Adaptive Interfaces for Supporting Design by Example

Brian Lee, Scott R. Klemmer, Savil Srivastava
Stanford University HCI Group
Computer Science Department
Stanford, CA 94305-9035
[balee, srk, savil]@cs.stanford.edu

Ronen Brafman
Stanford University Multiagent Group
Computer Science Department
Stanford, CA 94305
brafman@cs.stanford.edu

ABSTRACT
Analogy plays an important cognitive role in reasoning and problem solving. One illustration of analogical cognition can be found in design practice, where viewing examples is an established technique for inspiration and learning. While digital information technologies have made it easier for designers to access examples of other designers' work, significant opportunities exist for selecting and presenting examples in a proactive fashion. In this paper, we introduce techniques for dynamically deriving interfaces for example-based design tools using decision-theoretic selection, designer specification, and end-user preference as inputs. This paper describes a manifestation of these techniques in the Adaptive Ideas web page builder, an HTML-based display platform for web page designers that leverages content metadata to automatically generate displays of examples. We present an evaluation of these techniques through a first-use study.

ACM Classification: H.5.2. [Information Interfaces]: User Interfaces—*Graphical user interfaces (GUI); Interaction styles*. D.2.2 [Software Engineering]: Design Tools and Techniques—*user interfaces*. H1.2. [Models and Principles]: User/Machine Systems.

General terms: Design, Human Factors, Algorithms

Keywords: Adaptive interfaces, design by example, decision theory, model-based UIs

INTRODUCTION
Analogy plays an important cognitive role in reasoning and problem solving [12]. One illustration of analogical cognition can be found in the use and reuse of examples, where people draw from one example of a subject to gain insight or information on another. Viewing examples of previous work is an established technique in many design disciplines: compendiums such as "The Big Book of Logos" [5] serve as highly regarded resources for inspiration.

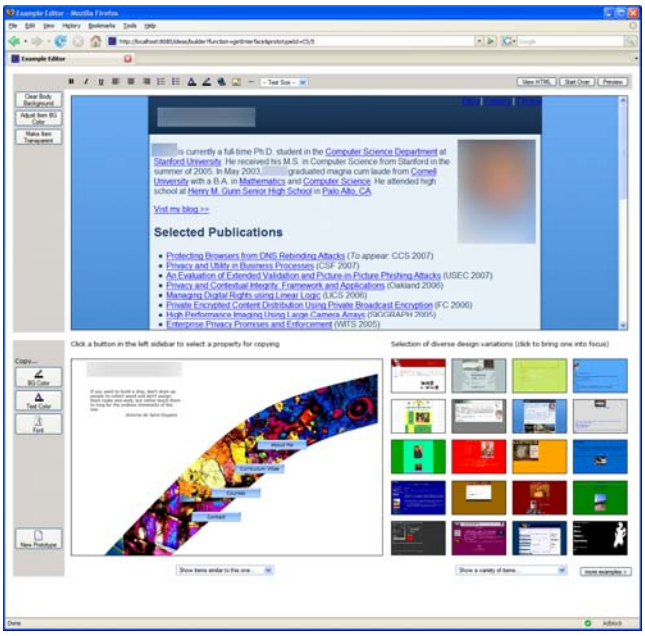


Figure 1. An adaptively generated web page builder, with examples displayed along the bottom of the interface. Interactive components are specified by a designer; content selection and interaction layout is performed algorithmically.

Digital information technologies have significantly improved users' ability to access a wealth of information: designers seeking to find examples of other designers' work need only open their web browsers. However, finding examples that are useful (representative, interesting, different, *etc.*) in a vast ocean of resources is difficult; designers may get lost among the possibilities, and many may not even know where to start.

We propose a novel application of decision-theoretic methods to the challenge of adaptively selecting, laying out, and navigating example artifacts to improve design practice. The hypothesis manifest in this work is that *proactive presentation and agile browsing of analogical information can increase awareness and serendipitous inquiry*.

The design of such an example-centered system raises several issues: How are examples provided to the system? How does the system choose what examples to show? How does one browse the chosen examples? How does one copy and /or modify features of existing examples in order to integrate them into one's own work?

