Variation in Element and Action: Supporting Simultaneous Development of Alternative Solutions

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

Kumiyo Nakakoji, Yasuhiro Yamamoto
Knowledge Interaction Design Lab, RCAST
University of Tokyo
4-6-1 Komaba, Meguro 153-8904, Japan
{kumiyo, yxy}@kid.rcast.u-tokyo.ac.jp

ABSTRACT
The complexity of many problems necessitates creating and exploring multiple, alternative solutions. However, current user interfaces do not cleanly support creating alternatives at a time when they are likely to be discovered: as users interactively modify data. This paper presents Parallel Paths, a novel model of interaction that facilitates generating, manipulating, and comparing alternative solutions. In contrast to existing approaches such as automated history capture tools, Parallel Paths emphasizes the *active*, *simultaneous* development of multiple, alternative solutions. We demonstrate this model of interaction in Parallel Pies, a user interface mechanism developed for image manipulation tasks that allows users to: easily create solution alternatives as they interact with a command; embed the alternatives in the same workspace; manipulate the alternatives independently or simultaneously as if they were the same object; and perform side-by-side comparisons of each. Results from an initial evaluation are presented, along with implications for future designs.

Categories & Subject Descriptors: H.5.2. [Information Interfaces and Presentation]: User Interfaces – Interaction styles

General Terms: Design, Experimentation

Keywords: Exploration, experimentation, what-if tools, interaction models, parallel exploration

INTRODUCTION
For many problems, a single "correct" solution does not exist. Instead, a variety of solutions can be developed, each possessing its own unique set of strengths and weaknesses. For example, there is no single, best solution for the design of a building, software system, or consumer product: new designs can always be created, and old ones improved [24]. These types of problems are often referred to as *ill-defined* problems [23,25], and span the range from design problems to relatively common tasks such as writing.

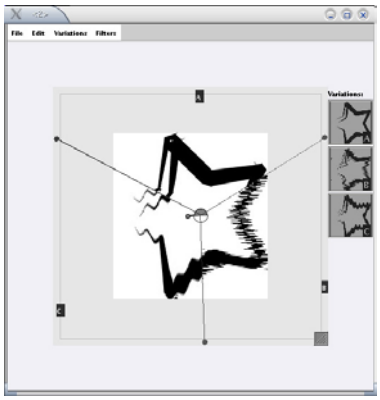

Figure 1. Parallel Pies allows users to embed and visualize multiple solution variations in the same space. Above the user has created and embedded three variations of a star in the same workspace.

The lack of one correct solution often leads to the development of multiple potential solutions, either as part of an explicit process, or out of necessity. For example, early in the problem solving process, experienced practitioners often intentionally create several rough ideas to better understand the problem and the client's needs [14,15,21]. Later, as ideas are instantiated, alternatives are also explored on an ad-hoc basis when it is unclear how to best implement individual elements of the solution [27].

While a number of tools have been developed to support the production of rough solution ideas early in the problem solving process (e.g., [9,13,16]), less attention has been paid to supporting the exploration of alternatives when actively operating on data in later implementation phases. In these later phases, users often need to slightly vary commands and their parameters to develop and explore alternative solutions to *elements* of the entire solution. For example, when manipulating data through a command dialog box, users may find several possible command

settings equally attractive and wish to pursue them in parallel. However, current interfaces do not afford the fluid generation of new alternatives at the point of operating on data. Instead, users must put the operation "on hold," duplicate their data, then re-establish the original context for each variation they wish to pursue. Furthermore, any variations produced become separate entities that must be manipulated independently. This limitation can be cumbersome when the variations produced are more similar than they are different, since similar variations likely require similar treatment as they are further developed.

To better support the practices of generating, manipulating, and comparing multiple solution instances, we introduce a novel interaction model called *Parallel Paths*. Parallel Paths raises the concept of a solution variation to a first-class object in the interface to allow users to:

- create new solution variations before, during, and after invoking a command,
- embed alternative solutions within the same workspace to facilitate direct comparisons, and
- manipulate the variations independently or collectively, as a whole.

