Visual & Textual Consistency Checking Tools for Graphical User Interfaces

Rohit Mahajan and Ben Shneiderman[1]
[1]*Department of Computer Science,*
Human-Computer Interaction Laboratory &
Institute for Systems Research
University of Maryland, College Park, MD 20742 USA
email: mahajan@cs.umd.edu, ben@cs.umd.edu

Abstract

Designing a user interface with a consistent visual design and textual properties with current generation GUI development tools is cumbersome. SHERLOCK, a family of consistency checking tools, has been designed to evaluate visual design and textual properties of interface, make the GUI evaluation process less arduous, and aid usability testing. SHERLOCK includes a dialog box summary table to provide a compact overview of visual properties of hundreds of dialog boxes of the interface. Terminology specific tools, like Interface Concordance, Terminology Baskets and Interface Speller have been developed. Button specific tools including Button Concordance and Button Layout Table have been created to detect variant capitalization, distinct typefaces, distinct colors, variant button sizes and inconsistent button placements. This paper describes the design, software architecture, and the use of SHERLOCK. An experiment with 60 subjects to study the effects of inconsistent interface terminology on user's performance showed 10-25% speedup for consistent interfaces. SHERLOCK was tested with four commercial prototypes; the corresponding outputs, analysis and feedback from designers of these applications is presented.

Index Terms: Graphical user interfaces, evaluation tools, consistency, textual and visual style, assessment tools, metrics.

1.0 Introduction & Previous Research

Consistency in User Interfaces follows the second law of thermodynamics. If nothing is done, then entropy will increase in the form of more and more inconsistency in your user interface

Jakob Nielsen

Graphical User Interface design is a complex and challenging discipline. It should be user-centered (Norman & Draper, 1986) and requires iterative design, usability testing, and evaluation (Shneiderman, 1992). GUI programming in recent years has become a major part of software development, and is minimally 29% of software development budgets (Rosenberg, 1989). Moreover, data analysis has shown that the user interface is 47-60% of the total lines of application code (MacIntyre, Estep & Sieburth, 1990) GUI design encompasses more than one third of the software development cycle and plays a major role in determining the quality of a product. Applying proper human factors techniques including early completion of user requirements definitions, usability prototype testing, and usability walkthroughs can significantly reduce the cost of identifying and resolving usability problems, and can save time in software development (Karat 1992).

With the emergence of powerful GUI development tools in the recent years, test applications can be created in few weeks. These interfaces may embody inconsistencies in visual design and textual properties which can't be detected by these powerful current generation development tools. Such inconsistencies can have a subtle and negative impact on the usability of the interface. Not only are more quality control and GUI test procedures required, but also new analytic and metric based tools are needed for the creation of cognitively consistent interfaces having a common “look and feel”.

We worked closely with General Electric Information Services to design and develop task-independent metrics to evaluate visual design and textual properties of user interfaces. We developed a single tool with multiple metrics to accomplish this task. The complexity of interpreting the results of a single evaluation tool led us to modify our evaluation approach by constructing smaller tools, each of which evaluates a few design aspects. Together these tools evaluated multiple design issues, forming a family of consistency checking tools (SHERLOCK). The reports generated by these mini-tools require less interpretation, thereby expediting the quick evaluation process and providing feedback to the designer. The designer must then decide whether the inconsistencies detected are relevant to the particular application.

1.1 Consistency and Evaluation

Defining Consistency: Consistency in design is an important aspect to be considered when creating user interfaces and is supported by most user interface experts. Shneiderman (1992) says that the first golden rule of dialog design is strive for consistency. Nielsen (1989) says that one of the most important aspects of usability is consistency in user interfaces. But, experts have struggled to define exactly “what consistency is?” and “how to identify good consistency?”. Reisner (1990) states that consistency is neither a property of the system nor the user, it is a relation between two languages, that of the system, as a designer intended it, and of the competent user. Wolf (1989) says that consistency means that similar user actions lead to similar results. A consistent user interface is an interface that maximizes the number of shared rules across tasks (Polson et al., 1986). Therefore, consistency is a relational concept and can't be defined by itself. A user interface is consistent or inconsistent with respect to something, which may be within the individual application or across a product family or for all the applications running on a particular system.

Consistency across the application in those components of the user interface which require human perception and cognitive mechanisms like visual scanning, learning and remembering is very important. Spatial organization which may include the organization of menus, placement of frequently used widgets, symmetry, and alignment of widgets is one of those components. Other components may include fonts, colors, common actions, sequences, terms, units, layouts, typography and more within an application program. Consistency is naturally extended to include compatibility across application programs and compatibility with paper or non-computer-based system. The sequence of pointing, selecting or clicking should be the same throughout the application (Smith et al, 1982) Consistency facilitates positive transfer of skills from one system to another leading to ease of use, reduced training time and improved retention of operating procedures (Nielsen, 1989; Polson et al, 1986).

Kellogg (1987) studied the impact of the conceptual dimension of consistency by prototyping a “consistent” version (common look and feel and conceptually consistent) and an “inconsistent” version (only common look and feel) of an interface. The results of the study, which incorporated a variety of measures like learning time, subjective satisfaction and more, showed that the “consistent” interface was better than the “inconsistent” (consistent in visual appearance and behavior only). Although visual appearance and behavior are very important aspects of consistency, they need to be combined with conceptual consistency in designing user interfaces.

GUI Guidelines & Task Analysis: The use of guidelines has been identified as important within the framework of a representation of the human-computer interface (HCI) by many experts (Harrison & Thimbleby, 1985). However most guidelines do not include specifications of consistency or properties of a consistent interface that produce positive transfer of skills. Frederiksen, Grudin and

Laursen (1995) demonstrated through an experiment that a consistency guideline, "The direction in which the contents of a window move when a mouse is used with a scrolling arrow at the edge of the window", which is strongly endorsed by the Apple Human Interface Guidelines (1992), may lead to poor design for certain tasks. This study showed that consistency guidelines should be applied cautiously to be in harmony with the user's task.

According to Grudin (1989) interface consistency is a largely unworkable concept and can sometimes work against good design. Grudin explained that consistency can be harmful with *Printing a folder of a directory* example. He considered the situation in which a folder containing several documents is selected and is followed by selection of print operation from the menu. The design question in this case is: What should be printed when the operation is executed? Informal user studies suggested that documents in the folder should be printed. However, system architects argued that a list of documents within the folder should be printed. A folder in this system was a list of pointers to documents. Since in this case, printing a document produced the contents of the documents and it was argued that printing a folder should similarly produce a copy of its contents i.e. list of documents in the folder. Although developers argued that consistency which was based on software architecture was important, the design was rejected in favor of consistency within system architecture. Therefore wrong dimension of consistency was chosen but was consistent with what users wanted. The bottom line is that the interface should be consistent with the user's task and fulfill the requirements of the targeted users.

GUI Terminology: Another important component of consistency is terminology used in the interface. Terminology used in interactive dialogs for text editing can sometimes be problematic for the user (Long et al., 1983). Inconsistencies in low-level interactions can be frustrating, for example programs that differ in the names of important commands (e.g., "quit" and "exit") (Grudin, 1989). A study done by Long, Hammond, Barnard and Morton (1983) showed that users in most text editing tasks refer first to objects in the text or in the system and then perform actions on them, so the use of proper terminology is an important factor in interactive dialog design. Several issues associated with the structuring of arguments with a set of commands were examined (Barnard et al., 1981) in three studies on consistency and compatibility in human-computer interaction. One of the implications of this study was that the users learned positionally consistent systems more readily when recurrent arguments were placed first.

Abbreviations are constructed to reduce typing and optimize the use of screen space, but can impose significant cognitive demands on a user in a new application. To create internally consistent design, one abbreviation algorithm should be used (Grudin, 1989)

Tools for Consistent Design: An important step in GUI design was taken by an Interactive Transition Systems (ITS) project (Wiecha et al, 1989). The ITS approach was to generate consistent interfaces automatically by the use of executable style rules. ITS provided a set of software tools to support four application development roles: an application expert, a style expert, an application programmer, and a style programmer. The ITS architecture divided the application into three parts, namely: application functions, a dialog manager, and views supporting user interfaces. This architecture helped to create a consistent interface for a family of applications and to create multiple consistent interfaces for a given application. ITS supported designers in generating consistent interfaces because it separated the application from the interface and linked them using executable style rules. ITS has generated interfaces for demonstration applications, but has not been used to create real-world applications.

Interface Evaluation Methods: Interface evaluation is a difficult and cumbersome process.and can be performed using various techniques. Evaluation of a software product's user interface using four techniques, namely heuristic evaluation, usability testing, guidelines and cognitive walk-throughs, showed that each has advantages and disadvantages (Jeffries et al, 1991) For instance, heuristic evaluation identifies more problems than any other method, but it requires UI expertise and several evaluators. Similarly, usability testing identifies serious and recurring problems, but requires UI expertise and has a high cost. The requirements for these powerful methods, which may include availability of working prototypes, test users, expert evaluators and time constraints are hindrances in applying these methods more frequently. The study also showed that usability testing, a powerful and effective evaluation method, is not good in evaluating design consistencies and missed consistency problems in its evaluation technique. Therefore, consistency checking tools are likely to be a beneficial complement to usability testing.

Furthermore, usability testing works best for smaller applications. It is practically infeasible to analyze every dialog box in an application with hundreds of dialog boxes with the current evaluation methods. Finding anomalies or differences while reviewing hundreds of dialog boxes is even hard for expert reviewers, who may fail to detect some flaws and inconsistencies. In contrast, automated evaluation tools can be used in early prototypes (or late iterations) and can detect anomalies across hundreds of dialog boxes. These automated tools, in addition to detecting anomalies, can make interfaces cleaner and easier to use.

