Providing High-level Control and Expert Assistance in the User Interface Presentation Design

by

Won Chul Kim and James D. Foley

GIT-GVU-93-09
March 1993

Graphics, Visualization & Usability Center

Georgia Institute of Technology
Atlanta GA 30332-0280

Providing High-level Control and Expert Assistance in the User Interface Presentation Design

Won Chul Kim

Department of Electrical Engineering and Computer Science
The George Washington University
Washington, DC 20052
kim@seas.gwu.edu

James D. Foley

Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
foley@cc.gatech.edu

ABSTRACT

Current user interface builders provide only low-level assistance, because they have knowledge of neither the application, nor the principles by which interface elements are combined effectively. We have developed a framework that unites the knowledge components essential for effective user interface presentation design. The framework consists of an application model (both a data model and a control model), a design process model that supports top-down iterative development, and graphic design knowledge that is used both to place dialog box elements such that their application dependent logical relationships are visually reinforced and to control design symmetry and balance. To demonstrate the framework's viability, we have constructed a tool based on encapsulated design knowledge that establishes high-level style preferences and provides expert assistance for the dialog box presentation design and menu structuring.

KEYWORDS: Automatic layout, knowledge-based tool, UI design process

1. THE DESIGN PROCESS

Every interactive computer application must have a user interface (UI) to mediate between users and application functions. UI builders [3, 5, 15, 22, 23] help designers by providing components, such as menus, dialog boxes, radio buttons and scroll bars, to put together user interfaces quickly and easily. However, they do not provide intelligent assistance for generating a presentation design. They facilitate the presentation design activities at too low a level; a designer still must follow written guidelines [1, 18, 21] and graphics principles [4, 9, 10, 16] to create a good design. Interpreting the guidelines unambiguously and applying generic principles to a particular design problem is itself a major challenge. Thus, such tools may simply allow a designer to assemble a poor UI quickly.

Our objective is to integrate expert assistance into the UI design process. The essential ingredient of expert assistance is knowledge. We have thus extended the knowledge base model of User Interface Design Environment (UIDE) [7, 8, 20] to encapsulate elements, relationships, and principles of design knowledge related to the organization and presentation of menus and dialog boxes. This extended knowledge base model [12, 13, 14] consists of (1) a *conceptual design representation* which is a high-level description of an application; (2) *style knowledge*, which provides default style attributes of interface widgets; (3) *organization structures*, which define the logical groupings of application commands and parameters; and (4) a *presentation design representation*, which represents the interface object type (what interface object to use), the style selection (how the object looks), and the layout decisions (where objects are placed). The integrated knowledge base model provides a framework in which a set of reusable design rules automates various steps of the UI design process.

Based on this knowledge base framework, we have developed a UI presentation design tool called DON [12, 13]. The tool has two stages. An *organization manager* uses a top-down design methodology to assist designers in organizing the information, and selecting appropriate interface objects and their associated attributes. A *presentation manager* lays out selected interface objects in a dialog box or selected menu items, in a meaningful, logical, and consistent manner.

A difficult challenge is to accommodate user preferences and design goals in the design process. Because it is unrealistic to automate completely the UI design process, we have modeled the process and have broken it down into stages corresponding to the logical steps a designer takes. With DON, a designer specifies high-level preferences, such as acceptable range of dialog sizes or consistent margins and spacing, and each design stage automatically accommodates the preferences. The designer can view the intermediate results of each design stage and can modify them interactively, or can repeat a stage with modified preferences.

Support of Organization

The design process starts with the conceptual design

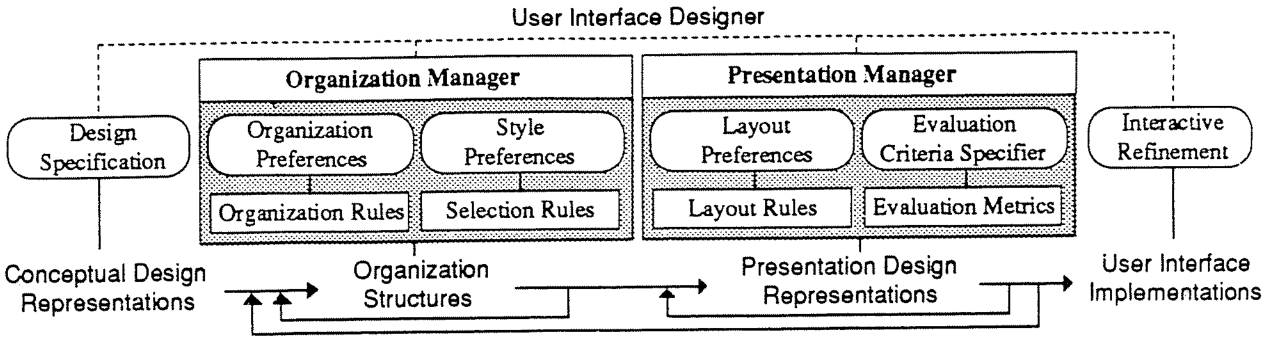


Figure 1 An overview of the design tool environment.

specification of an application. This representation of application entities and semantic relationships is used to aid the designer in determining the classifications and logical groupings of the application information to be presented to end users.

