Interactive Design Metric Visualization: Visual Metric Support for User Interface Design

James Noble
MRI, School of MPCE,
Macquarie University, Sydney
NSW 2109 Australia
kjx@mri.mq.edu.au

Larry L. Constantine
Professor of Computing Sciences,
University of Technology, Sydney
PO Box 123, Broadway, NSW 2007 Australia
larry@socs.uts.edu.au

Abstract

Interactive metric visualization is a novel approach providing complex, multi-dimensional feedback on the effects of layout changes in user interface designs. A graphical overlay, based on the underlying rationale of quantitative design metrics, provides immediate feedback, continually guiding designers toward improved layouts. Effective visual metaphors, colour coding, and dynamic updating enable designers to interpret and utilize more complex information than from simple quantitative data or static overlays. This technique is especially suited to accelerated design and development using modern visual development tools. An experimental prototype for this approach is described and initial experience is reported

1. Introduction

Much software development is conducted on a compressed time scale without the benefit of systematic user interface design, expert consultation, or usability testing. Time-boxed development and accelerated applications development life cycles have increased the pressure to deliver more software in less time [12, 14]. Commercial development tools supporting accelerated visual development, such as Borland's Delphi, IBM's Visual Age, or Microsoft's Visual Basic, have come to the aid of beleaguered developers. However, as these have become more powerful and easier to use, the very simplicity with which user interfaces can be created has invited ad hoc approaches that eschew design modeling or even organized problem solving.

Software and applications developers make most decisions regarding user interface design, and they are generally ill-equipped and ill-prepared for the task. To produce better, more usable systems in today's pressured environment, developers need simplified methods and tools that help them focus on essentials and make user interface design decisions more quickly, more confidently, and to better effect.

This paper describes a powerful new technique for visualizing information about the quality of user interface designs. Interactive metric visualization overlays graphical information about design quality on to the actual layout of a screen or dialogue box. Changes in the screen's layout are immediately reflected in this dynamic graphical overlay, providing continuous feedback as the developer tries various layouts.

2. Related work

The work described in this paper bases visualization of UI designs upon metrics for those designs.

2.1. Visualization in UI design tools

Visualization is becoming a common tool for software development. For example, algorithm animation systems are used to provide illustrations for computer science textbooks [3], and wide-spectrum program visualizations have been developed to produce multiple views of programs at multiple levels of abstraction [31].

Apart from directly displaying actual components making up an interface, existing interface design tools have made surprisingly little use of visualization. The most common additional visualization technique is to overlay layout information onto the editable display of components. For example, OPUS [25], Gilt [22], and Cardelli's interface builder [5] continuously display attachment points and lines representing the constraints used within their layout algorithms. A few design tools also display semantic information about the relationship between the interface and the rest of the program. For example, IBM's VisualAge overlays the interface design with lines linking components to the internal objects in the program with which they are associated.

Most closely related to the present work, Sears [34] suggested that visualization can help user interface

0-8186-7525-X/96 $05.00 © 1996 IEEE

designers and evaluators judge the likely efficiency of a particular user interface design. Sears overlaid static illustrations of interface designs with lines representing transitions between visual components within various usage scenarios. More frequent transitions were represented by thicker lines. A pilot study [35] found that these static displays improved judgements of relative efficiency.

2.2. User interface design metrics

Design metrics permit quantitative evaluation of software designs. Unlike subjective evaluations or objective performance measurements, design metrics can be evaluated before systems are implemented. Although design metrics in general are well established in software engineering research and practice [4, 21, 24] relatively little has been done with design metrics for evaluating user interface quality. Tullis [36] devised several simple metrics, such as Screen Complexity, Balance, and Density, based on the spatial distribution of interface features. Comber and Maltby [7, 8] refined this work into Layout Complexity, a metric based on the variance in sizes and alignments of visual features. Sears [33] developed a metric termed Layout Appropriateness (*LA*) that assesses the relative efficiency of a given layout for conducting a mix of tasks.