1.2 Evaluation Tools for Visual Design and Textual Properties

Automated tools for consistency checking are meant to replace the current manual consistency checking process which is complex, expensive, error prone, and time consuming. These tools can be made independent of platform and development environment, as visual and textual properties of the interface are independent of these factors. In developing a system to evaluate alphanumeric displays, Tullis (1983) derived six measures (Overall Density, Local Density, Number of Groups, Size of Groups, Number of Items, Layout Complexity) to describe screen formats of spatial arrays of characters. These measures were later incorporated into a Display Analysis Program to analyze displays on IBM PC and PC-compatible computers, to develop a tool for objectively evaluating the usability of any alphanumeric display format (Tullis, 1988a). This Display Analysis Program was developed with two systems created specifically to test these programs and then tested with the displays of seven previous studies (Tullis, 1988b). The results indicated that given a set of alternative screen formats to present some alphanumeric data, this program can accurately predict the relative search times and subjective ratings for the formats. “Number of groups” and “Size of groups” were found to be the most important display parameters in determining search time.

Streveler and Wasserman (1987) proposed novel visual metrics to quantitatively assess screen formats which have similarities with Tullis's Display Analysis Program. They proposed three basic techniques for analyzing screen formats: “boxing”, “hot-spot” and “alignment” analysis. A balance measure was also proposed by them which computed the differences between the center of mass of the array of characters and the physical center of the screen. These proposed metrics were not applied to any system to validate them. The applicability of Tullis's complexity proposition was later applied to the domain of interactive system design with findings strongly supporting their applicability (Coll & Wingertsman, 1990). Choosing screen density as the measure of complexity, it was found that Madd's Modified Discrepancy Hypothesis in psychology is applicable to user interface design, i.e. users performance and preference for screen complexity follows a U-shaped curve, with too little or too much complexity depressing preference and performance. In other words, humans have preference and affinity for medium complexity.

Furthermore, Kim and Foley (1993) used metrics as a constant for the design space and the layout style. They developed a tool which generated potential design specifications and guidelines for the metrics. Effectiveness of their metrics has not yet been evaluated. The evolution of GUIs with multiple typefaces, colors, new kinds of widgets etc. means that more analysis is required and new metrics need to be implemented.

The evolution of modern user interfaces, like multimedia interfaces, has sparked research in automated evaluation based on visual techniques. Vanderdonckt and Gillo (1994) proposed five visual techniques (Physical, Composition, Association and dissociation, Ordering, Photographic techniques) which identified more visual design properties than traditional balance, symmetry, and alignment. Dynamic strategies for computer-aided visual placement of interaction objects on the basis of localization, dimensioning and arrangement have been introduced (Bodart et al., 1994).These techniques of localization, dimensioning and arrangement were based on some of the visual techniques introduced by Vanderdonckt and Gillo (1994). Mathematical relationships were defined to improve the practicability, the workability and the applicability of the visual principles into a systematic strategy, but specific metrics and acceptance ranges were not tested. Visual techniques introduced for multimedia layout frames have only rarely been applied to commercial applications.

Sears (1993, 1994) has developed a first generation tool (AIDE) using automated metrics for both design and evaluation using Layout Appropriateness metrics. In computing the Layout Appropriateness the designer provides the set of widgets used in the interface, the sequence of actions to be performed by the user and how frequently each sequence is used. The appropriateness of a given layout is computed by weighing the cost of each sequence of actions by how frequently the sequence is performed. Layout Appropriateness can be used to compare existing layouts and to generate optimal layouts for the designer. AIDE has demonstrated its effectiveness in analyzing and redesigning dialog boxes in simple Macintosh applications and also dialog boxes with complex control panels in NASA applications. Currently, studies are being done by Comber and Maltby (1995) in assesing the usefulness of layout complexity metric in evaluating the usability of different screen designs. Mullet (1995) developed a simple layout grid forming the basis of a systematic layout program embodying a set of guidelines that make it easy to position related controls consistently across dialogs. Using this systematic re-structuring and redesign approach he showed that the GUI of the “Authorware Professional”, a leading development tool for learning materials in the Macintosh and Windows environments, could be easily redesigned to create a more coherent, consistent and less crowded layout.

In order to help designers identify inconsistencies before usability testing, automated consistency checking tools have been developed to evaluate visual design and terminology in user interfaces (Shneiderman et al., 1995; Mahajan & Shneiderman, 1995)

1.4 Scientific Experiments on Effects of Interface Inconsistencies

Chimera and Shneiderman (1993) performed a controlled experiment to determine the effects of inconsistency on performance. This experiment used two interactive computer systems at the National Library of Medicine which were an original inconsistent version and a revised consistent version. Compared to the original version, the revised version had consistent screen layouts and colors plus use of consistent task and domain oriented phrases for the description of menu items. The results showed that there was a statistically significant difference ($p<.01$) favoring the revised interface for five out of twenty tasks and only one task favored the original interface ($p<.01$). It was concluded that the revised interface yielded faster performance and higher satisfaction due to how information was displayed with respect to location, wording and color choices.

Bajwa (1995) studied the effect of inconsistencies in color, location and size of widgets (in this case buttons) on user's performance and subjective satisfaction. To test the hypothesis that inconsistency deteriorates performance and subjective satisfaction, four versions of a Billing System interface were created in Visual Basic. The original version was consistent in accordance with most windows applications. The other three versions were made inconsistent with respect to color, location, or size, so that every inconsistent version had about 33% of only one type of inconsistency. The experiment was divided into three phases, namely inconsistent color versus consistent interface, inconsistent location versus consistent interface and inconsistent size versus consistent interface. For every phase of the experiment, subjects used both the consistent and inconsistent version with 50% of the subjects using the inconsistent version first and the other 50% using the consistent version first. With the participation of 60 subjects, the results of the experiment showed that inconsistency in color, location and size of objects significantly effects user's performance by about 5%. Although studies have been done to check the effects of inconsistencies in the interface design on user's performance, no experiments have been done for terminology inconsistencies specifically.

2.0 Description of the Evaluation Tools

2.1 Metrics Evaluation Using Canonical Format:

The present research evolved from the concept of converting interface form files generated by Visual Basic into canonical format files and feeding them as input to the SHERLOCK. The canonical format is an organized set of GUI object descriptions. These object descriptions are enclosed in curly braces and embrace interface visual design and terminology information in a sequence of attribute-value pairs. The canonical format is advantageous because of its lucidity and extendibility. It can be easily modified to include any new attributes encompassing interface description information in the form files.

A translator program was developed to convert the Visual Basic form files into the canonical format. Another translator program was created to convert the Visual C++ resource files into the canonical format and is currently being tested. These canonical formats are platform independent and may be created for other interface development tools like Power Builder, Galaxy and XVT by writing a translator program for those tools.

2.2 Evaluation Tools

Development of SHERLOCK is an extension of previous work (Shneiderman et al., 1995) in which spatial and textual evaluation tools were constructed. These tools have been modified after evaluating sample applications and new tools have been integrated with them into a family of consistency checking tools leading to the evolution of SHERLOCK. Our focus was on evaluating only certain aspects of consistency in user interfaces which are task-independent and can be automated. One of the task-independent features evaluated by our tools is visual design which includes properties such as sizes of similar screens, placement of similar items, screen density, consistency in margins, screen balance and alignment. Consistency in other visual design properties such as fonts, font-sizes, font-styles and background and foreground colors has also been evaluated. Finally our evaluation includes checking for terminology inconsistencies, abbreviations, variant capitalization and spelling errors.

Dialog Box Summary Table

The dialog box summary table is a compact overview of the visual design of dozens or hundreds of dialog boxes of the interface. Each row represents a dialog box and each column represents a single metric. Typical use would be to scan down the columns looking for extreme values, spotting inconsistencies, and understanding patterns within the design.

Choosing the appropriate metrics set was the most important factor in the design of the dialog box summary table. The researchers at the University of Maryland generated a list of approximately 40 metrics which were constructed after reviewing the relevant previous literature, consulting with colleagues and using their GUI evaluation experience. A similar effort was taken on the GE side, where they brain-stormed and proposed their metric set based on their commercial software development experience. The two lists had many similar items and the lists were grouped into categories such as spatial layout, alignment, clustering, cluttering, color usage, fonts, attention getting, etc. The metric set has been revised several times after evaluating a series of interfaces. The metrics that were ineffective have been removed and others have been redefined and new metrics have been added. The modified column set of the dialog box summary table is explained below:

Aspect Ratio: The ratio of the height of a dialog box to its width. Numbers in the range 0.5 thru 0.8 are desirable. Dialogs that perform similar functions should have the same aspect ratio.

Widget Totals: Count 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, lists and combo boxes 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 for redesign.

Widget Density: The number of top-level widgets divided by the total area of the dialog box (multiplied by 100,000 to normalize it). High numbers greater than 100 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 box.

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 box, and across different dialog boxes.

Gridedness: Gridedness is a measure of alignment of widgets. This metric has been refined several times, but we have not been able to find a perfect metric to detect misaligned widgets. X-Gridedness counts the number of stacks of widgets with the same X coordinates (excluding labels). Similarly Y-Gridedness counts the number of stacks of the widgets with the same Y coordinates. High values of X-Gridedness and Y-Gridedness indicate the possibility of misaligned widgets. An extension of Gridedness is Button Gridedness where the above metrics are applied to button widgets.

Fig. 1 shows how the Gridedness metric works. The dialog box on the right half of the figure has all the three buttons (OK, No and Cancel) with the same Y-coordinates, so the Y-Gridedness in buttons is 1 and X-Gridedness in buttons is 0. On the otherhand, the dialog box on the left half of the figure has a misaligned Cancel button from the OK and No buttons which have the same Y-coordinate. Thus, the Cancel button has a different Y-value forming a different stack, this misalignment introduces a new stack in both X and Y direction increasing the Y-Gridedness to 2 and X-Gridedness to 1. So every time there is a misaligned button the X- and Y-Gridedness values are increased by 1.

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 box to the total widget area in the right half of the dialog box; and the vertical balance, which uses top area divided by bottom area. High values of balances between 4.0 and 10.0 indicate screens are not well balanced. The limiting value 10.0 represents a blank or almost blank (for example, a dialog box that has only one widget which is a button) dialog box.

Distinct Typefaces: Typeface consists of a font, font size, bold and italics information. Each distinct typeface in all the dialog boxes is randomly assigned an integer to facilitate quick interpretation.