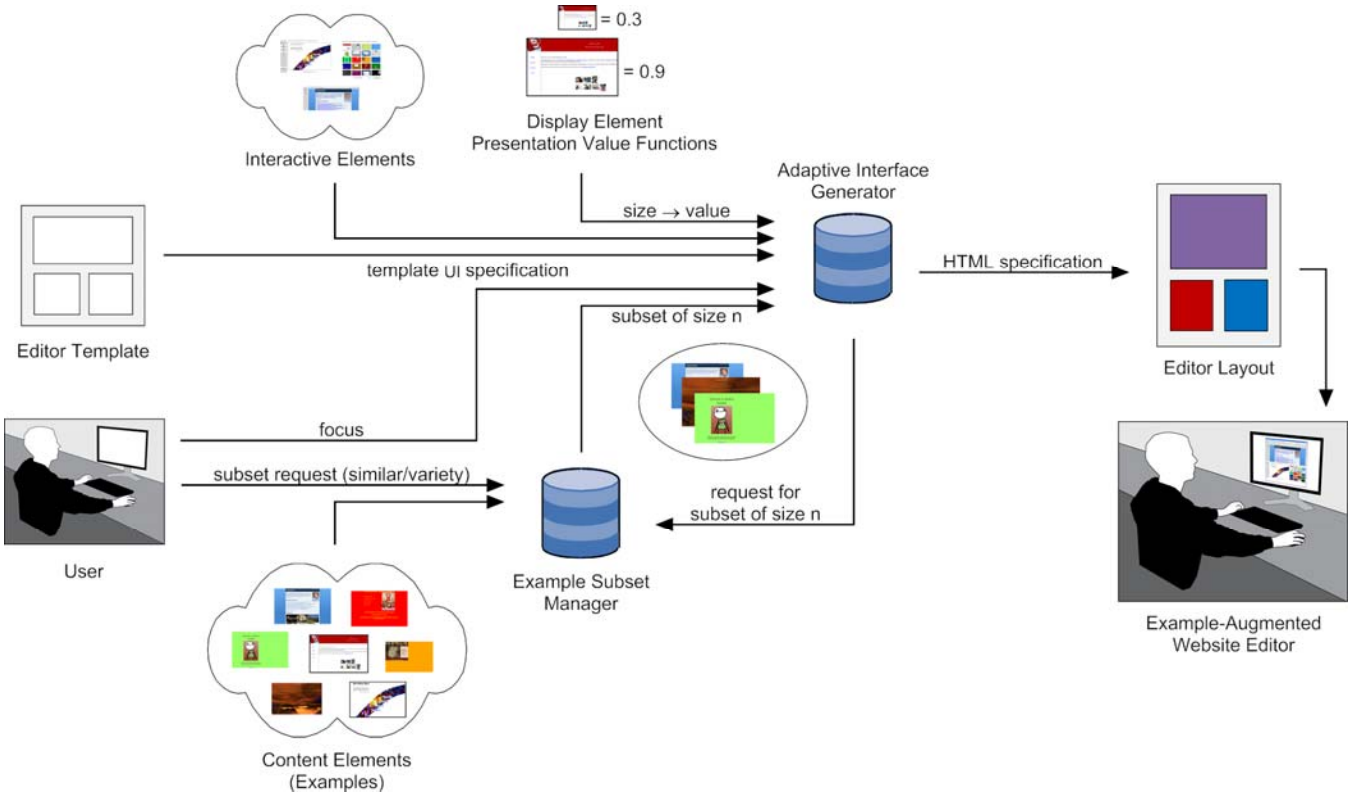


Figure 2. The Adaptive Ideas system uses subset selection algorithms and layout functions to automatically generate example-augmented interfaces.

Our research focuses on the second question: how to select and present interesting sets of examples in ways which are felicitous with current practice. In addition to choosing good examples, it is important to manage the dynamics of the user's attentional focus [12], as proactively displaying presentation information introduces additional potential for distraction and error.

This paper contributes a technique for dynamically deriving interfaces for example-based design tools using decision-theoretic selection, designer specification, and end-user preference as inputs. The algorithm is embodied in Adaptive Ideas, an HTML-based display platform which leverages content metadata and adaptive algorithms to automatically generate displays relevant to users' current activities (see Figure 1). The dataset in the Adaptive Ideas web page builder prototype is drawn from real homepages posted on the World Wide Web.

From a technical perspective, this research draws on prior work on model-based user interfaces [25] and adaptive interfaces [9], and in particular on the idea of casting interface generation as a constraint-based optimization problem [3, 11, 32].

The rest of the paper is organized as follows. We define the concerns of example subset selection and outline the dimensions that we use to analyze it. We next explain the Adaptive Ideas layout algorithm, implementation, and performance. We describe results of a first-use study examining the use of our example-based interface for website design. We conclude by discussing related and future work.

EXAMPLE SELECTION

The Adaptive Ideas web page builder seeks to assist users by displaying examples of existing pages. At its core is a subset algorithm that chooses which examples from the corpus to display to users so as to "maximize" estimated design value.

Design value is operationalized through two proxy measures: the *usefulness* of example content to the user's current task or request, and the *value* associated with the size of display elements in the overall interface. In this section, we elaborate on how content is selected for display.

Content Elements and Attributes

Content elements are example media that together comprise the dataset from which the adaptive algorithm selects. In Adaptive Ideas, they are existing homepages harvested from the web. Content elements have *attributes*, which are facets [31] that can be flat or hierarchical.

This paper focuses on using *visual properties* as attributes: background color, primary font, column layout, and visual density. Here, we manually assigned values for each page attribute; we believe that a production implementation could tractably assign them automatically.

Distance

At the heart of the Adaptive Ideas algorithm is the determination of *distance*, a metric that models the likeness of two attribute values. The distance between two attributes is a real number whose value depends on the properties of the attribute. For example, the distance function between two background colors is calculated by mapping the colors into a three-dimensional space (biconic HSB) and calculating the distance between the respective points in the color space. In contrast, the distance function used for fonts is a simple ternary function: 0 if the fonts are the same, 1 if they are both serif or both sans serif fonts, 2 if the fonts do not share serif characteristics.

Distance functions are used to compute two types of subsets of interest to the display of examples: *similarity* and *variety*.

Similarity

The goal of the similarity subset algorithm is to find a set of *n* objects most similar to a given object for a given attribute. We hypothesized that showing examples similar to a given example would be useful for designers who may have a exemplar in mind which is close to ideal, and are looking for subtle design variations.