The organization manager (OM), shown in Figure 1, implements the first step of the design process and embodies two main sets of rules: (1) *organization rules* that analyze the high-level conceptual design representation and design *organization preferences* to allocate commands to menus, and attributes and command parameters to dialog boxes; (2) *selection rules* that select the appropriate interface object types and their attributes. Various existing systems [2, 6, 26] demonstrate the feasibility of rule-based, automatic mapping of application descriptions to interface object representations. The logical relationships of application commands and parameters are represented explicitly in the organization structure, which provides the basis for feedback at the organization level and for interactive modification of automated decisions. The OM allows the designer to specify high-level preferences and to modify interactively interface object type mappings stored in the organization structure. The designer can iterate through the organization process until the content organization and interaction object mappings are complete and satisfactory.

Support of Presentation Design

The OM paves the way for the presentation manager (PM), also shown in Figure 1, to complete the presentation design. Four aspects of the PM are important: (1) it automates the actual layout of menus and dialog boxes; (2) graphic design principles are embedded in the *layout rules*, which automate the layout process; (3) designer-specified *layout preferences* provide effective means to control the presentation design details; and (4) the *evaluation metrics* provide the comparative criteria that the designer can use as a guideline to evaluate and constrain design alternatives. The PM supports iterative menu structuring. For dialog box design, a combination of iterative and generate-and-evaluate methods is supported.

In the following sections, we discuss the strategies for providing high-level design control and expert assistance for dialog box presentation design.

2. AUTOMATIC DIALOG BOX LAYOUT

Success in design of forms, including dialog boxes, involves analyzing the conceptual content, designing the graphic appearance, and evaluating the form. The major goals of dialog box layout is to reinforce visually the logical relationships among items, to use space efficiently, and to generate aesthetically pleasing designs.

The layout algorithm assumes a *dialog box tree*. This tree is generated automatically by the organization rules. A dialog box tree defines the logical organization structure of the widgets that have been assigned to each dialog box. *Leaf* nodes of a dialog box tree represent the contents and labels of selected interaction objects. Initially, the bound (extent) of all leaf nodes of a dialog box is obtained. The logical *group* nodes' bounds are unknown until the layout process is complete.

The layout algorithm works with the tree of bounds, as shown in Figure 2. The strategy is to organize the bounds (extents of selected interaction objects) recursively from the leaf nodes (shown in white) up to the *root* of the dialog box tree. The bottom-up layout strategy systematically reduces the complex layout problem into many smaller layout problems for each unknown group node in similar fashion to hierarchical composition of graph [17] and tree layout algorithms [25]. Figure 2 illustrates a possible solution based on the bottom-up layout strategy, where unknown subgroup node bounds and

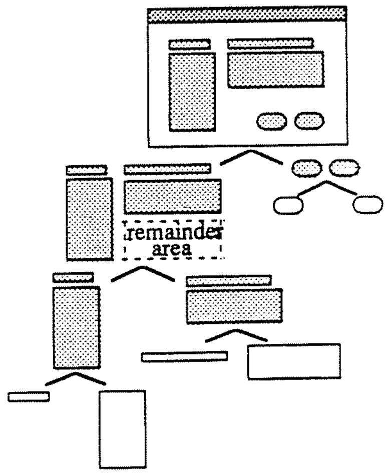


Figure 2 Illustration of the layout process, working with a tree structure of bounds.