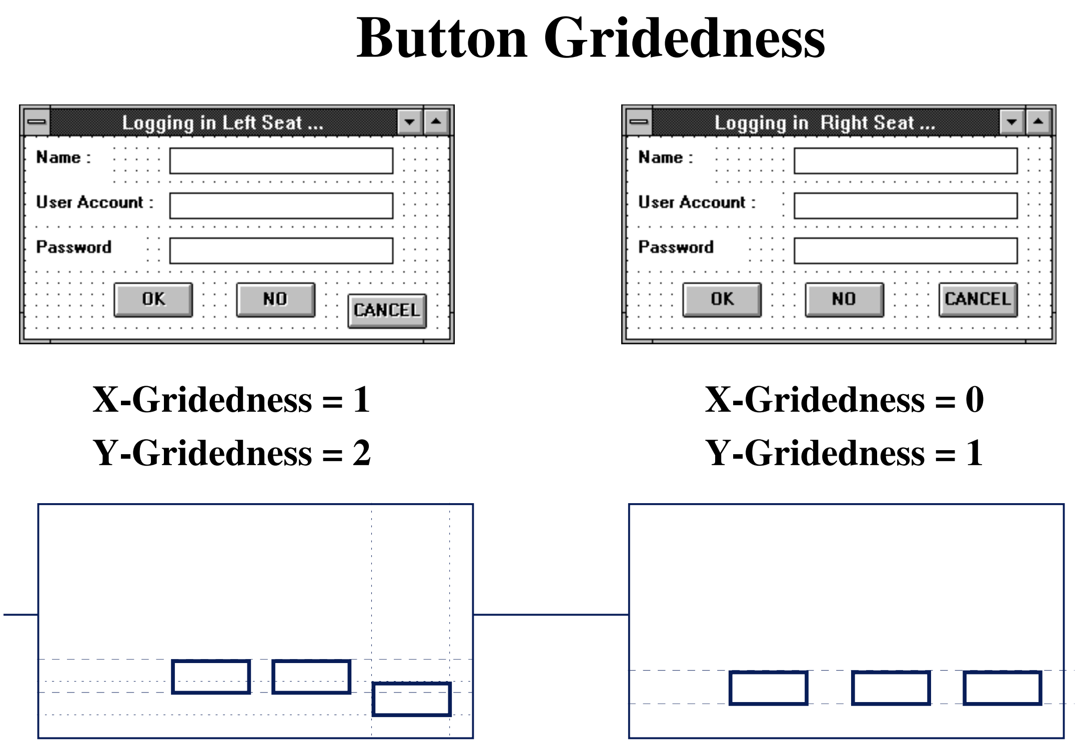


Fig. 1. Working of Button Gridedness

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 small number of typefaces should be used for all the dialog boxes.

Distinct Background Colors: All the distinct background colors (RGB values) in a dialog box are displayed. Each distinct color is 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 metric is to check if all the dialog boxes have consistent background colors. Multiple background colors in a dialog box may indicate inconsistency.

Distinct Foreground Colors: Similar to distinct background colors, displays all the distinct foreground colors in a dialog box. The purpose of this metric is to check if all the dialog boxes have consistent foreground colors.

Margin Analyzer

Margin analyzer is an extension of the dialog box summary table's margins metric. This analyzer calculates the most frequently occurring values of left, right, top and bottom margins across the interface and then lists margins in every dialog box which are inconsistent with these frequently occurring values. It also calculates what widgets of the dialog box need to be moved by how many pixels to make the margins consistent. The Margin analyzer tool depends on the fact that the most frequently occurring value of margins are the optimum margin values which the designer would have ideally used for consistency.

Concordance

The concordance tool extracts all the words that appear in every dialog box and helps designers with appropriate word use such as spelling, abbreviation, tense consistency, case consistency, passive/active voice etc. Occurrences of words in a different case are preserved as unique occurrences of words in a sorted list to point out the use of different case. The sort order used was aAbB...zZ so that the occurrence of "cancel" is not separated from "Cancel" or "CANCEL". The concordance tool has been broken down further to extract specific information related to spelling, abbreviation and case consistency to expedite the quick evaluation process.

Interface Concordance

The interface concordance tool checks for variant capitalization for all the terms that appear in buttons, labels , menus, etc. in every dialog box of the interface. This tool outputs strings which have variant capitalization, listing all the variant formats of the string and its dialog box sources. These variant forms are spelling differences and may be acceptable, but they may be something that should be reconsidered. For example the words "MESSAGES", "messages", "Messages" and "mesgs" are variant capitalization forms of the same word.

Button Concordance

As buttons are one of the most frequently used widgets performing vital functions like "Save", "Open", "Delete", "Exit" etc., checking consistency in their size, placement, typefaces, colors and case usage becomes more important. This tool outputs all the buttons used in the interface, listing the dialog boxes containing the buttons plus fonts, colors and button sizes. The button concordance identifies variant capitalization, distinct typefaces, distinct foreground colors and variant sizes in buttons.

Button Layout Table

Given a set of buttons that frequently occur together (for example, OK Cancel, Close, Help), if the first button in the set is detected in the dialog box then the program outputs the height, width and position relative to the first button of every button detected in the list. The relative position of every button detected in the set is output as (x + offset, y + offset) to the first button, where offset is in pixels. Buttons stacked in rows would yield a (x + offset, y) relative position and those stacked in columns would yield (x, y + offset). The Button Layout table identifies inconsistencies in button placement, inconsistencies in button terminology and variant button sizes locally within a dialog box and globally across all the dialog boxes. Some of the sample button sets are:

- `OK Cancel Close Exit Quit Help`
- `Start Stop Halt Pause Cancel Close Done End Exit Quit`
- `Add Remove Delete Copy Clear`
- `Cancel Close Exit`

Interface Speller

Interface Speller is a spell checking tool which reads all the terms used in widgets throughout the interface and outputs terms that are not found in the dictionary. The spell checking operation is performed within the code and all the possible misspelled words are stored in a file. This file can be reviewed by the designer to detect possible misspelled and abbreviated words which may create confusion for end users. The output is filtered through a file containing valid computer terms and default Visual Basic terms that may be flagged as spelling errors by the dictionary.

Terminology Baskets

A terminology basket is a collection of computer terms including their different tense formats which may be inadvertently used as synonyms by the interface designers. Our goal is to construct different sets of terminology baskets by constructing our own computer thesaurus and then search for these baskets in every dialog box of the interface. The purpose of terminology baskets is to provide interface designers with feedback on misleading synonymous computer terms, like "Close", "Cancel", "End", "Exit", "Terminate", "Quit". The program reads an ASCII file containing the basket list. For each basket all the dialog boxes containing any of the basket terms are output. Some of the idiosyncratic baskets are:

- `Remove Removes Removed Removing Delete Deletes Deleted Deleting Clear Clears Cleared Clearing Purge Purges Purged Purging Cancel`

Cancels Canceled Canceling Refresh Refreshed

- Item Items Entry Entries Record Records Segment Segments Segmented Segmenting Field Fields
- Message Messages Note Notes Letter Letters Comment Comments

3.0 Interface Evaluations

3.1 Testing the Evaluation Tools

The effectiveness of SHERLOCK tools has been determined by evaluating four commercial prototype applications developed in Microsoft Visual Basic. These applications included a 139 and 30 dialog box GE Electronic Data Interchange Interface, a 75 dialog box Italian Business Application, and a set of University of Maryland AT&T Teaching Theater Interfaces combined together into an 80 dialog box application. The analysis of the Italian Business Application is not discussed in this paper because its results detected inconsistencies similar to the other applications.

3.2 Evaluation Results, GE Interfaces

The 139 dialog box GE Electronic Data Interchange Interface was the first prototype evaluated. Although this was a well-reviewed and polished design, SHERLOCK detected some inconsistencies which may have otherwise been left undetected. Another small 30 dialog box GE interface was evaluated which also revealed inconsistencies.

Dialog Box Summary Table Analysis

Aspect Ratio: Aspect Ratio varied from 0.32 to 1.00 and some dialog boxes which performed the same functionality had different aspect ratio, which was an inconsistency.

Non-widget Area: Non-Widget Area varied from 2% to 97.5%. Some dialog boxes with low Non-widget area (5% to 15%) were candidates for redesign.

Widget Density: Widget Density varied from 14 to 271, but most of the high values were due to exceptions in the metric, as none of the dialog boxes had too many widgets in a small area.

Margins: Left, right, top and bottom margins were inconsistent within a single dialog box and were also inconsistent across the interface. For example, the average value of the left margin was 12 pixels, but the margin value ranged across the interface from 0 to 80 pixels.

Gridedness: Some high values of the Button Gridedness (3 or more) metric helped in detecting dialog boxes with misaligned buttons.

Area Balances: Dialog boxes were well balanced as the average value of Left/Right Balance and Top/Bottom Balance was 1.1 and 1.4 respectively.

Distinct Typefaces: Although most of the dialog boxes used a single typeface (MS Sans Serif 8.25 Bold), there were a couple which used more than three typefaces Altogether seven distinct typefaces were used.

Distinct Background & Foreground Colors: There was much variation in color usage among different dialog boxes, indicating inconsistency. The interface used a total of eight foreground and seven background colors (RGB Values).

Margin Analyzer

The margin analyzer successfully detected the dialog boxes which had margin values more than two pixels apart from the most frequently occurring value in both the applications. For each inconsistent value, it listed the widgets that need to be moved and by how many pixels to alleviate the inconsistency. In the case of the smaller application, the margin values were highly inconsistent, since most frequently occurring margin values had a maximum envelope of six to eight dialog boxes, making the other values inconsistent. These results showed that the design didn't adhere to any concept of consistent margins. There were some exceptions in Visual Basic (Visual Basic allowing widgets to extend beyond the area enclosed by the dialog box and size of label and text boxes being greater than size of text enclosed by them) beyond the capability of the tool to handle, leading to negative margins

Interface Concordance

The interface concordance tool spotted the terms that used more than one case across the application. For example, terms like “Messages”, “MESSAGES” and “messages” were detected by the interface concordance tool. Some of the other inconsistencies included terms like “Item”, “item” and “Item”, “Open”, “OPEN” and “open”.