Design metrics have particular appeal in the context of accelerated development because they permit quick and cost-effective assessment of user interface quality and early identification of potential usability problems. In connection with other rapid, low-cost usability techniques, such as usability inspections [11, 29], design metrics may offer significant gains for time-pressured professionals.

3. Metric visualization

A suite of five design metrics has been developed for guiding user interface designers and developers [15, 16]. The goal has been a coherent set of applied metrics that are simple to use, conceptually sound, and have a clear and transparent rationale connecting them to established principles of good design. The five metrics in this suite are Essential Efficiency, a measure of expected task efficiency, Task Visibility, a measure of accessibility of features, Layout Uniformity, a measure of visual order, Task Concordance, a measure of fit between layout and task structure, and Visual Coherence, a measure of semantic organization. The experimental prototype described here was designed to support these last three metrics.

3.1. Task concordance

Task Concordance (*TC*) is a measure of the fit between the expected frequency of various tasks and their relative difficulty using a given interface design. *TC* is computed as the rank order correlation (Kendall's τ) between tasks ranked by operational difficulty (e.g., steps or path length) and by anticipated frequency in use. It is related to Sears's Layout Appropriateness but, whereas *LA* is based on absolute transition probabilities between pairs of visual components, *TC* is based on expected relative frequencies of complete task scenarios or use cases [13]. Absolute values of transition frequencies can be reliably estimated only from substantial accumulated experience, so *LA* is better suited for redesigning deployed systems than for designing the first version of any interface. In contrast, relative frequencies of tasks are typically easier to estimate than their absolute frequencies, making *TC* better suited for developing new applications.

3.2. Layout uniformity

Layout Uniformity (*LU*) simply measures the graphical uniformity or orderliness of a user interface layout. Layout Uniformity is a simplification of Layout Complexity [7, 8], which derived from earlier work on typographical layout [2] and screen design [36]. Layout Complexity is argued to have a basis in information theory. Galitz [20], however, maintains that similar results are achieved simply by summing the number of visual components and the number of different top edge alignments and left edge alignments of components. Experience in practical comparisons of real-world designs suggests that height and width, as well as bottom and right edge alignments are also factors in subjective judgments of visual regularity. In practice, it has also proved convenient to follow the lead of Sears [33] and normalize to an index ranging from 0 to 100.

$$LU = 100 \bullet \left(1 - \frac{(N_h + N_w + N_t + N_l + N_b + N_r) - M}{6 \bullet N_c - M}\right)$$

where N_c is the total number of components on the screen, dialogue box, or other interface composite, and N_h, N_w N_t, N_l, N_b, and N_r are, respectively, the number of different heights, different widths, different top edge alignments, left edge alignments, bottom edge alignments, and right edge alignments of visual components. *M* is an adjustment for the minimum number of possible alignments and sizes that makes the value of ***LU*** range from 0 to 100: $M = 2 + 2 \bullet \lceil 2\sqrt{N_{total}} \rceil$, with $\lceil \; \rceil$ denoting "smallest integer greater than."

Layout Uniformity is based on the argument that highly chaotic arrangements are more difficult to use. Complete regularity is not the goal, however. Moderately high uniformity (*LU* = .5 to .9) characterizes good designs. Measuring only aspects of the spatial arrangement of visual components, LU is insensitive to the meaning or function of components and to any task characteristics.

3.3. Visual coherence

Visual Coherence (VC) measures how closely visual organization matches the semantic relationships among concepts associated with components. It is based on the principle that well-structured interfaces group semantically related components together. VC is an extension to user interfaces of the well-established software engineering complexity metric, cohesion [9, 37]. Operationalized in a variety of ways, cohesion has been widely applied in software engineering practice and research and has more recently been successfully extended into object-oriented software engineering [6, 18, 23]. An earlier attempt to apply cohesion to user interface design [28] suffered from difficulties in both definition and computation.