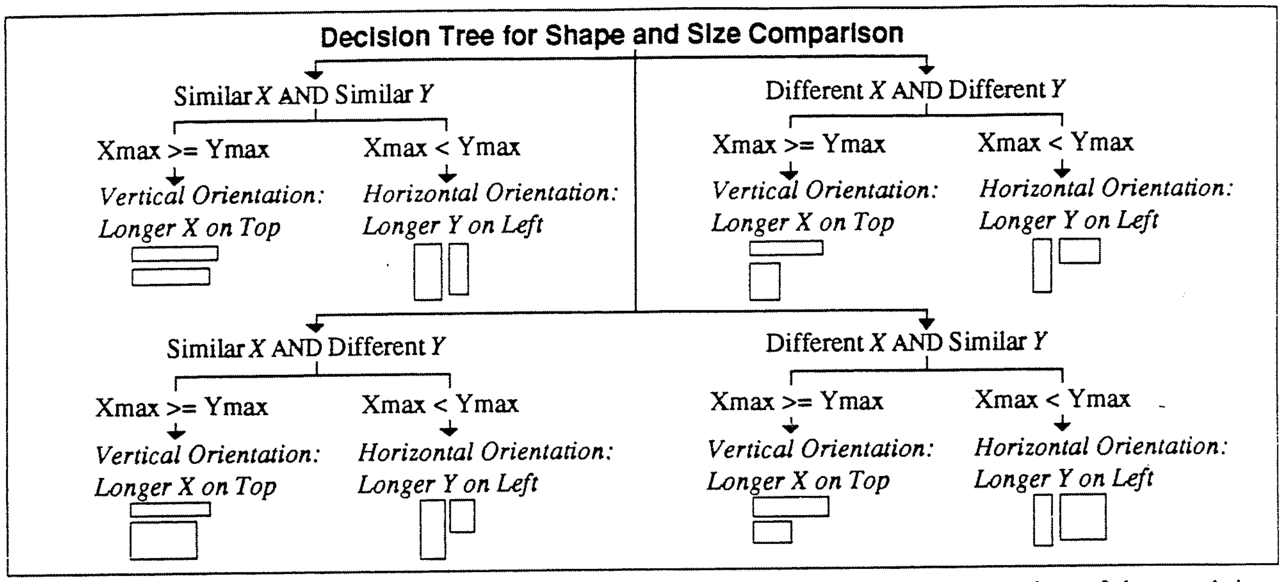

Decision Tree for Shape and Size Comparison
Similar X AND Similar Y
Xmax >= Ymax
Vertical Orientation: Longer X on Top
Xmax < Ymax
Horizontal Orientation: Longer Y on Left
Different X AND Different Y
Xmax >= Ymax
Vertical Orientation: Longer X on Top
Xmax < Ymax
Horizontal Orientation: Longer Y on Left
Similar X AND Different Y
Xmax >= Ymax
Vertical Orientation: Longer X on Top
Xmax < Ymax
Horizontal Orientation: Longer Y on Left
Different X AND Similar Y
Xmax >= Ymax
Vertical Orientation: Longer X on Top
Xmax < Ymax
Horizontal Orientation: Longer Y on Left

Figure 3 The *layout rule* decision tree for analyzing two rectangular items based on the comparison of shape and size.

locations are determined recursively until the shape and size of the root node are determined.

All the known bounds of a subgroup node are put into a *pool of bounds* to be laid out. Two bounds are selected, their arrangement is determined by a shape and size analysis (discussed in the following section), and the resulting group node consisting of two laid out bounds is thrown back into the pool as a single bound, with a remainder area that is maintained for possible use. This process is repeated for each unknown group node until all the bounds in the pool are exhausted. An important advantage of laying out two items at a time is that it ensures consistent alignments and gaps between items. Since the bottom-up layout is based on a logical tree structure, logically related items are arranged in close proximity, which partly ensures visual reinforcement of the logical structure.

Analysis of Shape and Size

Shape and size analysis is the central component of the layout algorithm that determines the alignment, orientation, and placement of selected pairs of bounds. The shape and size analysis is used to analyze the placement of a label with respect to its content. It is also used to categorize shapes of bounds and to formulate visual groupings based on similar shapes.

The decision tree for shape and size comparison for the two rectangular bounds is shown in Figure 3. The top level of the decision tree compares the similarity of the widths and heights of two bounds. The lower level of the decision tree compares the maximum widths and heights as the criteria to determine the orientation, alignment, and placement of the two selected bounds.

A designer can modify the similarity size ratio that is used by criteria for the shape and size comparison to determine similarity of items' lengths and widths. The default is set at 0.618, a value is derived from the golden proportion [4, 10, 16]. This option gives the designer additional fine control of the layout algorithm.

High-Level Analysis

To increase the efficiency of the algorithm and to handle special cases appropriately, DON incorporates a high-level analysis that examines many bounds at once, in contrast to the basic pair-wise comparison. The layout algorithm uses the high-level analysis to identify overflow situations, simple cases where the number of items is small, visual groupings based on similar shapes and sizes, and to assess appropriateness of a column-based design. DON handles these cases before applying the full layout algorithm.

When the area required for all the widgets exceeds the specified dialog box size, *overflow solutions* [12] are employed. For example, (1) the sizes of certain interaction objects can be reduced to the acceptable minimum, (2) the interaction object type can be changed to one that requires less space, and (3) margins and spacing can be tightened.

When the area required for all the widgets is less than the minimum dialog box size, DON suggests an increase in that minimum size, which results in a design with extra white space. The range of heights and widths for dialog boxes, as well as of limits on the size ratios, can be specified as style preferences.

With DON, the designer can explicitly specify and explore various alternative column-based styles, as shown in Figure 4. This feature allows the designer at all times to constrain layout structures to column-based arrangements of labels and dialog box contents, as suggested in the OPEN LOOK Application Style Guidelines [21]. The bottom two designs in Figure 4 are of the same dialog box representation. When a particular design is chosen—the lower-right design in this case—its

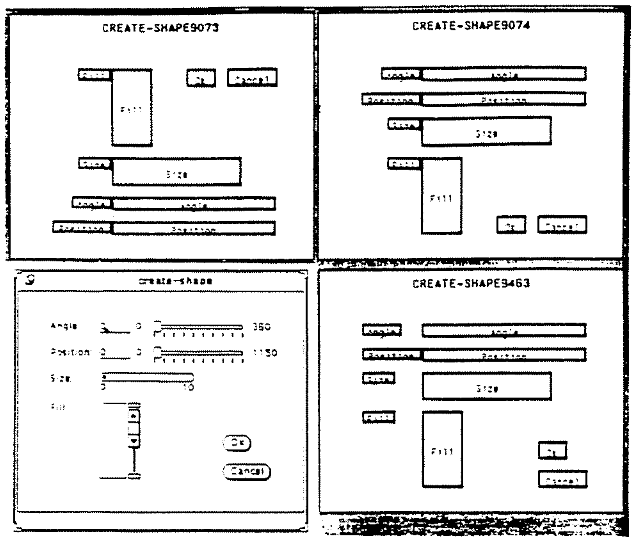


Figure 4 Alternative column-based designs for a dialog box with four items.