Button Concordance

GE Interfaces did not have any button labels which used more than one case. All the button labels used the title format and were therefore consistent. Also, all the buttons used the same typeface and foreground color. Button Concordance detected inconsistency in height and width of the buttons across the interface. The table below shows a portion of the button concordance output for the “Archive” button. Browsing across the columns of the table, we can see that the width of the “Archive” button varies between 65 and 105 pixels. All the buttons have a top margin of 0 pixels except one (file.find.cft) which has a top margin of 312 pixels. This is an inconsistency, since all the “Archive” buttons are placed at the top right corner of the dialog box except one which is placed at the bottom right corner. Button placement inconsistencies were detected in many other buttons including “OK”, “Cancel”, “Close”, “Find”, “Forward”, and “Print”

```
BUTTON     DIALOG      BUTTON     BUTTON    BUTTON     BUTTON POSITION
LABEL       BOX       TYPEFACE   FG_COLOR   (H , W)   LEFT   RIGHT   TOP
Archive   xref.cft       1           1       25,105    208     311     0
          file.cft       1           1       25,89     448      87     0
          file2.cft      1           1       25,73     360      72     0
          filefind.cft   1           1       25,73     408     142   312
          hold.cft       1           1       25,65     320      55     0
          in.cft         1           1       25,81     464      79     0
          out.cft        1           1       25,73     304      55     0
          sent.cft       1           1       25,81     344      78     0

DISTINCT TYPEFACES IN BUTTONS:
1 = MS Sans Serif 8.25 Bold No Label
DISTINCT FOREGROUND COLORS IN BUTTONS:
1 = Default Color
```

Interface Speller

The tool detected few misspelled terms, but many potentially confusing incomplete and abbreviated words such as “Apps”, “Trans”, “Ins” , “Oprs” were found in both the applications.

Terminology Baskets

The basket browser revealed some interesting terminology anomalies after analyzing the large 130 dialog box interface that led to reconsideration of the design. As shown below terms like “record”, “segment”, “field” and “item” were used in similar context in different dialog boxes. Other interesting inconsistencies included the use of “start”, “execute” and “run” for identical tasks in different dialog boxes . Also, the small 30 dialog box interface had terminology inconsistencies, such as using the terms like “Show”, “View”, and “Display” to perform similar tasks.

```
-----------------------------------------------------------------------
Basket: Entries, Entry, Field, Fields, Item, Itemized, Itemizing, Items
        Record, Records, Segment, Segmented , Segmenting, Segments
-----------------------------------------------------------------------
BASKET TERM        FORM CONTAINING THE BASKET TERM
Field              search.cft
Items              reconly.cft       reconly.cft       reconly.cft
                   reconly.cft       sendrec.cft       sendrec.cft
                   sendrec.cft       sendrec.cft       wastedef.cft
Record             ffadm.cft         profile.cft
Segment            addr.cft          search.cft
```

Button Layout Table

The most common button positional and terminology inconsistency was in the button set [OK Cancel Close Exit Help], the button labels "Cancel", "Close" and "Exit" were used interchangeably. Sometimes these buttons were stacked in a column on the top left corner of the dialog box and in other cases they were stacked in a row at the bottom of the dialog box and were either left, right or center aligned.

Output of the button detector set (OK Cancel Close Exit Quit Help) tested with the small 30 dialog box GE application is shown below. Inconsistency in height and relative button positions within a button set can be checked by moving across the rows of the table. Inconsistency in height and relative position for a particular button can be spotted by moving down the columns. For example, browsing the "OK" button column we found that the height of the "OK" button varied between 22 and 26 pixels and the width varied between 62 and 82 pixels. Also, scanning across the rows, we found that the relative position between "OK" and "Cancel" buttons varied in all the three dialog boxes in which they occurred together. In the dialog boxes "nbatch.cft" and "systinp.cft" the "Cancel" button was 20 pixels and 13 pixels below the "OK" button, but in the dialog box "admprof.cft" the buttons occurred next to each other in the same row. Also, both the buttons "Cancel" and "Exit" were used with the "OK" button essentially to perform the same task. This is a terminology inconsistency.

```
------------------------------------------------------------------------------
DIALOG BOX     OK          Cancel             Exit         Help
              (H,W)   (H,W) Rel. Pos.    (H,W) Rel. Pos.  (H,W) Rel. Pos.
------------------------------------------------------------------------------
admprof.cft   22,68   22,68  x+16, y                       22,68  x+98, y
checkpsw.cft  25,82                      25,82  x+18, y    25,82  x+116, y+1
nbatch.cft    25,62   25,62  x-1, y+20                      25,62  x, y+66
systinp.cft   26,72   26,72  x+1, y+13                     25,73  x+2, y+48
```

3.3 Evaluation of University of Maryland Interface

The 80 dialog box University of Maryland AT&T Teaching Theater Interface was a combination of different applications all designed for the students to use. Evaluation of this interface highlighted the intra-application inconsistencies that may exist among applications designed for the same user.

Dialog Box Summary Table Analysis

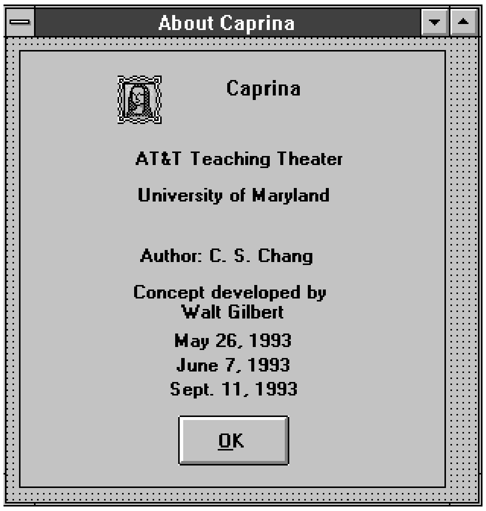
A portion of the dialog box summary table is shown in Fig. 3.

Aspect Ratio: Aspect Ratio, in general varied between 0.5 and 0.8. There were a few dialog boxes whose Aspect Ratio were on the lower side, (left_or_right.cft, ratefrm2.cft and zoom.frm.cft) and a few had a high aspect ratio of 0.9 or more. (about1.frm.cft, about2.frm.cft, delete.frm.cft and notice.frm.cft). All the About (Fig. 2), Cover and Exit dialog boxes had different aspect ratios. Although most of these dialog boxes belong to a different application, these applications have been designed for the same set of users and these inconsistencies in Aspect Ratio, especially in the dialog boxes with the same functionality, should not exist.

Widget Totals: Some dialog boxes had a high value of widget totals i.e. 70 or more widgets. This may indicate complexity in the dialog box.

Non-Widget Area: High values of Non-Widget area (above 90%) were found in some of the dialog boxes including main.frm.cft, syllabus.frm.cft, vdmdi3.frm.cft, winstat.frm.cft. This indicates that the use of screen space may not be optimum in the above cases. Dialog boxes filelist.frm.cft, winchat8.frm.cft and zoom.frm.cft had low Non-widget areas.

Widget Density: High values of widget density (around 150 or more) in the dialog boxes like ibm_az.frm.cft, rate.frm.cft and seat_uaz.frm may indicate that too many widgets are present in a small area. Those dialog boxes which had high widget density, but had Non-widget area of 40% or more may be acceptable.



About Caprina

Caprina

AT&T Teaching Theater

University of Maryland

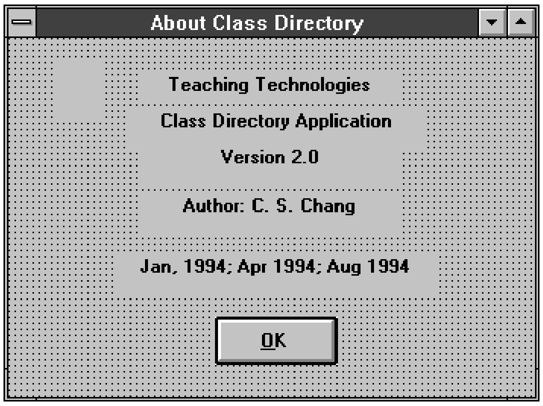
Author: C. S. Chang

Concept developed by
Walt Gilbert

May 26, 1993
June 7, 1993
Sept. 11, 1993

OK

About2.frm = 0.99



About Class Directory

Teaching Technologies

Class Directory Application

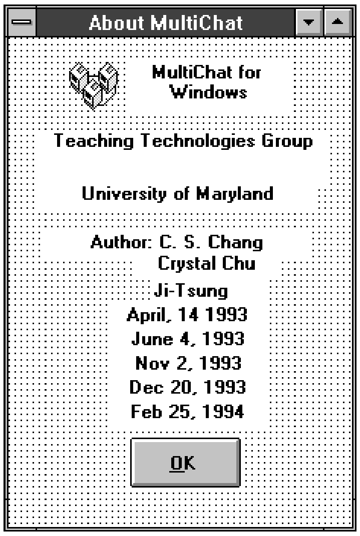
Version 2.0

Author: C. S. Chang

Jan, 1994; Apr 1994; Aug 1994

OK

Aboutbox.frm = 0.68



About MultiChat

MultiChat for
Windows

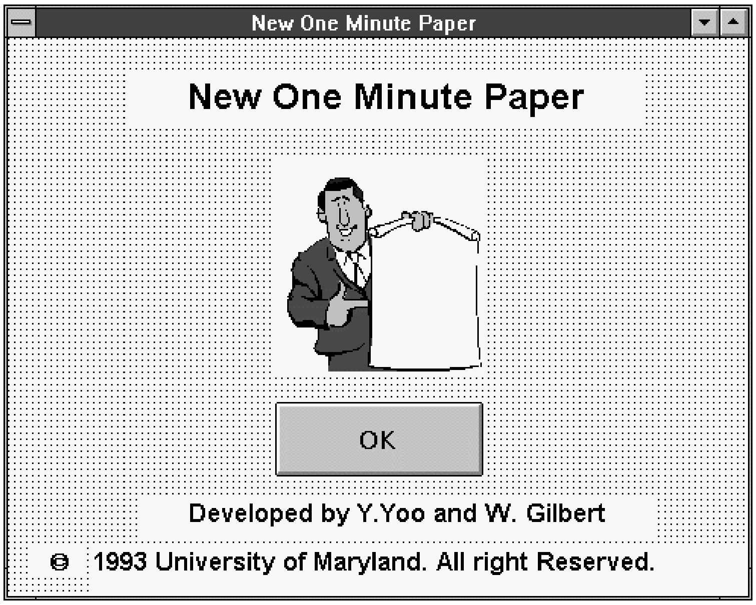
Teaching Technologies Group

University of Maryland

Author: C. S. Chang
Crystal Chu
Ji-Tsung
April, 14 1993
June 4, 1993
Nov 2, 1993
Dec 20, 1993
Feb 25, 1994

OK

About_m.frm =1.5



New One Minute Paper

New One Minute Paper

OK

Developed by Y.Yoo and W. Gilbert

© 1993 University of Maryland. All right Reserved.

