Investigating Layout Complexity

Tim Comber and John Maltby

Southern Cross University, Faculty of Business & Computing, P.O. Box 157, Lismore, N.S.W. 2480, Australia
Phone: +61-66-203{117, 724} – Fax: +61-66-221724
E-mail:{tcomber, jmaltby}@scu.edu.au
WWW: http://www.scu.edu.au/buscomp/compmaths/{tcomber, staff}.html

Abstract

This paper presents work-in-progress in assessing the usefulness of a layout complexity metric in evaluating the usability of different screen designs. The metric is based on the Shannon formula from communication theory. Initially the metric was applied to thirteen Windows applications where thirty subjects were asked to rank screens on the basis of "good" design. A significant negative correlation was found between the subjects' rankings and the complexity ratings, indicating that users do not like "simple" screens. For the next stage a pilot application, "Launcher", was developed in Visual Basic to calculate complexity and collected usability data. Seven subjects provided some evidence that a layout complexity metric could be of benefit to the screen designer. However, though Launcher proved useful in collecting data, some problems need to be overcome, namely more concise data collection and a better method for building screens, before more data can be collected. The final version of "Launcher" should provide conclusive evidence of the worth of the layout complexity metric as well as showing that usability metrics can be built into the design environment.

Keywords

Layout complexity, GUI, interface, usability.

Introduction

Computer systems usually rely on VDTs for essential interaction between humans and computers. Users' acceptance of a computer system and performance with that system can be greatly influenced by the presentation of information on the computer screen [Tullis88b]. Shneiderman [Shneiderman92] agrees, stating that successful screen design is essential to most interactive systems. However, despite the importance of screen displays, there are few empirical studies relating to modern, bit-mapped screens, [Tullis88a, Galitz93] even though clearly most new computer systems use some form of GUI [Nielsen90].

Authors of guidelines, e.g., [Mayhew92, Galitz93] admonish the interface designer to keep the interface simple and well-organised but does this apply to a GUI? Are simple interfaces the most usable? And, how can the designer know that a simple interface has been achieved?

One answer is to use complexity theory to provide a numerical measure of the quality of the layout design. The complexity metric provides a measure of the horizontal and vertical alignment of objects and their positional alignment [Bonsiepe-68]. Layout complexity has been applied to alphanumeric displays on computer terminals with results that do show an effect on usability [Tullis81, Tullis83, Tullis88a, Tullis88b] but no effort has been made to determine if complexity theory can be usefully applied to more complex GUI's even though screen design guidelines frequently recommend that design goals should be to minimise the complexity of a display or make screens as predictable as possible [Mayhew92, Galitz93, Shneiderman92].

The screens that Tullis studied only displayed information and his research looked at information retrieval and users' preference. GUI screens can display information but they also present a dynamic interface to the underlying software and tend to be object-oriented and event-driven.

Firstly a survey was used to determine whether complexity theory could be applied to GUI design and if indeed it measured some aspect of design "quality" [Comber-94]. This was followed by a pilot experiment [Comber95] with layout complexity as the independent variable and effectiveness, learnability, and attitude as the dependent variables. The dependent variables are collectively referred to as "usability". The final version of "Launcher" should provide conclusive evidence of the worth of the layout complexity metric as well as showing that usability metrics can be built into the design environment.

1 Complexity Theory

1.1 Shannon: Mathematical Measure of Information Flow

Shannon [Shannon62] investigated mathematical measures for the amount of information produced by a communication process consisting of n classes of event, where an event is the transmission of a specific "unit" of information. In an English language communication, for example, we might consider the letters of the alphabet to be the communication units, in which case n = 26 (slightly more if we include spaces and punctuation symbols). Shannon obtained a formula for H, the measure of uncertainty in the occurrence of a specific event in a sequence of events:

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

where:

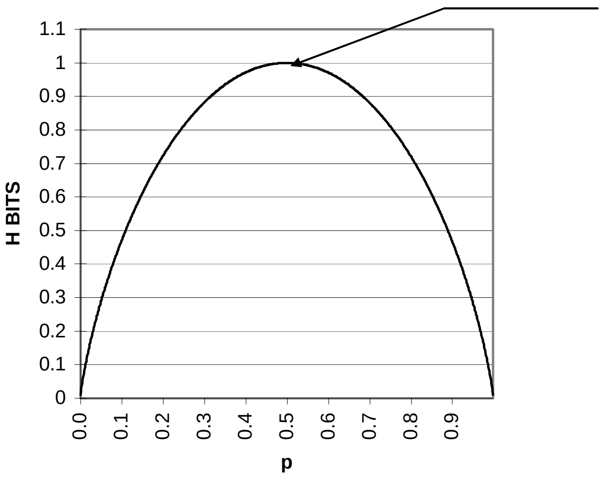
K = a positive constant
n = number of events
p_i = probability of occurrence of the *i*th event.

Shannon pointed out that the form of H is identical to that of entropy in statistical mechanics, where entropy is a measure of the disorder of a system that can be arranged in a large number of different ways. The meaning of H is best appreciated by considering a system with 2 event classes (equivalent to a 2-letter alphabet or a 2-word language). If in such a system the probabilities of each class of event are p and q, then (putting K = 1 for simplicity) the formula for H reduces to:

$$H = -(p\log p + q\log q) \qquad (2)$$

where q = 1 - p.

For this relationship, H is plotted in figure 1 as a function of p.



Entropy is a maximum if events in all classes occur with equal probability. ie when there is most uincertainty

Figure 1. Entropy in the case of two possibilities with probabilities, p *and (1 -* p*). (Modified from [Shannon62])*

It can be seen from figure 1 that there is the least uncertainty when the probabilities of one or the other event are highest and the most uncertainty when the probabilities are equal. Thus in a communication using a two-word language, the recipient is the most uncertain about which word is coming next if $p = q$, and the least uncertain if p = 0 or q = 0.

