MACE: A New Interface for Comparing and Editing of Multiple Alternative Documents for Generative Design

Loutfouz Zaman
University of Ontario Institute of Technology
Oshawa, Canada
loutfouz.zaman@uoit.ca

Wolfgang Stuerzlinger
Simon Fraser University
Vancouver, Canada
w.s@sfu.ca

Christian Neugebauer
University of Applied Sciences Bonn-Rhein-Sieg, Germany
christian@neugemail.de

ABSTRACT

We present a new interface for interactive comparisons of more than two alternative documents in the context of a generative design system that uses generative data-flow networks defined via directed acyclic graphs. To better show differences between such networks, we emphasize added, deleted, (un)changed nodes and edges. We emphasize differences in the output as well as parameters using highlighting and enable post-hoc merging of the state of a parameter across a selected set of alternatives. To minimize visual clutter, we introduce new difference visualizations for selected nodes and alternatives using additive and subtractive encodings, which improve readability and keep visual clutter low. We analyzed similarities in networks from a set of alternative designs produced by architecture students and found that the number of similarities outweighs the differences, which motivates use of subtractive encoding. We ran a user study to evaluate the two main proposed difference visualization encodings and found that they are equally effective.

CCS CONCEPTS

• **Human-centered computing → Graphical user interfaces.**

KEYWORDS

Alternatives; generative design; exploration; parallel editing; difference visualization.

ACM Reference format:

L. Zaman, W. Stuerzlinger, and C. Neugebauer. 2017. MACE: A New Interface for Comparing and Editing of Multiple Alternative Documents for Generative Design. In *Proceedings of DocEng '17, Valletta, Malta, September 04-07, 2017*, 10 pages.
https://doi.org/10.1145/3103010.3103013

1 INTRODUCTION

Parametric and generative design is a modern design technology in which a set of rules or an algorithm generates the output, such as an image, sound, architectural model, or animation. In parametric design, typically only values are changed. Generative systems can produce much more varied output and uses either a set of rules, a data-flow program, or even an algorithm as the underlying generative model to ensure that the generated output matches the goals, as specified as part of the model. Current examples include *Max/MSP* (cycling74.com), *NodeBox* (nodebox.net), *Grasshopper 3D* (grasshopper3d.com) and *Houdini* (sidefx.com). With parametric and generative technologies, it is possible to explore a much larger number of viable design options compared to what is possible with manual operations. However, how users can compare alternatives in such a large space of design options has not been investigated in detail.

Since designers routinely generate dozens of alternatives based on a single idea [36], basic techniques for comparing them, such as juxtaposition, synchronized zooming and panning, and uniform layouts, are not enough. This is especially true of environments that employ data-flow languages. Designers may want to superimpose their newer creations over earlier ones to see how their newer versions deviate from the previous version. They may want to become immediately aware how the parameters of subjunctive nodes differ between alternatives, or even want to see the differences in the networks' structures.

In this paper, we present *MACE* (Multiple Alternatives – Comparison and Editing), a novel user interface that facilitates the comparison of multiple alternatives in generative design which also includes new mechanisms that facilitate interactive editing in a difference visualization mode. We implemented these techniques as an extension of *GEM-NI* [36]. See Figure 1.

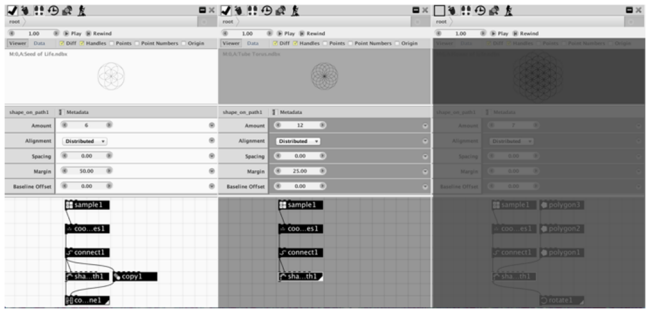

Figure 1. The "Flower of Life" worked example in *GEM-NI*.