We demonstrate the principles of Parallel Paths in the context of an image manipulation application augmented in the following ways:

- Command dialog boxes introduce a new option, *Add Variation*, which allows users to add the currently previewed result as a new variation to the document
- Variations are embedded directly within the same document and kept viewable through an interaction mechanism called *Parallel Pies* (Figure 1). This tool evenly divides the document to show portions of each variation side-by-side
- Commands are augmented to allow users to modify one or more variations simultaneously
- Any variation can be *duplicated* to create a new variation
- Each variation maintains a *complete history* of all its prior states, initially adopting the history of its source. Thus, users can duplicate a variation, then return to a previous state to pursue an alternative path

To motivate the need for this augmented model of interaction, we first describe characteristics of ill-defined problems and how these properties affect the problem solving process. We then review two studies of expert practitioners that argue for the need to develop multiple, potential solutions throughout the problem solving process. With this motivation in place, we examine current interface support and argue that existing tools do not sufficiently meet the need to generate and manipulate variations when working with precise data representations. To address this need, we introduce the Parallel Paths interaction model and present our implementation of this concept in Parallel Pies, an interface mechanism designed to support the production, manipulation, and evaluation of alternatives in an image manipulation application. Results from an initial round of user studies suggest that this interface mechanism can enhance workflow for a number of tasks. We conclude by discussing how these concepts may map to other domains.

ILL-DEFINED PROBLEMS AND SOLUTION VARIATIONS

Ill-defined problems arose as a focused topic of research in the 1960's when Walter Reitman made an important distinction between *well-defined problems*, and what he termed *ill-defined problems*, or problems with poorly defined operators and goals [23]. Prior to making this distinction, research in problem solving primarily concentrated on well-defined tasks, such as the studies conducted by Newell and Simon (summarized in [20]) that investigated how people solved problems with well-defined goals and states, such as cryptarithmetic. In drawing the line between these different types of problems, Reitman claimed that most real-world problems, such as design tasks, are ill-defined and qualitatively different from well-defined problems in form, complexity, and method of solution.

Subsequent research by design theorists, cognitive scientists, and psychologists has refined these distinctions and added weight to Reitman's claims through studies comparing problem solving methodologies for both types of problems (e.g., [8]). Rittel [24] observes that the lack of well-defined evaluation criteria means that solutions to ill-defined problems are neither right nor wrong, only better or worse. Thus, solutions can always be improved, and declaring a solution "done" is as much a function of the availability of resources (e.g., time, money) as it is a measure of the suitability of the solution to the task [25].

To help navigate the inherent thorniness of ill-defined problems, experienced practitioners rely on sets of strategies developed over time. One such strategy is to generate multiple, alternative solutions to cope with the lack of precise goals and evaluation criteria [4,14,15]. In two studies of expert practitioners, this practice was found throughout the problem solving process. We review these studies and their implications next.

Working with Variations in Practice

In a study of website design practices, Newman and Landay [21] found professional designers create a number of alternative solutions as part of an explicit, planned process in the initial problem solving stages. The variations produced were largely developed using informal tools and representations, such as hand-drawn pencil and paper sketches, and constituted "rough ideas" lacking specific details. The practices observed echo those found in

numerous other studies of design practices (e.g., see [14,15]).

A study conducted by Terry and Mynatt [27] analyzed the practices of expert users of an image manipulation application and also found users develop variations as part of the problem solving process. These variations were produced across a range of problem domains, including user interface design, image restoration, and artistic endeavors. However, in contrast to the type of variations reported in the website design study, the alternative solutions produced in this study differed in origin, size, and granularity. While the website designers produced rough variations as part of a planned process, this study found evidence of their production *throughout* the problem solving process, most often in response to difficulties encountered in implementation. Furthermore, these variations arose as users worked with, and *operated upon*, precise data representations, as opposed to informal representations such as sketches. For example, when designing a user interface, variations of button styles would be explored to determine which would work best with the overall theme of the interface. Because these alternatives targeted individual elements of the overall solution, they were substantially smaller in size and granularity than variations produced as part of an initial exploratory phase. In short, users demonstrated a need to vary their *actions and operations* in order to produce variations of *elements* of an overall solution.