Shannon lists the advantages for using the H quantity:

- H becomes zero when there is no uncertainty.
- For any number of events, H is at its largest and equal to log n when all the probabilities are equal.
- Where there are joint events H is less than or equal to the sum of the individual H.

- As the probabilities approach equality H increases.
- The entropy of a joint event is the uncertainty of the known event plus the uncertainty of the remaining event.
- Knowing the uncertainty of one event does not increase the uncertainty of another joint event.

1.2 Weaver's Contribution to Shannon's Theory

In his commentaries on Shannon's mathematical theories of communication, Weaver [Shannon62] points out that communication includes not only speech but also pictures, music, ballet and so on. A GUI can be viewed as a communication system between CPU and user (figure 2).

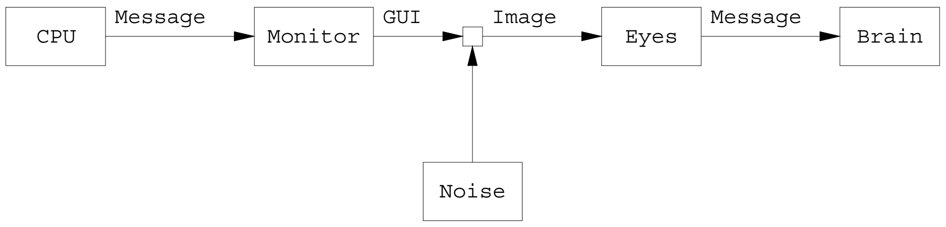


Figure 2. Diagram of a GUI communication system (after Shannon 1962)

Understanding this communication process has three levels (table 1). These levels overlap. It may appear that the theory only applies to the technical level but closer thinking reveals that problems at the technical level affect the semantics and effectiveness of communication. For example, a button with too small a font may not convey meaning and thus prevent the user from completing a task.

	Weaver	**GUI**
Technical	accurate transmission of data	layout, screen resolution etc
Semantic	attachment of meaning to the data	meaningful labels, error messages
Effective	changes in the recipient by the data	enables task completion

Table 1. Levels of the communication process

1.3 Information and Entropy

In Weavers' writings, information is thought of as a measure of the freedom of choice when selecting a binary event to send down a communication channel. This event can be either a single bit or a complete message. A channel capable of transmitting a single message from two alternatives is arbitrarily given an information value of unity: associated with this information value are the 2 possible messages, or meanings of the communication. Similarly, a channel capable of transmitting two binary messages has an information of 2 and 4 possible message combinations or 4 possible meanings; a channel with an information of 3 has 8 possible meanings, etc. According to Weaver, therefore, the information is proportional to $\log_2$ of the meanings. The entropy H of the transmission, however, depends upon the *probability* that a particular combination of messages will be sent at any given time. For a system with an information of unity, this will be according to Equation (2) and figure 1: if each message is equally likely to be transmitted, then $p = q$ in Equation (2) and H is a maximum. In general, for a system with an information of n, H is a maximum if all $2n$ messages are equiprobable.

For instance, in a GUI system consisting of a single check box, the user is free to check or de-check the box, resulting in the transmission of one of two alternative messages to the system. The information is thus unity and there are two possible system states. If there are three check boxes, then the system will have an information of 3 and 8 possible states.

However, the entropy of the system will depend upon the probability of occurrence of each check box state, and this in turn will depend upon the task being undertaken by the user and the nature of the GUI "language" being used [Maltby95a, Maltby95b]. Only if each of the *2*n check box states is equally likely to occur will H be a maximum.

These concepts apply in general to more complex situations. When a user runs a GUI based program, the GUI designer has used the basic building blocks of the GUI environment to communicate to the user. The user can start at one point and continue till a task is completed. When the user begins, any interaction object can be chosen, but once the first object has been chosen then probability can be used to indicate the next choice. For instance, if the user reads the label "Help" then the odds are high that the user would next press the "Help" button. The user's choice of the next object in a sequence is dependent upon the order of prior objects in the sequence.

Entropy in the physical sciences is a measure of the state of disorder of a system: the more disorder, then the higher the entropy. In communication theory, entropy describes the amount of uncertainty in the progress of a message. In a highly organised transmission the amount of information (entropy) is low and there is little randomness or choice.

The entropy of a message H can be compared to the maximum possible entropy H_{MAX} of the language to give the relative entropy. Subtracting this ratio from unity gives what Shannon calls the "redundancy" of the message, thus

$$R = 1 - H/H_{MAX} \qquad (3)$$

This is the amount of the message that is determined by the statistical rules of the message language and is not due to free choice. The loss of this amount of the message would not destroy the meaning of the message.

It is easy to conjecture just how much of a GUI interface is redundant. For instance, a common guideline is to place the "Exit" button on the bottom right-hand corner of the screen. If this guideline is followed then labelling the button "Exit" is redundant. If an icon is also placed on the button then that is redundant as well.

Unfortunately, the guideline is frequently ignored by designers and is not universally known to users. Therefore redundancy becomes important in a GUI, because users do not know the language.

Weaver points out that about 50% of the English language is redundant, that is about half of the letters or words used are open to the free choice of the user. Of course, one virtue of redundancy in the English language is that it allows the listener or reader to still get the meaning of a message even when some detail is missing e.g.:

1. Omit much words make text shorter.
2. Thxs, wx cax drxp oxt exerx thxrd xetxer, xnd xou xtixl maxagx prxttx wexl.
3. Thng ge a ltte tuger f w alo lav ou th spce. [Lindsay72, p.135].

Of course, it is harder to interpret a message with missing detail of this nature, and more effort must be made by the reader, but without redundancy in the language it would be impossible to interpret if any detail at all was missed out. A command language interface is a low entropy interface much like the third example for the English language.