GEM-NI [36] enables the user to quickly generate sets of alternative solutions through a multitude of techniques, edit them in parallel with undo capabilities, and supports merging of alternatives. Moreover, *GEM-NI* also provides GUI mechanisms to manage the set of alternatives. The work presented here

DocEng '17, September 04-07, 2017, Valletta, Malta

ACM ISBN 978-1-4503-4689-4/17/09…$15.00
http://dx.doi.org/10.1145/3103010.3103013

extends *GEM-NI* with new difference visualization techniques and with techniques that enable editing in difference visualization mode.

To support such comparisons better, we implemented the following explicit difference visualization techniques:

- a new interactive difference visualization that simultaneously compares more than one DAG (directed acyclic graph) against a given reference;
- two types of encoding: subtractive and additive to show/hide common elements for better difference readability and scalability;
- two levels of abstraction for subtractive encoding: alternative and node-focused difference visualization for the selected alternative or node to give the user the control over visual clutter;
- visualizing differences in parameters with the ability to synchronize the parameter state.

2 RELATED WORK

2.1 Generative Design

Generative design enables designers to create endless design variations based on a model. This model can vary just in parameters, be expressed as varying networks of computational nodes and/or constraints, or even full algorithms across alternatives. By structuring design concepts as models, it is possible to explore a much larger number of viable design options compared to what is manually possible. Generative modeling is a fast method for exploring design possibilities and is used in various design fields such as art, architecture, and product design. Then, the central role of the designer involves continuously modifying the generative model based on the resultant outcomes. Through this the designer navigates the solutions space. A very simple approach is to just have the user repeatedly select attractive solutions to zero in on desirable options [24]. Better approaches give the user more control.

To compensate for the lack of adequate software features to support easy and efficient exploration of a design space, current designers rely on strategies, referred to as idioms of use [32]. Some common idioms for comparing design alternatives are opening different file "versions" in different windows, or using layers [31]. Yet, current computational design tools still do not support the comparison of alternative solutions in an adequate manner, nor do comparison modes permit interactive editing. Moreover, such adaptations sometimes create more problems than they solve, e.g., when the file naming and window management overhead becomes large.

The user interface of *MACE* aids generative design through easy comparison of more than two alternatives through enhanced difference visualizations and by permitting editing in the difference visualization mode.

2.2 Difference and History Visualization for Graphs

History visualization, versioning, differencing and version merging graphs, trees and node-link diagrams are all related to the area of dynamic graph drawing, which deals with the problem of visualizing data that evolves linearly over time. Most of the work cited below targets generic undirected graphs and trees. Yet, generative models use directed edges to represent the data flow. Almost all the approaches below reduce the problem to showing only *pair-wise differences* between graphs or trees.

A difference map is a graph that encodes all of the differences between the node and edge sets between two graphs [3]. Such maps produce fewer errors when determining the number of edges inserted or removed from a graph evolving over time and were also preferred [5]. *MACE* employs a new variant of a difference map, which excludes nodes and edges common to the compared graphs.

Animation can also be used to compare graphs, e.g., [6, 29, 35]. Zaman et al. [35] demonstrated that animation was beneficial for graphs with non-matching layouts. This is not applicable for *MACE*, as *GEM-NI* synchronizes node positions across alternatives. Animation is better suited for showing gradual transitions, i.e., successive graphs that represent an evolving change of a single data set [13]. In our application domain, we deal with situations well beyond the evolution of a single graph. The closest relevant work compares multiple graphs at the same time. It is important to highlight that in generative design non-linear creation and editing of alternatives is the norm. In other words, generative design typically is ill described by a linear time flow. Thus, time/history-based difference visualization techniques such as animation and small multiples are not directly applicable to the comparison of alternative designs. *MACE* employs new graph difference techniques to illustrate the changes between alternatives.

Graham et al. [12] surveyed multi-tree visualization. They distinguish five methods of comparing nodes in two trees: edge drawing, coloring, animation, matrix representation, and agglomeration. Gleicher et al.'s [11] work offered a taxonomy of visual designs for comparison, which groups designs into three basic categories. They identify that all visual designs are assembled from the building blocks of juxtaposition, superposition and explicit encodings. Alper et al. [1] evaluated two techniques for weighted graph comparison and found that matrix representations are more effective than node-link diagrams. As our work involves directed and unweighted graphs, such techniques are not applicable. Layering is commonly used in diagram differencing and merging, e.g., [8, 34, 35] and superimposes multiple graphs. Yet, it can only handle a very small number of graphs simultaneously. Thus, it is most useful for showing pair-wise differences.