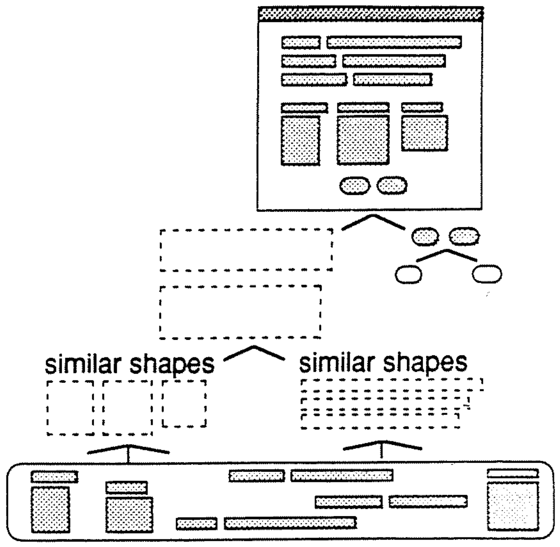


Figure 5 Groupings of bounds based on shape and size analysis.

design representation is mapped into the OPEN LOOK GUI components shown on the lower left.

DON also may recommend a column-based layout based on analysis (using rules such as the following) of the number of horizontally and vertically shaped contents, even if a designer has not explicitly requested such consideration:

```
If the number of horizontal shapes are greater than
        the sum of none-horizontal shapes,
then suggest column-based layout.
```

At each group node of a dialog box tree, the bounds are analyzed for possible grouping based on similar shapes and sizes. Figure 5 illustrates how the visual groupings are formed based on similar shapes. The row or column arrangements for a similar shape group are determined based on the shapes. When the majority of similar bounds is horizontally shaped, then a vertical column arrangement is recommended. When the majority is vertically shaped, then a horizontal row arrangement is recommended. In the interactive design mode, these automated decisions and recommendations can be overridden.

DON allows the designer to influence the overall organizational constraints on items by specifying:

- That the organization be column-based
- That a label location be used consistently (usually accompanies column-based design)
- That a specified visual balance tendency be used (top-heavy, bottom-heavy, left-heavy, or right-heavy)

These organizational preferences provide higher level constraints on the layout of all the items within a dialog box.

Iterative Design Method

An iterative design method, where alternative designs can be prototyped, implemented, refined rapidly and easily is advocated for developing good user interfaces. The iterative design cycle comprises these steps: (1) a designer specifies the layout constraints, (2) the system generates automatically a single design based on layout constraints, and (3) the system displays an outline of the layout structure and evaluation results. This cycle is short because only the outline (bordering bounds) of the layout structure is generated, and evaluation is based on this outline.

The high-level style customization capability combined with an iterative design strategy provides an easy to use tool that allows a designer to explore alternative designs quickly, without worrying about fine details. The functional user UI code is generated only when a designer is satisfied with the layout structure.

3. CUSTOMIZATION OF USER INTERFACE STYLE

DON provides an explicit method for specifying the layout constraints for each dialog box design instance. The layout rules achieve automatically the detail layout constraints specified. Consistency is realized when same layout constraints are applied for all dialog boxes; the framework makes it possible to set certain specifications global to all the dialog box instances. If the designer needs to specify exceptions, specific constraints for each dialog box instance can be overridden.

It is important that a design tool allow the designer to specify the fine detail layout constraints. The designer must have explicit control sufficient to generate a layout that fits the design goals particular to the current application. These layout constraints become input to the layout rules that automatically generate the dialog box designs.

Control of Margins and Spacing

Specification of consistent margins and spacing is tedious and difficult, even with an interactive layout tool. A simpler approach is to allow the designer to provide a high level specification of margins and spacing that can be applied consistently throughout the design.

Effective use of white space is the key to achieving attractive layouts. Various kinds of white space—such as inner, outer, top, foot margins, gaps separating items, and offsets between contents and labels—are controlled by the layout algorithm. Figure 6 shows the various spacings that a designer can specify.

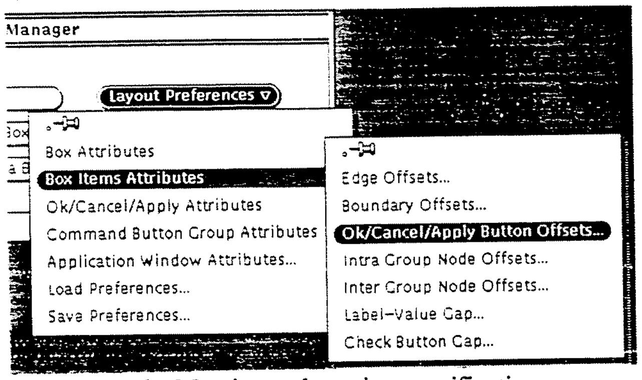

Manager
Layout Preferences
Box Attributes
Box Items Attributes
Ok/Cancel/Apply Attributes
Command Button Group Attributes
Application Window Attributes...
Load Preferences...
Save Preferences...
Edge Offsets...
Boundary Offsets...
Ok/Cancel/Apply Button Offsets...
Intra Group Node Offsets...
Inter Group Node Offsets...
Label-Value Gap...
Check Button Gap...