For example, in the Unix operating system, *cp* stands for copy, *ls -l* means give a long listing of the files in the directory. The commands are often abbreviated and there is frequently only one way to do things. This lack of redundancy is one feature that makes command languages difficult to learn and remember. In contrast, GUI's have a much higher redundancy. Often a task can be completed using different methods such as direct manipulation, menus or keyboard shortcuts.

However, it is important to remember that the entropy of Equation (1) can be increased both by increasing the number of classes in the GUI language (ie the number of symbols) and by increasing the probability of use of each class. This latter can only be achieved by design: a badly designed object will be infrequently used and a well designed object will be frequently used.

Weaver observes that the best measure of the capacity of a communication channel is the amount of information that can be transmitted not the number of symbols or classes. By analogy, the entropy of a GUI is maximised by having objects of few classes with all classes equally usable and reduced by having objects of many classes with a wide range of "usabilities".

1.4 Bonsiepe: Application of Complexity Theory to Typography

One statistical interpretation of entropy is that it is a measure of the disorder of the system. This interpretation provides a justification for Bonsiepe [Bonsiepe68] to use the Shannon formula as a measure of the order or complexity for the typographic design of a printed page.

Bonsiepe believed that mathematics could provide design with "a series of instruments for the conscious and controlled generation of forms" [Bonsiepe68, p. 204]. This idea is now being extended for computer supported design [Vanderdonckt-

94d, Sears93, Hudson93] for example. However, Bonsiepe does take it for granted that "order is preferable to a state of disorder" [Bonsiepe68, p. 205] and offers no justification other than that creating order is the business of designers. A related issue is how to recognise order, particularly in multimedia applications where objects may not have simple symmetries [Vanderdonckt94c].

Bonsiepe identifies two types of order; system order and distribution order. System order is determined by classifying objects according to common widths and common heights and distribution order is determined by classifying objects by their distance from the top of the page and from the left side of the page. This, of course, is based on the top-to-bottom, left-to-right pattern of reading evidenced in Western culture.

Bonsiepe's technique is to draw contour lines around each typographical object. The proportion of objects in each class is then used to determine the complexity C of the layout using a modified version of the Shannon formula. This C corresponds to Shannon's H, the measure of the uncertainty in the occurrence of an event. Bonsiepe's formula states that the complexity C of a system is given by:

$$C = -N\sum_{i=1}^{n} p_i \log_2 p_i \qquad (4)$$

where:

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

and:

N = total number of objects (widths or heights, distance from top or side of page)

n = number of classes (number of unique widths, heights or positions)

n_i = number of objects in the ith class

p_i = proportion of the ith class.

Bonsiepe tested the applicability of this formula by comparing two versions of a printed catalogue. It was found that the new version was 39% more ordered than the original version.

Subjective observation agreed with the mathematical theory, and the formula gave a measure of the difference in perceived "complexity" or "orderliness" between the new and old versions. In essence, Bonsiepe's work offers a justification for the grid system commonly advocated for the layout of printed documents, e.g., [Porter83] and for computer screens, e.g., [Hudson93].

1.5 Tullis: Complexity Theory Applied to Computer Screens

[Tullis83] reviewed the literature dealing with computer-generated, alphanumeric monochromatic screen displays to understand how formatting affected the processing of the information by the viewer. One metric he used was Bonsiepe's layout complexity. Minimising layout complexity with tables, lists and vertical alignment increases the user's ability to predict the location of items and thus improves the viewer's chance of finding the desired information.

In other words, Tullis was attempting to lower the entropy of the system; to lower the freedom of choice of the viewer. When Tullis applied Bonsiepe's technique to screens that had been identified in the earlier study [Tullis81] as narrative and structured, he found that the structured screen returned a lower complexity figure than the narrative screen.

Tullis [Tullis88b] later decided to determine if the complexity measure was a useful usability metric. Again using alphanumeric data, he prepared 26 formats that were viewed by ten subjects in different trials.

He found that layout complexity did not help in predicting the time it takes for a user to find information. This is an interesting result. If there is less uncertainty about the placement of objects then it should be easier to find information.

However, he did find that it was an important predictor of users' rating of the usability of screens. In a second experiment using different displays and subjects, Tullis [Tullis88b] attempted to predict the subjective ratings. He found that, along with other measures, layout complexity helped to predict the users' rating of the usability of the different screens.

2 Usability and Complexity

This research aims to develop a metric for evaluating object placements in a graphical user interface based on complexity theory or to put it simply "where is the best place to put things".

This metric, along with others already available, should be capable of being incorporated into the software environment so that the software developer can have immediate feedback on the layout quality of the GUI.

It is hypothesised that there is a trade off between usability (U) and complexity C with a relationship of the form U = f(C) where U is a maximum for some intermediate value of C (figure 3).

As the complexity figure becomes smaller, it becomes more difficult to distinguish different interface objects and the interface takes on an artificial regularity. On the other hand, the interface becomes more predictable. At the other extreme as the interface approaches maximum complexity, it looks artificially irregular.

What is more important, it becomes impossible for the designer to group objects with similar functions on the basis of size or position. However, the increase in entropy does mean that the user has more information and therefore more choice of operations.

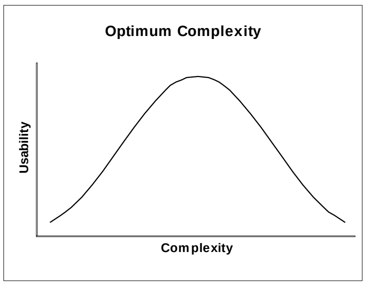


Figure 3. Relationship between complexity and usability

3 Research

3.1 Initial Investigation

Table 2 shows the results of applying Bonsiepe's technique to thirteen different Microsoft Windows applications. Four of the screens (MSRecorder, STW, Chartist and Rockford) are shown below demonstrating the range of complexity: figures 4, 5, 6 and 7. The total complexity, C, is given by C = $C_S + C_D$, where C_S and C_D are given by equation 4 with the p_i 's representing common widths and heights for C_S and positions on a page for C_D. The complexity per object C_O is also computed and is given by $C_O = C/N$.