Side-by-side views are a special case of graph difference visualization, which show only two versions. *DualNet* [27] visualizes sub-networks of node-link diagrams with side-by-side views. All work discussed below in this paragraph focuses only on trees, rather than graphs. Thus, most of these methods are not directly applicable to our context. *TreeJuxtaposer* [26] targets the comparison of large trees with side-by-side views. *TreeVersity* [15–17, 19] shows changes in topology and node values. The system uses glyphs that pre-attentively highlight changes and also highlights created and removed nodes.

TreeVersity2 [18, 19] enables the exploration of changes in trees over time. Guerra-Gomez et al. [19] identified and classified five types of tree comparisons. Except for the comparison of topological differences between two trees, none of the types of comparisons identified there apply to the DAGs used in *GEM-NI*. Instead of topological differences *MACE* identifies structural differences between two or more DAGs. Moreover, Guerra-Gomez et al.'s node types do not correspond to our context, as nodes in *GEM-NI* do not have categorical attributes, can contain data unsuitable for interpolation.

Another static visualization approach uses agglomeration. Graham and Kennedy [13] presented a DAG visualization to interact with a set of multiple classification trees to identify overlaps and differences between groups of trees and individual trees with up to six classifications. *Zoomology* [20] compares two classification datasets where two trees are merged into a single overview. Isenberg et al. [21] presented a new system that facilitates hierarchical data comparison in co-located collaborative environment using structural comparison through overlay. Their system dealt with up to six trees. *CandidTree* [22] merges two trees into one and visualizes location and sub-tree structure structural uncertainty. Yet, agglomeration is not applicable to (generative design) networks, as neither individual nodes nor sub-networks can be combined meaningfully into a hierarchy in DAGs.

Difference visualization is also used in software engineering, usually for UML diagrams. Förtsch et al. [9] presented a survey on solutions for differencing and merging of software diagrams and listed requirements for UML diagram versioning tools. Ohst et al. [28] introduced an approach that highlights common and specific parts of two diagrams. Zaman et al. [34] later demonstrated a system where graphs are displayed side-by-side with differences marked. Girschick [10] introduced a similar system, where eight colors were used to visualize eight different types of changes in class diagrams. A user survey for the Pounamu system [25] found positive feedback for their difference visualization, the support for incremental changes, merging, and the overall support for diagram-based design activities. Most of the approaches for UML diagram differencing are applicable to other forms of node-link diagrams and graphs. A single unified diagram for graph comparison was studied by Dadgari et al. [8]. They evaluated multiple graph differencing and merging techniques qualitatively and found that a translucent layer approach was preferred for simple pair-wise comparisons. A unified graph approach was also proposed by Andrews et al. [2]. A side-by-side approach for graph differencing was also investigated by Zaman et al. [35] in the *DARLS* system. *DARLS* displays two versions of a diagram side-by-side with differences marked, even if a node was moved.

Most directly related to our difference visualizations in *MACE* is Shireen et al.'s [30] conceptual prototype of a user interface for parallel work with design alternatives, which included a difference visualization. However, they only showed nodes that are common.

In summary, most previous work addresses the problem of showing pair-wise differences. Animation and small multiples were used for visual comparisons of more than two DAGs. Yet, this is only effective for content that has evolved linearly. None of these approaches can handle the visual comparison of a dozen or more different DAGs that have evolved non-linearly or in parallel.

3 MACE

Here, we present novel extensions of the original *GEM-NI* system [36]. to enable comparison of multiple alternatives.

The user can switch between the normal (viewing) mode, where typical interaction occurs for creating and editing alternatives, and the "*diff mode*", where *differences* of all alternatives are displayed against a chosen *reference*. The user can use any alternative as the reference. To get into this mode, one alternative must be designated by the user as the reference through a GUI button, clicking on a menu button, or a key short cut. The other alternatives (further referred to as *compared alternatives*) are then compared to it. In diff mode, differences are visualized in all three views (output, parameter and network), as applicable. The *diff mode* is fully interactive. For visualizing differences between the networks, we use two different approaches: subtractive and additive encoding.