Adaptive Ideas uses a simple algorithm to derive similarity subsets: it calculates the distance of all objects from the given object and sort them in ascending order of distance, taking the first *n* items (see Figure 3).

Variety

The goal of the variety subset algorithm is to find a set of *n* objects that represent a "diverse" subset of objects along a given attribute axis. We hypothesized that showing a well-selected variety of examples along a given attribute would give designers a better feel for the overall attribute space and thus provide better inspiration.

This raises the question of what defines a "well-selected" variety: one that shows off all possible values of the given attribute, one that represents the distribution of the underlying dataset, or one following some other formula? For instance, the majority of websites in our prototype dataset have a background color of white. An algorithm that tries to represent the distribution of the dataset would contain mostly white web pages (Figure 3c); in this case, an algorithm that focuses on displaying a variety of possible values may be more desirable, allowing users to see the full design space (Figure 3d). On the other hand, such an algorithm may emphasize outliers or unusual points in the design space.

The Adaptive Ideas framework takes a spaced stochastic approach to selecting a representative variety. First, the system picks a random example from the dataset as a starting point. Next, a random example is selected from the remaining elements in the dataset which are at least ε distance away from all of the elements selected thus far, where ε is a spacing function defined on a per-attribute basis.

The choice of ε has significant influence on the behavior of the spaced stochastic algorithm. When ε is zero or small relative to the design space, this algorithm degenerates to the completely random case (Figure 3c). As ε gets larger relative to the space, the algorithm has fewer elements from which to choose, and thus risks not filling up the space. We select a large ε, such that the theoretical maximum number of elements chosen is close to *n*.

The algorithm continues picking elements until either (1) *n* elements have been selected or (2) there are no legal elements remaining, *i.e.*, every remaining unselected element is less than ε distance away from an element in the selected subset. If more elements are needed (case 2), the system selects elements at random from the full set of remaining elements until *n* have been chosen. On balance, ε guarantees that distinctly different values for the given attribute will be represented in the variety set, while filling out remaining elements randomly implies that some of the under-

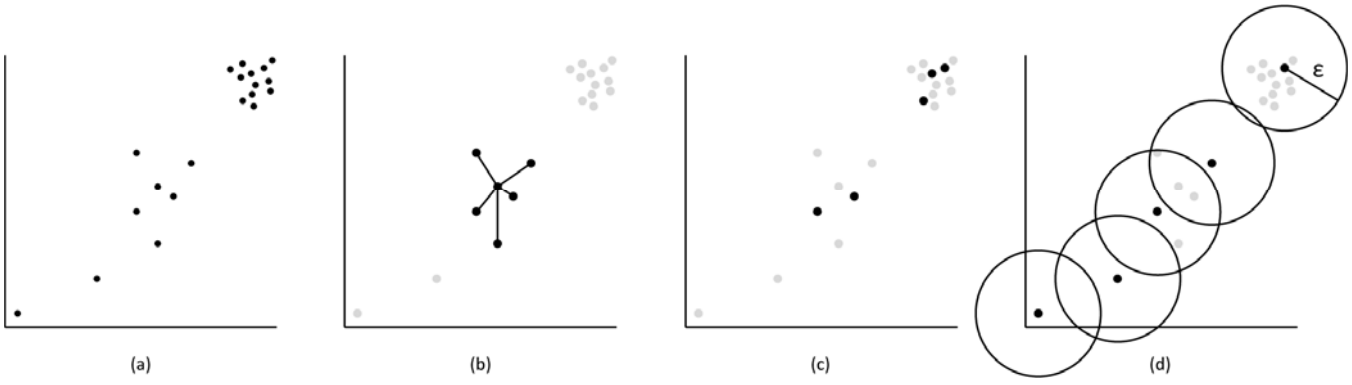


Figure 3. Illustration of Adaptive Ideas subset selection algorithms. (a) Two-dimensional representation of examples laid out in an attribute design space. Note that the space is "chunky," that is, a large fraction of the examples can be found in one area. (b) Similarity algorithm, which chooses the *n* closest examples to the given example. (c) Naïve variety algorithm, which randomly chooses *n* elements from the design space. Note that large areas of the design space are not represented by any chosen examples. (d) Adaptive Ideas variety algorithm, which chooses examples at least ε distance away from other chosen elements. Although the underlying distribution is partially hidden, users are shown a larger portion of the valid design space.

lying distribution of values will be reflected. A variant would be to iterate over successively smaller values of ε until enough legal elements are found; this would further emphasize the breadth of the design space.

INTERFACE LAYOUT

The Adaptive Ideas system recognizes two categories of display elements: *content* elements (examples) and *interactive* elements.

Interactive elements are interface units which provide some function or expose a service; examples include traditional GUI elements such as buttons and sliders, and more complex elements such as HTML editors. Interactive elements may include content, but are distinct from content elements because their behavior is dynamic rather than static.

The Adaptive Ideas system uses a combination of designer specifications and adaptive techniques to perform interface layout. Designers may create XML-based *templates* to partially specify the appearance and behavior of an example display. Similar to Damask [16], templates allow interface designers to specify grouping and layout of content and interactive elements in a device-independent fashion. Using templates, designers can also embed interactive web components which are not part of the Adaptive Ideas system but which provide other services, such as email and calendaring systems. This ability to include any HTML snippet that represents interactive content introduces a mash-up approach to creating adaptive displays.