It is seen that there is a large variation in complexity figures for the thirteen displays, with the complexity of the most complex display screen (from Rockford) being some 66 times greater than the complexity of the least complex screen (from Microsoft Recorder).

Application	Obj. No. N	System comp. C_S	Dist. Comp. C_D	Total C	Ratio C_O
MSRecorder	5	10.46	13.22	23.68	4.74
MSCalendar	17	76.59	101.28	177.87	7.54
Arachnid	60	96.22	388.89	485.11	8.09
MSCardfile	11	36.82	53.35	90.17	8.20
STW	23	50.75	122.60	173.35	8.60
Chartist	31	85.67	199.94	285.61	9.21
MSSolitaire	14	64.24	69.44	133.68	9.55
MSObjectPackager	23	89.73	143.55	233.28	10.14
ObjectVision	20	57.19	114.82	172.01	10.46
MSPaintbrush	61	239.61	467.89	707.50	11.60
MSWord	74	305.86	591.79	897.65	12.13
MSExcel	79	312.55	656.06	968.61	12.26
Rockford	104	582.42	989.56	1571.98	15.12

Table 2. Comparison of thirteen different screens in ratio order

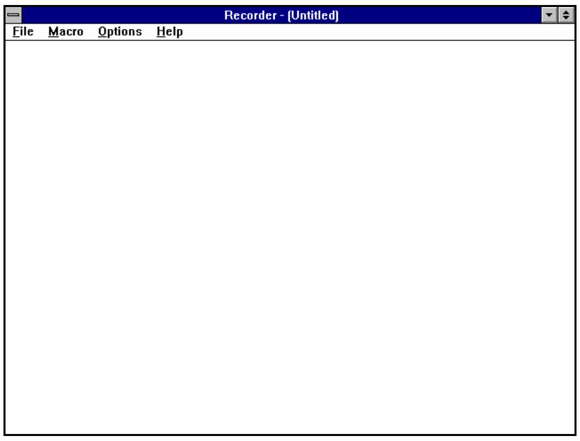
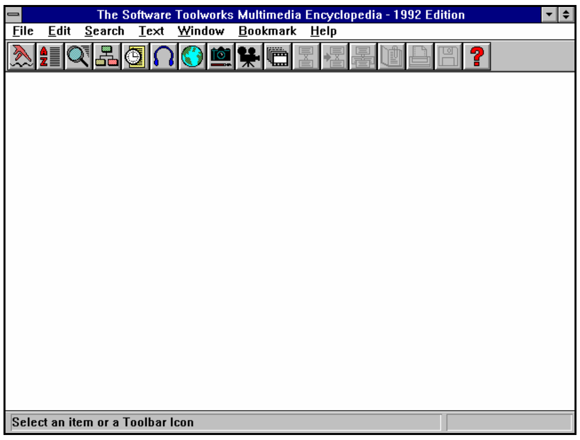


Figure 4. MSRecorder - Microsoft Recorder


The Software Toolworks Multimedia Encyclopedia - 1992 Edition
File Edit Search Text Window Bookmark Help

Select an item or a Toolbar Icon

Figure 5. STW: Software Toolworks Multimedia Encyclopedia

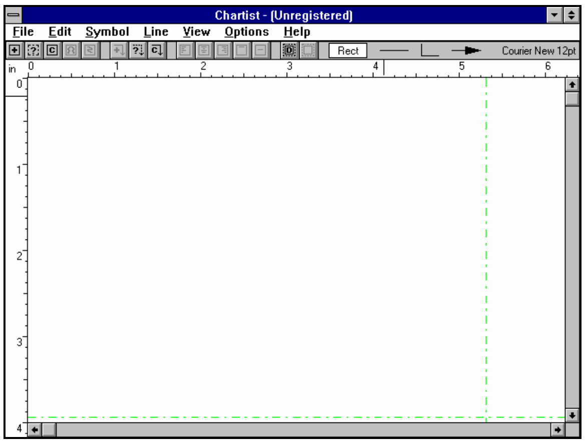
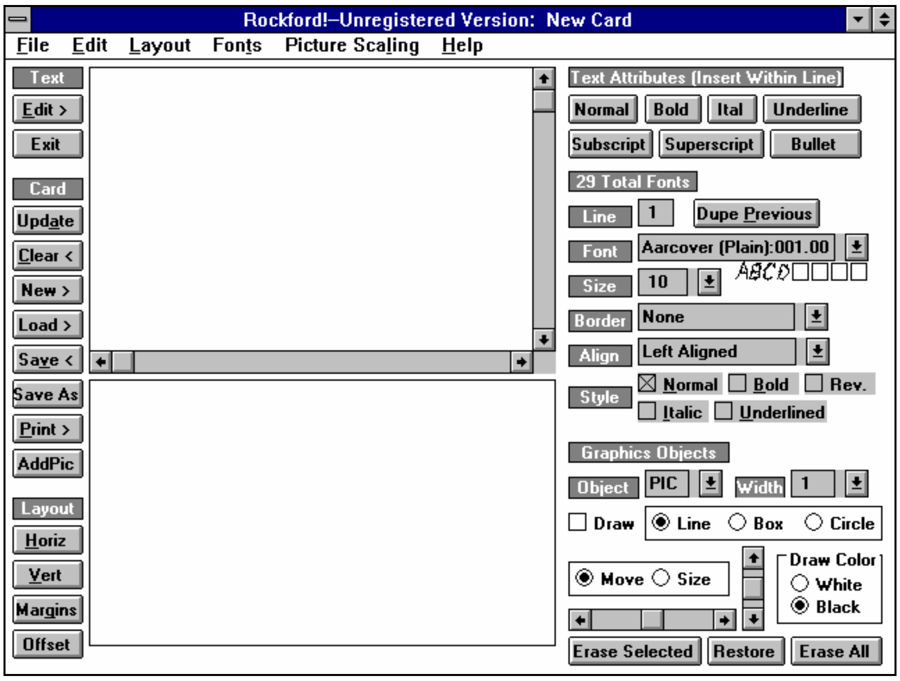

