CAR-TR-763 May 1995
CS-TR-3451
ISR-TR-95-51

Evaluating spatial and textual style of displays

Ben Shneiderman*, Richard Chimera and Ninad Jog
Ren Stimart†, David White†
Human-Computer Interaction Laboratory
Department of Computer Science
Institute for Systems Research*
University of Maryland, College Park, MD 20742-3255 USA
ben@cs.umd.edu
General Electric Information Service†
401 N. Washington Street, Rockville, MD 20850 USA

Abstract
The next generation of Graphic User Interfaces (GUIs) will offer rapid access to perceptually-rich, information abundant, and cognitively consistent interfaces. These new GUIs will be subjected to usability tests and expert reviews, plus new analysis methods and novel metrics to help guide designers. We have developed and tested first generation concordance tools to help developers to review terminology, capitalization, and abbreviation. We have also developed a dialog box summary table to help developers spot patterns and identify possible inconsistencies in layout, color, fonts, font size, font style, and ordering of widgets. In this study we also explored the use of metrics such as widget counts, balance, alignment, density, and aspect ratios to provide further clues about where redesigns might be appropriate. Preliminary experience with several commercial projects is encouraging.

1. INTRODUCTION AND LITERATURE REVIEW

Designing a user interface is a complex process (Hix & Hartson, 1993; Shneiderman, 1992). It begins with analysis of the users and their tasks, goes through creative stages in which key screens are designed and reviewed, and proceeds with detailed design of hundreds or thousands of screens, dialog boxes, form fill-in layouts, output formats, visual information presentation, help screens, tutorials, etc. Usability testing can begin early and be repeated with larger groups of users as the design becomes more stabilized and complete. The development process has been sped up in remarkable ways by the presence of user interface management systems and powerful development tools. It is possible to build running systems with an elaborate design in weeks and make refinements in hours.

While powerful tools enable designers to create excellent systems rapidly, designers can still produce poor designs. Commercial pressures are forcing many novice designers to turn out larger, more numerous systems at a more rapid pace, so concerns about quality are greater than ever. Furthermore, when several designers contribute to a large project coordination is needed to prevent unnecessary diversity. Quality control and acceptance testing procedures are being introduced in many organizations but system auditors are often at a loss for evaluation methods, criteria, and norms.

The most popular and effective methods appear to be usability testing and expert reviews. The dramatic expansion of usability testing has helped to improve designs, because designers are forced to work to a clear schedule and feedback from structured testing has proven to be powerful in revealing flaws early. Unfortunately usability testing cannot reveal what performance will be after months of usage and it is usually not possible for usability lab test participants to experience every dialog box. This “coverage” problem, a term borrowed from software testing, is increasingly important as systems grow in complexity and size. By contrast, expert reviews can be effective in coping with the coverage problem by diligent examination of each dialog box, but reviewers may differ in their opinions and can hardly be expected to notice all differences, omissions, or flaws when there are hundreds or thousands of dialog boxes. A further problem with usability testing and expert reviews is that they are relatively costly and time consuming compared with automated evaluations and interface metrics.

The criteria for excellent interface design are still emerging from creative graphics designers and from the results of empirical studies. Guidelines documents from Apple (1992), IBM (1991), Microsoft (1993), and others are a first step, but many design issues are not addressed by these already voluminous books (Brown, 1988; Galitz, 1989; Marcus, 1992). While there will always be room for innovative designs, there is a growing need for methods that enable ordinary designers to create effective systems reliably, on-time, and on-budget.

Interface development is greatly facilitated with widely used tools such as Visual Basic (Microsoft Corp.), Reality, or PowerBuilder, and more complex cross-platform systems such as Galaxy (Visix Corp.), XVT (XVT Corp.), or Open Interface (Neuron Data). These tools can facilitate standardization across platforms and provide the software infrastructure for new evaluation tools. Software tools to assist designers are being implemented by practitioners for their products while researchers have begun to develop some exploratory systems that help automate the design process (Kim & Foley, 1993; Sears, 1994).