About_n.frm = .84

Fig. 2. Aspect Ratio Inconsistencies

Margins: Left margins varied from 0 to 192 pixels, although the most frequently used margin values were between 8 to 16 pixels. A quarter of the dialog boxes had left margin values of 0 pixels and a few had high values above 70 pixels. Right margins varied from 0 to 381 pixels. In some cases high values of the right margins were not a problem, like the case when the dialog box only had labels and buttons which are in general center aligned. Top margin varied from 0 to 56 pixels and was more consistent than left and right margins. Similarly, bottom margins were more consistent than left and right margins with values clustered between 8 and 30 pixels.

```
No.  Dialog      Aspect  -WIDGET--    Non-  Widget   -----M A R G I N S------   ----GRIDEDNESS-----   -Balances--   Distinct      D
    Name          Ratio   TOTALS    Widget Density    Left   Right Top Bottom   Top Level  Buttons   Area Ratios    Typefaces     B
                 (H/W)  All  Top-    Area   widget/          (pixels)             X    Y    X    Y     Horiz  Vert
                             Level     (%)   area                                                      (L/R) (T/B)

 30 form9.cft     0.74   19    13    88.7      69       0   381     0    17       3    3    0    0     10.0  2.4          4 9

 31 frmcompaz.cft 0.79    5     4    67.5      11     104    97    56    69       1    2    0    1      1.0  1.4         9 13

 32 frmcompu.cft  0.79    5     4    67.5      11     104   101    56    69       1    2    0    1      1.0  1.4         9 13

 33 frmhand.cft   0.48    6     5    67.5      33      32    31    32    31       1    2    0    1      1.0  1.5         9 13

 34 frmlogin.cft  0.85    8     7    77.1      25      96    96    40    64       1    1    0    1      0.6  1.7         9 13

 35 frmlogo.cft   0.38    9     8    18.3      25       0   223     0     0       2    2    1    1      0.8  1.3     4 9 16 17

 36 frmmatch.cft  0.50    7     6    70.0      60      16    50    24    43       1    1    0    1      0.8  1.3            4

 37 frmquesaz.cft 0.42    6     5    75.6      25      56    54    48    61       0    1    0    1      1.1  1.5         9 13

 38 frmquesu.cft  0.45    6     5    73.9      23      72    14    48    74       0    1    0    1      0.8  1.3         9 13

 39 graph.cft     0.55   14     4    57.0      22       0     0     0     7       2    3    0    1      1.0  1.0          7 8

 40 grid3.cft     0.89    5     4    42.5      23      21    15    13    60       2    1    1    0      1.5  1.2            4


Maximum           1.60  102   101   100.0     184     192   381    56   276       9   13    2    3     10.0 10.0
Minimum           0.13    0     0     0.0       0       0     0     0     0       0    0    0    0      0.0  0.0
Average           0.73   14     9    57.5      48      19    52    11    29       2    2    0    0      1.8  1.6

DISTINCT TYPEFACES:

1 = Arial 13.5 Bold                        11 = Symbol 13.5 Bold
2 = Symbol 9.75 Bold                       12 = MS Sans Serif 24 Bold Italic
3 = Arial 8.25 Bold                        13 = MS Sans Serif 13.5 Bold
4 = MS Sans Serif 8.25 Bold                14 = MS Sans Serif 18
5 = System 9.75 Bold                       15 = MS Serif 30 Bold
6 = Arial 15.75 Bold                       16 = Arial 18 Bold
7 = MS Sans Serif 9.75 Bold                17 = Symbol 8.25 Bold
8 = MS Sans Serif 16.5 Bold                18 = Times New Roman 24 Bold Italic
9 = MS Sans Serif 12 Bold                  19 = Times New Roman 30 Bold Italic
10 =MS Sans Serif 13.5


DISTINCT BACKGROUND COLORS:                DISTINCT FOREGROUND COLORS:

1 = ffffff                                 2 = ffffffff80000005
2 = ffffffff80000005                       3 = ffffffff80000008
5 = c0c0c0                                 4 = 0
6 = ff                                     6 = ff
8 = e0ffff                                 7 = ff0000
10 =c00000                                 9 = c000c0
12  404040                                 10 =c00000
14 =ffffffff8000000f                       11 =ffff
                                           13 =808080
                                           15 =c000
```

Fig. 3. A Portion of Dialog Box Summary Table

Gridedness: Most dialog boxes had well aligned widgets, with low X-Gridedness and Y-Gridedness values (1 or 2). Some dialog boxes which had high values of Gridedness (4 or more) like picture3.frm, ratefrm2.frm, seat_uaz.frm, topic.frm, tqmain.frm, winstat.frm required minor alignment changes. A small number of dialog boxes like cover.frm, qmain3.frm, mulq.frm, omp.frm had higher values of Button Gridedness due to misalignment of buttons by a few pixels.

Area Balances: Dialog boxes like dynaset.frm, dyngrid.frm, form9.frm and winstat.frm had high balance values indicating that they may not have well balanced screens.

Distinct Typefaces: In total 19 distinct typefaces were used which is very high. This shows that different designers worked on different applications without following any guidelines. It is recommended that the applications should be modified to decrease the use of too many typefaces. Some of the dialog boxes that used four or more different typefaces are about.frm, cover.frm, coveraf.frm, coveruf.frm and frmlogo.frm.

Distinct Background Colors: This application uses 8 background colors. This may be attributed to the fact that different dialog boxes had different styles which means specific groups of designers worked on particular applications.

Distinct Foreground Colors: Altogether the application used 15 different colors (both background and foreground). The number of foreground colors used was 10 which is high.

Interface Concordance

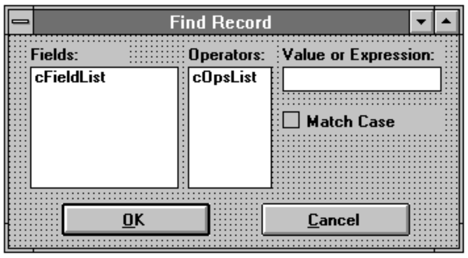
There are a few terms which used different case across the application like Cancel, cancel, CANCEL, Delete, DELETE and more. Designers need to check whether these are inconsistencies or not.

Button Concordance

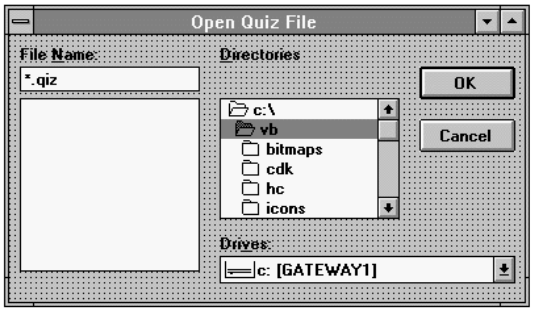
The following are the inconsistencies detected by the Button Concordance Tool:

- Designers used six distinct typefaces in Button Labels which is inconsistent. A single typeface should be used for all button labels.
- Designer used three distinct foreground colors in button labels which is inconsistent. Like the typefaces, a single foreground color should be used for all the button labels.

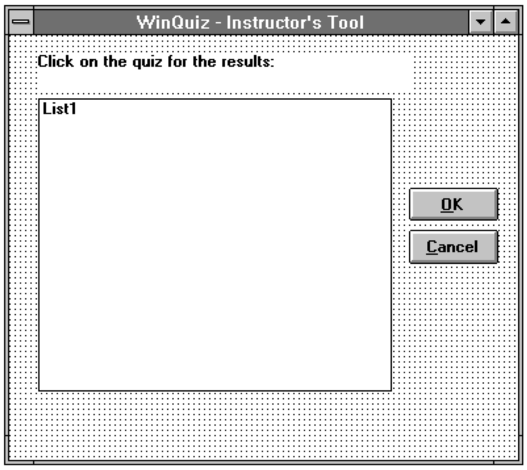
```
BUTTON        DIALOG      BUTTON     BUTTON    BUTTON     BUTTON POSITION
LABEL          BOX       TYPEFACE   FG_COLOR   (H , W)   LEFT   RIGHT   TOP
Exit         attapp94.cft   1           2       56,120   464       56   240
             cover.cft      2           2       57,153   680      182   536
             coveraf.cft    3           2       41,89    448        0   392
             coveruf.cft    3           2       41,89    448       85   392
             frmhand.cft    4           3       49,105   384       31   136
             frmlogin.cft   4           3       41,97    248       96   256
             winstat.cft    6           1       49,113   368       70   424

EXIT         delete.cft     1           1       41,97    280       45   352
             syllabus.cft   1           1       33,137   856        7   512

Left SAVE    feed.cft       5           2       33,81    272      647   448
Left Save    omp.cft        5           3       33,97    112      796   456
Right SAVE   feed.cft       5           2       33,89    648      263   448
Right Save   omp.cft        5           3       33,97    544      364   456
SAVE Left    mulq.cft       5           2       33,97    256      653   488
SAVE Right   mulq.cft       5           2       33,97    504      405   488

DISTINCT TYPEFACES IN BUTTONS:        DISTINCT FOREGROUND COLORS IN BUTTONS:
1 = MS Sans Serif 8.25 Bold           1 = Default Color
2 = MS Sans Serif 18                  2 = ffffffff80000005
3 = MS Sans Serif 13.5                3 = 0
4 = MS Sans Serif 12 Bold             4 = ff0000
5 = MS Sans Serif 9.75 Bold
6 = MS Serif 12 Bold
```