Iterations vs. Variations

It is critical to note that solution variations observed in both studies are not the same as *iterations* of a solution. Because iterations and variations play functionally different roles in the problem solving process, we wish to draw a sharper distinction between these two classes of solutions.

For the purposes of this paper, we define *variations* to be a user-designated set of distinct, *alternative solutions* to a given problem at a point in time, while we define *iterations* to be versions of the *same solution* at different points of revision. Variations arise in moments of ambiguity and uncertainty, when the problem solver has inadequate information to choose one solution over another. Their role, then, is to uncover the advantages and disadvantages to a variety of approaches: lessons learned from developing each variation feed back into, and further inform, the problem solving process. For example, in the website design study, designers developed multiple variations to both explore the design space and to learn the client's true needs and desires. In the image manipulation study, variations arose at implementation time, when it was unclear how to best solve a sub-problem.

Iterations, as we define them here, represent different stages of evolution for the same solution. Returning to the website example, once the client's wishes were better understood, a single design was chosen then implemented. Iterations represent instances of that design as it is gradually implemented and revised, and are a natural by-product of developing and revising a single solution.

While we make a distinction between these two different types of solution instance, we are not claiming they are mutually exclusive. For example, a previous iteration can simultaneously be considered an iteration and an alternative solution by a user. The critical difference is the *role* the solution instance plays for the user in the problem solving process at a specific point in time.

The differences between variations and iterations was noticed in the website design study, and led to separate recommendations for interface support structures: tools that help users create and "manage different *variations* of design ideas" (italics theirs), and tools that track the history and evolution of the overall solution. Building on these separate recommendations, we examine current interface support for variations and show aspects currently unaddressed by existing tools.

Iteration vs. Variation: Holes in Interface Support

Current mechanisms and methods for working with multiple solution instances (both variations and iterations) can be broadly divided into the following categories:

1. Prototyping and sketching tools that encourage the rapid production of rough solution ideas (e.g., [9,13,16])
2. Tools that track and/or save snapshots of the entire document. These include history tracking tools external to the application (e.g., version control systems such as CVS), enhanced history mechanisms strongly integrated with the user interface (e.g., [1,5,6,11,12]), and basic operations that allow users to save copies of their document ("Save As…")
3. Ad-hoc user strategies of embedding alternatives directly within the document. For example, creating a new layer for each variation of a graphical element in an image manipulation application, or selectively commenting out sections of code when developing software
4. Enhanced preview mechanisms and "what-if" tools (e.g., [10,19,28]) that provide a broader view of potential future states

While each of these tools or methods can be used in the service of exploring variations, the image manipulation study suggests a number of ways they can be improved. In particular, existing tools: do not allow users to designate variations when modifying their data; lack strong cues about relationships between variations; and cannot simultaneously manipulate all variations at once. We explain each limitation in turn.

During implementation, practitioners may unexpectedly encounter interesting alternatives as they manipulate their

data. For example, when interacting with a command's dialog box, several different parameter settings may appear promising. When these alternatives are discovered, users may wish to actively pursue them to see if they result in a better overall solution. Users may also need to pursue alternatives when they have difficulty solving a particular problem.

Current computer interfaces make it difficult to instantiate and follow multiple threads as users actively modify data. Enhanced previewing tools such as Side Views [28] or Design Galleries [22] provide a broad overview of possibilities (with some capability to view several steps ahead), but the previews are transient – users must ultimately commit to only one modification of their data. Traditional dialog boxes share this same limitation. As a result, users must manage the task of instantiating multiple variations before invoking a command (e.g., taking a snapshot of the data before operating on it), or after invoking a command (e.g., backtracking to a previous state to pursue an alternative). While this process does not prevent users from exploring variations, it does force them to repeatedly reestablish the context that provided the initial source of inspiration.