While automated layout holds promise for standardized situations, in more complex situations simple automated evaluations can provide feedback to designers, even at early stages of development. Many of the guidelines documents include recommendations about appropriate numbers of menu items, colors, widgets, etc. Sometimes there is experimental support for these recommendations but often they are based on only subjective judgments and thoughtful analyses. Streveler and Wasserman (1987) proposed novel visual metrics such as symmetry, balance, percentage of screen used, and average distance between groups of items, but they did not apply or test these notions.

Tullis (1988a, 1988b) carried these ideas further and implemented a system for evaluating the visual displays from character based interfaces only. He implemented metrics for overall and local density (based on number of characters filled), grouping (number of groups and their sizes), and layout complexity (vertical and horizontal alignment). His metrics were partially validated in a useful series of studies and his tool was distributed. The shift to graphic user interfaces with multiple font sizes, three dimensional widgets, etc. means that new analyses and metrics are necessary.

The availability of more graphic design features has raised interest in spatial properties such as balance, symmetry, regularity, alignment, proportion, horizontality, simplicity, economy, neutrality, unity, grouping, predictability, sequentiality, etc. These and a dozen more were identified and discussed in the context of traditional and multimedia layouts (Vanderdonckt & Gillo, 1994). These properties were intended to serve as a basis for an automatic placement tool (Bodart et al., 1994), however, specific metrics and acceptable ranges were not tested. Other efforts at automatic layout may lead to useful tools for some situations (Feiner, 1988; Kim & Foley, 1993; Byrne et al., 1994), but there is very little experience with substantial commercial applications.

Esthetically pleasing layouts are important, but the layouts should also match the sequence and frequency of the users's tasks. The term “layout appropriateness” (Sears, 1993, 1994) was chosen to convey the correspondence between layout and task. Layout appropriateness requires more inputs concerning usage patterns, but it is far more powerful in providing reliable evaluations and can even be used to generate layouts that would be optimal with respect to the metric of distance traversed or other metrics. Early testing has demonstrated its effectiveness in analyzing simple dialog boxes and complex control panels from NASA applications.

These preliminary efforts have all been helpful in identifying potential metrics and evaluation tools that could be used in the context of modern user interface building tools that generated graphic user interfaces. We sought to take these ideas from the laboratory into field testing and develop tools for professional developers working for General Electric Information Systems.

2. OUR METHODS

Our research expanded from single screen analyses, towards evaluations across the dozens or hundreds of dialog boxes found in many user interfaces. We focused on consistency across screens and on feedback to designers to guide them to issues that might require further analysis. In earlier work (Chimera & Shneiderman, 1993) we demonstrated that consistency in color, terminology, layout, instructions, etc. does make a difference in users's perceptions, performance, and subjective satisfaction. While consistency is a complex concept and sometimes violations are appropriate, some aspects of consistency checking are candidates for automation. It would seem appropriate for designers to preserve spatial properties such as position of similar items, size or aspect ratios of related dialog boxes, minimal wasted space, consistent margins, and aligned, balanced layouts. Similarly visual properties of text such as color, fonts, font sizes, font styles, and justification of labels would be more acceptable if they were used consistently. Finally terminological consistency, and standard spelling, abbreviations, and capitalization seems important in simplifying an interface for novice and expert users who are first time, intermittent or frequent users.

Our goal was to give designers rapid feedback as they developed their designs and steer them to examine certain screens in detail to see if there was a need for improvements. We wanted to provide a kind of medical lab report (like a blood test) for a set of dialog boxes that revealed potential anomalies but did not prescribe cures. To do this we used the descriptions of dialog boxes generated by tools such as Visual Basic. We developed a canonical format for dialog box descriptions that becomes the input to our evaluation programs. Other development tools produced different descriptive outputs, but we assumed that a knowledgeable developer could write a conversion program to put the information in the canonical format.

We developed two reports: a dialog box summary table that gave a compact overview of spatial and visual properties. Each row described a dialog box and each column a metric. The second report, a concordance, was built by extracting all the words that appeared in every dialog box and sorting them in one file with references to where they came from.

2.1 Dialog box summary table

The dialog box summary table was intended to provide designers with a compact overview of the dozens or hundreds of dialog boxes that had been designed by the single or multiple designers in a project. Each row represents a single dialog box and each column represents a single metric. Typical use would be to scan down the column looking for extreme values, spotting inconsistencies, and understanding patterns within the design.

