let us represent goals by using curly braces, so that scroll up is written {scroll up}. Given this starting point, it is then clear that in Microsoft Windows the goal of {scroll up} can be reached by performing one of the following operations:

1. click on the scroll up icon at the top of the scroll bar;
2. slide (by dragging) the elevator on the scroll bar in the up direction;

Let us identify each of these operations by using two letter mnemonics in square brackets. Thus we will let [su] stand for the BWO scroll up icon and [ve] stand for the BWO slide vertical elevator. Using this nomenclature, the relationship between basic windows operations, basic windows actions and primitive user actions for these two methods of reaching the goal {scroll up} is as shown in Figure 7. Each method shown requires a different BWO and each BWO requires various combinations of PUAs. In the first method, the mouse is clicked upon the scroll up icon; in the second method, the vertical elevator is dragged upwards.

$$\{\text{scroll up}\} \leftarrow \begin{cases} [\text{su}] \leftarrow < c > \leftarrow\sim [\text{scroll up icon}]\ \text{Mv}\wedge \\ [\text{ve}] \leftarrow < d > \leftarrow\sim [\text{vertical elevator}]\ \text{Mv}\ \sim(x,y)^{*}\ \sim(x',y')\ \text{M}\wedge \end{cases}$$

Figure 7: decomposition of the goal {scroll up}

Proceeding in this manner, we identify 26 basic window operations as listed in Table 4. This list is clearly not exhaustive and is subject to addition and refinement through further investigation.

The window object classes and the basic window operations required to manipulate window objects to perform tasks are listed in Table 4. The basic window action or actions necessary to perform each operation are given in the last column of this table. In this column, alternative BWAs for the same BWO are indicated by the | symbol. Thus

[d]|[c] indicates that the desired BWO can be achieved by either dragging or by a single click. This particular alternative occurs for the BWO [so], where the operation can be performed by opening the menu and then dragging the mouse pointer to the required option in a single operation, or by clicking on a menu option of an already open menu.

BWO	BWO code	BWO	BWO code
close folder	[cf]	scroll down	[sd]
close window	[cw]	select icon	[si]
scroll left/right	[he]	scroll left	[sl]
make active	[ma]	select option	[so]
move object	[mo]	scroll right	[sr]
maximize window	[mw]	select text	[st]
minimize window	[nw]	scroll up	[su]
open folder	[of]	type text	[tt]
open menu	[om]	view directories	[vd]
open window	[ow]	scroll up/down	[ve]
position cursor	[pc]	execute icon	[xi]
resize frame	[rf]	execute option	[xo]
restore window	[rw]	execute task	[xt]

Table 4: basic window operations

Different BWOs can be performed on the same WOC. For example, a program group icon can be moved [mo] or opened into a window [ow]. In this case, these different operations require different basic window actions: to perform [mo] a <d> must be executed; to perform [ow] a <cc> must be executed. But this is not always the case. For the program group icon, both [mo] and [rw] can be performed with the same basic window action, that of <d>. However, the action is performed in different ways. In order to move the group icon, the drag must be performed on the icon itself; in order to restore the program group window, the drag must be performed on the program group menu.

Note also that different BWOs can be performed to achieve the same functional goal and obtain the same response from the GUI. For example, it is possible to execute a program from the Program Manager by performing any one of 3 different operations: [xi], or the combined operations [si] + [so] or [so] + [tt] + [xt]. It is possible to execute a program from the File Manager by performing the operation [xo], and in the case of an application such as Microsoft Office, it is also possible to execute a program by performing the operation [xt]. It is not altogether clear at this stage how different BWOs should be selected and classified; this is still the subject of on-going research.

Given that we can identify a finite number of basic window operations, it becomes possible to record the sequences of such operations which are necessary to perform specific tasks. Thus the sequence of operations of the file copy task described in steps 1 through 9 and depicted in Figures 1 though 6 is

[xi][mw][of][vd][mo][mo][ma][cw][mw], as detailed in Table 1. This includes those operations required to return the system to its original state.
In order to analyse the complexity of this task, it is necessary to know the initial and digram frequencies which dictate the structure of a Windows dialogue.