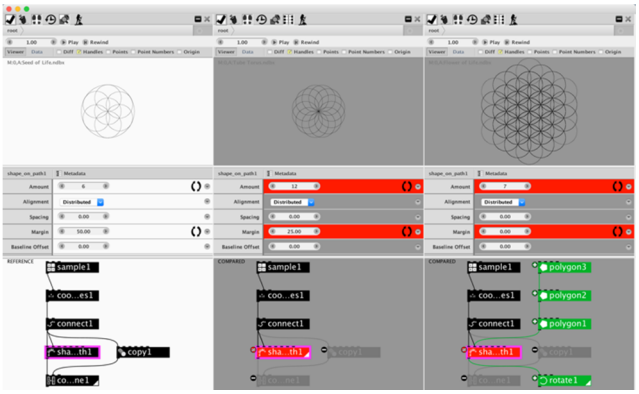

Figure 2. Difference visualizations of the "Flower of Life" example using additive layering.

3.1 Additive Layering

Additive layering is a direct adaptation of the layering technique from previous work [35] to multi-network comparisons. The word "layering" highlights the fact that this is a form of superposition. The word "additive" refers to additive encoding, which was identified by Gleicher et al. [11]. Figure 2 demonstrates this technique. The leftmost alternative was chosen as reference. The nodes SAMPLE1, COORDINATES1 and CONNECT1 are unchanged between the three alternatives. The differences start to appear with SHAPE_ON_PATH1, which is the next node in these networks. In the center and right alternatives, this node differs from the reference node in two parameters: Amount and Margin. These parameters are highlighted in red to indicate the difference in the parameter view. The state of a parameter can be synchronized between non-idle alternatives by clicking on the synch button in the parameter field. This is an extension of selective merging [36] to a lower level form of merging differences. A "≠" sign is displayed to the top left of the

node to emphasize this further through an inequality metaphor. The node is also highlighted in red. Nodes COPY1 and COMBINE1 with the corresponding connectors are not present in both the center and right alternatives. This is shown using transparency. A "–" sign in the top left of these nodes, which emphasizes this further through a reduction/subtraction metaphor. New nodes POLIGON1 (2 and 3), ROTATE1 and new connectors, which don't exist in the reference, are highlighted in green. A "+" sign is displayed in the top left of the node to emphasize this through an addition metaphor. The word "reference" or "compared" appears in the top left corner of the network view of each alternative as an additional aid to distinguish between the reference and compared alternatives. Deleted connectors relative to the active compared alternative are also displayed in the reference using transparency. This works for both additive layering and subtractive layering described below. See an example in Figure 8, where four deleted connectors are shown in the reference network relative to the compared active network.

3.2 Subtractive Layering

Our design for subtractive layering is in part based on the subjunctive interface proposed (but not implemented) by Shireen et al. [30]. Unlike in additive layering, common unchanged nodes are not shown in compared alternatives (with one exception, discussed below). See the example in Figure 3. Only nodes with modified states are shown, which are either changed, deleted or new nodes. Connectors to and from unmodified nodes are instead shown on the nodes in the reference as dashed curves, which emphasizes that these connectors cross network boundaries. In addition, unmodified common nodes are highlighted in the reference in yellow. Drawing every single connector would add significant clutter. Due to that, and after trying different variations, we came up with a design that minimizes showing redundant connectors through a simple rule: we draw cross-network connectors only for nodes which cannot be connected to the other nodes within the compared alternative because these other nodes are hidden. E.g., in the center alternative in Figure 3 the COMBINE1 and SHAPE_ON_PATH1 nodes are both shown. Therefore, a connector between them is only drawn in the center alternative as a solid line, and not between COMBINE1 in the center alternative and SHAPE_ON_PATH1 in the reference.