The order of the rows was initially alphabetical and the dialog box summary table was printed on paper, but other orders based on groupings of dialog boxes functionally (so that all the dialog boxes related to installation or printing might be seen as a group) might be useful. Viewing the dialog box summary table within an electronic spreadsheet would be logical. The order of the columns was less clear to us and we simply appended new columns as the software was created. A compact presentation which squeezed as many columns as possible across a wide printout was seen as advantageous.

The choice of the metrics was our most critical issue. The University of Maryland and the General Electric groups brainstormed independently for a week, consulting with colleagues and generating two lists with approximately 40 proposed metrics each. The specific items were grouped into categories such as consistency, spatial layout, alignment, clustering, cluttering, color usage, fonts, attention getting, etc. The two lists had many similar items and categories so we were encouraged. A second independent brainstorming session was used to choose an ordered list of metrics for implementation. Highly ranked items were to be ones that we expected to have high payoff and be easy to implement.

The implementation, written in C++, revealed problems in obtaining the required values in a complete and consistent manner. Definitions of the metrics were revised, special conditions were handled, and bugs were resolved one by one as the columns emerged. The current columns are explained and a portion of the dialog box summary table is below:

Dialog Name: Name of the file in which dialog is contained.

Aspect Ratio: The ratio of the height of a dialog to its width. Numbers in the range 0.3

thru 1.7 are desirable.

Widget Totals: Counts of all the widgets and the top level widgets. Increasing difference between all and top level counts indicates greater nesting of widgets, such as buttons inside containers.

Non-Widget Area: The ratio of the non-widget area to the total area of the dialog, expressed as a percentage. Numbers closer to 100 indicate high utilization, and low numbers (< 30) indicate possibilities of redesign.

Widget Density: The number of top-level widgets divided by the total area of the dialog (multiplied by 100,000 to normalize it). High numbers indicate that a comparatively large number of widgets are present in a small area. This number is a measure of the 'crowding' of widgets in the dialog.

Margins: The number of pixels between the dialog box border and the closest widget. The left, right, top and bottom margins should all be approximately equal to each other in a dialog, and should also be the same across different dialogs. Dialogs that contain widgets which extend beyond the dialog's bounds (e.g., lists) give rise to negative figures for bottom margins.

Gridedness: The ratio of the total number of widgets in a dialog to the number of distinct x or y positions that the widgets have. This gives rise to distinct x-axis and y-axis measures for gridedness. If all the widgets in a dialog have distinct values for the x-coordinate of their positions, the x-gridedness will be 1. A number greater than 1 is evidence of grouping. If the x-gridedness is greater than the y-gridedness in a single dialog, indicates that widgets are stacked into columns rather than rows.

Area Balances: A measure of how evenly widgets are spread out over the dialog box. There are two measures: a horizontal balance, which is the ratio of the total widget area in the left half of the dialog to the total widget area in the right half of the dialog; and the vertical balance, which uses top area divided by bottom area. Dialogs in which all widgets are vertically centered have a horizontal balance of 1 (Left Area = Right Area). In general we expect the horizontal balance to be greater than 1 because many dialogs typically consist of large-size widgets in the left and top halves, and small widgets (such as buttons) at the right and bottom.

Distinct Typefaces: Typeface consists of a font, font-size, bold and italics information. Each distinct typeface in all the dialog boxes is randomly assigned to an integer and is described in detail at the end of the table. For each dialog box all the integers representing the distinct typefaces are listed so that the typeface inconsistencies can be easily spotted locally within each dialog box and globally among all the dialog boxes. The idea is that a distinct typeface should be used for all the dialog boxes. Occurrence of too many typefaces within a dialog box may not be desired.

Distinct Colors: (This column is not shown below because of lack of space) All the distinct background colors in a dialog box are displayed. Each distinct color in all the dialog boxes has been randomly assigned to an integer for display and comparison convenience and is described in detail at the end of the table. The purpose of this metrics is to check if all the dialog boxes have the same background colors. Multiple background colors in a dialog box may indicate inconsistency.