The Adaptive Ideas algorithm decides how to visually present the display elements by selecting a *layout style*. The layout style is a function of the output display D and template T, and is specified as a set of tuples of content elements, positions, and sizes:

$$\langle e, x, y, w, h \rangle$$

where e is an element, x and y are the element's position in this layout, and w and h are the width and height in pixels of the element. In the Adaptive Ideas implementation, all elements are allocated rectangular regions.

Content elements are laid out in special interactive regions called *adaptive information grids*. The template dictates where these grids should be rendered, though their sizes are dynamically chosen in the same fashion as other elements.

Presentation Value

A key consideration when choosing a layout is deciding the size to render display elements. Larger items are generally easier to read and select, and therefore correspond to higher attentional value than smaller items. (In social settings, research suggests that people correlate size with permissibility in reading other's content [26].) However, increased space for one element necessarily implies less space for another. We encode this tradeoff in a *presentation value function*: $p(e,w,h,D)$. This function estimates the utility of presenting a display element e at a given size (w, h) in a given display D.

In general, larger sizes receive higher presentation scores and smaller sizes receive lower scores. However, the relationship between size and value is non-linear, and varies by element type. (One could extend the Adaptive Ideas architecture so that this relationship is defined on a per-instance basis, perhaps using metrics similar to those of Suh *et al*. [24].) Figure 4 shows an example of how content presentation scores are calculated.

We derived the presentation value functions for the current Adaptive Ideas system by assessing utility of the various element types (text, images, interactive widgets) at different

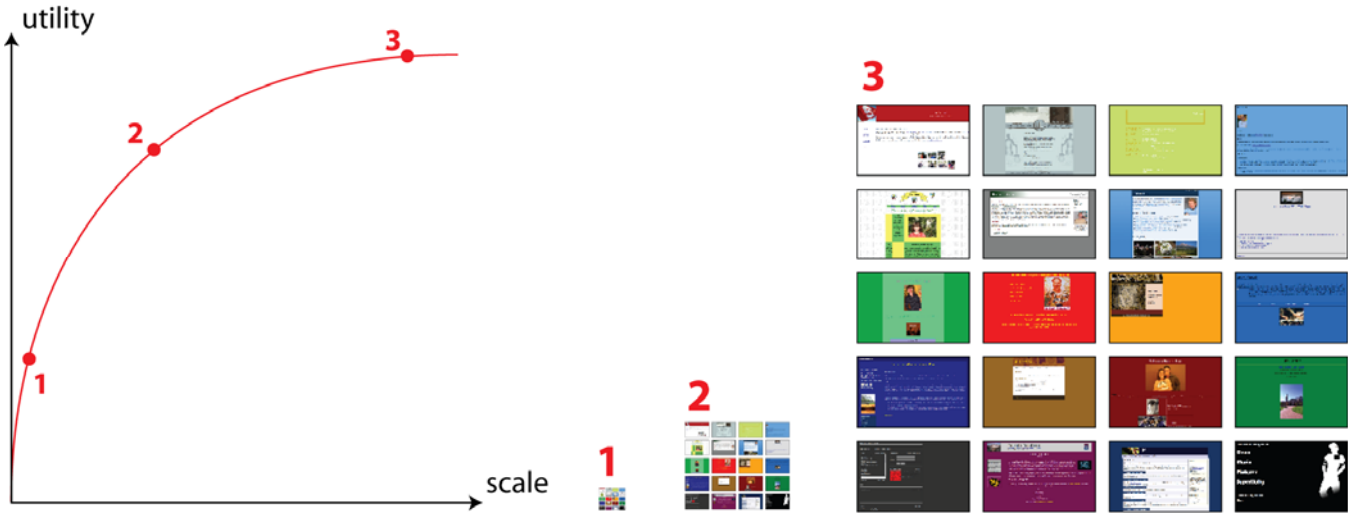


Figure 4. The Adaptive Ideas system assigns presentation value functions for content elements. *Left:* Graph of sample presentation values for example images. *Right:* A grid shown at very small (1), small (2), and large (3) sizes. Note that the images are readable when larger, still understandable when smaller, but not generally useful at the smallest sizes, modeled by the significant drop in presentation values at the latter.

sizes, then hand-tuning the functions and their parameters. One special case is the adaptive information grid, which has a presentation function equal to the sum of the presentation values of the content elements it contains.

Relevance

Another important piece of the Adaptive Ideas layout algorithm is *relevance*, which models the value of an interactive element in a given situation.

Relevance is determined with respect to a user's *focus*. This focus (or foci, if more than one) is specified either *implicitly* when selecting content to view or edit, or *explicitly* by requesting a similarity or variety subset. The goal of using this concept of focus ensures that the interface only displays elements relevant to the user's current activities, so as not to distract her from the task at hand.

To estimate the usefulness of seeing an interactive element d given the current focus or foci F, the Adaptive Ideas system uses a *display relevance function*: $r(d,F)$. Generally, elements that the user has explicitly requested receive higher scores from r than elements which are being shown peripherally; elements which are not needed in the current interaction state are assigned an r score of zero. These functions are also defined by hand on a per-element basis, but may be configured by designers or adapted through use.