History tracking tools facilitate the tracking of individual variations as they are made, but do so by storing separate, self-contained snapshots for each variation. Storing variations as separate snapshots has the effect of reducing the immediacy of the variations from the work environment: each variation must be loaded into its own separate window, creating competition for attention and screen space. Furthermore, semantic relationships between variations can be lost as the number of snapshots increases. For example, after exploring a number of alternatives, users may want to work with only a handful of the variations. However, while history tracking tools encode the lineage of a variation, they provide few facilities to indicate semantically meaningful relationships between the stored data. As a result, the most noteworthy solution instances may be scattered amongst intermediate stages of little interest.

Storing variations as distinct snapshots also has a more subtle effect: it forces users to choose which snapshot to use for continued development of other portions of the solution. That is, users may need to further revise other elements of the overall solution independent of the content that varies from alternative to alternative. However, since interfaces do not allow users to directly modify multiple variations simultaneously, users must either manually replicate any future revisions, or merge the differences. Notably, this problem does *not* exist when variations are embedded within the same document: Since there is only one document instance, users do not need to concern themselves with synchronizing changes to other content in the same document.

In summary, current tools provide a number of critical services for managing multiple solution instances, but they lack facilities for efficiently working with variations later in the design process: users cannot designate alternatives at the point of manipulating the data; semantic relationships between variations are not recognized by the interface; and variations cannot be simultaneously manipulated. To address these issues, we combine some of the advantages and affordances of the existing tools to synthesize an augmented model of interaction that further raise the status of a variation as a first-class object in the interface.

PARALLEL PATHS MODEL OF INTERACTION

Parallel Paths is an enhanced model of interaction that improves upon existing models of interaction by providing more explicit support for generating, manipulating, managing, and comparing solution variations. While current interfaces require commitment to only a single set of parameters when applying a command, Parallel Paths relaxes this requirement by allowing users to add *all* interesting results discovered. Each alternative added is embedded directly within the same workspace to increase accessibility and facilitate comparisons. Furthermore, users can operate on the variations simultaneously or individually, as necessary. Together, these enhancements create an environment that promotes and scaffolds deeper, more sustained explorations of alternatives, especially when interacting with data representations and commands that leave no room for ambiguity or indecision.

IMPLEMENTING PARALLEL PATHS

To illustrate the application of these principles, we describe the construction an image manipulation application enhanced with the concepts of Parallel Paths.

Generating and Designating Variations

There are three conceptually different situations in which a user may discover it is necessary to explore alternatives:

1. Before a command is invoked. In this situation, the user realizes that the current state of the problem will necessitate exploring a number of alternatives
2. While interacting with the command. At this point, users may discover a number of interesting variations, or be unable to find one that perfectly fits the problem
3. After a command has been applied. In this case, the results obtained are not as hoped, though not without value. This realization may come immediately after invoking a command, or several steps later, when it becomes clearer that earlier actions must be refined

To support the generation of variations under these three conditions, users must be able to duplicate a document, create a variation while interacting with a command, and revisit past states of a document without losing the current state.

Document Duplication

In our application, users can duplicate the current document by invoking the "Create New Variation" command from a pull-down menu. The document, including its history, is copied and embedded within the same workspace. Visualization of the multiple variations is handled by the Parallel Pies tool, described below.

Adding Variations within Commands

To support the creation of variations while modifying data, users can insert the currently previewed result as a new variation to the workspace. Command dialog boxes now feature an "Add Variation" command (Figure 2) that duplicates the document, applies the command with its current settings, and inserts the result in the workspace.

Exploration of variations while manipulating data is further enhanced in our application by incorporating Side Views [28], an enhanced previewing mechanism that automatically generates sets of previews for each command parameter. Adding Parallel Paths to this tool facilitates more sustained exploration by enabling users to selectively add any interesting results to the workspace.

Duplicating Lineages and Skating Through Time

To support the creation of variations after a command has been applied, each variation maintains a history of all prior states and commands leading to its current state. When users duplicate a particular variation, its lineage is also duplicated. The copied history enables users to create variations after applying a command: when a result is not what was anticipated, but still worth keeping, users can duplicate the current state then non-destructively return to a previous state through a function we call *skating*.