This table reveals some interesting anomalies that led to reconsiderations of designs. The test user interface had about 140 dialog boxes and was a well-reviewed and polished design. Very few obvious bugs appeared but many interesting questions were raised as we reviewed the detailed analysis. For example the varying aspect ratios were a surprise and irregular margins were a sign of lack of coordination. The gridedness values did lead to some review of layouts, but we are not yet sure about how to refine this measure. The balance ratios were effective in finding unusual layouts which will be reconsidered. The unusual variety in typefaces in `contacts.cft` were a surprise and it turned out to be the work of a specific designer who had created other dialog boxes of the application with his distinctive style. Similar surprises occurred in the distinct typefaces and colors columns.

No.	Dialog Name	Aspect Ratio (H/W)	–WIDGET-- TOTALS All	Top-Level	Non-Widget Area (%)	Widget Density widget/area	--- M A R G I N S --- (pixels) Left	Right	Top	Bottom	Gridedness X	Y	–Balances-- Area Ratios Horiz (L/R)	Vert (T/B)	Distinct Typefaces
1	aboutedi.cft	0.49	6	5	74.4	76	64	30	8	6	1.0	1.0	1.0	0.7	1
2	actlog.cft	0.67	16	14	-0.0	60	0	6	0	-241	2.0	1.4	1.1	0.4	1
3	addexp.cft	0.43	3	2	46.0	38	8	33	8	17	1.0	1.0	1.1	3.2	1
4	addfamdf.cft	0.77	25	13	28.3	74	8	26	8	4	1.3	2.6	1.0	0.7	1
5	addr.cft	0.73	47	8	23.9	36	8	26	8	4	1.1	2.7	1.1	0.9	1
6	addrbk.cft	0.84	45	29	15.5	177	0	13	0	6	1.7	2.2	1.1	0.8	1
7	addsec.cft	0.50	7	6	32.9	84	8	23	8	9	1.2	2.0	1.5	0.8	1
8	addseg.cft	0.63	7	6	42.4	103	16	23	8	12	1.0	2.0	1.4	0.6	1
9	addstand.cft	0.41	3	2	60.9	44	24	38	24	12	1.0	1.0	1.0	2.1	1
10	admpwd.cft	0.70	14	6	31.9	63	16	21	8	5	1.0	2.0	1.0	0.7	1
11	adrmsg.cft	0.74	28	14	23.2	61	8	21	8	7	1.4	2.3	1.0	0.5	1
12	adrmsg2.cft	0.74	28	14	23.0	61	8	21	8	6	1.4	2.3	1.0	0.5	1
13	adrmsg3.cft	0.76	28	14	24.7	60	8	13	8	7	1.4	2.3	1.0	0.5	1
14	advsched.cft	0.82	4	3	35.6	42	16	23	16	13	1.0	1.5	1.0	1.3	1
15	afile2.cft	0.66	4	3	47.7	57	16	33	8	13	1.0	1.5	1.0	1.6	1
16	alert1.cft	0.47	5	4	42.4	129	8	18	8	4	1.0	1.3	1.0	1.5	1
17	archive.cft	0.57	23	14	44.6	86	8	33	8	26	1.6	1.8	0.9	0.6	1
18	archok.cft	0.60	13	12	48.8	130	8	11	8	6	1.5	3.0	1.1	1.1	1
19	asgnfam.cft	0.49	12	11	50.8	82	16	7	8	3	1.2	2.2	1.0	0.4	1
20	autoff.cft	0.42	10	9	38.7	87	16	23	8	4	1.1	3.0	1.2	0.8	1
21	autofile.cft	0.39	10	5	34.1	74	16	31	16	4	1.2	1.7	1.3	0.7	1
22	autoupd.cft	0.37	8	5	52.6	89	16	26	8	4	1.0	1.7	1.1	1.1	1
23	backnow.cft	0.49	12	11	56.4	102	24	24	8	14	1.4	2.8	0.8	1.0	1
24	btmail.cft	0.48	3	2	50.3	109	8	20	8	10	1.0	1.0	1.0	3.1	1
25	buildcl.cft	0.58	4	3	38.8	56	8	18	8	11	1.0	1.5	1.0	1.7	1
26	cc.cft	0.44	3	2	76.5	76	32	49	16	15	1.0	1.0	1.1	1.4	1
27	chgstat.cft	0.43	3	2	47.6	87	8	20	8	7	1.0	1.0	0.9	2.9	1
28	ckdoc.cft	0.55	3	2	47.9	47	8	31	8	12	1.0	1.0	1.0	3.5	1
29	conhost.cft	0.47	3	2	46.7	55	8	13	8	14	1.0	1.0	0.9	3.8	1
30	connect.cft	0.61	17	16	49.7	142	16	31	16	2	1.8	1.8	0.6	1.9	1
31	contacts.cft	0.73	105	13	-358.8	47	8	12	8	-1403	1.6	1.9	1.3	0.3	1234
32	create.cft	0.71	92	12	0.2	44	0	14	0	9	4.0	1.3	1.0	0.9	1
33	dbback.cft	0.37	14	5	45.2	65	16	29	16	11	1.0	1.7	1.1	0.8	1
34	dearch.cft	0.53	15	14	48.6	119	8	11	8	5	1.8	3.5	1.3	0.8	1
35	dearch2.cft	0.59	10	9	41.1	68	24	26	0	16	1.1	4.5	1.5	1.2	1
36	dearchok.cft	0.63	10	9	35.2	63	16	26	8	12	1.0	4.5	1.6	1.0	1
37	delconf.cft	0.41	6	5	63.2	118	16	14	16	7	1.7	1.0	0.9	2.7	1
38	dociduti.cft	0.69	29	3	23.8	14	16	28	16	16	1.5	1.5	1.3	1.0	1
	...														
Maximum		1.00	170	31	97.5	271	80	56	24	27	4.4	4.5	6.2	8.6	
Minimum		0.32	3	2	0.0	14	0	0	0	0	1.0	1.0	0.3	0.0	
Average		0.60	17	8	35.3	86	11	19	7	7	1.6	1.7	1.1	1.4	