ADAPTIVE CALCULATIONS

The Adaptive Ideas system (see Figure 2) receives the following inputs from the system and the environment: content elements, interactive elements, output display, and a design template. The algorithm searches the space of possible layouts and selects the layout with the maximum estimated utility for the given output display.

Estimated Value of Display Elements

For displays of content elements in information grids, we use a simple algorithm for indicating order: a *row-major ordering* (left-to-right, top-to-bottom) where the starting item is at the top left.

In this formulation, an element's absolute location has no effect on its estimated value. A more complex model would assign different values if an element appeared in the center or the side, near the top or near the bottom of the interface.

Given a focus F, the estimated value of an interactive element at a given size is a multiplicative function of its relevance to the given focus and its presentation value at the given size:

$$s(e,w,h,F) = p(e,w,h) \times r(e,F)$$

A low *presentation* or *relevance* value will result in a low score, even when the other input value is high: a highly relevant item is of little value if it is unrecognizable, and a prominently displayed item is not valuable if it is not relevant to the user's current task or state.

Estimated Value of a Layout

Given a focus F, the estimated value of a layout is the sum of the estimated values of all elements displayed in the presentation:

$$s(F,L) = \sum_{e \in L} s(e,w,h,F)$$

where w and h are the width and height of element e in layout L.

This function presumes that the contributions of a given element are *independent* of the presence or absence of other elements. Though we account for some of this in our selection algorithms, we recognize that this assumption may not be valid for all situations: there may be interactions between different elements that may either increase (*e.g.*, due to synergies) or decrease (*e.g.*, due to clutter or overlap) the estimated value of a presentation. Computing such relationships has been researched in other domains with highly structured metadata [32] but is nontrivial when dealing with freeform and less structured data.

Finding the Optimal Interface Layout

Assuming no additional constraints beyond the requirement to fit all selected items on the screen, and using the current model of presentation scores, this problem can be viewed as a two-dimensional variant of the knapsack problem. This is a difficult computational problem, and an active area of research [17]. As we anticipate the existence of additional constraints, we believe that optimization algorithms for this problem will be an important topic of research.

We use dynamic programming (caching the results of subproblems; in this case, partial layouts) and branch-and-bound methods to conduct the search. To boost performance, we also perform *discrete calculations* for layout: instead of evaluating every possible integer width and height, we iterate through possible dimension values in five-pixel increments.

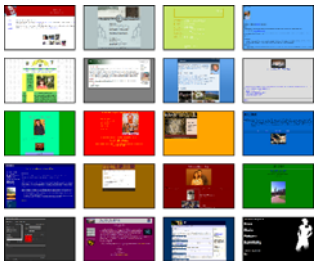
Optimizing layout is simplified when laying out content elements in an information grid. As fractional displays of content elements are useless, the algorithm needs only to search through a small range of discrete size settings, specifically sizes that result in an exact integer number of elements either across or down for a given size. Finding the best set of content items to display at a given size then becomes a greedy search, linear in the number of elements.

Intuitively, the information presentation problem is a tradeoff between showing a smaller number of items at larger sizes and showing a larger number of items at smaller sizes. The Adaptive Ideas framework quantifies this tradeoff neatly and succinctly, enabling quick and efficient evaluation of candidate interfaces.

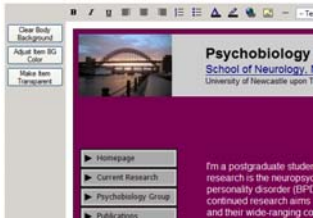
SCENARIO

We illustrate the envisioned use of the Adaptive Ideas web page builder with an example scenario. Elaine Marsh is a 21-year-old economics student, starting her senior year.

Studious and reserved, Elaine spends much of her time outside the classroom serving as vice president of the Alpha Beta Gamma honor society and volunteering as a tutor at a local high school. Elaine wants to make a homepage that details her undergraduate activities, including class projects, research papers, and leadership positions. Her vision for the page includes a mature, sophisticated design and a slightly conservative feel.

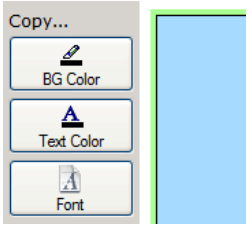


She opens the Adaptive Ideas web page builder and is presented with a variety of possible starting points for her website. She browses through them, looking for a design that she thinks is appropriate. Elaine chooses a two-column design with a purple background.

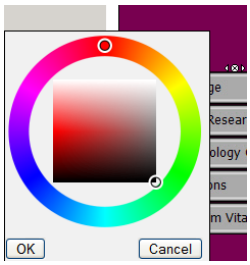


The interface displays her selection in the editing area. Links and buttons indicate features that the user can copy and where to paste them. A text note reminds Elaine that she can browse more examples and copy elements from each of them to her prototype, or she can edit things manually using the controls along the top of the interface.

Elaine decides that the background color isn't exactly what she would want, so she selects "Show a variety of background colors;" the interface presents several examples spanning different hues, saturations, and brightnesses. Elaine selects one of the blue examples and then clicks "Show examples similar to this one"; a set of blue and purple examples is displayed. She sees an example with a tasteful light blue background that she fancies. She clicks on "background color", clicks on the blue of the example, and clicks a third time on the prototype to replace the purple background with the new blue.