Chartist - [Unregistered]
File Edit Symbol Line View Options Help
Rect
Courier New 12pt
in
0
1
2
3
4
5
6

Figure 6. Chartist



Line 1
Font Aarcover (Plain):001.00
Size 10
Border None
Align Left Aligned
Style [✓ Normal] Bold Rev. Italic Underlined
Object PIC
Width 1
Draw [✓ Line] Box Circle
[✓ Move] Size
Draw Color White [✓ Black]

Figure 7. Rockford

3.1.1 Discussion

Both system order and distribution order are difficult to calculate manually. One good empirical measure of complexity might be the time it took to analyse an application. The more complex the layout of an interface, the more difficult it can be to determine the class of object.

Ideally a development environment such as Borland's IDE or the Visual Basic editor would calculate the size and position of objects and return a complexity figure automatically. Shneiderman [Shneiderman92] points out the lack of a computer program to do these computations for text screens though his recent work is attempting to remedy this [Shneiderman95].

3.1.2 Conclusions

The simplest measure of the layout complexity of a GUI screen is to count the number of objects. A screen with more objects is more complex than one with fewer objects. This does not take into account the difference between an ordered display and one where objects are scattered. The number of objects is also determined by the functionality of the interface.

An application that provides more functions needs more objects. Clearly layout complexity measures something but the question remains: does layout complexity matter? In other words, how is usability affected by interfaces exhibiting differing degrees of layout complexity?

3.2 Screen Complexity and User Design Preference in Windows Applications

3.2.1 Method

Both Bonsiepe and Tullis have indicated that designs with high values of C are less desirable than designs with low values of C; this would also intuitively seem to be the case. On this basis, it would be expected that if users were asked to rank application screens in order of "goodness" of their design, then the ranking would be similar to that given in table 3, i.e., Microsoft Recorder would be considered to be the best design and Rockford the worst.

A survey was therefore carried out to determine whether Bonsiepe's technique would provide a predictive measure for users' ranking of different designs. Subjects were recruited from the local campus (both students and staff) and from off-campus. All subjects were volunteers and no rewards were offered.

The survey took between 5 and 10 minutes to conduct. A grey-scale 300dpi laser print was made of each screen and inserted in a plastic envelope. They were asked to sort the screen prints from best design to worst design, with no ties. No attempt was made to define what was meant by "goodness" of design, this interpretation being left up to the subject.

3.2.2 Results

There was found to be a significant agreement in screen rankings among all thirty subjects, with Kendall's coefficient of concordance giving $W = 0.25$ and $\chi^2 = 91.1$ at a significance level of < 0.00005. The results indicate that there was a common interpretation of "goodness" of design. However, the distribution of the results was unexpected.

The least complex screen for either C or C_O is from MS Recorder. This screen was ranked as being the second worst design by 12 out of the 30 subjects. The most complex screen for either C or C_O is from Rockford. This screen was ranked as being the best design by 4 subjects, although 9 other subjects ranked it as the worst.

The rankings by user perception are compared with the rankings by complexity in table 3 and in figure 8. Whilst the rankings by C_O show some positive agreement with the rankings by C, it is seen that there is lack of such agreement between either of these rankings and the rankings by user perception. The Spearman correlation between ranking by perception and ranking by C gives a negative coefficient of $r_s = -0{\cdot}52$ at a significance level of 0.07, and a Spearman correlation between ranking by perception and ranking by C_O gives a negative coefficient of $r_s = -0{\cdot}47$ at a significance level of 0·11. Both of these correlations indicate that users show a greater preference for the more complex screens.

Application	ID	Mean Perceived Rank	Rank by total C	Rank by C_O
MS Paintbrush	1	4.4	10	10
MS Excel	2	4.5	12	12
MS Word	3	5.2	11	11
MS Solitaire	4	5.5	3	7
STW	5	6.0	6	5
Arachnid	6	6.1	9	3
ObjectVision	7	6.8	5	9
Chartist	8	7.0	8	6
MS Calendar	9	7.9	4	2
Rockford	10	8.1	13	13
MS ObjectPackager	11	9.5	7	8
MS Recorder	12	9.6	1	1
MS Cardfile	13	10.3	2	4

Table 3. Expected ranks compared to mean ranks

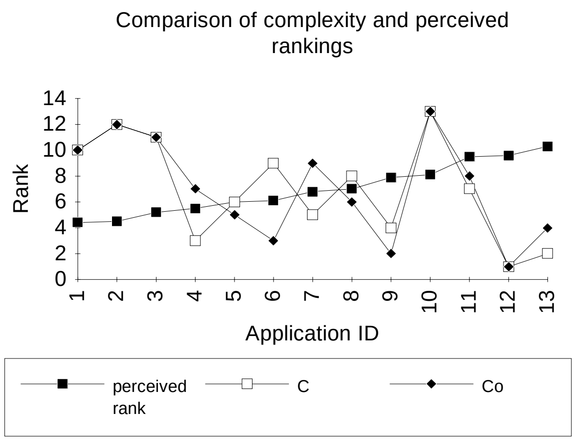


Figure 8. Mean ranks compared to expected ranks

3.2.3 Discussion

The expectation based upon the work of Tullis and Bonsiepe was that good layout design strives to be simple. This was not borne out by the results. A number of applications, including Microsoft Word and Excel, received rankings opposite to that expected. This suggests that users prefer more complex layouts.

There are clearly a number of problems with comparing screen designs for different applications. Some users reported being more familiar with certain designs and judged them better because of familiarity, suggesting that screens may be judged to be "good" because users can map them to what they know the applications can do. However, as we have seen, the results show a high degree of correlation in screen rankings between all thirty subjects ($\chi^2 = 91.1$, $\alpha = 0.00005$), indicating that familiarity with the screen was not a major factor in the ranking.