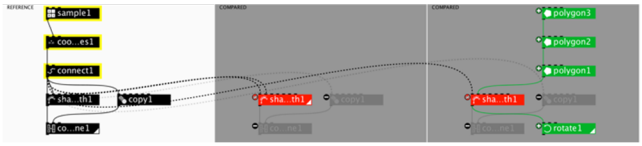

Figure 3. Difference visualizations of the "Flower of Life" example using subtractive layering.

3.2.1 Abstraction of Connectors. To reduce clutter further, we provide two additional mechanisms: alternative and node focused visualizations. In Figure 4a, the user selects the center alternative as active, which hides cross-network connectors to the right alternative. The user then selects SHAPE_ON_PATH1, which hides all the connectors which do not involve this node (Figure 4a).

Figure 4. (a) alternative-focused visualization, (b) node-focused visualization.

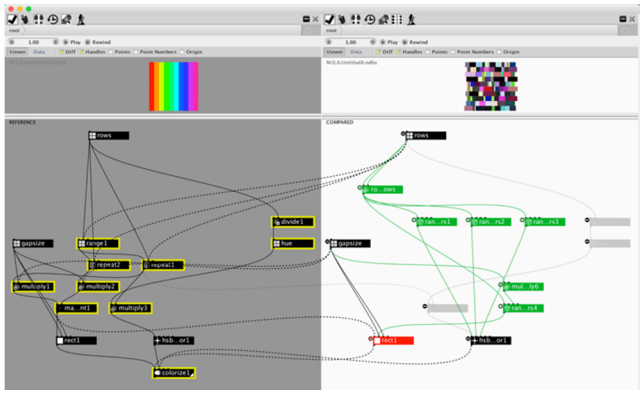

Figure 5. Showing differences in connectors between unchanged nodes.

3.2.2 Showing Differences in Connectors between Unchanged Nodes. We also implemented a technique to illustrate differences in node *connections* that involve one or more unchanged nodes. In this visualization, we reveal unchanged, common nodes that have one or more changed connections. Figure 5 shows an example, where nodes ROWS, GAPSIZE and HSB_COLOR1 are shown in the compared alternative (contrary to the rule of hiding common unchanged nodes in subtractive layering). This is done to highlight that in ROWS one outgoing connector was deleted and two new connectors were added, in GAPSIZE one outgoing connector was added, and in HSB_COLOR1 one incoming connector was deleted and three were added.

3.2.3 Design Motivation and the Research Hypothesis. We employ subtractive encoding—a complementary approach to the additive encoding identified by Gleicher et al. [11]. Subtractive encoding removes *common* nodes from the compared graphs. This technique reveals and/or highlights changed, unchanged, added and removed nodes in the compared network relative to the reference and could reduce visual clutter in the compared network. Thus, if there are no differences between a reference and compared network and there is nothing to show in the compared network, visual clutter is non-existent or minimal. On the other hand, if the overall number of changes is substantial,

then the difference visualization might visually "overwhelm" the content. To characterize this, we propose using the relative percentage of nodes and edges shown in each difference visualization between two alternatives as an approximation to quantify relative difference, D_{rel}:

$$D_{rel} = \frac{n_{\neq} + n_{+} + n_{-} + e_{+} + e_{-}}{n_{ref} + e_{ref}}$$

Here, $n_{\neq}$ is the number of changed, n_{+}—the new and n_{-}—the deleted nodes in the compared network, n_{ref} is the total number of nodes in the reference, e_{ref} is the total number of connectors in the reference, e_{+}—the new and e_{-}—the deleted connectors in the compared network. If there are no changes, $D_{rel} = 0$. If the number of displayed nodes and edges in the compared view is equal to or exceeds the number of nodes in the reference, $D_{rel} \geq 1$.

For subtractive layering to be effective, the average D_{rel} of typical designs must be low. We believe that alternatives for a design problem will likely show substantial similarities due to the *shared goal*. Thus, we expect fewer differences among data-flow networks for alternatives compared to the number of similarities. To test this hypothesis we computed these numbers for the dataset of the second user study conducted on *GEM-NI* [33]. For the outcomes, we performed all pairwise comparisons of all alternatives to the first design. Averaging across all participants yields a low difference of $\mu_{D_{rel}} = .29$, $\sigma_{D_{rel}} = .22$, $N = 20$. Given that the average is substantially closer to zero than one, this suggests that showing differences instead of commonalities is an appropriate design choice for difference visualization across multiple alternatives.

