Recognizing Creative Needs in User Interface Design

Michael Terry, Elizabeth D. Mynatt
Everyday Computing Lab, GVU Center
College of Computing, Georgia Tech
Atlanta, GA 30332
{mterry, mynatt}@cc.gatech.edu

ABSTRACT
The creative process requires experimentation, the exploration of variations, and the continual evaluation of one's progress. While these processes are frequently non-linear and iterative, modern user interfaces do not explicitly support these practices, and instead impose a linear progression through tasks that is a poor fit for creative pursuits. In this paper we use data from three case studies, and draw upon Schön's theory of reflection-in-action to illustrate specific deficiencies in current user interfaces when used in creative endeavors. We then develop a set of guidelines for user interface design and demonstrate their application in three designs intended to support tasks in the domain of image manipulation.

Categories & Subject Descriptors: H.5.2 [User Interfaces] – graphical user interfaces (GUI), interaction styles; I.3.4 [Graphics Utilities] – graphics editors

General Terms: Design, Human Factors

Keywords: Creativity, open-ended tasks, non-linear interaction model, Side Views, on-demand previews, image manipulation

INTRODUCTION
The creative process requires much effort to define, refine, and realize a creative vision [4, 13, 14]. As computational systems increasingly enter into the creative process, end-users, developers, and researchers strive to understand and define the ways these systems can support and enhance the creative process. For example, Shneiderman outlines a number of high-level user interface guidelines intended to support innovation, such as "what-if" tools or histories that can be recorded, reviewed, and replayed [14], while other work investigates specific systems that target certain aspects of the creative process, such as the collaborative nature of creative work [11], or the early, formative stages of design (e.g., [6, 9]).

The work we discuss in this paper seeks to support the creative process in the moment-to-moment interactions a user has with the user interface as she acts on her data. Accordingly, our focus is on the lower-level interactions likely to be found in the use of *any* tool, such as the act of choosing a command and its parameters, branching to explore variations in depth, and evaluating one's current position to understand what to do next.

In this paper we report on ongoing research to support the creative process in the domain of image manipulation. We begin with three case studies that serve to demonstrate typical needs of users as they progress through open-ended tasks. These studies clearly affirm the need for the user to be able to *experiment*; to *explore variations*; and to *evaluate* past, current, and potential future states. These processes closely parallel those described by Schön's theory of reflection-in-action [13], which we use to frame our observations and to further our understanding of what is required to support progression through an open-ended task.

We then turn to an analysis of the ways interfaces support or hinder creative processes. We find that user interfaces often fall short of explicitly supporting the experimental, non-serial nature of the creative process, and instead impose a fairly restrictive interaction model that we term the *Single State Document Model*. The Single State Document Model requires a document to be in one, and only one, state at any particular time, thereby imposing a serial, linear progression through a task that is at odds with the "messy," highly iterative creative process.

From the unsatisfied needs of users, we synthesize guidelines for user interface design, referring to related efforts in the process. Three designs illustrate application of these guidelines: Side Views, on-demand previews of commands that appear within a tool-tip's pop-up window; Parameter Spectrums, a tool for choosing parameters that shows a spectrum of previews for a parameter's range of values; and the Design Horizon, a workspace that explicitly supports multiple versions of a document, multiple views on data, and previews of potential future states. We conclude with initial reactions to functional and paper prototypes of our designs, and future work.

IMAGE MANIPULATION: THREE CASE STUDIES
The three case studies below illustrate typical work practices of users engaged in open-ended tasks using a popular image manipulation application. These studies serve to highlight user needs in open-ended tasks, thus suggesting opportunities for interface design.

Each study consisted of an interview to understand individual work practices, followed by the interviewee demonstrating one of their typical tasks.

Newspaper Image Control Desk: Image Toning
Our first case study centers on the task of image toning (i.e., color correction) at a newspaper. A former employee of a major newspaper described the task of preparing photos for the newspaper's printing press at the newspaper's image control desk, and demonstrated the process on a sample image.

The image control desk's charter is to improve the quality of the images before printing, without altering their editorial content. This process includes cropping and sizing the image, and adjusting its colors to make the photo print well on newsprint. Because newsprint tends to soak up ink in unique ways, new employees must continually monitor how their images print the next day, to cater their toning process to the newspaper's printing press and paper.