find.frm

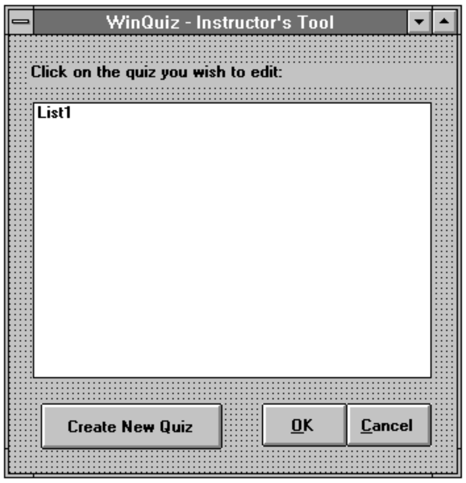


opensav.frm



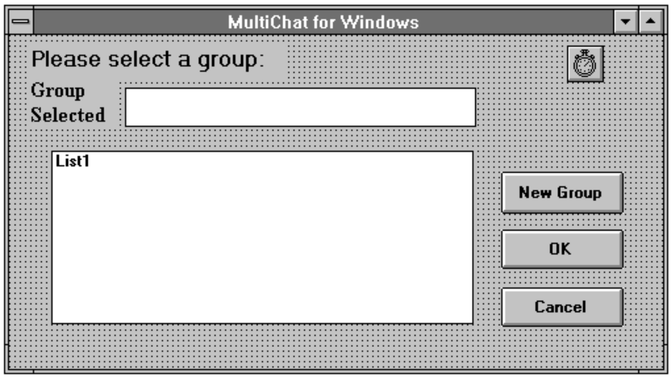
Click on the quiz for the results:

grid3.frm



Click on the quiz you wish to edit:

listque2.frm



Please select a group:

list2.frm

Fig. 4. Button placement inconsistencies in OK and Cancel buttons.

- Button sizes are inconsistent across the application. For example, the height of the "Cancel" button varies between 24 and 49 pixels and width varies between 57 and 122 pixels. Other buttons that have inconsistencies in button sizes include "OK", "Done", "Exit", "No", "Previous" and "Start" and more.
- Buttons like "OK", "Cancel", "Done", "Exit", " No" used different case across the

application which is an inconsistency. Also, the designers used the button labels “Save Left” and “Save Right” in some dialog boxes and “Left Save” and “Right Save” in others which is an inconsistency.

- The button positions metric detected many inconsistencies. For example, the “Cancel” button had a different right button position for every dialog box. In the case of the “Close” button, the left position was 8 pixels in two dialog boxes and was 291 in the third, indicating that the “Close” button is left aligned in the first case and right aligned in the second case. Similar inconsistencies existed in buttons like “Done”, “Exit”, “OK” and more.

Interface Speller

The spell checking tool detected various abbreviations and a few misspelled terms. The following were the spelling errors detected: “qiz”, “veryfying”, “peronal” and “btrieve”.

Terminology Baskets

The output from the basket [Browse, Display, Find, Retrieve, Search, Select, Show, View] shows that “Display”, “View” and “Show” have all been used in this application. Also, both “Find” and “Search” have been used. Similarly the output from the basket [Cancel, Clear, Delete, Purge, Refresh, Remove] indicates that the terms “Cancel”, “Delete”, “Clear”, “Refresh” and “Remove” were all used in the application. Designers need to check whether these are inconsistencies.

Button Layout Table:

The Button Layout Table revealed inconsistencies in button sizes and placement within a dialog box and across the interface. For example, the button set [OK Cancel ... Exit Help] revealed inconsistencies in the sizes of the “OK” “Cancel” and “Help” buttons. Also, the ”Cancel” and “Help” buttons were in some cases placed next to OK buttons in a row and other times stacked below the “OK” button in a column, with the distance between these buttons varying from 0 to 40 pixels. Fig. 4 shows the dialog boxes in which button placement inconsistencies of “OK” and “Cancel” buttons were detected by the Button Layout Table and Button Concordance Tool.

3.5 Conclusion

Evaluation of the four applications using SHERLOCK tools helped us to determine those tools which were most successful in detecting inconsistencies and those that were less successful. The dialog box summary table had limited success in detecting inconsistencies. Only certain metrics of the dialog box summary table like Aspect Ratio, Margins, Distinct Typefaces, Distinct Foreground and Background Colors were successful in finding inconsistencies. Most of the extreme values computed by the metrics like Non-Widget Area, Widget Density and Area Balances were due to the limitations of SHERLOCK or the Visual Basic development tool and were not real inconsistencies. These metrics were modified several times to deal with exceptions and further work is required to validate these metrics. The Button Concordance and the Button Layout Table proved to be the most useful tools and were able to detect inconsistencies in the size, position, typeface, color and terminology used in buttons. The Interface Concordance and the Interface Speller tools were successful in detecting terminology inconsistencies like variant capitalization, abbreviations and spelling errors. The Terminology Basket tool helped in detecting misleading synonym terms in many cases. In summary SHERLOCK, was successful in detecting major terminology inconsistencies and certain inconsistencies in visual design of the evaluated interfaces.

SHERLOCK is a collection of programs that require detailed knowledge to use effectively. Additional programming and a graphic user interface would be necessary to make it viable as a software engineering tool. However, the source code and documentation that exists are available (http://www.cs.umd.edu/projects/hcil).

3.6 Limitations of SHERLOCK

SHERLOCK evaluation is limited to certain visual design and terminology aspects of the interface. Issues like efficiency in screen layout including proper placement of widgets on the dialog box, violations of design constraints, and the use of inappropriate widgets types are not evaluated by SHERLOCK. Other evaluation methods, such as usability testing and heuristic evaluation, are needed to locate typical user interface design problems such as inappropriate metaphors, missing functionality, chaotic screen layouts, unexpected sequencing of screens, misleading menus, excessive demands on short-term memory, poor error messages, or inadequate help screens.

4.0 Feedback From Designers

Output from the tools and the screen shots of the interface along with the analyses were forwarded to the developers and designers to elicit feedback.

4.1 GE Interfaces

We worked closely with the people at GE Information Services to get feedback on the effectiveness of SHERLOCK tools, as these tools were being iteratively refined. The feedback was positive on the evaluation output with suggestions for modifications at every refinement stage.

The feedback suggested that the outputs of the dialog box summary table were simple for the designers to interpret, as they were able to detect inconsistencies by scanning down the columns for extreme values, indicated by the statistical analysis at the bottom of the table. They recommended that we develop some "goodness" measures for the metrics after analyzing more applications. We have succeeded partly in assigning measures to certain metrics after analyzing the four applications. The detection of the use of multiple typefaces and colors by SHERLOCK is one of the inconsistencies they otherwise would have missed. Inconsistent margins was another dimension that would have been slipped through testing, if not detected by SHERLOCK. Although, most of the dialog boxes in the GE interface neither had a cluttered or crowded layout, Non-Widget Area and Widget Density were two of the metrics which the GE designers agreed were important in determining the layout of the interface.

The incorporation of a spell checking tool in SHERLOCK had a positive response from the designers, since none of the current GUI building environments on the PCs have a built in spell checker. Separate detailed analysis of each metric of the dialog box summary table was recommended by the designers for the future implementations. One of the steps taken in this direction was the development of the Margin Analyzer tool to indicate inconsistencies in margins and the way to rectify those inconsistencies. Another tool that had positive results in detecting inconsistencies was the button concordance tool. The extreme variations in button sizes detected by this tool that included instances with no two same label buttons having the same size across the application, raised concerns within GE design team. The button typeface, size and placement inconsistencies detected by the Button Concordance tool were corrected by the GE designers after reviewing the evaluations performed by SHERLOCK.

Inconsistencies in relative positioning of button labels and button terminology detected by the Button Layout table were modified by the GE designers using the output. Use of misleading terminology was another dimension explored by the terminology basket tool which helped GE designers in rectifying a few terminology inconsistencies which would have been missed otherwise. Overall the use of SHERLOCK helped to modify the layout, visibility and terminology of GE interfaces by detecting many small inconsistencies.

4.2 University of Maryland Interface

Since this application was a combination of various small applications, the output was given to two different design groups to elicit a broader spectrum of feedback. The first group included the developers of a portion of the interface and the other was the designers responsible for all the applications together.

Developers' feedback on the dialog box summary table was positive for some metrics. They showed interest in the ability of the dialog box summary table to detect the typeface and color inconsistencies in their application. When asked for a possible explanation of these inconsistencies, they explained that different designers worked on different portions of the application, with very few guidelines on visual design. Similar reasons were given for other inconsistencies including different aspect ratio's for functionally similar screens and the use of inconsistent margins.

Designers liked the statistical analysis at the end of the metric table with mean, maximum and minimum values and wanted an additional function that listed the optimum values for the metrics. Many of the terminology inconsistencies detected by the Button Layout Table and the Terminology Basket tool were valid inconsistencies which they will take into consideration for the next version of the application.

5.0 Software Design

SHERLOCK is a set of 7 user interface consistency checking programs which were implemented in about 7000 lines of C++ code, and developed on the SUN SPARC Stations/UNIX platform. In order to evaluate a Graphical User Interface using SHERLOCK, its interface description files need to be converted to a canoncial format [see section 2.1]. These canonical format files are the only input required by the SHERLOCK evaluation tools. SHERLOCK was designed to be a generic GUI consistency and evaluation tool.

5.1 Translator and Canonical Format Design

Translator programs are specifically designed for a particular GUI development tool and converts its interface description(resource) file to a canonical format.[see section 2.1]. Design of the data structure for the translator depends on the format of the interface resource file. Two translators were created, one for Visual Basic 3.0 and the other for Visual C++ 4.0 using a lexical scanner generated by FLEX (Fast Lexical Analyzer Generator) which is a tool for generating programs that perform pattern matching on text. Using the lexical scanner, attribute value strings are detected and converted to the appropriate format as defined by the canonical format. All the dimensional coordinates are converted to pixels and other platform and application dependent values.