1 = MS Sans Serif 8.25 Bold 2 = MS Sans Serif 8.25 3 = MS Sans Serif 9.75 Bold Italic
4 = MS Sans Serif 8.25 Bold Italic 5 = Arial 8.25 Bold 6 = MS Sans Serif 18 Bold
7 = MS Sans Serif 9.75 Bold

Minimum, maximum, and average values were computere for the metrics. Dialog boxes with extreme values should be examined as candidates for redesign.

A second part of the dialog box summary table (shown below) displays information on frequently used buttons: OK, Cancel, Help and Close. The columns enabled us to spot the highly inconsistent sizes and relative placements of these buttons in this application.

Presence of OK and Cancel Buttons: If a dialog has OK or Cancel buttons, their height and width in pixels are printed. The idea is that they should have the same sizes, and designers can verify the presence of these fundamental controls.

OK and Cancel Button Relative Positions: For dialogs that have both OK and Cancel buttons, this metric indicates their relative position. If the Cancel button is to the right of the OK button, the offset in pixels is printed as x + offset, else if it is below the OK button, it is printed as y + offset.

Help and Close Button Sizes: If a dialog has Help or Close buttons, their height and width in pixels are printed. The size of the buttons should be consistent.

No.	Dialog Box Name	OK Button (height,width)	Cancel Button	Relative Position	Help Button (height, width)	Close Button
1	aboutedi.cft	25,89				
2	actlog.cft					25,123
3	addexp.cft	25,97				
4	addfamdf.cft	25,73	25,73	y + 7	25,73	
5	addr.cft	25,81	25,81	y + 7	25,81	
6	addrbk.cft					25,89
7	addsec.cft	25,73	25,73	y + 7	25,73	
8	addseg.cft	25,73	25,73	y + 7	25,73	
9	addstand.cft	25,89				
10	admpwd.cft	25,65	25,65	y + 7	25,65	
11	adrmsg.cft	25,57	25,57	y + 7	25,57	
12	adrmsg2.cft	25,57	25,57	y + 7	25,57	
13	adrmsg3.cft	25,57	25,57	y + 7	25,57	
14	advsched.cft	25,97				
15	afile2.cft	25,65				
16	alert1.cft	25,97				
17	archive.cft	25,73	25,73	x + 23	25,73	

2.2 Concordance

The idea of the string concordance output is to list all occurrences of words that appear in labels, buttons, menus, user messages, etc. throughout the user interface canonical format file. Designers can use the concordance to identify many aspects of appropriate word use such as spelling, case consistency, passive/active voice, noun/verb choice, etc.