3.2.4 Interactive Editing. We believe it would be trivial to describe how interactive editing works with additive layering. So, we will focus on the subtractive encoding instead. To enable interactive editing with subtractive layering we employ our "reveal-to-edit" feature. The interface mechanics we discuss here are in part based on the subjunctive interface proposed by Shireen et al. [30].

3.2.5 Worked Example. Consider the same design scenario as in the worked example of *GEM-NI* [36]. This time imagine Ann wants to recreate the three designs with subtractive layering enabled. She first creates the design on the left (Figure 1). She then creates a clone of the design (Figure 6a) and enables the diff mode by setting the left design as the reference (Figure 6b). Subtractive encoding hides all visualizations that are common between the original (left) and the clone (right). Therefore, the network view of the clone becomes empty. She then selects SHAPE_ON_PATH1, COPY1 and COMBINE1. By clicking anywhere in the network view of the clone and holding down a modifier, she then reveals these three nodes (Figure 6c). She then sandboxes the clone, changes two parameters of SHAPE_ON_PATH1 ("Amount" and "Margin") and deletes COPY1 and COMBINE1. This is how she arrives at the "Tube Torus" (Figure 6d). She then creates a clone of "Tube Torus" (Figure 7), sandboxes it and completes the final design as described in original worked example [36].

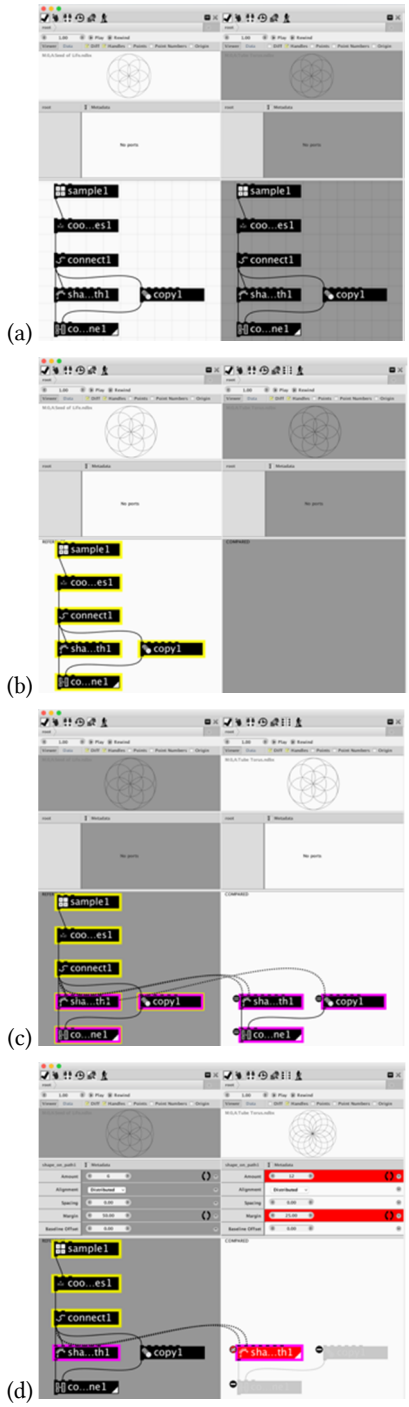

Figure 6. Editing alternatives using interactive difference visualization with subtractive encoding. (a) The original and a clone; (b) subtractive encoding; (c) SHAPE_ON_PATH1, COPY1 and COMBINE1 revealed in the clone for further editing; (d) "Amount" and "Margin" changed in SHAPE_ON_PATH1; COPY1 and COMBINE1 are deleted.