Skating allows users to traverse the timeline of a variation without needing to invoke "undo." This additional functionality addresses a number of issues. First, undo is often overloaded in its use, taking on many responsibilities beyond simply correcting errors. For example, users may create a variation, then wish to return to a previous state to explore an alternative path. Using undo to navigate a history for this purpose is risky, since an alternative may be accidentally lost when undoing to a previous state. Skating reduces this risk by offering an independent operation for navigating history.

Second, skating helps reduce potential ambiguity surrounding the interpretation of undo when working with multiple variations in the same workspace. For example, should "undo" undo the last action performed in the whole application, or the last action applied to a particular variation? These problems are similar to those encountered in the design of Flatland, a whiteboard application that hosts self-contained workspaces that can also interact with one another [5]. To avoid ambiguity, skating explicitly fulfills the role of retrieving past states.

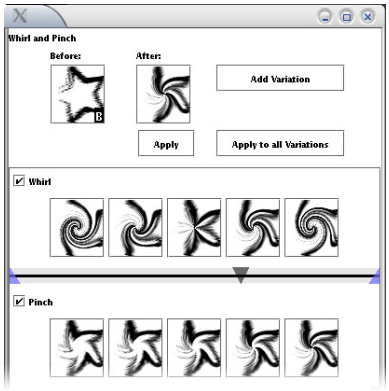

Figure 2. Command dialog boxes are augmented with the capability to add the current result as a new variation (top right button), or to apply the command to all variations simultaneously (bottom bight button).

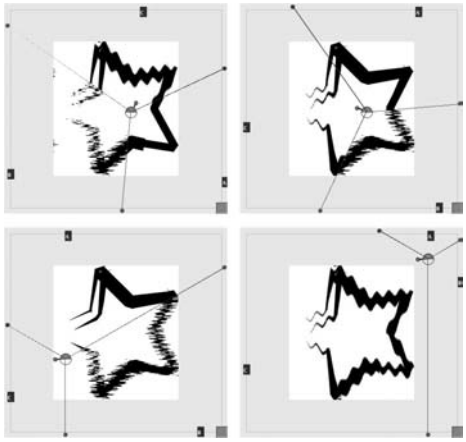

Figure 3. The Parallel Pies tool can be rotated and moved to selectively show all or part of a variation. A paddle extending from the hub controls the angle of rotation.

Parallel Pies: Supporting Direct Embedding and Comparison of Multiple Variations in a Workspace

One of the advantages of the ad-hoc practice of manually embedding variations within the document itself (for example, two versions of a paragraph, one after the other) is that the variations are ready-at-hand: there is no intermediate layer required to load or save them, they are highly accessible, and they lend themselves to basic comparisons. As previously discussed, they also allow other parts of the solution to be manipulated independent of the variations: changes do not need to be "merged" or duplicated as they would be if separate document instances were created for each variation in a history tracking system.

Building on this concept, our interface embeds the variations directly within the same workspace and offers Parallel Pies to selectively show "slices" of each variation (Figures 1, 3). Parallel Pies divides the document by creating wedges that radiate outwards from a central hub, with each variation mapped to a single wedge. The hub can

be repositioned and rotated to reveal different areas of the individual variations (Figure 3). A "gutter" surrounding the image provides space for the hub to be dragged off to the side; when pulled into the gutter, the increased wedge size affords a larger view of a variation, allowing users to focus on only one at a time. Notably, using other schemes to divide the space (such as a grid), would not have this same affordance.

Parallel Pies also acts as the mechanism by which users select a variation when they need to modify only one at a time. Users "select" a variation by allocating it the most screen space (for example, by pulling the hub to the side). The choice of this mechanism was driven through user testing that revealed that users expected their actions would be applicable to only the most visible variation. A more detailed description of this issue is detailed in the evaluation section.