There is a short format and a long format of the string concordance. Both formats create a file that is an ascii table with multiple columns. The first column of both formats lists individual words one per line sorted in alphabetical order. Occurrences in different case are preserved as unique occurrences of words, and are listed in the sorted list after one another so that uses of different case is clearly pointed out. The normal sort order is a..zA..Z, but this would separate occurrences of "find" from "Find" or "FIND" and so our program sorted words by aAbB...zZ.

The short format lists the word and number of times the word appears. The long format identifies the files in which the word appears (see below). The word "Message" appears 18 times in the files whose names follow it, "Message:" appears 2 times, and further down the list the term "messages" (uncapitalized) appears once and "msgs" appears once. These variant forms are spelling errors and may be acceptable, but they may be something that should be reconsidered.

```
Message                        18
              addr.cft                     addr.cft                     dociduti.cft
```

```
              docsearc.cft                docsearc.cft                docsearc.cft
              docsort.cft                 docsort.cft                 famdef.cft
              ffadd.cft                   in.cft                      moreinfo.cft
              moreinfo.cft                moreinfo.cft                moreinfo.cft
              profile.cft                 profile.cft                 profile.cft
Message:                    2
              profile.cft                 remfam.cft
MessageIDs                  1
              dociduti.cft
Messages                    4
              archive.cft                 autofile.cft                autofile.cft
              profile.cft
messages                    1
              dbback.cft
msgs                        1
              dbback.cft
```

3. TESTING OUR METHODS

Our testing has included applying the metrics to a prototype application, reviewing the results for concept validity, and gathering reactions from developers. The prototype with 140 highly varied dialog boxes was a GE Information Services' Electronic Data Interchange application. The user interface was written in Microsoft Visual Basic, independently of our efforts to create metrics to evaluate the spatial, and textual aspects of displays. The prototype simulated typical actions to show what the user would see. It served as the portion of the functional specification to which the final product was designed. It also was used in an early usability test to confirm the design concepts.

A translator was written to convert Visual Basic .FRM files into the canonical format that could then be inputted to the evaluation program. Screen shots were also taken of all the dialog boxes in the prototype and printed out. Output from the metric evaluation program was then scanned for patterns and anomalies that were compared to the screen printouts. Several iterations of generating output, comparing to the screen printouts and reworking the program took place until a stable and accurate set of metrics were produced. These metrics were then shown to developers and quality assurance people at GE Information Services for preliminary feedback.

We are in the process of repeating the testing with a GE Information Services' commercial product. This application (also written in Visual Basic) contains a larger and more complex set of dialog boxes. The output of the metric evaluation program will then be given to a number of developers and quality assurance specialists for feedback.

4. CONCLUSIONS AND FUTURE WORK

As the complexity of GUI's increase, developers are finding they need more help in the analysis and testing of their designs. Quality assurance is also finding it increasingly difficult to adequately test all aspects of current user interfaces. Initial feedback on the metric evaluation program from these groups at GE Information Services indicate a definite perceived value in such tools. And while the feedback was positive on the concept and initial output, several issues and suggestions were raised. In its current format, the output needs to be manually scanned for anomalies and patterns. There is a desire to have these highlighted by the tool.

For developers:
- prescriptive directions on how to correct the interface problems. While in some cases, it is obvious what to do such as when there are multiple capitalization strategies for the "Cancel" button, or different fonts and button sizes are being used. Easier still would be a message telling them to use x font with a particular capitalization as it is encountered. The more difficult cases like widget density or gridedness is more of a mystery.
- a tool which is interactive that displays the problems as they are encountered. This may be beyond current processing speeds of most PC's, but intuitively it seems right to correct the problem as they go rather than discovering it in a printout and then finding their way back to the location in the application.
- a tool that is usable in all of their development environments, i.e. if they are using C++ or some of the new cross platform development tools, they want the same capability as we showed them with Microsoft Visual Basic. This supports the canonical format approach and implies a need to have a translator from whichever environment they are using.

For Quality Assurance people:
- a summative tool that checks across the entire application and reports back those areas that have problems.
- an indicator of the severity of the problem. While any item uncovered as an issue is probably worth trying to resolve, they are worried more about those items that have a large user impact than those that do not.
- a check for consistency of wording and layout. To do it manually is becoming very difficult. There are so many different things to look at that it is often a challenge just to make sure they have seen every dialog box. They would also like the evaluation tool to validate against the design specification.