WOC	Possible BWOs	Required BWAs	WOC	Possible BWOs	Required BWAs
title bar	[ma]	<c>	program icon	[xi]	<cc>
	[mo]	<d>		[si]	<c>
close button	[cw]	<cc>		[mo]	<d>
	[om]	<c>	scroll up icon	[su]	<c>
	[so]	<d>|<c>	scroll down icon	[sd]	<c>
minimize button	[nw]	<c>	scroll left icon	[sl]	<c>
maximize button	[mw]	<c>	scroll right icon	[sr]	<c>
restore button	[rw]	<c>	hor. elevator butn.	[he]	<d>
window	[ma]	<c>	ver. elevator butn.	[ve]	<d>
window frame	[rf]	<d>	disk icon	[vd]	<c>|<cc>
active menu item	[so]	<c>	folder/file icon	[of]	<c>|<cc>
prog. group icon	[si]	<c>		[cf]	<c>|<cc>
	[om]	<c>		[mo]	<d>
	[so]	<d>|<c>	tool icon	[xt]	<c>
	[ow]	<cc>	document	[pc]	<c>
	[mo]	<d>	text	[st]	<d>
	[rw]	<d>		[xo]	<cc>|<c+c>
	[mw]	<d>	keyboard	[tt]	<k>

Table 5: window object classes and corresponding operations and actions

CONSTRUCTION OF A TRANSITION PROBABILITY MATRIX FOR BASIC WINDOW OPERATIONS

Comprehensive data has yet to be collected in order to determine initial and digram frequencies of each BWO. However, a pilot study of 462 BWOs over several different tasks was made by the author logging his own operations during normal work sessions on a computer. The tasks included file management sessions and document production using Microsoft Word, Excel and Powerpoint. The study yielded the frequency information in Tables 6 and 7.

cf	cw	he	ma	mo	mw	nw	of	om	ow	pc	rf	rw	sd	si	sl	so	sr	st	su	tt	vd	ve	xi	xo	xt	total
1	14	0	3	8	5	6	21	0	2	103	1	0	48	1	3	25	1	25	18	78	2	2	7	5	83	462

Table 6: initial frequencies for basic window operations

The initial frequencies of occurrence of the various BWOs are as shown in Table 6. Note that during these short sessions there are 3 operations that were never performed by the user, these being [he], [om] and [rw]. This may be a reflection of the user's windows language style as well as a reflection of the very small sample size.

The digram transition frequencies are given in Table 7. In this table, the frequency of occurrence of each BWO is dependent upon the previous BWO; thus each cell contains the frequency of occurrence with which a BWO *j* follows a BWO *i* in the sample sequence of 462 BWOs. The data can be normalised to produce a table of transition probabilities by applying the condition

$$\sum_{j} p_{ij} = 1 \qquad \text{for all } i$$

to each row.

Pi(j)													j														
		cf	cw	he	ma	mo	mw	nw	of	om	ow	pc	rf	rw	sd	sl	sl	so	sr	st	su	tt	vd	ve	xl	xo	xt
	cf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	cw	1	2	0	0	0	0	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	4
	he	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ma	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	mo	0	0	0	3	2	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
	mw	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
	nw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
	of	0	0	0	0	1	0	0	2	0	1	0	0	0	10	0	0	1	0	1	0	0	1	0	0	3	1
	om	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ow	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	pc	0	0	0	0	0	0	1	0	0	0	22	0	0	0	0	0	3	0	0	0	62	0	0	0	0	15
i	rf	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	rw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	sd	0	0	0	0	0	0	0	9	0	0	3	0	0	23	0	1	1	0	2	5	1	0	1	0	2	0
	sl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	sl	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
	so	0	1	0	1	0	0	0	1	0	0	6	0	0	1	1	0	1	0	6	1	3	0	0	0	0	3
	sr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	st	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	8	0	0	0	3	0	0	0	0	13
	su	0	1	0	0	0	0	1	3	0	0	1	0	0	2	0	0	1	0	1	5	1	0	0	0	0	2
	tt	0	0	0	0	0	0	0	1	0	0	56	0	0	5	0	0	2	1	5	1	0	0	1	0	0	6
	vd	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ve	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
	xl	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1
	xo	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	0	0	1	0	0	0	0	0	0	0
	xt	0	6	0	0	2	2	1	1	0	0	13	0	0	3	0	2	2	0	8	6	7	0	0	1	1	29