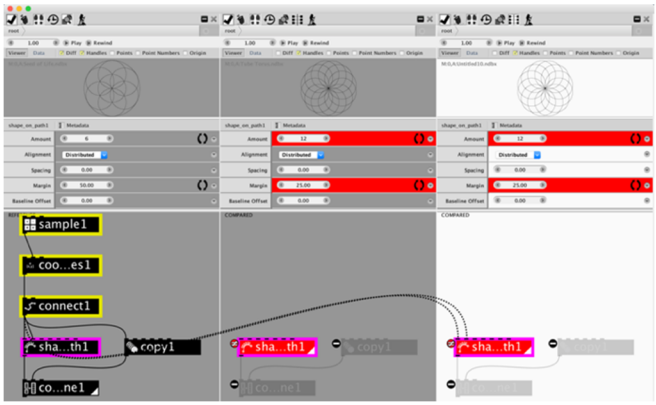

Figure 7. Editing alternatives using interactive difference visualization with subtractive encoding: cloning of a design.

3.3 Showing Differences in the Output View

To illustrate the differences in the output of the generative design, the geometry produced by the rendered node of the reference is displayed transparently in the output view in a bottom layer for all compared alternatives. This directly superimposes the geometry of compared alternatives over the reference to enable simple visual comparisons. Figure 8 demonstrates an example where the design in the compared view superimposes over the reference view. To deal with cases where this is visually too intrusive, we provide an option to disable this functionality by unchecking the corresponding "diff" checkbox. Figure 8 shows an example with subtractive layering, however, the technique works in combination with both subtractive and additive layering.

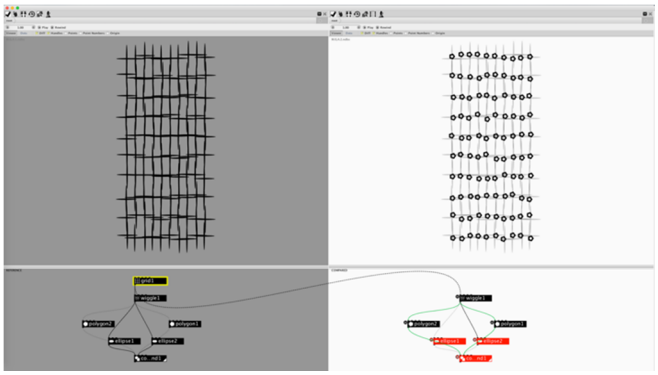

Figure 8. Demonstration of difference visualization in the output view and in connectors in the reference networks view.

4 USER STUDY

In our previous user studies we evaluated how *GEM-NI* supports creativity. In those studies participants engaged in creative tasks and ranked *GEM-NI* against an unenhanced version of *NodeBox* [33, 36] through a psychometric survey. Experts also ranked participants' designs in terms of quality [33]. In the work presented here, we ran an empirical user study to compare subtractive against additive layering for difference visualization in *MACE*. The goal of the user study was to investigate the hypothesis that subtractive encoding is more effective for visualizing differences of data-flow networks of designs for a common design goal.

4.1 Participants

Twelve participants (one female) were recruited for the study. Participants were between 18 and 48 years old with an average (μ = 28.42) and had on average 17.75 years of experience using a desktop/laptop computer. Three were left handed, but all chose to use the mouse with their right hand for the experiment. Six participants indicated that they were regular users of data comparison and differencing tools such as the Track Changes feature in Microsoft *Word* and the Unix *diff* tool, three were familiar with these tools, but were not using them, two had used them in the past, and one never heard of the concept before. Three participants had experience using either data-flow programming or generative design tools. Ten participants used diagrams in their current line of work, two used them in the past.

4.2 Apparatus

The user study was conducted on a MacBook Pro laptop with a USB wheel mouse and a 27" external 2560×1440 display.

4.3 Experimental Design

Half of the participants evaluated the additive layering first, while the other half evaluated the subtractive layering first. We used a repeated measures design with one between factor (techniques order: subtractive layering first vs. additive layering first) and one within factor (differencing technique: subtractive layering vs. additive layering).

The dependent variables were trial completion time and the error rate (measured as the number of attempts).

4.4 Procedure

Participants were presented with 15 sets (tasks) of alternatives one after another in the fixed order. Previous work [35] had identified that the presentation order does not seem to affect the outcome of this type of experiment substantially. In each of these sets, the alternative highlighted in white was compared to the reference alternative, which for simplicity was always chosen to be the leftmost alternative in every set. All the alternatives starting from the second one were compared to the reference. The participants had to identify the changes for each of these comparisons.