Figure 6 Margins and spacing specifications.

Control of Location and Orientation of OK and Cancel Buttons

Consistency in screen design is an important design factor that affects the usability of an interface, and thus of the application. A designer may want to force a consistent location of the OK and Cancel buttons in all dialog boxes. The Confirm (OK and Cancel) button group is considered separately based on either *specified* or *analyzed* location, alignment, and orientation.

Alignment:
(left, center, right, equal-size, none)
Orientation:
(above-below, left-right, none)
Placement:
(top-left, center-left, bottom-left,
top-right, center-right, bottom-right,
top-center, bottom-center,
distribute-top, distribute-bottom,
distribute-left, distribute-right, none)

Figure 7 Possible alignment, orientation, and location specifications for the OK and Cancel button group.

Possible options that a designer can specify explicitly to generate consistent Confirm button group location are shown in Figure 7. If preferences are not specified, then the layout rules analyze the overall content size, shape, and remaining space available to determine the location and orientation. The Confirm button group is considered after all content item bounds are laid out. Two general cases are considered: (1) the Confirm button group as a whole in any orientation does not fit in the remainder area (examples shown in Figure 8); (2) the remainder area is large enough to hold the Confirm button group (examples shown in Figure 9).

Figure 8 shows a set of few alternative Confirm button group

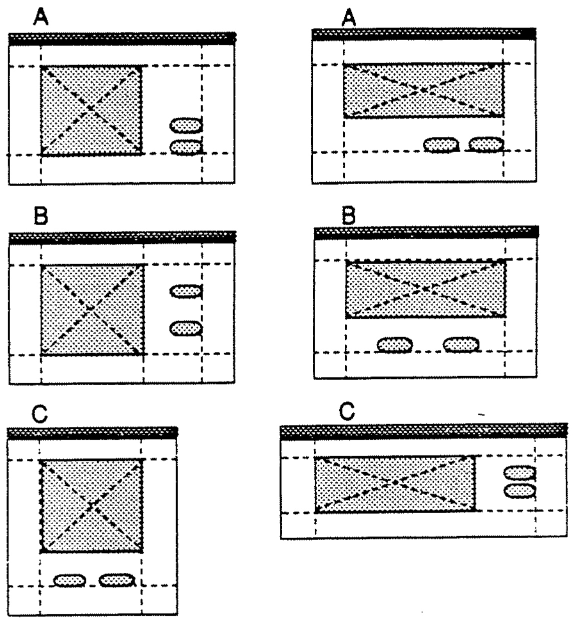


Figure 8 Determination of location and orientation for the OK and Cancel button group.

locations and orientations based on (1) vertically shaped content bound (left column) and (2) horizontally shaped content bound (right column). There are many other possible designs, not shown, with different combinations of location and orientation. Some combinations are rejected by the system because they result in awkward layout, as judged by the built-in evaluation criteria discussed in Section 4. The system defaults to a design A for both shapes considered in the example. The Confirm button group is the last logical item with which user interacts. Thus, the bottom-right location is an appropriate default location [4].

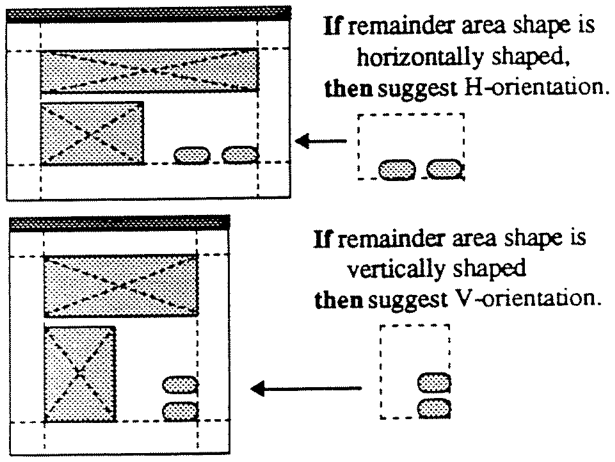

If remainder area shape is horizontally shaped, **then** suggest H-orientation.
If remainder area shape is vertically shaped **then** suggest V-orientation.

Figure 9 Determination of orientation for the OK and Cancel button group based on the shape of the remainder area.

The two examples in Figure 9 show the system default based on the shape of the remainder area. The horizontal orientation is preferred for the horizontally shaped remainder area; the vertical orientation is preferred for the vertically shaped remainder area.

Encapsulation of Style Knowledge

DON assists a designer in establishing a default style for interface object classes; this default persists, and is reusable for different design instances. This idea is similar to that used in Interaction Transaction Systems [26], which incorporates reusable style knowledge by allowing style experts to write style rules that represent both graphic arts and human factors decisions. The style knowledge consists of hierarchically structured classes of interface objects that take advantage of inheritance of attribute values. The advantages of modeling the style knowledge in this fashion are the following abilities:

- To provide default and ranges of acceptable attribute values to all interaction objects
- To allow a designer to modify defaults to generate customized styles
- To allow a designer to specialize style objects for special purposes
- To allow a designer to package a set of style attribute values

The style knowledge facilitates a designer's control of the presentation style for individual interface objects. For example, during the layout process, the layout rules use the minimum acceptable sizes to reduce the size of the interaction objects when there are too many items to fit into the specified dialog box size.