Visual Coherence of a visual group (e.g. a dialogue box or window) is the ratio of the total "relatedness" of pairs of visual components to the total number of pairs. Total VC for a screen or dialogue is the sum of the VC for each level of grouping.

$$VC = 100 \bullet \left(\frac{\sum_{\forall l} G_l}{\sum_{\forall l} N_l \bullet (N_l - 1)/2} \right)$$

with $G_l = \sum_{\forall i,j \mid i \neq j} R_{i,j}$

where N_l is the number of components in group l and $R_{i,j}$ is the semantic relatedness between components i and j in group l, $0 \leq R_{i,j} \leq 1$. In practice, semantic relatedness can be simplified to $R_{i,j}$=1 if components i and j are "substantially related", $R_{i,j}$=0 otherwise. This formula correctly favours organizing user interface components into subgroups, but only so long as the groups make sense, that is, they enclose substantially related components. Preliminary research [17] supports the ability of this metric to discriminate designs in terms of both user preferences and ease of use.

To identify "substantially related" pairs, a subjective affinity clustering [10] can be performed. An object-oriented approach to semantic clustering is also under investigation.

3.4. Interactive visualization

Interactive metric visualization embodies metric-based, interactive, multi-dimensional conceptual feedback in real time. Interactive visualization reveals concerns captured by the metric suite explicitly in terms of the mental model used by the designer, that is, the layout of the interface itself. Instead of simply calculating and displaying numerical values or their graphical equivalent, visualization draws attention to those aspects of designs that affect the metrics by displaying graphical information connected with the underlying conceptual or theoretical basis of the metric. Visualization provides information that is easier to interpret and likely to be more valuable in guiding the developer toward improved designs. Since visualizations are dynamically updated as the designer manipulates interface features, the effects of any changes upon the design metrics, as well as the design itself, are immediately apparent.

Some interface design systems, such as HYDRA [19] or AIDE [34], employ a knowledge base of design constraints or rules and inform designers when these are violated. In contrast, visualization assists designers in critiquing their own designs. Designers decide which rules to follow and which to ignore but receive feedback on the effects of their decisions. In practice, good design balances conflicting concerns, and most sets of rules or metrics are contradictory. For example, when using AIDE, the designer modifies the interface layout by adjusting rule weights and requesting a redesign. Using our approach, the designer works in one seamless process, and receives continuous visual feedback about design quality without changing their focus of attention.

Our own initial experiments suggest that displaying the numeric value of design metrics may actually be a deterrent to good design because, instead of attending to the overall structure of the layout and the broader design issues, designers can become too focused on "the numbers," maximizing the numerical values even at the expense of better layout.

For similar reasons, practical tools should provide simultaneous feedback in multiple dimensions. Multi-dimensional feedback requires careful choice and fine tuning of display techniques to allow complex information to be presented in a manner that makes sense to the designer and does not becoming visually overwhelming or confusing.

Dynamic display is the heart of the approach. In this technique, instead of post-design calculation of metrics or static overlay of information on a fixed drawing or "screen-shot," the user interface is "live" and can be manipulated by the designer while metric feedback is maintained. In our experiments, even very complex networks of relationships become decipherable when the designer is able to move visual components and see the effect on the display. This effect must be seen to be appreciated fully.