C&C'02, October 14–16, 2002, Loughborough, Leic, United Kingdom.

To achieve consistent results, employees rely on a set of color-toning heuristics they develop through experience. An overarching goal in the process is to apply as few changes as possible, because each change causes original pixel data to be lost. Therefore, individual operations must be chosen with care so that the *total number* of operations is minimized: Each operation should contribute as much as possible toward a better image, without causing side-effects that make subsequent corrections difficult or impossible. These constraints necessitate a highly iterative process that often requires users to try out several different approaches to achieve the best result. Employees thus make back-up copies of their work as they go along to allow them to branch and explore possibilities in depth.

As part of the iterative process, each operation is evaluated in a number of ways after it is executed. "Undo" and "redo" are repeatedly invoked in quick succession to "flash" the current and previous versions of the image on the screen. This technique enables comparisons to be made in time, rather than in space (i.e., side by side). Users also "zoom in" and pan the image to analyze specific portions of the image in detail. If the specific portions are acceptable, users will then zoom out to perform a more holistic analysis of the effect of the most recent command. At times they will also evaluate their work by examining the separate color channels of an image. Collectively, these techniques enable users to evaluate the results of their actions at different scales, varying levels of detail, and via multiple representations.

When results prove unsatisfactory, users undo to the previous version, unless they believe their current strategy seems to have hit a dead end, in which case they will restore a previously saved version.

Interactive, Multimedia Software: User Interface Design

Our second case study looks at how a professional artist uses the image manipulation program to help produce interactive, multimedia websites and CD-ROMs. Like the newspaper employee, a part of her work deals with image toning and preparing images for use in interactive software. In this case study, however, we focus on the portion of her work involving the design of graphical user interfaces.

When designing interfaces, the artist uses the image manipulation application primarily as a drawing and painting program. Typically, her process consists of creating a large, blank canvas and drawing a number of variations of particular components (such as buttons) side-by-side on the same canvas. The variations enable her to discover and develop the right "look and feel" for the particular project, and their side-by-side placements on the canvas enable direct comparisons. The art director with whom she works also creates variations of her work, but uses a different mechanism to contain the variations. Rather than create a large canvas, she creates new layers in the document to hold each version, and selectively makes layers visible to toggle between versions. Accordingly, multiple versions cannot be compared side-by-side, but the use of layers enables different versions to be quickly switched in and out within the context of the larger product.

Once the variations have been generated, the artist meets with the art director to discuss and choose promising versions to be developed further. From these individual elements a coherent interface is created, but like the earlier versions, this composite interface undergoes more changes before arriving at a final state. To support experimentation at these stages, the artist and her art director again utilize extra layers in the image, but also duplicate files to hold the multiple versions. Thus, these users explicitly generate variations and store them in large canvases, multiple layers, and multiple files.

Amateur Artist: Pen Drawing Coloring Before Painting Wood

Our final case study follows an amateur artist using the image manipulation application to color a scanned-in pen drawing of a science-fiction scene. His goal is to decide on a color scheme before using real paints to paint a wood cutout based on the drawing. Since he has little experience painting in color, he is using the image manipulation program to help him iterate through variations before committing to using real paints. Unlike users in the previous case studies, he is a novice user of the program.

To paint the pen-drawing, the artist uses the application to fill regions with a color. His process entails applying a color to a region or set of regions, then sitting back and evaluating the changes. If they prove satisfactory, he continues; otherwise, he undoes the operation and tries another variation. At times he also prints out a grayscale version of the image to reflect on the overall contrast of the image; this practice provides him with another representation of his work for the purpose of evaluation.

Like the users in the other case studies, the artist creates multiple versions by copying files. His file system reflects this process of experimentation: folders are labeled with the type of experiment (e.g., grayscale-only, versions based on a palette used by Matisse, etc.) and files are named according to version number.