The visualization provided by Parallel Pies works particularly well when the differences between images are relatively minor. When variations are more distinct, the ability to reposition the hub in the gutter helps reduce visual clutter and confusion by showing only one variation at a time.

Selective Manipulation of One or More Variations

Users can choose to modify a single variation, or all variations at once. An "Apply" button in a command's dialog box (Figure 2) affects the most visible variation without dismissing the window. Users can modify all variations at once by pressing the "Apply to All Variations" button.

Distinguishing Between Variations

Moving to a model of interaction that allows and recognizes multiple potential states at one time necessitates a few other changes to the interface. Most obviously, it requires additional feedback to the user so they know what variations they have created, and which they will effect when interacting with the command. Accordingly, we provide a number of cues throughout the interface.

A thumbnail-based summary of all variations is placed on the right side of the window (Figure 1). As an additional cue, we decorate variations with tags throughout the interface to help users differentiate between them when they are visually similar. Small black boxes with a unique letter are positioned over the variations' thumbnails on the side of the window, in the "before" view of a preview, and on the edge of variations' slices in the workspace.

EVALUATING PARALLEL PATHS AND PARALLEL PIES

To evaluate the concepts of Parallel Paths, we performed think-aloud sessions with three expert users of image manipulation applications. These sessions lasted approximately one hour each. For each, we demonstrated the application's new features, then asked the user to explore the interface while working on an image. No specific goal was given for users to accomplish. Two types of qualitative data were gleaned from this evaluation: usability information and the appropriateness of this interaction model for enhancing workflow.

In general, the users responded favorably to the new additions. Users commented that the overall feature set would be particularly well suited to tasks such as image toning and print work, which often require developing a number of variations for an image before finding one that prints well. One user suggested a use of the system we had not previously considered, namely, cel animation. Individual cels of an animation share similarities to variations: each is unique, though much of the content is shared from cel to cel. Using our system would allow users to more easily compare differences between adjacent cels, and would also allow them to apply changes to many cels at the same time.

While most users understood the new model of interaction, a number of issues were uncovered in earlier designs that increased the time required to fully understand and use the new capabilities. These centered on the need to be able to distinguish between variations. In our initial design, we did not explicitly label the variations, leading to a problem of correspondence within the interface: users could not always match variations between the thumbnail view, the document, and the command's previews. This led us to add unique tags to all representations of the variations in the interface.

Earlier designs also employed alternative mechanisms for choosing which variation to manipulate. We briefly describe the evolution of this selection mechanism and the lessons learned in its development.

In our initial design, users clicked on pie slices to select which slice(s) to modify. While this approach had the advantage that users could directly click on the item to manipulate it, it also had the consequence of adding an extra layer of selections: users could now select variations as well as individual objects in the document.

The next mechanism we implemented allowed users to select the variation from within the command's dialog box. Buttons in the shape of arrows below the "before" preview let users cycle between variations. However, users did not readily understand the buttons' functionality, nor could they easily discover this method of choosing which variation to manipulate.

Our current implementation builds on a behavior that emerged through testing, namely moving the pie's hub to the side to concentrate on one variation at a time. When the interface showed only one variation at a time, users expected that commands would only affect that variation, since others were not visible. Therefore, we adopted this convention and modified commands to update their previews accordingly.

RELATED WORK

In his work developing the subjunctive interface [17,18], Aran Lunzer has similarly identified the need to be able to explore multiple paths simultaneously. The subjunctive interface answers this need by allowing users to select multiple values of interest for a command's parameters, then execute the command consecutively for each variation. For example, in a cannon simulation [18], users can select multiple values for objects' properties (such as the angle of the cannon), then run the simulation to watch all possibilities execute simultaneously. The concepts of Parallel Paths builds on many of the concepts put forth by the subjunctive interface, while providing a set of concrete user interface mechanisms and conventions to support the generation, manipulation, and comparison of variations.

A number of systems have facilitated the process of exploring variations by reducing the work required to track and store those variations. Enhanced history mechanisms, such as those found in [5,6,11,12], automatically record the evolution of a user's work through a tight integration with the work environment. Similar systems have been developed with an eye towards supporting collaborative editing of documents (for example, Timewarp [6]).