Table 7: digram frequencies from a sample of 462 basic window operations

THE ENTROPY EQUATION

We next require some method of quantifying the uncertainty in outcome at any point in a Markov chain of BWOs. For this we utilise a formula from communication theory due to

Shannon and Weaver (1949). Consider a message transmitted using a message alphabet consisting of *n* different types of symbol, where at any given point in the message the symbol types have the probabilities of occurrence p_1, p_2,...p_n. Shannon and Weaver proposed that the uncertainty in occurrence of a given symbol type at any particular place in such a message is given by $H(p_1, p_2,...p_n)$, where

$$H = -K\sum_{i=1}^{n} p_i \ln p_i \qquad ...(1)$$

and *K* is a positive constant. This form of *H* is identical to that of entropy as defined in statistical mechanics, where entropy is a measure of the *disorder* of a system which can be arranged in a large number of different ways.

By considering a long sequence of *N* symbols, Shannon and Weaver also obtain an expression for what they call the entropy of an information *source*. In the first-order approximation of a message transmitted from such a source, there will be a high probability that there will be p_i *N* occurrences of each *i*th symbol. It follows that the probability of the message will be approximately:

$$p = \prod_{i=1}^{n} p_i^{p_i N}$$

where there are *n* different possible symbols.

$$\therefore \quad \ln p = N\sum_{i=1}^{n} p_i \ln p_i \qquad ...(2)$$

Substituting for H from equation (1) and re-arranging gives

$$H_s = -\frac{K \ln p}{N} \qquad ...(3)$$

It is of interest to determine the *maximum* entropy that could be associated with the same source. This will occur when all the p_is are equal and given by $p_i = 1/n$. From equation (1) this is:

$$H_{MAX} = -K\sum p_i \ln p_i = -K \ln\frac{1}{n} = K \ln n \qquad ...(4)$$

The *relative* entropy of the information source is then given by:

$$H_R = \frac{H_s}{H_{MAX}} = -\frac{\ln p}{N \ln n} \qquad ...(5)$$

ENTROPY OF A WINDOWS SCREEN DISPLAY

A modification of equation (1) has been utilised by Bonsiepe (1968) to provide a measure of the complexity of a typographically designed page. Bonsiepe classified a page of text and graphics in terms of objects of different sizes. Objects were classified by width, height, and x and y spatial coordinates. By considering the proportion of objects in each class Bonsiepe calculated an aggregate value for *H*; this was then interpreted as a measure of the complexity of a given page layout. The same method has also been applied by Tullis (1983) to a layout of text on a computer terminal and by Comber and Maltby (1994) to objects in a Microsoft Windows graphical user interface.

In order to demonstrate the approach used by these workers, and to further clarify the nature of H, we will briefly consider how H might be calculated for a GUI screen which consists of different *classifications* of on-screen objects. Setting K equal to unity for convenience, the entropy of the screen display H_d is given by:

$$H_d = -\sum_{i=1}^{C} p_i \ln p_i \qquad ...(6)$$

where C = number of different classes of screen objects, and

$$p_i = \frac{n_i}{N}$$

where n_i = number of objects in class i and N = total number of objects in all classes. p_i now represents the proportion of screen objects in a particular class, rather than the probability of their occurrence.

To demonstrate possible uses of equation (6), consider static displays of objects on a windows screen which are classified by size and shape. Some examples are shown in Figure 8.