The designer can bundle together these high-level style preferences and the layout constraints to develop a customized style guide that satisfies the written style guides [1, 18, 21] yet is suited to a particular application using a particular toolkit.

4. THE GENERATE AND EVALUATE STRATEGY

The generate-and-evaluate method of design facilitates exploration by allowing a designer to view many automatically generated dialog box layouts. The designer can not only view multiple layout structures simultaneously, but also obtain various evaluation metrics on specific criteria. The designer can use the metrics to compare many alternative designs. DON further enhances exploration of the design space by combining the advantages of both the iterative design strategy and the generate-and-evaluate strategy. After selecting the generated layout structure from multiple design alternatives, the designer can use an iterative strategy to fine tune the detailed visual attributes.

Evaluation Criteria

A design tool needs quantitative measures to analyze the spatial properties of the dialog box layout. Streveler and Wasserman developed quantitative metrics [19] for analyzing digital screen layouts, and Tullis developed metrics [24] for evaluating the visual layout or design of alphanumeric displays. DON incorporates several evaluation criteria, such as symmetry, balance, dialog box size ratio, dialog box size range, and negative space percentage calculations to help a designer apply graphic design principles in dialog box layout. The algorithms for the evaluation metrics have been developed based on graphic design principles [4, 10, 11, 16].

A design achieves *balance* by placing elements of similar weight, but not necessarily precisely the same size, in relationship to one another such that there is weight at the bottom of layout as well as the top, and to the left and right to balance the whole [4]. A significant concept that contributes to balance is that an object located a greater distance away from the center will have a higher visual weight than will a more central item. A similar idea holds for two items placed at opposite ends of a fulcrum: a smaller item placed farther away from the middle is balanced by a bigger item placed closer to the middle.

To simplify the evaluation score calculation, we convert the actual extent of items laid out in a dialog box to 1s, and we convert the white space to 0s. This step is common to several evaluation calculations. The bitmap resolution can be increased for higher accuracy at the cost of computation time, by setting the parameter to the bitmap conversion function.

The formula for left-right balance calculation is as follows (top-bottom balance is computed similarly):

$$\text{Total weight for each side} = \sum 1\text{s} * f(\text{distance away from the center})$$

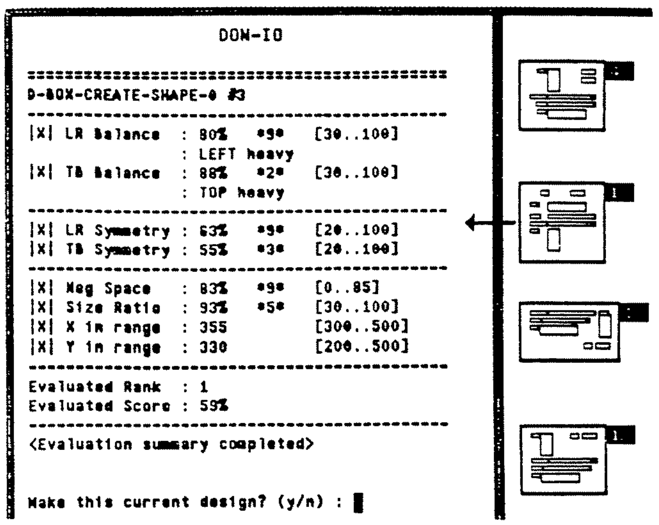
$$\text{Left-right balance} = \frac{\text{total weight of less heavy side}}{\text{total weight of more heavy side}}$$

The method of calculating the *symmetry* of layout is straightforward. We can think of symmetry as folding a layout in half and checking all the matched areas covered by the elements on each side.

Proper use of white space is critical to an attractive layout. Nevertheless, when screen space is constrained, a designer may need to minimize *negative space* (unused white space). We compute the negative space by calculating the percentage of the white space against the total space available.

The evaluation metrics for a selected dialog box design are shown in Figure 10. From right to left, (1) a designer-specified acceptable range in each evaluation criteria is shown by brackets; (2) a designer-specified weight, which can range from 1 to 9, for each evaluation criteria is shown by the numbers surrounded by asterisks; (3) computed evaluation scores for each criterion are shown in percentages (the higher the number, the greater the balance and symmetry); each score is marked by "X" that satisfies the specified range. The left-right balance score for the example in Figure 10 is 80 percent (the left side is heavier) and the top-bottom balance score is 88 percent (the top is heavier).