UNDERSTANDING USER PRACTICES IN LIGHT OF SCHÖN'S THEORY OF REFLECTION-IN-ACTION

Our observations of the three users and their work practices echo the work practices described by Schön in his theory of reflection-in-action [13]. Reflection-in-action refers to the process by which skilled practitioners tackle an open-ended problem, and can roughly be described by the following steps: a framing of the problem (i.e., an attempt to understand and define the problem), making a move (acting on the framed problem), reflecting on the move (evaluating the consequences and implications of the move), and repeating the entire process. What is noteworthy in this theory is the notion that a practitioner does not apply a standard approach to a problem. Rather, he calls upon a repertoire of past experiences to derive an initial hypothesis, which he tests with small experiments and actions. Furthermore, as he makes a move, he simultaneously transforms the problem into a more desirable state while generating new understandings of the situation by the way it reacts to the move. As Schön puts it, he enters into a conversation with the problem, making a move, analyzing how it "talks back" to him, and responding accordingly.

The case studies revealed many instances of reflection-in-action. In particular, we noted users *experimenting* to better understand the problem and their available options; *generating variations* to approach the problem from multiple angles; and continually *evaluating* their efforts to reflect on their progress and inform their future actions. We discuss each of these activities in more detail now.

Near-Term Experimentation

When faced with an open-ended task, users need to engage in near-term experimentation, which we define as those efforts intended to discover and instantiate the next move. In many cases, users don't know exactly which command to invoke, nor the best parameters for the chosen command, so they must experiment. This experimentation can take the form of trying and undoing multiple commands, or scanning or tweaking a command's parameters.

In our case studies, near-term experimentation played a vital role in individual tasks. Image toning, especially, requires users to experiment with various commands and settings before finding an optimal action that moves them closer to their goal, without causing unnecessary side-effects.

This near-term experimentation closely resembles the process of reflection-in-action as Schön describes it. Users would make a hypothesis about what to do next, and test their hypothesis by invoking a command and adjusting its settings to achieve the imagined effect. As the command's effect became known to the user (either through a command invocation or a preview), the user would either accept the command, tweak the parameters more, or undo it and try another tact. It is important to note that all users, not just novice users, needed to engage in experimentation with commands and their settings – their tasks were not ones that lent themselves to a simple application of predetermined actions.

Variations

While near-term experimentation aids the user in choosing and applying commands, there are times when deeper exploration of alternatives is warranted. The artist designing user interface elements in our study is one example: the artist would generate multiple variations of a specific component by creating them side-by-side on a large canvas, confer with the art director, and iterate on promising versions to arrive at an acceptable solution. Similarly, the artist who was painting a line drawing created various thematic colorations of his line drawing, and grouped these together in named folders in the file system.

Users generate variations sequentially (fully developing one before creating another), or in parallel (working through the steps of creating alternatives at the same time). In our observations, it seemed that when individual files were used to hold variations, they were produced sequentially, while variations "in place" (i.e., in the same document), were more likely to occur in parallel.

Generating variations enables the individual to better understand the problem, its boundaries, and potential solutions. This practice is a common methodology in the design, as it enables an individual to engage in what Schön describes as a "conversation" with the materials and the problem.

Evaluation

As users progress through their task, they need to evaluate their progress. Evaluation happens in both the short- and long-term, for example after performing a near-term experiment, or in the process of generating variations. Though evaluation is a critical aspect of both near-term experiments and the generation of variations, we list this activity independently because of its prominence in our observations.

The users in our case studies performed critical evaluations at every step of their work. In image toning, users would invoke a command, then examine the entire image in detail after the command's application. Sometimes users would evaluate the result on its own, with nothing to compare to, but more frequently they would quickly undo/redo the action to compare the result in time. They would also, on occasion, load a previous version in a separate window to perform direct comparisons between versions further separated in time.

Users would also evaluate their moves through various representations. The representations serve to emphasize certain aspects of their current state, or to provide alternative views into the document. For example, the amateur artist used grayscale printouts to suppress colors when evaluating the contrast. In image toning, separate color channels would be examined to provide an alternative view of the current image.

The evaluative process gives the problem the chance to "talk back" to the person. It is the moment in which the individual reassesses the problem and their understanding of it, before making the next move.

Supporting the Process of Reflection-in-Action

Schön points to a number of tools employed by a practitioner to support reflection-in-action. We note two here, *virtual worlds*, and multiple *representations*.

Schön describes the use of a "virtual world," within which the practitioner can easily generate and test hypotheses: "Virtual worlds are contexts for experiment within which practitioners can suspend or control some of the everyday impediments to rigorous reflection-in-action." [13, p. 162] In our studies, the use of a large canvas or extra layers to hold multiple versions both serve as virtual worlds for the users.