5.2 SHERLOCK Design

The family of consistency checking tools was implemented using different sets of classes in the C++ programming language. The data structure for these tools was designed in accordance with the canonical format input files. The SHERLOCK data structure was designed to be flexible, extensible and customizable to changes that may be made by expansion of the canonical format files. SHERLOCK has a sequential modular design and can be divided into the follwing subsystems.

- Widget Store Subsystem
- Dialog Box Subsystem
- String Processing Subsystem
- Spell Checker subsystem
- Button Processing Subsystem

6.0 Effects of Terminology Inconsistencies on User's Performance and Subjective Satisfaction

6.1 Introduction

An experiment was designed to test the hypothesis that terminology inconsistencies reduce performance speed and subjective satisfaction The experiment considered only one aspect of inconsistency, misleading synonyms.. We developed a GUI in Visual Basic for the students to access the resources of University’s Career Center. Three versions of the interface were created, the first one with no terminology inconsistency. The second version had a medium level of terminology inconsistency (on average, one inconsistency in terminology was introduced for each task and each task had an average of four screens). The third version had a high level of terminology inconsistency (50% more inconsistent terms than the medium inconsistent version). The resulting 2 X 3 experiment had two independent variables which were level of expertise and the type of the interface (no inconsistency, medium level of inconsistency and high level of inconsistency). The levels of expertise were no prior training and five minutes of training. For all the six phases of the experiment, users were given the same task list and their task completion time and subjective satisfaction was evaluated. For each treatment 10 subjects were selected, with a total of 60 subjects for the whole experiment. The results showed that the user's performance is significantly affected by terminology inconsistencies.

6.2 Interface Design

All the screens of the Career Center interfaces had a consistent visual design. Only terminology inconsistencies were introduced in the medium and high inconsistency version.

- In the medium inconsistency version, on average one terminological inconsistency was introduced for every task. Since the subjects were told to perform seven tasks, the interface had seven terminology inconsistencies. These inconsistencies included changing the heading of the dialog box from “Questions” to “Inquiries” or changing the widget labels from “Workshops” to “Seminars”. Also, menu items were changed from “Career Counseling” to “Career Advising” and “View” to “List”. Inconsistency in button labels were also introduced by changing “OK” and “Abort” to “Forward” and “Discard” in the case of a particular task.
- In the high inconsistency version, 50% more terminology inconsistency was introduced than the medium inconsistent version. On average, the high inconsistency version had one or two terminology inconsistencies per task with a total of 11 terminology inconsistencies. These inconsistencies included changing the“OK” button to “Done”, “Return” or “Forward” depending upon the task being performed and the “Abort” button to “Discard” ,”End” or “Suspend”, and “View Interviews” to “Interviews”. Inconsistencies in menu items were introduced by changing “Workshops” to “Seminars” plus inconsistencies in widget labels were introduced by changing “Major” to “Area”.

6.3 Hypothesis

- The task completion time for the interface with no terminology inconsistency will be significantly lower than the interfaces with medium and high levels of terminology inconsistency in the case of subjects with no prior training of the system.
- The difference in task completion time for the interface with no, medium and high levels of terminology inconsistencies would decrease when subjects are given five minutes of training with the system, prior to the execution of the tasks.
- The subjective satisfaction will be significantly higher for the interface with no terminology inconsistency as compared to those with medium and high levels of terminology inconsistency.

6.4 Subjects

A total of 60 subjects were used in the experiment out of which 30 were given 5 minutes training on the no inconsistency interface. The subjects chosen were students at the University of Maryland and were frequent computer users and were comfortable using the mouse and Windows 3.1 operating system.

6.5 Materials

The experiment was run on a 100MHz Pentium machine with a 17" color monitor having a 1024 X 768 pixel resolution with 256 colors. A set of instructions was provided in writing and was also explained to the subjects verbally before beginning the experiment. The subjects were asked to fill out a modified version of the Questionnaire (QUIS) (Chin et al, 1988) after completing the experiment.

6.6 Task List

The subjects performed the seven tasks which are:

- Register for Workshop II
- Set appointment with any one of the career counselor for Nov. 8 for 10:00 to 10:30 am.
- View Part-time job openings in Computer Science Major in Maryland state.
- Submit the following question to the Career Center: Do counselors at the career center perform mock interviews ?
- Request Graduate School information for Masters program in Business Management in any one of the listed universities.
- Register for an interview with any one of the listed companies on Nov. 20th using any one of the open slots.
- Cancel Registration for Workshop II

6.7 Procedure

Administration: Subjects were asked to read the instructions before starting the experiment and to sign the consent form. All the subjects were shown the different functionalities of the interface by browsing through the list of menu items. The subjects who were trained were shown the functionality of all the dialog boxes by opening each of the menu items. They were also allowed to use the interface for two minutes to experience the interface. The subjects in the training group were all trained using the same version (no terminology inconsistency) of the interface and then were tested using appropriate versions.

Grading: Task completion times for each task were measured using an electronic stop watch with an accuracy of 1/100th of a second. Any time taken by the subject to ask questions during the experiment was excluded from the total task completion time. Error rates were observed to facilitate the derivation of results, but no measurement were done. The subjective satisfaction questionnaire had 19 questions.

6.8 SHERLOCK Analysis

The terminology inconsistencies in the Career Center application can be detected by the SHERLOCK Terminology Baskets tool. These baskets with user predefined synonym terms are used to find misleading synonyms in the application. For example, shown below are the output of terminology baskets tool using the basket “Browse”, “Browsing”, “Query”, “Search” and “Searching” when applied on the no and high inconsistency versions respectively. Comparing the outputs for this basket for both versions we find that the no inconsistency version of the interface only uses the term “Search” and “Searching”, but in the high inconsistency version terms “Browse”, “Browsing” and “Query” were used instead of the term “Search” in various parts of the application.

```
--------------------------------------------------------------------------------
Browse          Browsing        Query           Search          Searching
--------------------------------------------------------------------------------
TERM                        DIALOG BOXES CONTAINING THE TERM
Browse
Browsing
Query
Search
                  coop.frm.cft      fulltime.frm.cft     intern.frm.cft
                  parttime.frm.cft
Searching
                  coop.frm.cft      fulltime.frm.cft     intern.frm.cft
                  parttime.frm.cft

--------------------------------------------------------------------------------
Browse          Browsing        Query           Search          Searching
--------------------------------------------------------------------------------
TERM                       DIALOG BOXES CONTAINING THE TERM
Browse            coop.frm.cft
Browsing
                  parttime.frm.cft
Query
                  parttime.frm.cft
Search
                  fulltime.frm.cft      intern.frm.cft
Searching
                  coop.frm.cft          fulltime.frm.cft      intern.frm.cft
```

Following are the other terminology baskets used in detecting terminology inconsistencies:

- `Abort Discard End Exit Suspend`
- `Major Field Area Program`
- `OK Done Apply Submit Send Forward`
- `Question,Questions,Inquiry,Inquiries`
- `Counselor,Counseling,Advisor,Advising`
- `Workshop, Workshops, Seminar, Seminars`

Level of Expertise	No Inconsistency	Medium Inconsistency	High Inconsistency
Without Training	239.0 sec. (61.0)	287.4 sec (42.6)	312.7 sec (88.25)
With Training	204.0 sec. (41.7)	217.4 sec (50.6)	270.7 sec (30.5)

Table 1. Average Task Completion Time and Standard Deviation (10 subjects per cell & 7 tasks per subject)

6.9 Results

The experimental results summarized in Table 1 shows that the no inconsistency version had a faster average task completion time than the medium and high inconsistency versions. Also, the average subjective satisfaction ratings for the no inconsistency version was higher than the medium and the high inconsistency versions of the interface. Overall, the performance improved when training was administered to the subjects, as the average task completion time was lower for all the three versions of the interface when training was provided.

The 2 X 3 ANOVA (Analysis of Variance) was used to determine whether the interface types (versions) and the level of expertise have statistically significant effects on the task completion time and subjective satisfaction measured across the three treatments(no, medium and high level of terminology inconsistency) and two training levels (with prior training and without prior training). There was a statistically significance difference for task completion time by expertise ($F(1,54) = 12.38$, $p < 0.05$)and interface type ($F(2,54) = 8.21$, $p < 0.05$) but no interaction effect.This implied that for the task completion time, the effect of the level of expertise is not dependent on the inconsistency level of the interface. Therefore, performance of both the experts and novices reduces with increase in terminology inconsistencies. In summary, terminology inconsistencies decrease user's performance regardless of the user`s level of expertise. For the subjective satisfaction, no statistically significant effects.

6.10 Discussion

In relation to the task completion time, the ANOVA identified that the terminology inconsistencies introduced in each version of the interface significantly slowed the user's performance. These effects can also be observed in the task completion time difference between the averages of each treatment. In the treatments with no training the average task completion time for the medium and high inconsistency treatments were 20% and 30% more than the no inconsistency treatment. Similarly in the block with training administered, the average task completion time for medium and high inconsistency treatments were 7% and 34% more than the no inconsistency treatment.

The level of expertise, according to ANOVA significantly effected the user's performance. On average, the training decreased the task completion time by 14%, 24% and 13% in no, medium and high inconsistency versions respectively. Although the subjective satisfaction ratings for the medium and the high inconsistency versions were less than the no inconsistency version, the ANOVA analysis found no statistically significant results. It is difficult to obtain statistically significant differences in preference scores for between groups design, because subjects do not see the other versions. A future within subjects study might elicit stronger preference differences.

6.11 Conclusion

The results of this experiment, along with the experiment done by Bajwa (1995) support the encouragement to "strive for consistency" and including consistency as one of the prime guidelines when designing user interfaces. Also, the terminology inconsistencies introduced in the design of the experimental interfaces were detected by the SHERLOCK Terminology Basket Tool, verifying the tool's capability.