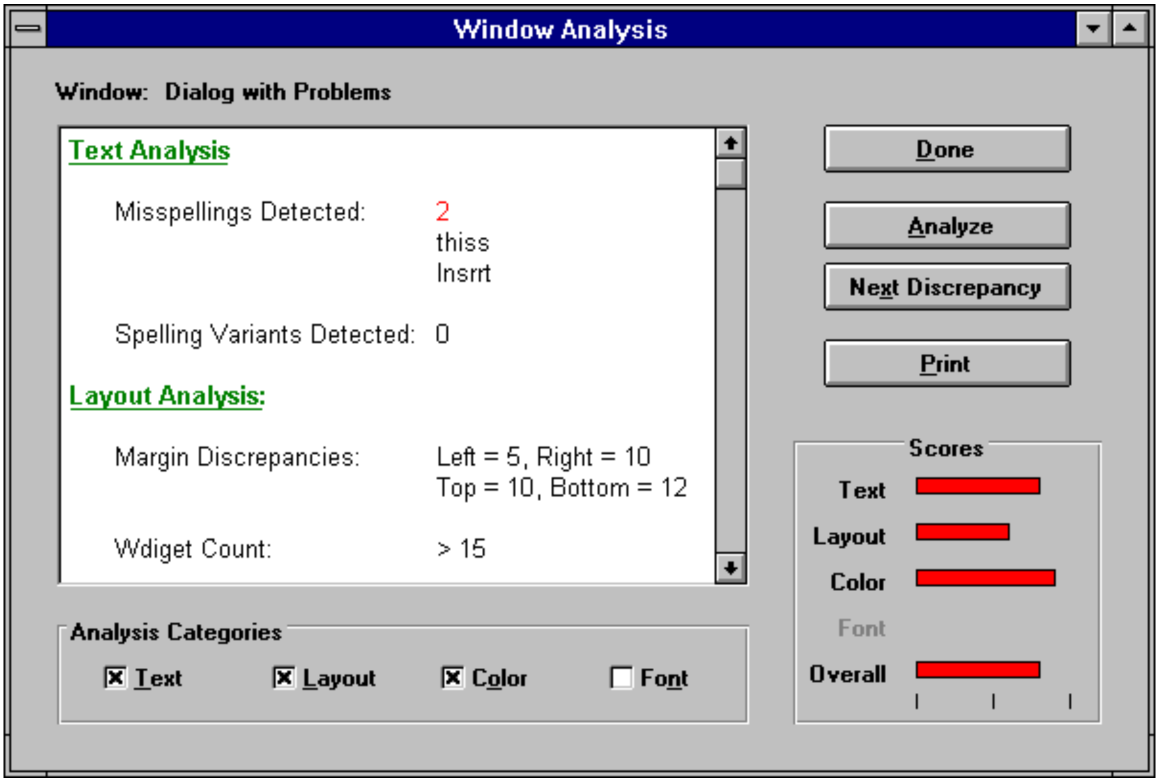
There are many measures that were not attempted in this first effort. We needed to start someplace and the metrics we chose represented a variety of things to look at so we could test the concept. We know from this initial exploration that we are touching only the tip of the iceberg which are the aspects of displays that could be evaluated automatically. Where it is practically possible, assessment against industry standards should be undertaken. We need to expand the number of metrics to get better measures on usage consistency and if possible get at conformity to an organization's "look and feel" for their products.

Perhaps the hardest part for future refinement of the evaluation tools is to provide the "goodness" measures for the metric values. Its clearly what is needed for those not schooled in human factors. In many cases it is acceptable if there is not a clear, research supported recommendation. Educated judgments will suffice to provide the rules for development and to gain the consistency across applications.

The next steps are underway to develop a tool that provides some of how the developers would like to use the metrics thus far established. Below is a sample layout.

The tool will allow developers and quality assurance specialists to view on-line summative metrics for multiple dialog boxes and metrics for individual dialog boxes all with anomaly highlighting. They would receive feedback from whatever dialog box has focus by pressing the "Analyze" button. Once the dialog box has been analyzed the reviewers could walk through each set of discrepancies or review the scores for that dialog box.