Schön also observes that practitioners make use of various representations of the problem. These alternative representations let a person focus on some details, while ignoring others. We noticed use of alternative representations by the amateur artist when he created printouts, or when individuals examined their work using separate color channels.

SUPPORTING AND HINDERING MOMENT-TO-MOMENT INTERACTIONS

In this section, we provide concrete examples of the ways user interfaces support or hinder the processes of near-term experimentation, the generation of variations, and evaluation. We first discuss the Single State Document Model, an interaction model that strongly influences the way the user progresses through a task when using a user interface.

The Single State Document Model

We define the *Single State Document Model* as the interaction model that recognizes and requires a document to be in one, and only one, state at any particular time. This model necessitates a serial, linear progression through a task where each step replaces the current state with a new state.

While most user interfaces implement a version of this interaction model, it is typically a poor match to the non-linear, experimental processes characteristic of creative endeavors. Although it is difficult to analyze the evolution of user interfaces, a likely explanation for the mismatch between many content-creation tools and the process of open-ended exploration is the narrow interpretation of task analysis in the design of user interfaces. The decomposition of a task into its sequential steps is evident in many user interfaces, for example, a wizard interface that leads the novice user through a series of steps. However, taken too literally, these interfaces disregard the iterative and exploratory processes that make up creative processes. They are over-tuned to *production* tasks.

The poor fit of this interaction model to desired work practices causes users to develop specialized strategies to compensate for deficiencies in the interface. For example, users must make copies of their documents to support the generation of variations. We will see other instances of the Single State Document Model asserting itself as we discuss each activity in turn.

User Interfaces and Near-Term Experimentation

In Support of Near-Term Experimentation

Near-term experimentation is comprised of choosing a command and its parameters, and evaluating the effect of the command.

User interfaces facilitate the process of choosing a command through a hierarchical organization of text-based menu commands grouped according to functionality. Icons on toolbars or palettes provide compact, abstract visual representations of a command, while online help, such as tool-tips or contextual help, offer additional hints when selecting a command.

The capability to "undo" enables users to experiment with the knowledge that they can easily revert to a previous state if the experiment fails. Similarly, when multiple, competing alternatives exist (i.e., the user identifies more than one way to approach a problem), undo allows users to sequentially try and undo commands to find the best one.

Interactive previews assist when choosing parameters. Controls that afford direct manipulation (e.g., a slider widget, rather than a numerical input box) create a tight feedback loop useful for finding the "sweet spots" for a particular parameter. This form of experimentation is much faster than trying and undoing commands.

Limitations in Support of Near-Term Experimentation

Existing user interface designs allow users to discover commands, but the terse textual descriptions and compact icons do not accurately convey a command's precise effect. When a user is unfamiliar with a command or the interface, they must enter a try-undo cycle for the sake of discovering the available functionality. This try-undo cycle is distinct from the acts of trying and undoing a command in the process of reflection-in-action: The former case serves to uncover an interface's functionality (because it is not adequately conveyed by the user interface), while the latter case is an experiment.

When faced with competing alternatives, the Single State Document Model makes it difficult to make comparisons because only one command may be invoked at a time. Thus, users must engage in the try-undo cycle to successively try/undo commands, mentally noting their respective effects. Alternatively, users must explicitly create copies of their data to create multiple versions on which they can apply the desired commands. These limitations impose additional burdens on users, and reduce the likelihood of users performing near-term experiments.

Comparisons between previews (when available) is also difficult in current interfaces because many commands have parameters that generate a large space of possibilities to consider. For example, a dialog box that has two parameters each spanning 100 discrete values creates 10,000 different possible combinations. However, with only one preview into the total number of possibilities, users must scan the ranges of values to find desirable combinations.

Finally, while previews are often available for commands with dialog boxes, few to no previews are available for interactive, direct manipulation tools, such as the paintbrush. While some guides may be present (such as an outline for the cursor indicating the brush size) users are offered no other preview of the effect such a tool will have until they operate directly on their data. Furthermore, parameters cannot be varied once interaction has begun (a notable exception is a pressure-sensitive drawing tablet which translates pressure into the intensity of applied effect). These restrictions make it difficult for true reflection-in-action, because the user must first contend with setting the parameters as originally desired, before analyzing the effect of their actions.