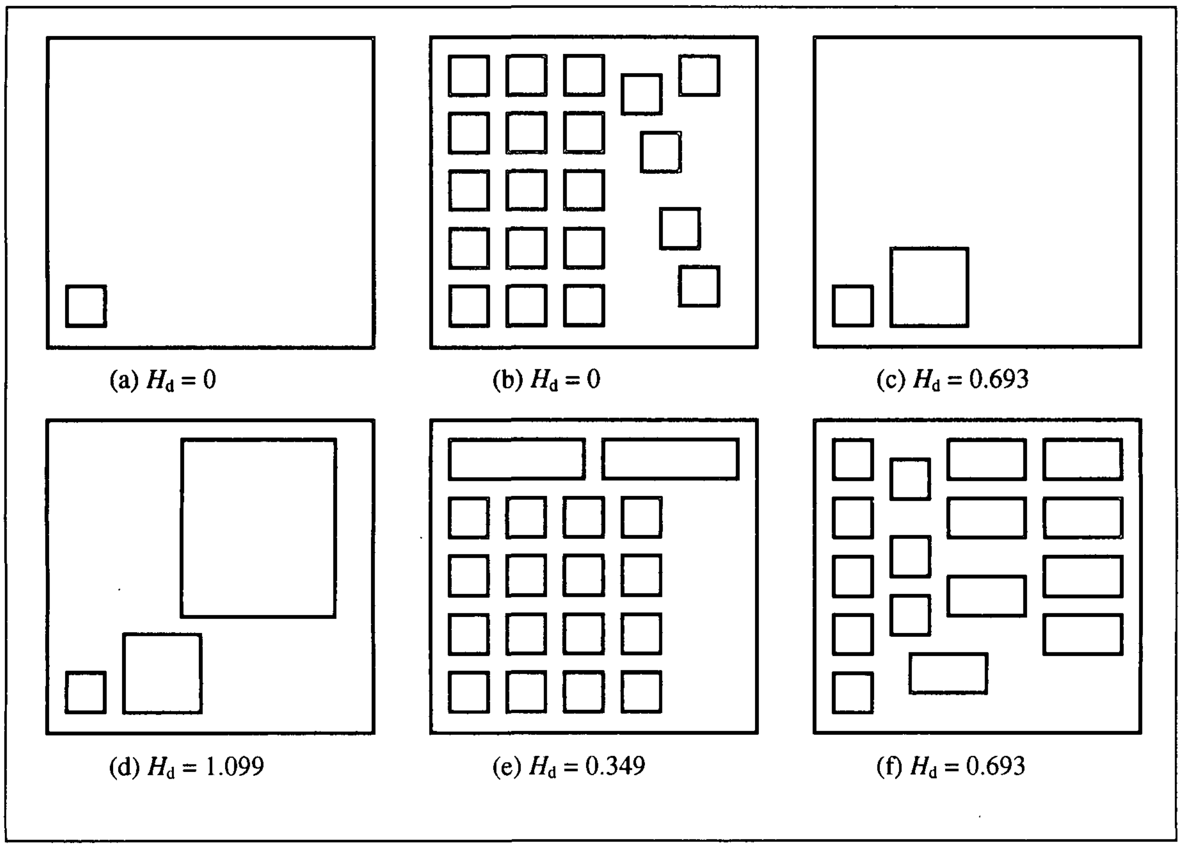


Figure 8: H_d values for screens of objects classified by size and shape

Each of the screens (a) to (f) in this figure contains a number of objects of different classes. The H_d values are determined in the following manner. Screens (a) and (b) contain only 1 class of object and so we have $C = 1$. In both cases n_1 / N is also 1 and so H_d is zero: both of these screens thus have the same minimum entropy. In contrast screen (c) has 2 classes, with 1 object in each. In this case H_d is given by

$$H_d = -\left(\frac{1}{2}\ln\frac{1}{2} + \frac{1}{2}\ln\frac{1}{2}\right) = 0.693$$

This is the maximum value of H_d possible with 2 classes; thus the entropy of a 2 object screen such as screen (a) is considerably greater is than the entropy of a 20 object screen such as screen (b). Clearly, if we add another class of object as in screen (d), the entropy will further increase, thus:

$$H_d = -\left(\frac{1}{3}\ln\frac{1}{3} + \frac{1}{3}\ln\frac{1}{3} + \frac{1}{3}\ln\frac{1}{3}\right) = 1.099$$

Since it is the *proportion* of objects in each class that is the determining factor, it is clear that for screen (e) we will have some value of H_d which is between the values of H_d for screens (b) and (c), thus:

$$H_d = -\left(\frac{2}{18}\ln\frac{2}{18} + \frac{2}{18}\ln\frac{2}{18}\right) = 0.349$$

The final example of screen (f) has equal numbers of objects in 2 classes; H_d is therefore the same as for screen (c), i.e. 0.693.