Our first attempts led to lengthy outputs of uncertain merit, but as we refined our choices of metrics the outputs became more provocative and productive. New ideas flowed more easily and new metrics, output formats, and theories of automated evaluation emerged. We are still at the beginning phases, but see that there is potential for these evaluation tools since they are quick and simple to apply, and they reveal interesting properties of complex designs.



ACKNOWLEDGEMENTS

We appreciate the support for this project from GE Information Services and the Maryland Industrial Partnerships program. We are grateful for draft comments from Vic Basili, Catherine Plaisant,and Anne Rose, and for programming assistance from Rohit Mahajan.

REFERENCES

Apple Computer, Inc. (1992), *Macintosh Human Interface Guidelines*, Addison-Wesley Publishing Co., Reading, MA.

Bodart, F., Hennebert, A.-M., Leheureux, J.-M., and Vanderdonckt, J. (1994), “Towards a dynamic strategy for computer-aided visual placement”, In Catarci, T., Costabile, M., Levialdi, S., and Santucci, G. (Editors), *Proc. Advanced Visual Interfaces Conference ‘94*, ACM Press, New York, 78-87.

Brown, C. M. (1988), *Human-Computer Interface Design Guidelines*, Ablex Publishing Co., Norwood, NJ.

Byrne, M., Wood, S., Sukaviriya, P., Foley, J., and Kieras, D. (1994), “Automating Interface Evaluation”, *Proc. of CHI’94*, ACM, New York, 232-237.

Chimera, R. and Shneiderman, B. (1993), “User interface consistency: An evaluation of original and revised interfaces for a videodisk library”, In *Sparks of Innovation in Human-Computer Interaction* (B. Shneiderman, editor), Ablex Publishers, Norwood, NJ, 259-271.

Feiner, S. (1988), “A grid-based approahc to automating display layout”, *Proc. of Graphics Interface ‘88*, 192-197.

Galitz, W. O. (1989), *Handbook of Screen Format Design: Third Edition*, Q. E. D. Information Sciences, Inc., P. O. Box 181, Wellesley, MA 02181.

Hix, D. and Hartson, H. R. (1993), *Developing User Interfaces: Ensuring Usability Through Product & Process*, John Wiley & Sons, New York, NY.

IBM (1991), Systems Application Architecture: Common User Access, Advanced Interface Design Reference, IBM Document SC34-4290-00, Cary, NC.

Kim, W. and Foley, J. (1993), “Providing high-level control and expert assistance in the user interface presentation design”, *Proc. of CHI’93*, ACM, New York, 430-437.

Marcus, A. (1992), *Graphic Design for Electronic Documents and User Interfaces*, ACM Press, New York, NY.

Sears, A. (1993), “Layout Appropriateness: A metric for evaluating user interface widget layouts”, *IEEE Transactions on Software Engineering 19*, 7, 707-719.

Sears, A. (1994), “Using automated metrics to design and evaluate user interfaces”, DePaul University Dept of Computer Science Technical Report #94-002, Chicago, IL.

Shneiderman, B. (1992), *Designing the User Interface: Strategies for Effective Human-Computer Interaction: Second Edition*, Addison-Wesley Publ. Co., Reading, MA.

Streveler, D. and Wasserman, A. (1987), “Quantitative measures of the spatial properties of screen designs”, *Proc. of INTERACT ‘87*, Elsevier Science, Amsterdam, 125-133.

Tullis, T. S. (1988a), “Screen design”, In Helander, M. (Editor), *Handbook of Human-Computer Interaction*, Elsevier Science, Amsterdam, The Netherlands, 377-411.

Tullis, T. S. (1988b), “A system for evaluating screen formats: Research and application”, In Hartson, H. Rex, and Hix, Hartson, *Advances in Human-Computer Interaction: Volume 2*, Ablex Publishing Corp., Norwood, NJ, 214-286.

Vanderdonckt, J. and Gillo, X. (1994), “Visual techniques for traditional and multimedia layouts”, In Catarci, T., Costabile, M., Levialdi, S., and Santucci, G. (Editors), *Proc. Advanced Visual Interfaces Conference ‘94*, ACM Press, New York, 95-104.