3.2.4 Conclusions

The most interesting result is the degree to which people like complex interfaces. At first glance this is counter-intuitive but further thought indicates that people usually do tend to judge a tool by its perceived functionality. This research suggests that it would be a good idea for interface designers not to open an application with a simple interface. Having shown that layout complexity is both measurable for GUI's and that at least one aspect of usability, attitude, is affected by the metric, the next stage was to determine the metric's utility by building an application and measuring the effect of layout complexity on usability.

3.3 Evaluating Usability of Screen Designs with Layout Complexity

3.3.1 Launcher

Usability has been defined as consisting of effectiveness, learnability, flexibility, and attitude [Lindgaard94]. The pilot experiment was designed to test three of these components of usability; effectiveness, learnability, and attitude.

The pilot consisted of a simple application, *Launcher* (figure 9), running under Microsoft Windows that calculated layout complexity for each design iteration. Launcher was originally designed as an example application for a Visual Basic tutorial and provides an alternative to the Window's "Program Manager" and "File Manager". Visual Basic (VB) was chosen to build the application and collect data as it could provide the necessary information about the dimensions and positions of most objects. It also could be used to track the user's progress with a task, keeping a record of each event and time taken.

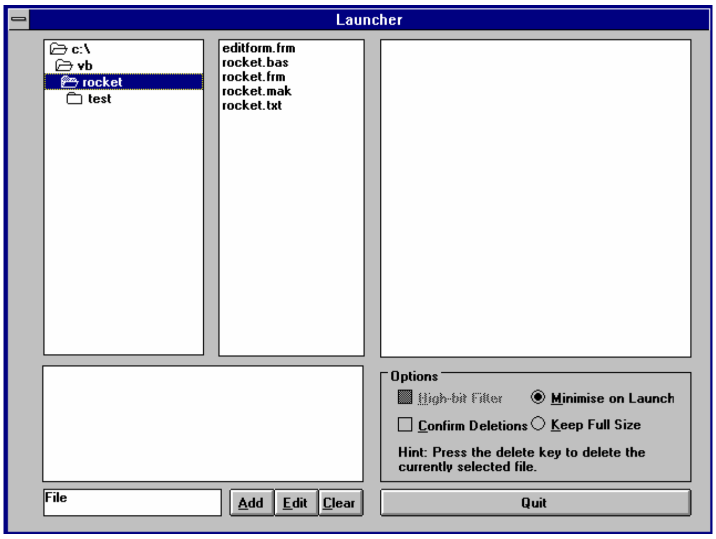
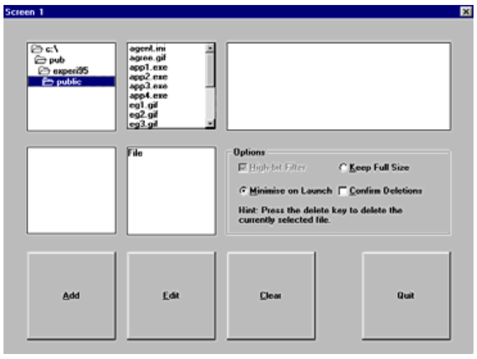


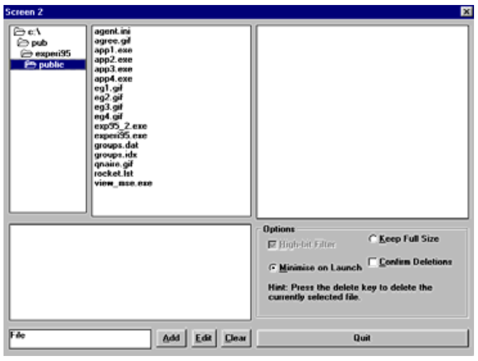
Figure 9. The application, "Launcher", used in this experiment

3.3.2 Screen Layouts

Four different screen layouts were designed, each with a different complexity score (figures 10, 11, 12 and 13).

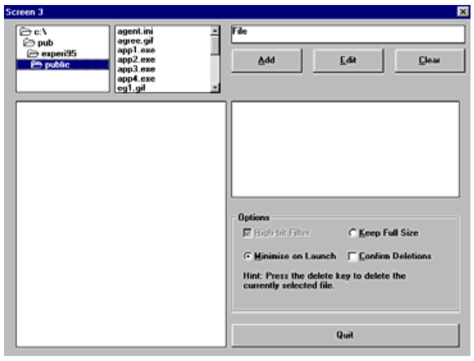


Hint: Press the delete key to delete the currently selected file.

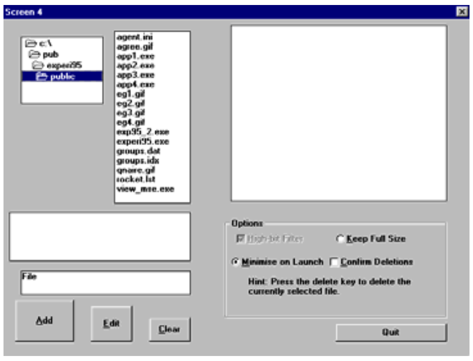


Hint: Press the delete key to delete the currently selected file.

Figure 10. Screen 1 - Complexity equals 156 Figure 11. Screen 2 - Complexity equals 170



Hint: Press the delete key to delete the currently selected file.



Hint: Press the delete key to delete the currently selected file.

Figure 12. Screen 3 - Complexity equals 186 Figure 13. Screen 4 - Complexity equals 228

The screen with the lowest score consisted of objects arranged in a neat grid with almost uniform sizes. The next two screens consisted of almost normal layouts and the final screen had every object with a different size and position.

Table 4 shows the complexity ratings for each of the four screens used in the experiment. The theoretical minimum was not achievable in VB, when using different objects, as some objects could not be resized to match other objects ie objects in VB have a fixed size relationship to other objects.