Towards Better Support of Near-Term Experimentation

User interfaces should provide unambiguous, authentic representations (e.g., previews) of available commands before the user has to actually commit to invoking the command. Furthermore, these previews should afford comparisons, to aid the user when faced with competing alternatives. Because screen real estate is limited, these previews can appear on demand.

To assist users in selecting parameters, interfaces should provide a *set* of views into the space of possibilities, rather than a single preview. Two complimentary approaches seem possible: presenting an evenly distributed spectrum representing a parameter's range of values, or presenting the parameter space in a semantically meaningful way. For example, an initial view into a parameter space could show the points most likely of interest to the user. Design Galleries [10] is a system that does just that. For its target tasks (e.g., lighting a 3D scene), it will search the space of possibilities to find a set of images perceptibly distinct from one another. The user simply needs to choose the one most similar to the effect desired, which he can then fine-tune to his needs.

Interfaces should offer a dedicated space in which to perform near-term experiments, without needing to modify the document nor its data. This capability is sometimes called a "what-if" tool, but use of this term has usually referred to larger experiments, such as simulations [14]. Instead, this space should be an instantiation of what Schön calls a virtual world, a place to test out hypotheses. VisualAge for Java [1], and the ART system [12] both provide such spaces, the former to test out snippets of Java code, the latter to work on composing and arranging elements of a document. Such experimental spaces lower perceived risks of "trying something out," thereby encouraging experimentation.

Some input devices, like a pen and tablet, allow the user to interactively vary some settings for direct manipulation tools as they are used. Extending this capability to all tools, without requiring a specialized input device, would help remove the guesswork of choosing settings for these tools, and enable reflection-in-action. Furthermore, allowing the user to change the parameters after-the-fact would remove the need to undo and retrace one's work. Editable Graphical Histories [8] is one system that demonstrates this capability: Users are presented with a graphical history of their actions and can modify past steps in place, without needing to undo to a previous state.

User Interfaces and Variations

In Support of Variations

Mechanisms that let users duplicate their data, such as the "Save As" command (which creates a copy of their current document) implicitly support the generation of variations. Users can copy their data, then act on each copy independently to create variations.

Variations can also be explored in place (i.e., side-by-side in a document) if the application's document space can grow indefinitely. For example, users in our case study used multiple layers in the document to hold variations, or created a large enough canvas to hold multiple versions.

"Undo" enables a transient form of exploring variations. Users can go down a path, and if unsatisfied, they can undo to a previous state.

Limitations in Support of Variations

Most user interfaces do not explicitly support the process of branching, and the Single State Document Model does not allow a document to be in more than one state at a time. While duplicating files through the file system or version control software may enable multiple versions to co-exist, interfaces still treat each as a separate, self-contained entity.

Using "undo" to generate variations is problematic because of its transient nature: only one version exists at a time, so direct comparisons are not possible unless the user saves copies in the process.

Towards Better Support of Variations

Variations may come about intentionally (e.g., creating several versions next to one another in a document), or unexpectedly (e.g., in the course of working, the user reaches a point and decides it would be worthwhile to back up and try another strategy). Accordingly, interface support for variations should recognize these two ways of generating multiple versions.

Interfaces should allow extra spaces to be created on demand to hold multiple versions of a document, or portions of it. For example, one can imagine working on a problematic paragraph in a word processor and creating a space to hold a new version right next to the original paragraph. All versions would be accessible while editing, though only one designated version would appear in formal representations (such as a printout). This capability can already be found in the image manipulation program we analyzed, as well as experimental systems like ART [12]. ART allows multiple versions of elements of a document to exist; layers signify which elements appear in the final version.

User interfaces should capture users' actions, make these histories persistent, and provide tools to search and navigate past states. A large body of work already exists analyzing how to enhance histories (e.g., [2, 5, 8]), but the challenge for supporting variations is to provide tools that enable a user to easily traverse histories *and* to reload multiple versions at the same time (something not possible in the Single State Document Model). Recent work on an enhanced history system for website design moves us closer to this possibility [7].