In the context of a windows screen, we can interpret these figures in several ways; they can be interpreted as a measure:

(a) of the degree of disorder of the screen;
(b) of the complexity of the screen in the context of different categories of information.
(c) in the uncertainty in choice of a windows object class by a user;

Interpretation (a) leads to the conclusion that Figures 8(a) and 8(b) have the same amount of disorder, and that Figures 8(c) and 8(f) have the same amount of disorder. Also, Figure 8(c) is more disordered than Figure 8(e), because there is a greater proportion of one object than the other. Although these disorder rankings do not appear to be intuitively the case, we shall see shortly that these interpretations may have validity in the context of different classes of user.

In terms of interpretation (b), a user may well see a screen consisting of a large number of uniform objects (such as icons) to be less complex than a screen consisting of a smaller number of objects of a variety of types (such as icons, windows, scroll bars and menus). Comber has suggested that a novice user would view all objects in a windows screen as separate entities, thus seeing a complex screen, whereas an expert user would be more likely to group items of a similar type (such as options of a menu) into an entity which is seen as a single object. Thus an expert user may see Figure 8(f) as Figure 8(c), which has the same H value. These assumptions need to be tested by collecting data on different classes of users.

Interpretation (c) suggests that it might be profitable to look at how entropy changes along different task *paths* by comparing different options open to the user, such as whether to perform an [xt] or an [so] operation.

We will now apply this method of determining display entropy to the file copy task summarised in Table 1. The results are given in Table 8.

	steps								
WOC	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
title bar	0	2	2	2	3	3	3	3	2
close button	0	2	2	2	3	3	2	3	2
minimize button	0	2	2	2	3	3	3	3	2
maximize button	0	2	1	1	2	2	2	2	1
restore button	0	0	1	1	1	1	1	1	1
window	0	1	1	1	2	2	2	2	1
window frame	0	2	2	2	3	3	3	3	2
active menu item	0	8	8	8	8	8	8	8	8
program icon	2	1	0	0	0	0	0	0	0
scroll up icon	0	1	1	1	0	1	1	1	1
scroll down icon	0	1	1	1	0	0	1	0	1
scroll left icon	0	1	1	1	0	0	1	0	1
scroll right icon	0	1	1	1	0	0	1	0	1
horizontal elevator button	0	1	1	1	0	0	1	0	1
vertical elevator button	0	1	1	1	0	1	1	1	1
disc icon	0	3	3	3	6	6	3	6	3
folder/file icon	0	36	36	39	4	35	39	37	39
Total number classes C =	1	16	16	16	10	12	16	12	16
Total number objects N =	2	65	64	67	35	68	75	70	59
Display entropy H_d =	0	1.45	1.75	1.12	1.72	1.41	1.31	1.56	0.94

Table 8: Display entropies and number of objects in each class for steps 1 to 9

It is seen from these figures that the screen in step 1 (Figure 1) possesses the minimum display entropy and that the screens in step 3 (Figure 2) and 5 (Figure 4) possess the maximum. We can liken the screen in step 1 to Figure 8(a), in step 3 to Figure 8(f) and in step 5 to Figure 8(d). This latter case is of interest, since it highlights that H_d is a measure of disorder or complexity of object *class*, the high value in this case being due to the additional window which has been opened. We might postulate that a novice user would find the screen in step 5 the most confusing, *because it offers the most even distribution of different object classes*, thus demanding of the user the maximum amount of knowledge about the interface. In contrast, the entropy of the screen in step 4 (Figure 3) is significantly lower, even though it contains a greater number of different classes, because of the *uneven distribution of different object classes*. In this screen there is a predominance of one particular class, that of folder/file icon. We might deduce that there is thus more certainty as to its *purpose*. This deduction leads us to consider ways in which the entropy of basic windows operations can be determined.

ENTROPY OF BASIC WINDOWS OPERATIONS

The above determinations of screen entropy relate to static displays in a similar (but not identical) manner to the work of Bonsiepe and Tullis. In this paper we further extend the application of entropy to an analysis of the basic windows operations required to complete a specific task, and hence determine what we will call the *operation* entropies H_0 associated with each screen. We do this by substituting the initial and digram probabilities from Tables 6 and 7 into equation (6). This enables us to determine the entropy of a windows operation at any point in a Markov chain of basic windows operations. Consider one again the file copy task described in Table 1 and depicted by the screens in Figures 1 through 6. Given the sequence of BWOs necessary to complete this task, it is possible to calculate the entropy H_0 of each operation by applying equation (6) and hence, since entropy is an additive quantity, to determine the total entropy of the task. The results of this determination are given in Table 9, which also includes the values of the display entropies and the total the number of windows objects present in each step. The entropy and object number data is plotted against step number in Figure 9. In this graph, the number of objects N has been normalised for comparison purposes. Although there are some similarities in trend between all three parameters as we move from screen to screen, the correlation between N and each type of entropy is low, being $r = 0.66$ for H_d and $r = 0.62$ for H_0. It is clear that entropy is measuring something more significant than simply the number of objects on a given screen.