Complexity Scores for 17 Objects						
	Theory min.	Scr. 1	Scr. 2	Scr. 3	Scr. 4	Theory max.
C	71	156	170	186	228	272
%	0%	42%	49%	57%	78%	100%

Table 4. Complexity scores for 17 objects

3.3.3 Procedure

Seven experienced computer users volunteered to take part in the pilot study. Each subject was asked to read an ethics disclaimer and answer some basic questions about computer usage and experience. They were then requested to select a file, add it to a list, change its name and quit for each screen. At the completion of the first stage they were asked to indicate their preferences for the different screens.

They were given the choice of looking at printed copies of the screens or selecting images of the screens. The application was designed to record the time it took users to complete each step in a task and to record any errors. The subjects were then asked to run through the experiment again thus giving a second set of data for the same task and screen. It was expected that any improvement in performance for the second run would indicate an interface that was more learnable and memorable.

3.3.4 Results and Discussion

Each subject scored 1 if the screen was completed correctly and 0 if any mistake was made. This provided a simple measure of error rates. Table 5 shows the percentage correct for each screen design and for each run of the experiment.

	Percentage error-free screens			
Run	Screen1	Screen2	Screen3	Screen4
First	29%	71%	100%	71%
Second	43%	86%	71%	71%
Mean	36%	79%	86%	71%

Table 5. Percentage of screens that were completed without errors

There were no errors for Screen 3 in the first run and in the second run Screen 2 had the least errors. The two screens at either end of the complexity scale exhibited more errors, however the results for Screen 1 were confounded by confusion about the task and which object to choose.

The total time spent on each screen is presented in table 6. It can be seen that there was an overall improvement in task completion time from the first run to the second run. Screen 1 and Screen 4 were slower to complete. The times for Screen 1 were possibly affected by the same problems as mentioned previously.

Total Time Spent (s)					
Run	Scr. 1	Scr. 2	Scr. 3	Scr. 4	Total
First	221	165	147	145	678
Second	133	125	129	148	535
Total	354	290	276	293	

Table 6. Time spent completing the task for each screen and run

The subjects were asked to choose the most preferred screen (table 7). The votes for re-design are shown as minus figures to highlight the negative nature of the statement. Some subjects changed their minds on the second run. The reasons for this were not explored.

User Preferences					
	Scr. 1	Scr. 2	Scr. 3	Scr. 4	None
Attractive	4	3	3	4	0
BestDesign	2	4	7	1	0
EasyToUse	5	0	7	1	1
ReDesign	-7	0	-1	-6	0
Rating	4	7	16	0	1

Table 7. Users evaluation of the different screen designs

Both the least and most complex screens were rated poorly even though more users found them attractive. It is also interesting, that even though it was a small homogenous group, there was still quite a divergence of preferences.

Summary				
Usability	Scr. 1	Scr. 2	Scr. 3	Scr. 4
Complexity	156	170	186	228
Error-free	36%	79%	86%	71%
Time	354	290	276	293
Rating	4	7	16	0

Table 8. Summary of usability

Table 8 summarises the results. The screens with a mid-range complexity, screens 2 and 3, rate better overall than the screens at either end of the complexity scale. However these results do need to be treated cautiously because of the small number of subjects and the limited number of screens.

Visual Basic did prove a useful tool for calculating complexity though there were some problems. It was also useful for collecting data about the user's interaction with the application.

However one shortcoming in this pilot was using different forms for each screen. The results from this pilot showed differences in usability between screens differing in complexity. Graphic design manuals [Galitz93] stress the importance of using a grid to layout objects. Complexity theory offers a means for determining if objects have indeed been laid out in a grid but is a perfect grid pattern the best way to layout a screen?

The least complex screen, which most followed an exact grid, was not the most usable though the limited number of subjects, tasks and screens do suggest treating the results with caution.

The application will be extended to present more screens and more tasks and a wider cross-section of users will be involved in the next iteration. Extra metrics will also be added including "layout appropriateness" [Sears93], percentage white space and sampling of mouse pointer position to determine whether the user has "wandered" looking for the correct button.

Conclusion and Further Research

The designer of a GUI application is exposed to many guidelines, standards and rules [Vanderdonckt95c]. How the screen is actually designed depends on the designer's interpretations of these rules. A popular admonition to interface designers is to keep the screen simple and well organised [Mayhew92, Galitz93, Hix93].

In his influential and popular book on interface design, Galitz [Galitz93, p. 244] asserts that graphical interfaces can reduce usability because of the "variety and complexity" of interface objects. He indicated that an important requirement of users is that screens have "an orderly, clean, clutter-free appearance" [Galitz93, p. 60] to not reduce usability. Shneiderman [Shneiderman92, p.315] even goes so far as to state that "dense or cluttered displays can provoke anger". These authors have in common an idea that the interface designer agrees with them in what makes a simple, ordered interface. This research attempts to quantify this concept to enable objective design decisions to be made.

There are two groups that require a method of evaluating GUI applications.

1. Application designers need to be able to choose between competing layouts.
2. Prospective purchasers need to be able to compare different applications for design quality.

Bonsiepe's technique enables the designer to compare two versions of the same application and allows for an objective measure of their complexity. For this to be a practical technique would require the development environment to calculate the complexity figure as manual calculations are slow and prone to error. Recent work [Shneiderman95] shows it is possible to produce reports on the usability of an interface but we believe that it is a better approach to give feedback to the designer whilst work is in progress.

To this end, the layout complexity metric developed in this paper, and other metrics such as Tullis's measures [Tullis81, Tullis83, Tullis88a, Tullis88b], Kim's symmetry and balance [Kim93] and Sear's layout appropriateness [Sears93], could be implemented as functions that can be added to the Visual Basic software being developed and removed when development is complete. This will enable designers to modify their design "on-the-fly", according to the values of these metrics as continually provided during the interface design process.