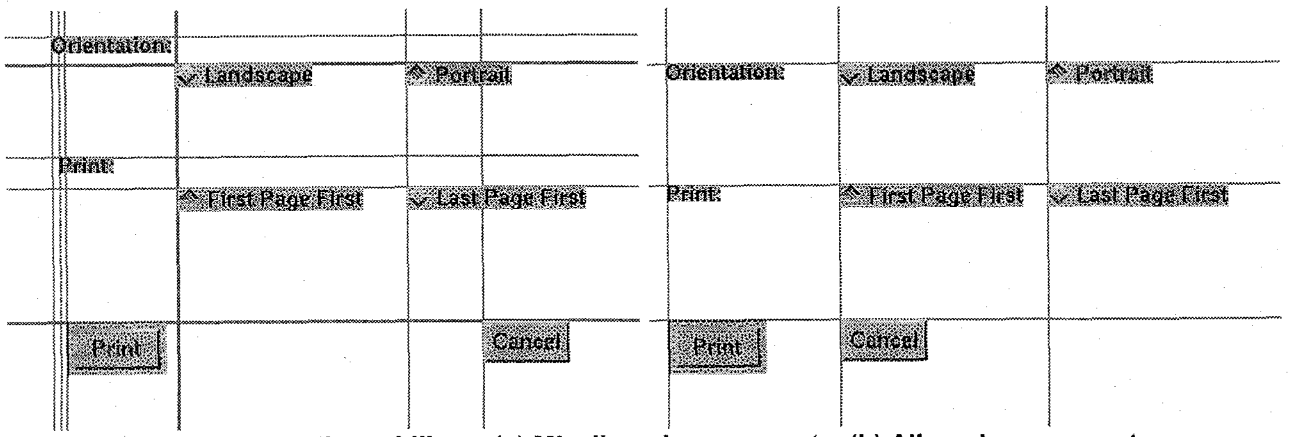

Figure 1. Adaptive gridlines. (a) Misaligned components. (b) Aligned components.

4. Demonstration prototype

The Visual Usability Engineering Interactive Tool (VueIt) was created as a simple proof-of-concept prototype and platform for experimentation with interactive metric visualization. It was written in Tcl/Tk [32], based upon a simple design diagram editor [30]. The prototype supports dynamic rearrangement of interface components and entry of relevant metric information by direct manipulation. In this version, interface components are not active and not editable; they can only be repositioned on the interface.

Because interactive metric visualization supplements actual layout display, we concentrated on metrics that are not immediately apparent from the layout itself. For example, such basic visual metrics as horizontal and vertical balance and lack of negative space used in DON [27] and AIDE [34], are omitted. Lack of balance or misuse of negative space are typically obvious from inspection. DON and AIDE employ such metrics because they automatically generate layouts for ranking or modification by human designers. In our approach, layout is manual, so such simple metrics are easily taken into account by designers.

Three metrics from the suite described above, *VC*, *LU*, and *TC*, are each interactively visualized in distinct ways.

4.1. Visualizing layout uniformity

Layout Uniformity (*LU*), which measures the regularity of spatial layout, is straightforward to visualize. Variance in component sizes is visually obvious, but the number of different top edge and left edge alignments can be highlighted through a grid of lines tracing the top and left edges of every component (Figure 1a). These *adaptive gridlines* move as interface components are moved. When two components are aligned along an edge, the gridlines for that edge merge, simplifying the display (Figure 1b).

Adaptive gridlines are related to Gilt's graphical tabs [22] as both provide lines to which interface components can be aligned. Garnet's graphical tab lines are superficially more powerful, as components can be aligned to tabs at edges and centers and moving a tab line moves the widgets aligned to it. However, tab lines and adaptive gridlines differ in purpose and function. Tab lines assist designers with mechanical alignment details, while adaptive gridlines visualize alignment as an issue in Layout Uniformity. Tab lines are created explicitly by the designer, whereas adaptive gridlines are created and displayed automatically for each interface component. Similarly, tab lines are fixed until the designer explicitly moves them, whereas adaptive gridlines automatically adapt to the positions of components.

4.2. Visualizing visual coherence

Visual Coherence *(VC)* measures the way a design groups interface components, compared with semantic groupings in the underlying application. A component is considered strongly related to every other component in its semantic group, and weakly related to others. *VC* has been visualized initially with *grouplines* to link semantically related components. In Figure 2, for example, three groups of components are identified, each component linked to others in its group with thick grouplines of unsaturated colour. As the dialog is rearranged, grouplines move with the components. In the layout of Figure 2a the Print and Orientation labels have been exchanged, resulting in a multicoloured tangle of grouplines. Figure 2b shows the dialog with the labels repositioned correctly; grouplines link three separate clusters of components.