User Interfaces and Evaluations

In Support of Evaluations

User interfaces support evaluations through multiple perspectives and views, such as different levels of zoom, and through alternative representations, such as separate color channels. These evaluations can happen alone, or in comparison to something else (such as another document, or a previous version). For example, we mentioned that a favorite technique of users was to use "undo/redo" quickly in succession to compare the current and the prior state.

Limitations in Support of Evaluations

The biggest hindrance we noted to performing evaluations in current user interfaces was the inability to easily perform side-by-side comparisons of multiple versions or representations of a document. Since the Single State Document Model restricts the user to only one version of a document at a time, side-by-side comparisons necessitate the user making copies of their document.

Evaluation is also hindered when it requires the user to change their data to achieve an alternative representation. For example, the amateur artist was unable to view his painting in grayscale unless he invoked a command to remove the colors, or printed out a copy on a black and white printer.

Towards Better Support of Evaluations

Alternative representations are a critical component to the evaluative process of reflection-in-action. Interfaces should thus allow the user to create multiple views and representations of their data, in its current, past, and potential future states. For example, users should be able to preview multiple commands at the same time, to help choose the best one. These additional views should be persistent, when required, and dynamically update as the user acts on the document. Furthermore, alternative views should not require users to change the state of their data to achieve the new view.

Magic Lenses [3] is an example of a user interface tool that provides these types of capabilities. Each lens represents a specific function: As users pass a lens over their data, the view of their data is transformed by the function represented by the lens.

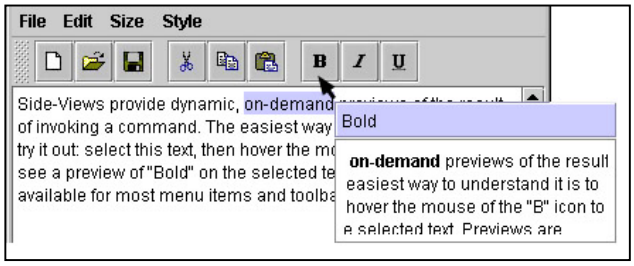

Figure 1. Side Views provide a preview of a command using a copy of the data to be affected. In this figure, a preview of "bold" is shown in a word processor

Users can thus peer into the future and evaluate how their data may be modified, without actually invoking a command.

DESIGNS

The three designs presented below demonstrate the application of the guidelines developed in the last section. As we are motivated to develop systems that can have immediate value to real users, our designs are conservative in some respects to aid in the integration with existing applications and work practices. For example, they assume a basic WIMP interaction model, but are not tied to, nor limited by it. We present the three designs separately, though together they form a complete system.

Side Views

Side Views are enhanced tool-tips that preview the result of invoking a command within the tool-tip pop-up window. The command is shown applied to a copy of the working data, providing an authentic forecast of a command's effect, rather than a generic, "canned" preview. As an example, when the tool-tip for the "Bold" command appears in a word processing application, the Side View shows the selected text with bold formatting applied in the tool-tip window, instead of just the command's name (see Figure 1).

Previews are also displayed when passing over controls that can be directly manipulated. For example, as the user moves her cursor over the range of values represented by a slider, the Side View appears to show a preview of the value under the cursor. To support side-by-side comparisons, the most recent Side View remains visible when a new Side View appears.

Side Views serve to support near-term experimentation and evaluation by displaying authentic previews of potential future states. Users gain an unambiguous view of how a command will affect their data, can understand potential side-effects associated with invoking a command, and can more easily choose among competing alternatives. Furthermore, Side Views preserve existing strategies of representing commands and options (e.g., through text-based menus), while providing more informative displays on demand.

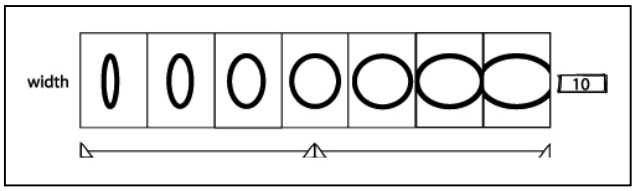

Figure 2. A Parameter Spectrum that varies the width of an oval. The center knob on the slider varies the value, while the triangles on the ends can be moved to vary the range of values displayed

Parameter Spectrums

Parameter Spectrums are snap-in replacements for traditional slider controls and previews. Instead of a single preview, Parameter Spectrums create a spectrum of previews over the range of values for a parameter. Figure 2 displays an example of a Parameter Spectrum for a parameter that varies the width of an oval.