Step	Task	Number of Objects N	Display Entropy H_d	Operation Entropy H_o
1	[xi]	2	0	1.6138
2	[mw]	65	1.45	3.1192
3	[of]	64	1.75	2.8832
4	[vd]	67	1.12	3.2477
5	[mo]	35	1.72	1.7918
6	[mo]*	68	1.41	3.3286
7	[ma]	75	1.31	3.4012
8	[cw]	70	1.56	2.6701
9	[nw]	59	0.94	1.3108

Table 9: H values for each screen display and BWO in a file copy task
(At * file copy task is completed)

The arrow in Figure 9 indicates the last operation required to complete the task. The total entropies of the operation up to this point are $H_d = 7.45$ and $H_0 = 15.93$. The operations after this point are those required to return the system to its original state and the total entropies of the complete task are $H_d = 11.30$ and $H_0 = 23.32$.

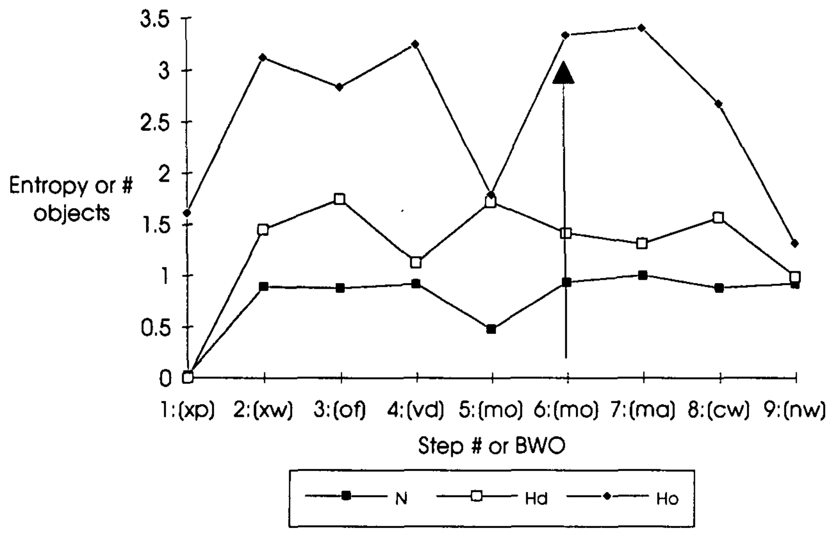


Figure 9: H versus BWO for a file copy task

Figure 10 highlights the differences between the 3 parameters by plotting the *change* in N, H_d and H_0 as a function of each transition as the user goes from screen to screen. The greatest differences occur in the transitions between the screens of steps 4 to 5 and steps 5 to 6, where H_d and H_0 change in opposite directions. In the transition from step 4 to 5, H_d increases while H_0 decreases; in the transition from step 5 to 6, these changes are reversed. Once again the key screen is that of step 5: in terms of *possible* user operations, H_d indicates that this is a complex screen; however, in terms of *probable* user operations, H_0 indicates the opposite, because there is a high probability that having just opened a window, the user will then move it to a new position. Perhaps H_d is an appropriate measure of complexity for the novice user and H_0 is an appropriate measure of complexity for the expert user.