4.3. Visualizing task concordance

Task Concordance (*TC*) measures the fit between expected task frequency and task difficulty. For graphical

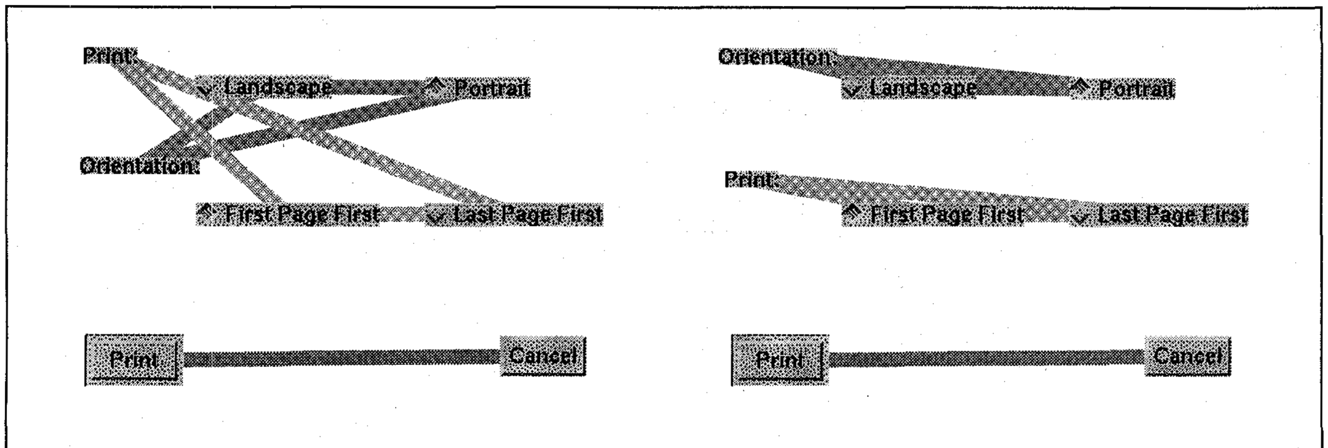

Figure 2. Grouplines. (a) Weak semantic groups. (b) Strong semantic groups.

interfaces such as dialogue boxes, the length of the path through which the user must move a pointer is one reasonable estimate of task difficulty. *TC* is visualized with directed lines in saturated colours drawn on top of interface components. These *pathlines* connect components in task order; line weight (thickness) represents task frequency. For example, in Figure 3, four task scenarios are superimposed on a print dialog box, each starting in the upper left corner. The layout in Figure 3a is obviously not particularly efficient; Figure 3b illustrates an alternative with shorter pathlines revealing the improvement.

Interactive visualization of *TC* differs from the visualization of transition frequencies [34, 35]. Where transition lines merge information from different scenarios into transition frequencies between components, pathlines preserve information about the structure of each task scenario. Multiple tasks traversing the same components are shown with multiple pathlines, thus either thick lines or bundles of lines highlight critical transitions. Because each pathline corresponds to one task scenario, each use case can be traced individually by the designer. Task scenarios can also be animated by the tool moving a simulated mouse pointer along the pathline or flashing each link in turn.

4.4. Multi-dimensional visualization

The real strength of interactive visualization is that a number of metrics can be displayed at once (Figure 4). Each metric is visualized using a separate visual encoding and in a separate layer. From front to back these are: task pathlines, interface components, and adaptive gridlines and coherence grouplines. The layers reflect the semantic content of the metrics. Pathlines, which represent users' interactions with the interface, are drawn in front of interface components. Adaptive gridlines, which provide structural information about component layout, are in the same layer as components but behind pathlines. Grouplines, which display information about underlying application semantics, are drawn behind components, closest to the application. Because of the layering, information on each metric is displayed and distinguishable without excessively obscuring the overall design of the dialog. When directly manipulating interface components, designers can determine the effects of