DON uses evaluation metrics to constrain the design space and the layout style, and to compare and evaluate alternative designs. The high-level constraints—such as margins, spacing, and the preferred location and orientation of OK and Cancel buttons—are reflected in the generation scheme to enhance designer control in constraining the resulting design.



```
                    DON-IO

============================================
D-BOX-CREATE-SHAPE-0 #3
--------------------------------------------
|X| LR Balance   : 80%   *9*   [30..100]
                 : LEFT heavy
|X| TB Balance   : 88%   *2*   [30..100]
                 : TOP heavy
--------------------------------------------
|X| LR Symmetry : 63%   *9*   [20..100]
|X| TB Symmetry : 55%   *3*   [20..100]
--------------------------------------------
|X| Neg Space    : 83%   *9*   [0..85]
|X| Size Ratio   : 93%   *5*   [30..100]
|X| X in range   : 355         [300..500]
|X| Y in range   : 330         [200..500]
--------------------------------------------
Evaluated Rank  : 1
Evaluated Score : 59%
--------------------------------------------
<Evaluation summary completed>


Make this current design? (y/n) : █
```

Figure 10 Evaluation results of a selected dialog box.

Design Browser

The design browser displays alternative designs that are generated automatically, and allows the designer to interact directly with these visual representations. The design browser maintains the design variants, internally, by instantiating instances of a dialog box design schema that stores left-right, top-bottom relations of bounds, exact locations of bounds, and evaluation scores of a design.

Four important aspects of the design browser are shown in Figure 11: (1) multiple layout structures can be viewed and compared, (2) designs that satisfy all the specified evaluation criteria are indicated by a bold boundary, (3) a numerical ranking based on the total evaluation score is displayed next to each visual representation of a dialog box, (4) the designer can select alternative choices to refine a selected design iteratively. DON reports detailed evaluation scores corresponding to a dialog box design when the designer clicks directly on the visual representation.

DON ranks multiple designs based on the total evaluation score, which DON calculated using specified weights for each evaluation criterion. A designer can request a certain number of designs that satisfy the specified criteria. A designer can also modify the weights, and request recalculation of the total score and reranking of multiple designs. This functionality provides an additional method to compare and evaluate multiple designs, complementing the visual feedback of the display of multiple designs.

5. IMPLEMENTATION

DON is implemented with the Automated Reasoning Tool (Inference Corp.) [11] on workstations (Sun Microsystems). The final designs from DON can be read into the Devguide [22], an interactive UI design tool, for further refinement. The presentation design representation is generic in that additional implementation models [14] can be developed that will translate it into different toolkits.

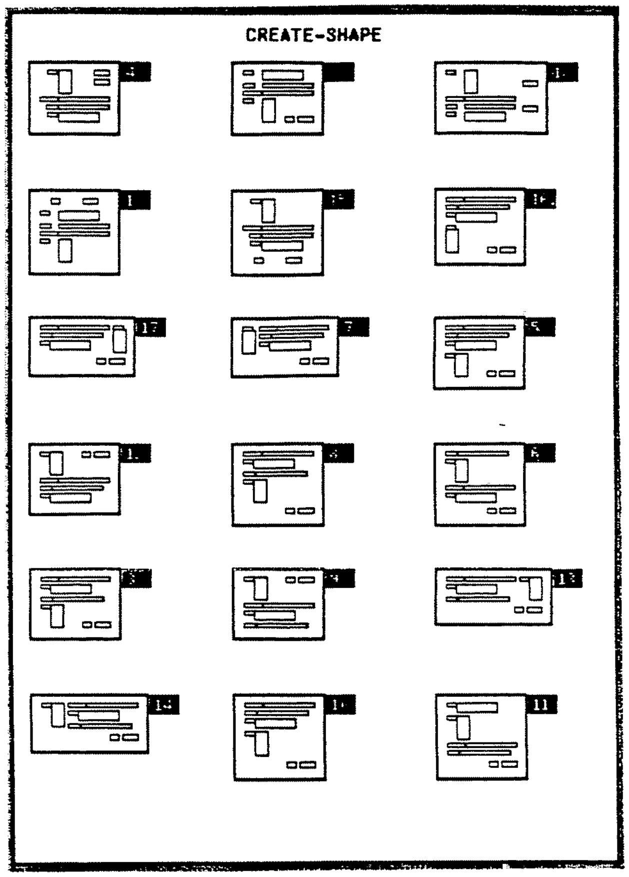


Figure 11 Automatically generated multiple designs, shown by the design browser.

6. SUMMARY

DON represents a step forward in UI development tools. It shows how we can provide high-level control and expert assistance in the UI presentation design process. These capabilities assist a designer in generating UI designs and provide easy ways for a designer to specify constraints. DON applies graphic design principles in the dialog box layout process, and uses evaluation metrics to improve visual clarity, consistency, and aesthetic appearance.

Work can be done to refine the layout rules to use more semantic information. We are exploring combining shape and size analysis with more extensive analysis of parameter relationships, such as parameter sequencing and dependencies. Visual attribute analysis can be combined with interaction object type analysis to place similar types next to each other when there are no substantial differences in the shape and size.

To enhance DON's usability, we can develop more visual tools based on the framework, such as a graphical tool that allows a designer to view the layout in progress, and to interact directly with the dialog box tree nodes to specify constraints.

The knowledge-based framework developed can be used as the basis for integrating top-down and bottom-up approaches to UI design. By sharing representation of various stages of UI design, rules can facilitate mapping from either direction. To build a true link from DON, which is a top-down iterative design tool, to the interactive UI builders, we must capture the reasons for modifications made by the designer and incorporate them back into a set of the layout rules for use during future designs.