Elaine still isn't completely satisfied with the background color, so she clicks on the color widget at the top of the interface, and selects the background of the prototype. A color wheel pops up, allowing her to tweak the blue slightly, making it a touch lighter. She then clicks on the text of the prototype web page, replacing the example's name with hers, and filling in some of the page navigation with her categories.



She continues to browse examples looking for inspiration. A page that uses the Georgia font catches Elaine's eye; after some consideration, she switches the prototype to Georgia, both for readability and style, and alters the font size using the manual controls. She adds a head-and-shoulders picture of herself to the top-left corner of the page. Next to the picture, she places a prominent link to her resume. Satisfied, she uploads the page to a server.

IMPLEMENTATION

We have implemented these selection and layout algorithms in the form of an example-based application for web design. The current prototype generates HTML interfaces using Java Servlets and AJAX for additional interactivity. Our testbed implementation leverages a collection of approximately 250 homepages harvested from the web.

Internally, three components drive Adaptive Ideas: a *subset manager*, an *adaptive interface generator*, and a *web page proxy*. The subset manager takes a content request (similarity or variety, number of items), reads metadata for all the examples from the Adaptive Ideas database, and returns an appropriate subset of the elements. The adaptive interface generator takes as input a set of content elements, a display template, and a set of output properties, and returns an HTML layout. The web page proxy identifies the properties of example websites in the focus pane (see below), taking a click coordinate and returning the requested style feature at that point.

Externally, the Adaptive Ideas website editor prototype interface contains three interactive components: the web editor, the example pane, and the focus pane.

The *example pane* is an adaptive information grid which displays a set of examples. When the interface is first started, only the example pane is shown (see Figure 5, left). To begin, the user is presented with examples representing a variety of background colors. The user navigates through the examples by clicking *next* and *previous* buttons at the bottom right and bottom left. The user may also request to see a variety of elements along a different dimension by clicking a drop-down box at the bottom of the component.

The user selects an example to modify by clicking on an example on the initial screen. This brings up the example page in the *web editor* (see Figure 5, center) at the top of the page. The web editor is a WYSIWYG HTML editor, implemented using Mozilla Firefox's design mode, which allows the user to manually edit the page.

Once the user has selected an example to edit, clicking on another example in the example pane brings that example

page into view in the *focus pane* (see Figure 5, right). From the focus pane, the user may copy features from the example to their prototype by selecting a feature, clicking on a point on the example to copy the feature from that point, and clicking on a point in the prototype to paste the feature at that point. The user may also request to see examples similar to the example in focus by clicking a drop-down box at the bottom of the component.

EVALUATION

We conducted a first-use study of Adaptive Ideas to assess the usefulness of our example-based interfaces for design practice. The study group comprised nine participants. Participants had the following educational backgrounds: three from Computer Science, four from Engineering, and two from Humanities disciplines. Participants' ages ranged from 24 to 30; six were male, three female. All of the participants were frequent web users; two of the participants self-rated as experienced or expert web designers; the others had little to no experience designing websites.

Participants were seated at a workstation with the Adaptive Ideas web page builder. Sessions began with a demonstration of the capabilities of the web page builder. Participants were then asked to create websites for two different personas (one was described in the Scenario section), using a different variation of the builder interface for each. The *standard features* variant disabled the similarity and variety features and sorted the examples randomly: users could view all examples, but could only browse them using the next and previous page controls. The *adaptive features* variant offered the full set of controls described in the Implementation section. Personas and interfaces were varied across participants using a Latin square ordering.

Study Results

In our post-study survey, participants found the general presentation of examples highly useful (mean=4.5, median =4.5 on a 5-point Likert scale, σ=0.53), and appreciated the ability to borrow features directly from example web pages (mean = 3.9, median = 4, σ = 0.83). Participants found the adaptive browsing features to be helpful in finding examples, indicating the variety tool to be most useful while exploring the design space (mean=4.4, median=5, σ=1.01), although the similarity tool was also welcomed (mean=3.9, median=4, σ=0.78). In general, participants did not find the examples to be distracting (mean=2.2, median=2, σ=0.83).

Responses were less conclusive on whether it was easier to navigate examples using the adaptive features interface (mean=3.7, median=4, σ=1.22). However, observations and server logs revealed that several participants resorted to long stretches of "linear" browsing while using the standard features interface, during which they clicked on many examples in a row in order to examine them. In the adaptive interface, users selected fewer items for larger viewing. This may indicate that the similarity and variety tools lent themselves to more efficient exploration. Participants also expressed a desire for more dimensions along which to sort and browse examples, particularly aesthetic attributes such as formality.

Novice versus expert use

In post-study interviews, users with little to no design experience particularly approved of having examples integrated into the design tool. All of them expressed particular appreciation for the ability to request variety subsets.

The two self-rated experts differed strongly in their opinions of the use of examples for website design. One of the expert participants found the limitations of the example-borrowing interface annoying and thought the examples were both unhelpful and distracting to the design task, wasting valuable screen space. The other experienced participant commented that the browsing of examples worked well with her personal strategy for this type of design task: "That's my philosophy of designing websites: I like to find a template or exemplar that I think is good and then tweak it

Figure 5. Screenshots of the Adaptive Ideas web page builder. *Left:* Initial grid of examples. As the user has not yet selected an example to modify, the editor and focus panes have zero *r* values, so those components are hidden. *Center:* Interface after the user has selected an example to modify; the selected example is loaded into the web editor, with more examples along the bottom of the screen. *Right:* Interface after the user has selected an object on which to focus (light blue image, bottom left) and requested to see similar items (grid of pastels, bottom right).

by hand." This suggests that future instances of example-adaptive interfaces should allow power users to disable example display at their discretion, but also hints that design by example interfaces may be valuable even for experienced users.