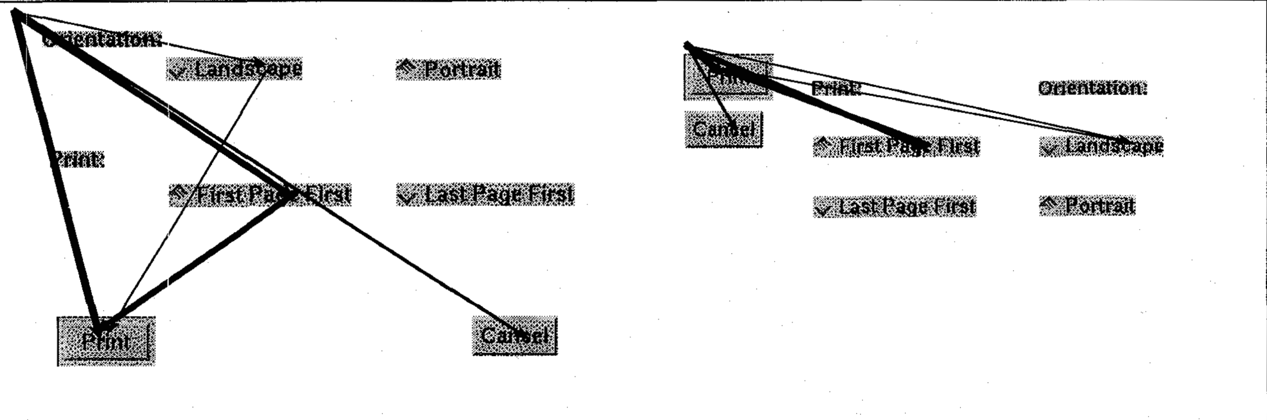

Figure 3. Pathlines. (a) Long task paths. (b) Shorter task paths.

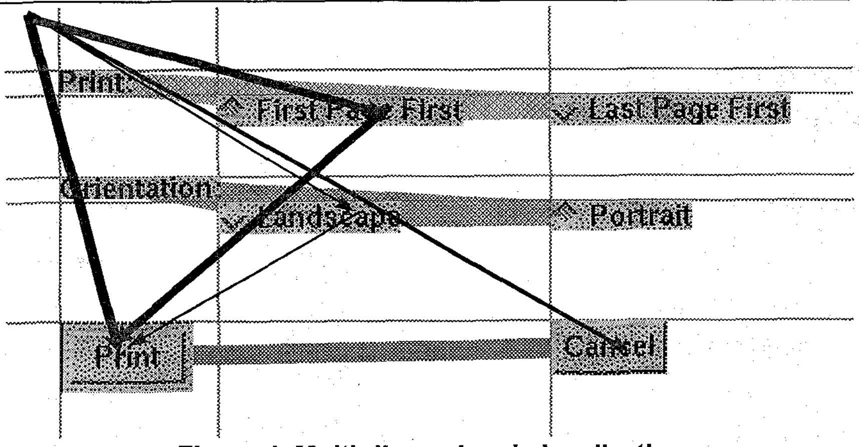

Figure 4. Multi-dimensional visualization

changes on the interface layout and all three metrics simultaneously.

5. Ongoing work

The current prototype is intended as a platform for research into the technique of interactive metric visualization. Laboratory experiments to validate the metrics, and to investigate how designers use metric visualization and what impact it might have on their work are planned. This research will eventually explore and compare a variety of metrics and visualization schemes.

Informal experiments by the authors and students at the University of Technology, Sydney have already been used to refine the technique and the prototype tool. Several working hypotheses have already emerged from these initial investigations. Display of numeric values of a metric may encourage optimization of a particular parameter at the expense of broader design considerations, and interactive visualization of a single metric could have similar effects. On the other hand, multiple numeric values for metrics are likely to be difficult to interpret and utilize. Informal experiments suggest that interaction with dynamically updated visualizations aids in the separation, interpretation, and utilization of complex metric data when it is appropriately layered and encoded. However, when grouplines for Visual Coherence are incorporated with pathlines, displays of complex interfaces can become a visual birds nest of lines. An alternative technique for visualizing semantic grouping has been prototyped using colored screens that overlay components with tints corresponding to semantic groups.