ACKNOWLEDGEMENTS

The Graphics and User Interface Research Group at George Washington University provided intellectual environment supporting this work. Thanks to Shelly Heller, Jon Mckeeby, Lucy Moran, and Lyn Dupre whose comments helped to improve the clarity of the paper. Special thanks to Samuel Molina for his contribution in formulating evaluation metrics. Financial support was provided by NSF Grant IRI-88-13179 and DMC-84-205-29, Siemens Corporation, and Sun Microsystems.

REFERENCES

1 Apple Computer, Inc. *Human Interface Guidelines: The Apple Desktop Interface*, Addison-Wesley, Reading, MA 1990.

2 Arens, Y., L. Miller, S. Shapiro, and N. Sondheimer. Automatic Construction of User-Interface Displays, *AAAI-8 Proceedings*, St. Paul, MN, August 1988.

3 Cardelli, L. Building User Interfaces by Direct Manipulation, in *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, ACM, New York, 1988, pp. 152-166.

4 Conover, T.E. *Graphic Communications Today*, West Publishing Company, St. Paul, MN 1985.

5 Cossey, G. *Prototyper*, SmetherBarnes, Portland, OR, 1989.

6 de Baar, D., J. Foley, and K. Mullet. Coupling Application Design and User Interface Design, in *Proceedings of CHI'92-SIGCHI Computer-Human Interaction Conference*, ACM, New York, 1992, pp. 259-266.

7 Foley, J., W. Kim, K. Murray, and S. Kovacevic. UIDE-An Intelligent User Interface Design Environment, in Sullivan, J and S. Tyler (eds.), *Intelligent User Interfaces*, Addison-Wesley, Reading, MA, 1991, pp. 339-384.

8 Foley, J., W. Kim, S. Kovacevic, and K. Murray. Defining Interfaces at a High Level of Abstraction, *IEEE Software*, 6(1), January, 1989, pp. 25-32.

9 Galitz, W. *Handbook of Screen Format Design*, Q.E.D. Information Sciences, Inc., Wellesley, MA, 1981.

10 Hurlburt, Allen. *Layout: the design of the printed page*, Watson-Guptill Publications, New York, 1977.

11 Inference Corp. *ART Reference Manual*, Inference Corporation, Los Angeles, CA, 1987.

12 Kim, W. Knowledge-Based Framework for an Automated User Interface Presentation Design Tool, *Ph.D. Thesis*, Department of Electrical Engineering and Computer Science, George Washington University, Washington, D.C., 1993.

13 Kim, W., and J. Foley. DON: User Interface Presentation Design Assistant, in *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*; ACM, New York, 1990, pp. 10-20.

14 Kim, W., and J. Rhyne. Knowledge Base Model for Automated User Interface Design, *Report GWU-IIST-89-29*, Department of Electrical Engineering and Computer Science, George Washington University, Washington, D.C., 1989.

15 Linton, M., J. Vlissides, and P. Calder. Composing User Interfaces with InterViews, *IEEE Computer*, Feb.. 1989.

16 Marcus, A. *Graphic Designfor Electronic Documents and User Interfaces*, Addison-Wesley, Reading, MA, 1992.

17 Messinger, E., L. Rowe, and R. Henry. A Divide-and-Conquer Algorithm for the Automatic Layout of Large Directed Graphs, *IEEE Transactions of Systems, Man, and Cybernetics*, Vol. SMC-21, No. 1, 1991, pp. 1-12.

18 Open Software Foundation. *OSF/Motif Style Guide*, Revision 1.0, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

19 Streveler, D., and A. Wasserman. Quantitative Measures of the Spatial Properties of Screen Designs, in *Proceedings INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction*, Vol. 1, Elsivier Science Publishers, Amsterdam, 1987, pp. 125-133.

20 Sukaviriya, P., J. Foley, T. Griffith. A Second Generation User Interface Design Environment: The Model and The Runtime Architecture, *INTERCHI'93, Conference on Human Factors in Computing Systems*, ACM, New York, 1993.

21 Sun Microsystems, Inc. and AT&T. *OPEN LOOK GUI Application Style Guidelines*, Addison-Wesley, Reading, MA, 1990.

22 Sun Microsystems, Inc. *Open Windows Developer's Guide 1.1, Reference Manual*, Part No. 800-5380-10, Revision A, Mountain View, CA, June 1990.

23 Szczur, M., and P. Miller. Transportable Applications Environment (TAE) PLUS, in *Proceedings of OOPSLA'88, Object-Oriented Programming Systems, Languages and Applications Conference*, ACM, New York, 1988.

24 Tullis, T. A Computer-Based Tool for Evaluating Alphanumeric Displays, in *Proceedings INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction*, Vol. 2, Elsivier Science Publishers, Amsterdam, 1987, pp. 123-127.

25 Walker II, John. A Node-positioning Algorithms for General Trees, *Software-Practice and Experience*, Vol. 20(7), 1990, pp. 685-705.

26 Wiecha, C, W. Bennett, S. Boies, and J. Gould. Generating Highly Interactive User Interfaces, in *Proceedings of CHI'89-SIGCHI Computer-Human Interaction Conference*, ACM, New York, 1989, pp. 277-282.