Other history-tracking tools put less emphasis on tracking every intermediate state, and instead focus on making it easy to selectively store salient versions of the document as it is developed. For example, a bookmark facility attached to a 3D modeling tool [29] provides a thumbnail browser of past states, while Adobe Photoshop's [1] "snapshot" feature allows users to save and reload different instances of the document. External version control systems such as CVS also provide capabilities to track the evolution of a document, but are more loosely coupled with most interfaces.

Each of the history tools described above provides valuable services when working through a complex problem, but are built with the assumption that only one variation can be open and manipulated at a time. Parallel Paths extends these concepts by allowing users to simultaneously interact and operate on the variations.

In addition to history *tracking* tools, a number of systems have been constructed that allow *editing* of a history. For example, Graphical Editable Histories [12] allows users to modify previous commands, with changes propagating down the timeline. These tools are especially well suited for revising past decisions, and would nicely compliment Parallel Paths' ability to generate variations of solutions.

See-Through Tools, such as Magic Lenses [2], and enhanced previewing mechanisms (e.g., [10,22,28]) afford rapid exploration of possibilities, but do not allow the user to easily spin off interesting results into new variations.

Several commercial applications include tools that afford side-by-side comparisons of two alternatives within the workspace. These tools offer "split-screen" previews that divide a document to show both unmodified and modified content. PixelNance [3] provides split-screen previews with a visualization tool that can be operated similarly to Parallel Pies: a movable, rotating divider line allows users to selectively view portions of the original and modified image while adjusting a command's parameters. Sonic Foundry's Vegas [26] also provides simultaneous viewing of the original and previewed result in the same document, but allows users to arbitrarily define a rectangular preview region. While these tools share a common visualization mechanism with Parallel Pies, the function of the tools differ: The existing tools focus on offering transient previews, while Parallel Pies allow users to view and manipulate a set of *persistent* alternatives.

Finally, in the domain of image manipulation, applications such as the GIMP [7] or Photoshop [1] provide the ability to assign individual graphical elements to separate layers. The image manipulation study [27] found layers used as an ad-hoc storage facility for variations. While this strategy lets users store variations, it lacks the benefits of a set of tools dedicated to generating, manipulating, and comparing variations. For example, it does not address the need to create new variations while interactively manipulating data.

FUTURE DIRECTIONS

We have argued for the need to more easily generate, manipulate, and compare separate, viable solution alternatives when working with precise data formats. To that effect, we have proposed augmenting user interfaces with capabilities to designate, and therefore accumulate, variations before, during, and after manipulating data. We have demonstrated these principles in an image manipulation application augmented with Parallel Pies, a tool suited to working with multiple variations in the same workspace.

The ultimate goal of this research is to design and develop computational tools that allow users to more easily experiment when solving ill-defined problems. This initial effort has focused on the domain of image manipulation and its highly graphical, visually-based tasks. While the tools developed have been matched to this problem domain, we believe there is opportunity to apply the basic principles to other domains, such as writing or software development. Our experience in building these tools, as well as some of the authors' experiences building tools to support the writing process [19], provide some insight into how this transfer may occur. In particular, it seems clear that strict adherence to the WYSIWYG style of presenting data is not necessary, and may in fact be counter to facilitating exploration. This argument has been made before, particularly with respect to the need for sketch-like tools (e.g., [13]), but we feel it applies equally well to interfaces devoted entirely to manipulating precise data representations, whether the data are images, text, or some other format. The challenge moving forward is to understand how these principles generalize across task

domains so a user's document can always retain a feeling of plasticity, even when containing highly precise data formats.