A Parameter Spectrum is comprised of a traditional slider widget, a series of previews above the slider, and bounding controls that vary the portion of the parameter range previewed. When the Parameter Spectrum first appears, the spectrum shows an even interpolation of values across all possible values for a parameter. Through user interaction, the boundaries can be adjusted to show a specific range of values. For example, if the user clicks on one of the previews, the value associated with that preview becomes the new chosen value, and the neighbors of the selected preview "push out" to the edges of the spectrum to form the new boundaries for the range displayed (see Figure 3). The effect is akin to "zooming" in to see a finer interpolation of values around a chosen value. The user may also manually adjust the bounding controls to vary the portion of the parameter showed within the spectrum.

Multiple parameters are represented by their own Parameter Spectrum. Each spectrum varies only in the dimension represented by the parameter. For example, in Figure 3, the parameters for the thickness, width, and height of an oval all vary only in their respective dimensions. When the user chooses a new value for any of the parameters, the other Parameter Spectrums update their previews to reflect the newly chosen value. This behavior enables users to more clearly visualize the interrelations between parameters.

Parameter Spectrums support near-term experimentation and facilitate evaluations as the user formulates her next move.

Design Horizon

The Design Horizon is a space intended to complement the "normal" document window, and is ideally located in a second monitor. Within this space users can place snapshots of their work to support the creation of variations. Persistent, dynamic views of their data can also be placed there. These active views can either be alternative representations of their data (e.g., zoomed-in views), or Parameter Spectrums for user-chosen commands (see Figure 4).

The Design Horizon's main area is a "global" space whose contents persist across documents and sessions. Dynamic views in this global space continually update their views to reflect the active document and its current state. Each file also creates its own workspace within the Design Horizon's global workspace; the file's workspace and its contents open and close with the file. Dynamic views in a file's workspace are tied to the current

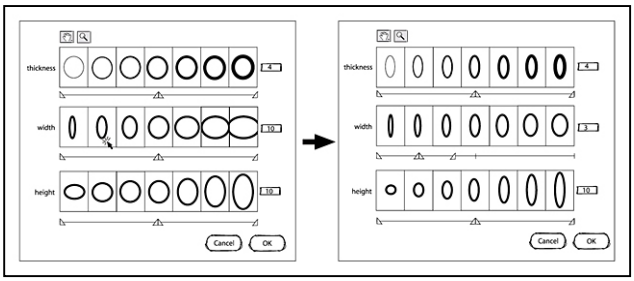

Figure 3. Parameter Spectrums for customizing an oval. The user clicks on the desired width, and the previews for the thickness and height parameters update their previews to reflect the new width

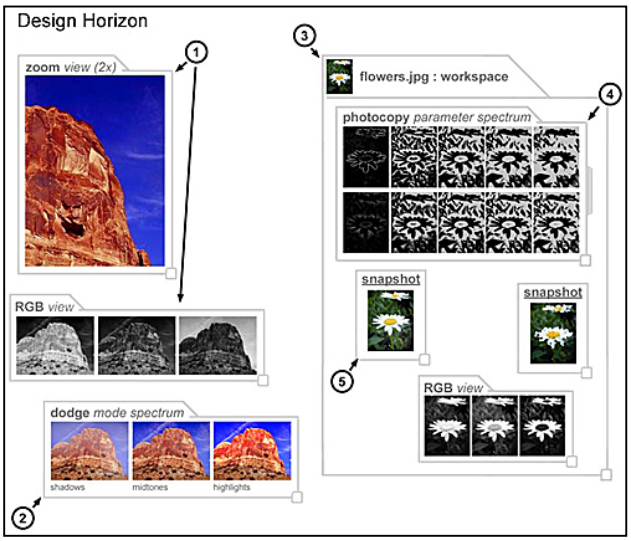

Figure 4. Design Horizon 1) Multiple views onto the active document 2) A Parameter Spectrum showing potential settings for a direct manipulation tool (dodge) 3) Workspace tied to a specific file 4) Parameter Spectrum for "photocopy" filter, tide to a specific file 5) Snapshot stored with a specific file

version of that document, rather than the active document in the application.