However, the most important aspect of this research is to determine the relevance of these metrics to usability and to the ergonomic criteria, such as compatibility, consistency, workload, adaptability, dialogue control, representativeness, guidance and error management, which are known to lead to efficient and user friendly interfaces [Farenc95]. It might then be possible to provide a composite usability index from relationships such as the one suggested in figure 3. Eventually, a user interface development environment could be developed that automates part of the generation of a particular GUI and then lets an evaluation module compute these metrics and the associated index.

If the usability index and related metrics were provided to the prospective purchaser of software it would allow for an objective comparison of the interfaces. Ideally, the measures would be calculated either for some standard subset of the interface or for all screens in the interface. It would then be possible for software publishers to state the results of usability tests as a selling point.

Acknowledgements

We would lack to thank the CADUI'96 anonymous reviewers for their helpful comments and suggestions. We would also like to thank Jean Vanderdonckt for encouraging us to submit to this conference.

References

[Bonsiepe68] Bonsiepe, G.A., *A Method of Quantifying Order in Typographic Design*, Journal of Typographic Research, Vol. 2, 1968, pp. 203-220.

[Comber94] Comber, T., Maltby, J. R.*, A Formal Method for Evaluating GUI Screens*, in Doctoral Consortium of ACIS'94 (Melbourne, 1994), Dept. of Information Services, Monash, Australia, 1994.

[Comber95] Comber, T., Maltby, J. R., *Evaluating Usability of Screen Designs with Layout Complexity*, in Proceedings of OZCHI'95 (Wollongong: CHISIG), 1995.

[Galitz93] Galitz, W.O., *User-Interface Screen Design*, Q.E.D. Information Sciences, Wellesley, 1993.

[Hix93] Hix, D., Hartson, H.D., *Developing User Interfaces - Ensuring Usability Through Product and Process*, John Wiley & Sons, New York, 1993.

[Hudson93] Hudson, S.E., Hsi, C.-N., *A Synergistic Approach to Specifying Simple Number Independent Layouts by Example*, in Proceedings of the Conference on Human Factors in Computing Systems INTERCHI'93 « Bridges Between Worlds » (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), ACM Press, New York, 1993, pp. 285-292.

[Kim93] Kim, W.C., Foley, J.D., *Providing High-level Control and Expert Assistance in the User Interface Presentation Design*, in Proceedings of the Conference on Human Factors in Computing Systems INTERCHI'93 « Bridges Between Worlds » (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), ACM Press, New York, 1993, pp. 430-437.

[Lindgaard94] Lindgaard, G., *Usability Testing and System Evaluation*, Chapman & Hall, London, 1994.

[Lindsay72] Lindsay, P., Norman, D., *Human Information Processing*, Academic Press, New York, 1972.

[Maltby95a] Maltby, J.R., *Operational complexity of direct manipulation tasks in a windows environment*, Australian Journal of Information Systems, Vol. 2, No. 2, May 1995, pp. 30-49.

[Maltby95b] Maltby, J.R., *Towards a language for GUI dialogues*, in Proceedings of the CHISIG Annual Conference OZCHI'95 (Melbourne, 27-30 November 1995), Helen Hasan, Cathy Nicastri (Eds.), Australia, pp. 30-35.

[Mayhew92] Mayhew, D.J., *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, 1992.

[Nielsen90] Nielsen, J., *Traditional dialogue design applied to modern user interfaces*, Communications of The ACM, Vol. 33, No. 10, October 1990, pp. 109-118.

[Porter83] Porter, T., Goodman, S., *Manual of Graphic Techniques*, Charles Scribner's Sons, New York, 1983.

[Sears93] Sears, A., *Layout appropriateness: A metric for evaluating user interface widget layout*, IEEE Transactions on Software Engineering, VoL. 19, No. 7, 1993, pp. 707-718.

[Shannon62] Shannon, C.E., Weaver, W., *The Mathematical Theory of Communication*, University of Illinois, Urbana, 1962.

[Shneiderman92] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (2nd ed.), Addison-Wesley, Reading, 1992.

[Shneiderman95] Shneiderman, B., Chimera, R., Jog, N., *Evaluating spatial and textual style of displays*, Technical report No. CAR-TR-763, CS-TR-3451, ISR-TR-95-51, HCIL, University of Maryland, 1995.

[Tullis81] Tullis, T.S., *An Evaluation of Alphanumeric, Graphic, and Color Information Displays*, Human Factors, Vol. 23, No. 5, 1981, pp. 541-550.

[Tullis83] Tullis, T.S., *The formatting of alphanumeric displays: a review and analysis,* Human Factors, Vol. 25, No. 6, 1983, pp. 557-582.

[Tullis88a] Tullis, T.S., *A system for evaluating screen formats*, in « Advances in Human-Computer Interaction «, H.R. Hartson, D. Hix (Eds.), Ablex Publishing, 1988, pp. 214-286.

[Tullis88b] Tullis, T.S., *Screen design*, in « Handbook of Human-Computer Interaction », M. Helander (Ed.), Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1988.

[Vanderdonckt94c] Vanderdonckt, J., *Information Presentation for Multimedia Business Oriented Applications*, Workshop on Standardisation of Multimedia User Interface Design during meeting of ISO/TC 159/SC 4/WG 5 (Paris, 29 August-2 September 1994), September 1994.

[Vanderdonckt94d] Vanderdonckt, J., Ouedraogo, M., Yguietengar, B., *A Comparison of Placement Strategies for Effective Visual Design*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 125-143. http://www.info.fundp.ac.be/cgi-bin/pub-spec-paper?RP-94-019

[Vanderdonckt95c] Vanderdonckt, J., *Tools for Working with Guidelines*, Tutorial #12 notes, 6th International Conference on Human-Computer Interaction HCI International'95 (Yokohama, 10 July 1995), 1995.