REFERENCES

1. Adobe Photoshop. http://www.adobe.com
2. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., and DeRose, T.D. Toolglass and Magic Lenses: The see-through interface. In *Computer Graphics*, 27 (Annual Conference Series), 1993, 73-80.
3. Caffeine Software, Inc. http://www.caffeinesoft.com
4. Cross, N. Design cognition: Results from protocol and other empirical studies of design activity. Chapter 5 in *Design Knowing and Learning: Cognition in Design Education*. Eastman, C., McCracken, M., and Newstetter, M. (eds.). Elsevier Science, 2001, 79-103.
5. Edwards, W.K., Igarashi, T., LaMarca, A., and Mynatt, E.D. A temporal model for multi-level undo and redo. In *Proceedings of UIST 2000*, 31-40.
6. Edwards, W.K., and Mynatt, E.D. Timewarp: Techniques for autonomous collaboration. In *Conference Proceedings on Human Factors in Computing Systems* (CHI 1997), 218-225.
7. The GNU Image Manipulation Program (GIMP). http://www.gimp.org
8. Goel, V., and Pirolli, P. The structure of design problem spaces. *Cognitive Science*, 16(3), 1992, 395-429.
9. Groos, M.D. and Do, E.Y.-L. Ambiguous Intentions. In *Proceedings, ACM Symposium on User Interface Software and Technology* (UIST '96), 183-192.
10. Jankun-Kelly, T.J., and Ma, K.L. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics*, 7(3), 2001, 275-287.
11. Klemmer, S.R., Thomsen, M., Phelps-Goodman, E., Lee, R., and Landay, J.A. Where do web sites come from? Capturing and interacting with design history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI 2002), 1-8.
12. 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, 257-275.
13. Landay, J., and Myers, B. Sketching Interfaces: Toward More Human Interface Design. In *IEEE Computer*, 34(3), March 2001, 56-64
14. Lawson, B. *Design in Mind*. Architecture Press. 1997.
15. Lawson, B. *How Designers Think: The Design Process Demystified*. Architectural Press. 1997.
16. Lin, J., Newman, M.W., Hong, J.I., and Landay, J.A. Denim: Finding a tighter fit between tools and practice for web site design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI 2000), 510-517.
17. Lunzer, A. Choice and comparison where the user wants them: Subjunctive interfaces for computer-supported exploration. In *Proceedings of IFIP TC, 13 International Conference on Human-Computer Interaction* (INTERACT '99), 474-472.
18. Lunzer, A. Towards the subjunctive interface: General support for parameter exploration by overlaying alternative application states. In *Late Breaking Hot Topics, IEEE Visualization 1998*, 45-48.
19. Nakakoji, K., Yamamoto, Y., Reeves, B.N., and Takada, S. Two-Dimensional Positioning as a Means for Reflection in Design. *Design of Interactive Systems* (DIS 2000), 145-154.
20. Newell, A., and Simon, H.A. *Human Problem Solving*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.
21. Newman, M., & Landay, J. Sitemaps, Storyboards, and Specifications: A Sketch of Website Design Practice. In *Proceedings of Designing Interactive Systems* (DIS 2000), 263-274
22. Marks, J., Andalman, B., Beardsley, P. A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pifster, H., Ruml, W., Ryall, K., Seims, J., and Shieber, S. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 1997, 389-400.
23. Reitman, W.R. *Cognition and Thought*. John Wiley & Sons, Inc., 1965.
24. Rittel, H.W.J., and Webber, M.M. Planning Problems are Wicked Problems. Chapter in *Developments in Design Methodology*, 1984, 135-144.
25. Simon, H. The structure of ill-structured problems. *Artificial Intelligence*, 4:181-203, 1973.
26. Sonic Foundry. http://www.sonicfoundry.com
27. Terry, M. and Mynatt, E.D. Recognizing creative needs in user interface design. In *Proceedings of the Fourth Conference on Creativity & Cognition*, 2002, 38-44.
28. Terry, M. and Mynatt, E.D. Side Views: Persistent, on-demand previews for open-ended tasks. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (UIST 2002), 71-80.
29. Verlinden, J.C., Igarashi, T., and Vergeest, J.S.M. Snapshots and Bookmarks as a Graphical Design History. In *Proceedings of International Design Conference 2002*, Dubrovnik, Kroatia, 567-572.