Users add commands or snapshots to the Design Horizon by dragging and dropping them in the desired workspace (i.e., the global or file workspace). Double-clicking a snapshot or a Parameter Spectrum loads the document or Parameter Spectrum, respectively. Users can also interact with the Parameter Spectrum directly in the Design Horizon and apply its effect, if desired.

The Design Horizon intends to support the non-linear nature of creative endeavors without losing the conceptually simple model of the Single State Document Model: The Design Horizon compliments the Single State Document Model by creating a space specifically intended to hold multiple versions, and to display multiple views of potential future states (through the Parameter Spectrums). Users can thus manipulate their data and see how their document would change if any of the commands represented by the Parameter Spectrums were applied. The ability to place persistent views of commands on the Design Horizon also

allows users to customize their workspace with the most frequently used commands.

FUTURE WORK

We are in the process of implementing and evaluating an enhanced version of Side Views (see [15]) that incorporates many of the ideas presented in the three individual designs discussed in this paper into a single tool. This implementation is within an image manipulation application based on the open-source GNU Image Manipulation Program (http://www.gimp.org). Initial reactions to working versions have been positive, but more thorough testing of this new user interface mechanism is needed to understand its specific strengths and weaknesses in real-world use.

REFERENCES

1. IBM VisualAge for Java. http://www.ibm.com/software/ad/vajava/
2. Berlage, T. A Selective Undo Mechanism for Graphical User Interfaces Based on Command Objects. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 1., Issue 3 (September 1994). pp. 269-294
3. Bier, E., Stone, M., Pier, K., Buxton, W., & DeRose, T. Toolglass and Magic Lenses: The See-Through Interface. In *Proceedings of the 20th Annual Conference on Computer Graphics*, August 1993, pp. 73-80.
4. Csikszentmihalyi, M. *Creativity: Flow and the Psychology of Discovery and Invention*. HarperCollins Publishers, New York, NY. 1999.
5. Edwards, K., Igarashi, T., LaMarca, A., and Mynatt, E. D. A Temporal Model for Multi-Level Undo and Redo. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, November 2000, pp. 31-40.
6. Klemmer, S.R., Newman, M., Farrell, R., Bilezikjian, M., & Landay, J. The Designer's Outpost: A Tangible Interface for Collaborative Web Site Design. In *CHI Letters, The 14th Annual ACM Symposium on User Interface Software and Technology: UIST 2001*. 3(2), pp. 1-10.
7. Klemmer, S.R., M. Thomsen, E. Phelps-Goodman, R. Lee, J.A. Landay, Where Do Web Sites Come From? Capturing and Interacting with Design History. *CHI Letters, Human Factors in Computing Systems: CHI2002*. 4(1).
8. Kurlander, D., and Feiner, S. A Visual Language for Browsing, Undoing, and Redoing Graphical Interface Commands. In *Visual Languages and Visual Programming*. S.K. Chang (ed.). Plenum Press, New York, NY., 1990, pp. 257-275.
9. Landay, J., and Myers, B. Sketching Interfaces: Toward More Human Interface Design. In *IEEE Computer*, 34(3), March 2001, pp. 56-64.
10. Marks, J., Andalman, B., Beardsley, P. A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., & Shieber, S. Design Galleries. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, August 1997, pp. 389-400.
11. Nakakoji, K., Yamamoto, Y., & Ohira, M. A Framework that Supports Collective Creativity in Design using Visual Images. In *Proceedings of the Third Conference on Creativity and Cognition*, October 1999, pp. 166-173.
12. Nakakoji, K., Yamamoto, Y., Takada, S., & Reeves, B. Two-Dimensional Spatial Positioning as a Means for Reflection in Design. In *Conference Proceedings on Designing Interaction Systems (DIS '00)*. ACM Press, pp. 145-154.
13. Schön, D. A. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, NY. 1983.
14. Shneiderman, B. Creating Creativity: User Interfaces for Supporting Innovation. In *ACM Transactions on Computer-Human Interaction*, Vol. 7., No. 1, March 2000, pp. 114-138.
15. Terry, M., Mynatt, E. Side Views: Persistent, On-Demand Previews for Open-Ended Tasks. To appear in *Conference Proceedings of UIST, 2002*.