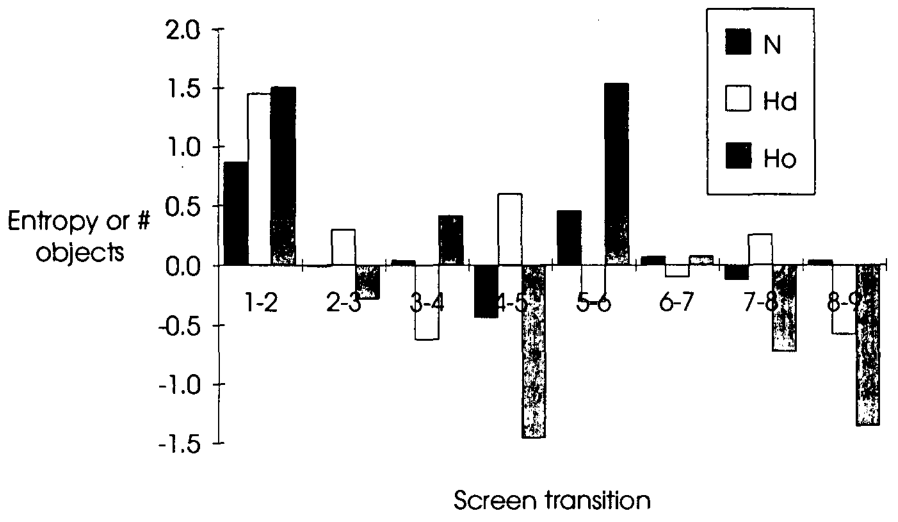


Figure 10: differences in N, H_d and H_0 as a function of screen transitions

We also note from both Figures 9 and 10 that the file copy task undertaken both starts and finishes with low H values. The opening values of H_d and H_o are a consequence of the fact that we chose the simplest possible type of opening screen. However, H_o decreases as the task approaches termination because the sequence of BWOs is a predictable one.

ENTROPY OF A WINDOWS LANGUAGE SOURCE

We can use equation (4) to determine to the first-order the entropy of a Windows language source by using the BWO frequencies given in Table 6. The total number of operations performed in the pilot survey is N = 462. From equation (3) we can also calculate:

$$\ln p = N \sum_{i=1}^{C} p_i \ln p_i = 1083.4$$

Substituting this and n = C = 26 into equation (5) yields a relative entropy of H_R = 72.0%.

DISCUSSION

Shannon and Weaver (1949) classify communication problems into the following three categories:

Category A:	the accuracy of transmission of symbols in a communication
Category B:	the precision with which the transmitted symbols convey their desired meaning
Category C:	the effectiveness with which the received meaning affects the receiver

If we consider our problem to be mainly that of quantifying the operational complexity of a GUI and of also providing a measure of its usability, then all three of these categories become relevant to GUI design and to a Windows dialogue. Category A involves issues not only of software accuracy, but also of interface design. Category B relates to the complexity and usability of an interface as seen by a user, quantities which are very dependent upon the user's interpretation of transmitted symbols (the windows object classes). Category C relates to the efficiency of a windows dialogue, which is dependent upon the effectiveness of a user's response to these same transmitted symbols. To give an extreme example, if a user mistakes a save icon for a close icon, and the application does not warn the user that the file is unsaved, then it is possible to lose an unacceptable amount of work.

In a discrete communication system, an information *source* is considered to select a desired sequence of *objects* (such as alphanumeric characters, picture or sound entities, or bits) in order to convey a particular message or amount of information. This sequence of objects is transmitted by the source to a receiver. The receiver translates the sequence into a form understandable by the information *destination*. If the destination is a human being, then the human being attempts to interpret the final *presentation* of objects. In this context, the word information has a wider meaning than message content. In terms of an information source, it relates to what *can* be transmitted rather than to what actually *is* transmitted. For sequential objects in a Markov chain, what can be transmitted is determined by the Markov transition probabilities. Thus when we determine an entropy value H_0 at a particular point in a Markov process, we are obtaining a measure of the uncertainty in the outcome of the next event in that process. This uncertainty can be a reflection of the freedom of choice of outcomes available to the information source, or it can be a measure of the predictability of outcomes as seen by the information destination.