RELATED WORK

This research draws on three areas of prior work: model-based user interfaces, automatic layout systems, and document scoring systems. We discuss each in turn.

Model-Based User Interfaces

The area of model-based user interfaces (*e.g.*, [21]) began with the interest of creating tools for specifying interfaces declaratively, through high-level semantics, rather than imperatively, by the pixel-level details of the implementation. Szekely [25] provides a retrospective overview of this field. The field slowed down in the early 1990s, likely because the desktop PC did not provide sufficient diversity to mandate a higher-level representation: the value of abstraction is derived from the lower margin costs of repurposing—with one platform, there was no amortization to be had.

Automatic Layout

Several projects have explored the automatic layout of interfaces and/or information. Perhaps the two most closely related systems are SUPPLE [11] and RIA [32], which examined constraint-based optimization approaches to interface adaptation. We apply a decision-theoretic strategy similar to that of SUPPLE and RIA, but with significantly different constraints. Adaptive Ideas addresses both user interface elements and visual information sources, and supports a mix of adaptive generation and designer specification. It also has the additional burden of rendering layouts interactively, potentially introducing interesting tradeoffs between optimality and performance. Finally, the RIA system dealt with highly structured, heavily faceted metadata [31]; its algorithms depended on an intricate understanding of the dimensions and their relationships. Adaptive Ideas is designed for less structured, more loosely related data.

Many techniques have been introduced for laying out and browsing large image collections. PhotoMesa [1], a zoomable image browser which encouraged serendipity using a 2D space-filling layout, inspired several design decisions in our implementation (*e.g.*, quantum elements). Saliency-based cropping methods [24] are another innovation that could be applied to later versions of our adaptive interface, posing interesting questions regarding presentation value functions for content. Our adaptive interface research extends this body of work by applying novel techniques in the context of large heterogeneous data sets.

The selection of what information is visible and its arrangement for the user has significant implications for the cognitive activities that are ready-at-hand [15], and the effective presentation of personal information has been the subject of considerable activity. Furnas's fisheye calendar [10] first introduced the idea of a *focus+context* visualization: the calendar item in focus was displayed larger and with local detail; non-focus items would correspondingly shrink. More generally, this example demonstrated how constraints can be effectively used to manage screen layout globally, and this present research is a continuation in that vein. Other research has explored book-like metaphors for information collections [4], and facet-based approaches to search [8]. Our approach draws strongly on faceted search; it distinguishes itself in that display elements are not constrained to be *only* those requested—elements with similarities to those requested may be displayed as a means of providing for serendipity in search and browsing.

Ambient displays have explored the use of spaces and surfaces for proactive presentation of information [6, 28, 30]. Our research follows up on this work by applying adaptive techniques to contextual displays. In particular, we are exploring the peripheral presentation of examples and other epistemic artifacts to encourage exploration.

Document Scoring

We turn to the question of the underlying algorithms and information model. As with prior work on information foraging [20], we seek to improve the *information scent* of interfaces. More precisely, the goal of this paper is to provide scents of potentially valuable information in addition to the specific information has requested. The use of *small steps* observed by Teevan *et al.* in their study of orienteering behavior [27] points to the value of providing scent via contextual information.

As the quantity of information we work with increases [18], and metadata becomes ever more prevalent [2], improved techniques for sorting this information are required. Adaptive user interfaces have proven particularly useful in managing personal information. Rhodes' Remembrance Agent demonstrated the use of richer types of metadata—most notably location—as a means for retrieving information [22]. Perhaps most similar to Adaptive Ideas is Horvitz *et al.*'s email ranking system [23], which employs decision-theoretic techniques to prioritize and rank emails that are likely to contain higher value information or be more urgent; this work was very inspirational in framing our approach. Haystack [14] takes a highly flexible approach to data presentation and user interaction that could easily integrate adaptive techniques to increase visibility.

The information model in this work draws on the idea of faceted metadata [31], the conceptually distinct dimensions of the metadata. Of particular value has been the recent research on lightweight techniques for labeling photographs with rich metadata [7, 19], and the use of those in information retrieval. Again, the difference with this work is that while we employ the same ontological mechanisms, the contribution lies in the use of this schema to enable proactive and adaptive display.

CONCLUSION

This work contributes an algorithm for dynamically selecting content for and generating layouts of example artifacts,

using a combination of decision-theoretic selection, designer specification, and end-user preference. Future work includes integrating other attributes (*e.g.*, page metadata such as creation time, title, and keywords; aesthetic properties such as genre and formality), and deriving design attributes and values programmatically. We also plan to investigate alternative representations for examples (*e.g.*, representative exemplars of example subsets), and examine example-based interactions that use more implicit cues from task activities to proactively display content.

ACKNOWLEDGEMENTS

We thank Ron Yeh, Wendy Ju, and Yoav Shoham for their research insights, and the National Science Foundation and the Wallenberg Global Learning Network for sponsoring this research (NSF IIS-0534662, KAW 2004.0184). We are also grateful to Intel for technology donations.