7.0 Recommendations

7.1 Recommendations for the GUI Application Developers:

The following guidelines are recommended as a step towards creating consistent interfaces:

- Consistent Aspect Ratio especially for dialog boxes having similar visual design and functionality.
- Non-Widget Area (white space) should be atleast 20% of total area enclosed by the dialog box.
- Consistent margins within and across dialog boxes.
- Widgets within a dialog box should be both horizontally and vertically aligned.
- Design with too many widgets in a small area should be avoided.
- Consistent background colors, foreground colors and typefaces.
- Consistent location and size of frequently used widget.
- Consistent terminology across dialog boxes
- Terminology in menu items should be consistent with the title of the dialog boxes which they open and with the labels of the widgets contained in those dialog boxes.
- Button Labels should be consistent across the interface, for example synonyms like "Abort", "Cancel", "Close" and "Exit" should not be used for similar tasks.
- Terminology should be consistent within the sequence of dialog boxes which are connected through button clicks.
- In addition to these consistencies within the dialog boxes, the interface should be consistent in terminology with the current commercial applications running on that system, but in accordance with the user's task domain.

7.2 Recommendations for Designers Looking for GUI Evaluation Metrics

- To evaluate screen complexity, use metrics like Non-Widget Area and Widget Density of the dialog box summary table. Explore the use of other metrics for dialog box crowdedness like Local Density metric developed by Tullis (1988b) and Hot-Spot metric by Streveler & Wasserman (1987).
- Use metrics like Aspect Ratio, Margins, and Balance to evaluate size of dialog boxes and create a more coherent and consistent layout.
- Metrics evaluation using additional visual techniques of Regularity, Proportion, Neutrality, Transparency and Grouping defined by Vanderdonck and Gillio (1994) should be explored.
- Develop metrics to check consistency in typefaces and colors across dialog boxes in general and for every widget type in particular.
- Develop a better metric for detecting misaligned widgets similar to the dialog box summary table gridedness metric and previously developed Layout Complexity and Alignment metrics (Tullis, 1988; Streveler & Wasserman, 1987).
- Develop metric to check consistency in size and location of specific widget types across dialog boxes. These metrics may be similar to those used in SHERLOCK's Button Concordance and Button Layout Table tools.
- Expand the Terminology Basket tool to detect misleading synonyms for specific widget types and across all the widgets.

- Implement tools similar to Interface Concordance and Interface Speller to detect variant capitalization, spelling errors and abbreviations.

7.3 Recommendations for Designers of New GUI Tools

The applicability of SHERLOCK's canonical format approach to other GUI development tools beyond Visual Basic was explored. The following are recommendations for designers of new GUI tools after analyzing existing tools like Visual Basic, Visual C++, Galaxy, and Tk/Tcl.

- Visual design and textual properties of dialog boxes including widget labels, widget coordinates, widget sizes, typefaces, colors and more should be stored in an ASCII resource file. Many existing GUI development tools store this information as binary files.
- If the GUI development tool allows the user to dynamically change the widget size and position within the code, these changes should be updated to the corresponding resource files.
- The GUI development tool should not allow widget in the dialog box to extend beyond the area of the dialog box.
- If possible, the GUI development tool should create default left, right, top and bottom margins for dialog box beyond which widgets may not be extended.
- The GUI development tool should promote consistency by creating default dialog box templates so that developers are aware of the positioning of frequently used widget.
- A spell checking tool should be incorporated in the GUI development tool.

8.0 Future Directions

8.1 Extension to SHERLOCK

- Analyze more interfaces developed in Visual Basic to further modify the dialog box summary table metrics. Furthermore, remove those metrics that are not successful in detecting any inconsistencies.
- Analyze interfaces developed in Visual C++ to validate the canonical format approach on which the design of SHERLOCK is based.
- Subdivide the dialog box summary table into smaller tools. This subdivision would expand the analysis of each metric by reporting any exceptions or additional information along with metric values to facilitate the interpretation of results.
- Link the dialog box summary table to a spread sheet software like Microsoft Excel that can graph the inconsistencies for every metric.
- Add a generic tool to SHERLOCK which can detect visual design and textual inconsistencies in any type of widget, including combo boxes, drop down boxes, text boxes, similar to the inconsistencies detected by button layout table and button concordance tools in button widgets.
- Expand the terminology thesaurus of SHERLOCK in button sets of the button layout table and baskets of the terminology basket tool.
- A new tool needs to be added to SHERLOCK that can separate dialog boxes with similar layout features and then use the other SHERLOCK tools for consistency checking.

8.2 Extension beyond SHERLOCK

SHERLOCK analysis is static and provides limited feedback to the user. Steps need to be taken to design a tool that incorporates SHERLOCK tools, but is dynamic in nature, where users can interactively modify the inconsistencies highlighted by SHERLOCK. Also, the future tool should be able to indicate the severity of the inconsistency, so that the designers can prioritize the modifications suggested by those tools. Perhaps the greatest challenge is to provide a “goodness” measures for the metric values. We have laid down the foundation for building the next generation generic GUI consistency checking tool to evaluate visual design and textual properties, but more work needs to be done to build a complete structure on this foundation.

Acknowledgments

Funding for this research was provided by General Electric Company. We would also like to thank Ren Stimart of GE for his help.

References

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

Bajwa, S.(1995), “Effects of inconsistencies on users performance and subjective satisfaction”, *Unpublished, Department of Computer Science, University of Maryland.*

Barnard, P., Hammond, J. , Morton, J., Long, J. and Clark, I. (1981), “Consistency and compatibility in human-computer dialogue”, *International Journal of Man-Machine Studies 15*, 87-134.

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.

Chimera, R. and Shneiderman, B. (1993), User interface consistency: An evaluation of original and revised interfaces for a videodisk library, In Shneiderman, B. (Editor), *Sparks of Innovation in Human-Computer Interaction*, Ablex Publishers, Norwood, NJ, 259-273.

Chin, J.P., Diehl, V.A. and Norman, K. (1988), Development of an instrument measuring user satisfaction of the human-computer interface. *Proc. of the Conference on Human Factors in Computing Systems, CHI'90*, ACM, New York, 213-218.

Coll, R., and Wingertsman, A. (1990), The effect of screen complexity on user preference and performance, *International Journal of Human-Computer Interaction 2,3*, 255-265.

Comber, T. and Maltby, J. (1995), Evaluating usability of screen design with layout complexity, *Proc. of the CHISIG Annual Conference*, Melborne, Australia (in print).

Frederiksen, N., Grudin, J. and Laursen, B. (1995), Inseparability of design and use: An experimental study of design consistency, *Proceedings of Computers in Context'95, Aarhus*, Aarhus University, 83-89.

Grudin, J. (1989), The case against user interface consistency, *Communications of the ACM 32*,10, ACM Press, New York, 1164-1173.

Grudin, J., and Norman, D. (1991), Language evolution and human-computer interaction, *Proc. of the Thirteenth Annual Conference of the Cognitive Science Society*, Hillsdale, New Jersey, 611-616.

Harrison, M.D. and Thimbleby, H.W. (1985), Formalizing guidelines for the design of interactive systems. *In Proc. of BCS HCI Specialist Group Conference HCI85*, 161-171.

Jeffries, R., Miller, J., Wharton, C. and Uyeda, K. (1991), User interface evaluation in the real world: A comparison of four techniques, *Proc. of CHI 1991*, ACM, New York, 119-127.

Kellogg, W. (1987), Conceptual consistency in the user interface: Effects on user performance, *In Proc. of Interact '87 Conference on Human-Computer Interaction*, Stuttgart.

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.

Karat, C-M., (1992), Cost-justifying human factors support in development projects, *Human Factors Society Bulletin 35*, 8.

Long, J., Hammond, N., Barnard, P., Morton, J. and Clark, I. (1983), Introducing the interactive computer at work: The user's view, *Behavior and Information Technology, 2*, 39-106.

Lynch, P. (1994), Visual design for the user interface, pt. 1 design fundamentals, *Journal of Biocommunications 21, 1*, 22-30.

MacIntyre, F., Estep, K.W. and Sieburth, J.M. (1990), Cost of user-friendly programming, *Journal of Fourth Application and Research 6, 2*, 103-115.

Mahajan, R. and Shneiderman, B. (1995), A family of user interface consistency checking tools: Design analysis of SHERLOCK, *Proc. of NASA Twentieth Annual Software Engineering Workshop*, *SEL-95-004*, NASA Pub., 169-188.

Mullet, K.(1995), Organizing information spatially, *Interactions*, July 95, 15-20.

Nielsen, H.(1989), Coordinating User Interfaces for Consistency Checking, Ed. Nielsen, J. Academic Press Inc., London.

Norman, D.A. and Draper, S.W. (1986), *User Centered System Design: New Perspectives in Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, N.J.

Polson, P., Muncher, E. and Engelbeck, G. (1986), A test of a common elements theory of transfer, *Proc. of CHI'86*, ACM, New York, 78-83.

Reisner, P. (1990), What is consistency?, *Proc. of the IFIP Third International Conference on Human-Computer Interaction, Interact '90*, Elsevier Science Publishers, B.V., North-Holland, 175-181.

Rosenberg, D. (1989), Cost benefit analysis for corporate user interface standards: What price to pay for consistent look and feel?, *Coordinating User Interfaces for Consistency Checking*, Ed. Nielsen, J. Academic Press Inc., London, 21-34.

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), AIDE: A step towards metric-based interface development tools, *Proc. of UIST '95*, ACM, New York, 101-110.

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

Shneiderman, B., Chimera, R., Jog, N., Stimart,R. and White, D. (1995), Evaluating spatial and textual style of displays, *Proc. of Getting the Best from State- of the-Art Display Systems '95*, London.

Smith, D.C., Irby, C., Kimball, R., Verplank, B.and Harslem, E. (1982), Designing the star user interface, *Byte 7, 4*, 242-282.

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. (1983), The formatting of alphanumeric displays: A review and analysis, *Human Factors 25*, 657-682.

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 (Ed.), *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.

Wiecha, C., Bennett, W., Boies, S. and Gould, J. (1989), Generating Highly Interactive User Interface, *Proc. of CHI'89,* ACM, New York, 277-282.

Wolf, R. (1989), Consistency as process, *Coordinating User Interfaces for Consistency Checking*, Ed. Nielsen, J. Academic Press Inc., London, 89-92.