Some tool enhancements are under consideration. Because of a limitation of the Tk Canvas widget, programming was simplified by using bitmaps captured from application programs as interface components. Having active, editable components would broaden the range of experiments possible with the prototype tool. Metric lines are used for visualization only and do not respond to input. A improved tool might, for example, overload adaptive gridlines by allowing them to be dragged, thus moving all the aligned components, as is possible with graphical tabs. To facilitate aligning interface components, optional snap-to-grid behaviour could be helpful [25]. A full-featured user interface design tool would allow metric visualizations to be turned on and off selectively and to be augmented with numeric values. Active, adjunct display of metric-based graphics could also be useful, for example, continuously updated histograms of path lengths or even Fitts Index of Difficulty for task scenarios. Indeed, a complete tool might support a broad range of user selectable metrics.

We are also continuing to refine the metrics themselves, and are investigating new metrics arising from visualization techniques. For example, in our experiments, users often attended to changes in path direction, manipulating the layout to reduce deviation from straight paths. This suggests a possible total-angle metric connected with smooth workflow and visual saccades. Some of the metrics, for example, Task Concordance, generalize to entire user interface architectures, but the current visualization approach does not support multiple windows. It remains to be seen how best to accomplish this or whether it will be useful. Task Concordance pathlines, like Sear's transition lines model tasks as a sequence of steps which must be completed in a fixed order. Practical task scenarios are typically partially ordered sequences that do not require all steps be performed in fixed order, moreover, the order can be influenced by component's positions on the interface. We plan to extend pathlines to handle partially ordered tasks.

Ultimately we envision integration of interactive metric visualization with visual development environments, like Borland's Delphi, permitting dynamic evaluation and improvement of actual user interfaces under development, not just of passive visual prototypes. The current experimental prototype is a stand-alone tool, although it could be adapted to share design descriptions with real interface builders. Since even the fully interpreted Tcl prototype was quite responsive on a 64M Sparc4, a compiled implementation of these techniques should be quite simple to integrate into commercial interface builders.

Acknowledgments

We gratefully acknowledge a grant from Constantine & Lockwood, Ltd., and the support of ISSC Australia, a joint venture of IBM Australia and Lend Lease Corporation.

References

[1] N. Bevan and M. Macleod. Usability Measurement in Context, *Behaviour & Information Technology, 13* (1-2): 132-145. 1994.

[2] G. A. Bonsiepe. A Method of Quantifying Order in Typographic Design, *J. Typographic Research, 2*: 203-220. 1968.

[3] Brown, M. *Algorithm Animation.* MIT Press. 1988

[4] D. N. Card and R. L. Glass. *Measuring Software Design Quality.* Englewood Cliffs, N.J.: Prentice Hall. 1990

[5] L. Cardelli. Building User Interfaces by Direct Manipulation, *ACM SIGGRAPH Conference Proceedings.* 1988.

[6] S. Chidamber and C. Kemerer. A Metrics Suite for Object-Oriented Design, *IEEE Trans. Software Eng., 20* (6): 476-493. 1994

[7] T. Comber and J. R. Maltby. Screen Complexity and User Design Preference in Windows Applications, in S. Howard and Y. K. Leung, eds., *OzCHI '94 Proceedings.* Canberra: CHISIG, Ergonomics Society of Australia. 1994

[8] T. Comber and J. R. Maltby. Evaluating Usability of Screen Designs with Layout Complexity, H. Hasan and C. Nicastri (eds) *OzCHI '95 Proceedings.* Canberra: CHISIG, Ergonomics Society of Australia. 1995

[9] L. L. Constantine. Segmentation and Design Strategies for Modular Programming. In T. O. Barnett and L. L. Constantine (eds.) *Modular Programming: Proceedings of a National Symposium.* Cambridge: Information & Systems Press. 1968

[10] L. L. Constantine. Software by Teamwork: Working Smarter, *Software Development, 1* (1), July 1993.

[11] L. L. Constantine. Collaborative Usability Inspections for Software *Software Development Conference Proceedings.* San Francisco: Miller Freeman. 1994.

[12] L. L. Constantine. In-Time Delivery. *Software Development, 2,* (7), July 1994.