In the context of a Windows language source, the entropy H_0 at any given point in a Markov chain of BWOs can be interpreted as a measure of the:

- uncertainty in choice by the novice user of the next BWO to transmit
- freedom of choice by the expert user of the next BWO to transmit
- complexity of the task being undertaken, as seen by the user
- ability of the Windows transceiver to predict the next BWO to be received

We have seen that the relative entropy of a Windows language source is $H_R = 72.0\%$. This figure can be interpreted as the amount of freedom experienced by the user in choosing a Windows operation to perform when undertaking a particular task. The complement of this quantity is the *redundancy* of the Windows destination, given by $R = 1 - H_R = 28.0\%$. This quantity is the proportion of the sequence of Windows operations which is determined by the statistical structure of the dialogue, and not by the free choice of the user. It is interesting to compare the redundancy of a Windows language source to the redundancy of the English language, which is about 50%. In other words, when composing a *meaningful* English sentence, an author has around 50% freedom in the choice of each successive word. Weaver notes that if the redundancy of the English language were much higher, it would be difficult to construct crossword puzzles, whereas if it was much lower, then it would be possible to construct three-dimensional crossword puzzles.

According to the data from the pilot study reported here, a Windows user has about 72% of freedom in the choice of each successive BWO. Whether we would expect a windows language to have a lower redundancy than a natural language such as English is an interesting question. In terms of freedom of choice of operations by the user, the redundancy of Windows is a reflection of the *versatility* of the interface. We might speculate that low redundancy is desirable for an expert user, who would wish for the freedom to complete a task in a variety of ways. However, such freedom may be an added complication for the novice user, because it increases the uncertainty in the choice of each operation to be undertaken in order to complete a given task..

CONCLUSIONS

A taxonomy of basic windows operations has been presented for the Microsoft Windows interface, enabling interface dialogue to be described as a Markov process. A pilot study has been undertaken to obtain a stochastic matrix for a Windows language source and this matrix has been used to calculate the entropy *H* of a specific Windows task and of the language source itself. It has been found that the relative entropy of the source and its redundancy are H_R = 72.0% and *R* = 28.0% respectively. Various interpretations of these parameters have been briefly discussed in the context of a GUI. It is concluded that additional research is necessary to further clarify the meaning of *H* and to develop methods by which the model can be applied to GUIs in general in order to quantify their operational complexity.

REFERENCES

Bonsiepe, G. A. (1968) A Method of Quantifying Order in Typographical Design, **Journal of Typographic Research** Vol 2, pp 203-220.

Card, S. K. and Moran, T. P. (1980) The Keystroke-Level Model for User Performance Time with Interactive Systems, **Communications of the ACM** Vol 23, pp 396-410.

Card, S. K., Moran, T. P. and Newell, A. (1983) **The Psychology of Human-Computer Interaction**, Hillsdale, NJ: Erlbaum.

Comber, T and Maltby, J. R. (1994) Screen Complexity and User Design Preferences in Windows Applications, **Proceedings of OZCHI '94**, CHISIG of the Ergonomics Society of Australia Annual Conference, Melbourne University (in press).

Galitz, W. O. (1989) **Handbook of Screen Format Design**, Wellesley, MA: QED Information Sciences, Inc.

Hartson, H. R., Siochi, A. C. and Hix, D. (1990) The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs, **ACM Transactions on Information Systems**, Vol 8(3), pp 181-203.

Marcus, A. (1992) **Graphic Design for Electronic Documents and User Interfaces**, New York: ACM Press / Addison-Wesley.

Mayhew, D. J. (1992) **Principles and Guidelines in Software User Interface Design**, Englewood Cliffs, NJ: Erlbaum.

Moran, T. P. (1981) The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems, **International Journal of Man-Machine Studies**, Vol 15, pp 3-51.

Payne, S. J. and Green, T. R. G. (1986) Task-action Grammars: a Model of the Mental Representation of Task Languages, **Human-Computer Interaction**, Vol 2, pp 93-133.

Powell, J. E. (1990) **Designing User Interfaces**, San Marcos, CA: Microtend Books.

Rivlin, C., Lewis, R. and Davies-Cooper, R. (1990) (ed) **Guidelines for Screen Design**, Oxford: The University Press.

Shannon, C. E. and Weaver, W. (1949) **The Mathematical Theory of Communication**, Urbana, Illinois: University of Illinois Press.

Schneiderman, B. (1992) **Designing the User Interface: Strategies for Effective Human-Computer Interaction**, Reading, MA: Addison-Wesley.

Tullis, T.S. (1983) The Formatting of Alphanumeric Displays: A Review and Analysis. **Human Factors**, Vol 25(6): pp 557-.679