All human subjects research was conducted under Stanford University IRB approved protocol 3392.

REFERENCES

1 Bederson, B. B. PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. UIST 2001: ACM Symposium on User Interface Software and Technology. pp. 71–80.

2 Berners-Lee, T., J. Hendler, and O. Lassila. The Semantic Web, Scientific American, May, 2001.

3 Brafman, R. I. and D. Friedman. Presentation Adaptation for Rich Media Messages. STRIMM Consortium Working Paper 2003.

4 Card, S. K., L. Hong, J. D. Mackinlay, and E. H. Chi. 3Book: a scalable 3D virtual book. CHI 2004: ACM Conference on Human Factors in Computing Systems. pp. 1095–98.

5 Carter, D. E., The Big Book of Logos: Watson-Guptill. 384 pp. 2001.

6 Churchill, E. F., L. Denoue, J. Helfman, and P. Murphy. Sharing multimedia content with interactive public displays: a case study. DIS 2004: ACM Conference on Designing Interactive Systems. pp. 7–16.

7 Davis, M., S. King, N. Good, and R. Sarvas. From context to content: leveraging context to infer media metadata. MM 2004: ACM International Conference on Multimedia. pp. 188–95.

8 Dumais, S., E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. SIGIR 2003: ACM Conference on Research and Development in Information Retrieval. pp. 72–79.

9 Fischer, G. User Modeling in Human–Computer Interaction. User Modeling and User-Adapted Interaction 11(1). pp. 65-86, 2001.

10 Furnas, G. W. Generalized fisheye views. CHI 1986: ACM Conference on Human Factors in Computing Systems. pp. 16–23.

11 Gajos, K. and D. S. Weld. SUPPLE: automatically generating user interfaces. IUI 2004: Proceedings of the 9th international conference on Intelligent user interface. pp. 93–100.

12 Gentner, D., K. J. Holyoak, and B. N. Kokinov, The Analogical Mind: Perspectives from Cognitive Science: M.I.T. Press. 520 pp. 2001.

13 Hutchings, H. M. and J. S. Pierce. Understanding the whethers, hows, and whys of divisible interfaces. AVI 2006: Proceedings of the Working Conference on Advanced Visual Interfaces. pp. 274–77.

14 Karger, D. R., K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A general-purpose information management tool for end users based on semistructured data. Proc. CIDR. pp. 13–26, 2005.

15 Kirsh, D. The Intelligent Use of Space. Artificial Intelligence 73(1-2). pp. 31–68, 1995.

16 Lin, J., Using Design Patterns and Layers to Support the Early-Stage Design and Prototyping of Cross-Device User Interfaces, Unpublished Ph.D. Dissertation, University of California, Berkeley, 2005.

17 Lodi, A. and M. Monaci. Integer linear programming models for 2-staged two-dimensional Knapsack problems. Mathematical Programming 94(2-3). pp. 257–78, 2003.

18 Lyman, P. and H. R. Varian, How Much Information? 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/

19 Naaman, M., S. Harada, Q. Y. Wang, H. Garcia-Molina, and A. Paepcke. Context data in geo-referenced digital photo collections. MM2004: ACM International Conference on Multimedia. pp. 196–203.

20 Pirolli, P. and S. K. Card. Information foraging. Psychological Review 106(4). pp. 643–75, 1999.

21 Puerta, A. R., E. Cheng, T. Ou, and J. Min. MOBILE: user-centered interface building. CHI 1999: ACM Conference on Human Factors in Computing Systems. pp. 426–33.

22 Rhodes, B. J. The wearable remembrance agent: A system for augmented memory. Personal Technologies 1(4). pp. 218–24, 1997.

23 Sahami, M., S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. AAAI 1998: Workshop on Learning for Text Categorization.

24 Suh, B., H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. UIST 2003: ACM Symposium on User Interface Software and Technology. pp. 95–104.

25 Szekely, P. Retrospective and Challenges for Model-Based Interface Development. DSV 1996: Design, Specification, and Verification of Interactive Systems. pp. 1–27.

26 Tan, D. and M. Czerwinski. Information Voyeurism: Social Impact of Physically Large Displays on Information Privacy. CHI 2002: ACM Conference on Human Factors in Computing Systems. pp. 748–9.

27 Teevan, J., C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a

study of orienteering behavior in directed search. CHI 2004: ACM Conference on Human Factors in Computing Systems. pp. 415–22.
28 Vogel, D. and R. Balakrishnan. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. UIST 2004: ACM Symposium on User Interface Software and Technology. pp. 137–46.
29 Weiser, M. The Computer for the 21st Century. Scientific American. pp. 94-104, 1991.
30 Wisneski, C., H. Ishii, A. Dahley, M. Gorbet, S. Brave, B. Ullmer, and P. Yarin. Ambient Displays: Turning Architectural Space into an Interface between People and Digital Information. COBUILD 1998: International Workshop on Cooperative Buildings. pp. 22–32.
31 Yee, K.-P., K. Swearingen, K. Li, and M. Hearst. Faceted Metadata for Image Search and Browsing. CHI 2003: ACM Conference on Human Factors in Computing Systems. pp. 401–08.
32 Zhou, M. X. and V. Aggarwal. An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. UIST 2004: ACM Symposium on User Interface Software and Technology. pp. 227-36.