[13] L. L. Constantine. Essential Modeling: Use Cases for User Interfaces. *ACM Interactions, 2* (2): 34-46, April 1995

[14] L. L. Constantine. Under Pressure. *Software Development, 3,* (10), October 1995.

[15] L. L. Constantine. Usage-Centered Software Engineering: New Models, Methods, and Metrics, Iin Purvis, M. (ed.) *Software Engineering: Education & Practice.* Los Alamitos, CA: IEEE Computer Society Press. 1996.

[16] L. L. Constantine. Measuring the Quality of User Interface Designs, *Software Development Conference Proceedings.* San Francisco: Miller Freeman. 1996.

[17] L. L. Constantine. Visual Coherence and Usability: A Cohesion Metric for Assessing the Quality of Dialogue and Screen Designs. *OzCHI'96 Proceedings,* Hamilton, NZ, 1996.

[18] D. W. Embley and S. N. Woodfield. Cohesion and Coupling for Abstract Data Types. *Sixth Annual Phoenix Conference on Computers and Communication.* (229-234) Phoenix, February 1987.

[19] G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl and T. Sumner. Embedding Computer-Based Critics in the Contexts of Design. *InterCHI'93 Proceedings.* ACM Press. 1993.

[20] W. O. Galitz. *It's Time to Clean Your Windows: Designing GUIs that Work.* New York: Wiley-QED. 1994.

[21] Gilb, T. *Software Metrics.* Cambridge, MA: Winthrop. 1977

[22] O. Hashimoto and B. A. Myers. Graphical Styles for Building User Interfaces by Demonstration. *UIST'92 Conference Proceedings.* ACM Press. 1992.

[23] B. Henderson-Sellers, L. L. Constantine and I. M. Graham. Coupling and Cohesion: Toward a Valid Metrics Suite for Object-Oriented Analysis and Design, To appear in *Object-Oriented Systems.* 1996.

[24] B. Henderson-Sellers. *Object-Oriented Metrics: Measures of Software Complexity.* Upper Saddle River, N.J.: Prentice Hall PTR. 1996.

[25] T. R. Henry, S. E. Hudson and G. L. Newell. Integrating Gesture and Snapping into a User Interface Toolkit. *UIST'90 Conference Proceedings.* ACM Press. 1990.

[26] I. Jacobson, M. Christerson and L. L. Constantine. The OOSE Method: A Use-Case Driven Approach In A. Carmichael (ed.) *Object Development Methods.* New York: SIGS Publications. 1994.

[27] W. C. Kim and J. D. Foley. Providing High-level Control and Expert Assistance in the User Interface Presentation Design. *InterCHI'93 Proceedings.* ACM Press. 1993.

[28] P. Kokol and I. Rozman and V Venuti. User Interface Metrics. *ACM SIGPLAN Notices.* 30(4). 1995.

[29] J. Nielsen and R. L. Mack. *Usability Inspection Methods.* Reading, Mass. Wiley. 1994.

[30] J. Noble. Scribble: A Design Tool with a Minimal Interface. *OzCHI'96 Proceedings,* Hamilton, NZ, 1996.

[31] J. Noble, L. Groves, and R. Biddle. Object-Oriented Program Visualisation in Tarraingím. *Australian Computer Journal, 27* (4). December 1995.

[32] J. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley. 1994.

[33] A. Sears. Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout, *IEEE Transactions on Software Engineering, 19* (7): 707-719. 1993.

[34] A. Sears. AIDE: A Step Toward Metric-Based Interface Development tools, *UIST'95 Proceedings.* New York: ACM Press. 1995.

[35] A. Sears. Visualizing Efficiency: A Technique to Help Designers Judge Interface Efficiency *CHI'96 Conference Companion.* New York: ACM Press 1996.

[36] T. S. Tullis. A System for Evaluating Screen Formats: Research and Applications, in R. Hartson and D. Hix (eds.), *Advances in Human Computer Interaction, Vol. 2.* Norwood, NJ: Ablex. 1988.

[37] E. Yourdon and L. L. Constantine. Structured Design. Englewood Cliffs, NJ: Prentice Hall. 1978.