Our procedure employed a hybrid approach based on the idea of using a dialog [4] for gathering responses. But unlike Archambault et al.'s work, in our user study the participants had to identify all changes in the networks until success or a timeout occurred, similar to other previous work [35]. Unlike that work [35], participants had to specify only the number of occurrences for each type of change, rather than explicitly selecting all changes. See Figure 9. After the participant successfully identified all the changes of each category (new nodes, new connectors, deleted nodes, deleted connectors and changed nodes), had entered their numbers in the dialog and clicked

"Validate", or after a timeout occurred, the next task was presented using the next alternative to the right. When the participant completed all the comparisons, the next set was then pre-loaded. Participants could use the mouse to select the numbers from the list menus, or to use the number and the <tab> key (for switching to the next field), whichever they felt more comfortable with. The combinations of the two input methods were also allowed. We also demonstrated how to zoom in and out, pan and reset the network view. This was necessary in tasks with many (5+) alternatives as they appeared too small for easy identification.

This procedure was repeated for both difference visualization techniques. For the subtractive technique, the dialog did not display the dashed connector image to avoid confusion. All the comparisons were made in the alternative-focused connector abstraction mode.

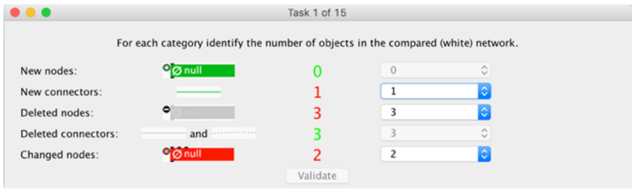

Figure 9. Task dialog state after an unsuccessful attempt.

4.5 Tasks

We picked 15 sets of generative designs with varying number of alternatives as tasks for this user study. The first set was the "Flower of Life" example with three alternatives (Figure 1), which was used as a practice set where each participant tested the two techniques during the initial exposure. The results obtained from this set were discarded. Sets 2-8 were picked from the dataset of second user study that we conducted on *GEM-NI* [33]. Sets 9-10 were created by a graduate student at Simon Fraser University. Sets 11-15 were picked from a generative design book [7].

Excluding the first two trials from the "Flower of Life" example that were discarded, a total of 38 comparisons with each technique remained. On average, each set contained 3.73 alternatives, with a maximum of 8 and minimum of 2. On average the number of nodes in the reference was 9.53, with 10.4 connectors. In the compared alternative, the average number of nodes was 10.49, with 11.12 connectors. On average the number of new nodes was 1.05, new connectors – 2.12, deleted nodes – 0.97, deleted connectors – 1.54, and changed nodes – 2.05.

4.6 Pilot Study

We recruited four participants for a pilot. This allowed us to fix the user interface issues and to improve the reliability of the collected data. Notably, we made changes to the dialog to prevent participants from re-submitting their response without making changes in all the fields where they made mistakes. We learned that it was essential to prevent participants from modifying responses which were entered correctly in previous attempts. Moreover, we identified that the task took a bit too long and was too challenging – networks where participants had to identify more than 20 new/changed/deleted nodes were particularly frustrating regardless of the technique. Thus, we modified one of the networks by grouping nodes to reduce the number of visible changes. In two tasks, we could not sensibly cluster the nodes, and so we removed them. In the pilot we confirmed that a timeout of 2 minutes was appropriate, which also agrees with previous work [35]. One participant stated that grid lines were making the task too confusing because it was difficult to tell them apart from deleted connectors, and so we disabled them. With these changes, we then performed the main user study.

4.7 Results

4.7.1 Trial Completion Time. The main effect of technique was not significant, $F_{1,10} = 3.45$, $p > .05$. On average, additive layering took 28.5s and subtractive 26.2s. We did not observe a main ordering effect on time, $F_{1,10} = 1.03$, $p > .05$. However, the interaction between order and technique was significant, $F_{1,10} = 51.96$, $p < .0001$ (power = 0.99 at $\alpha = 0.05$). A Tukey-Kramer analysis on the interaction between order and technique revealed among others that when additive layering was evaluated first, additive layering was much slower (35.4 s) than subtractive layering when subtractive layering was evaluated first (28.5 s). See Figure 10.

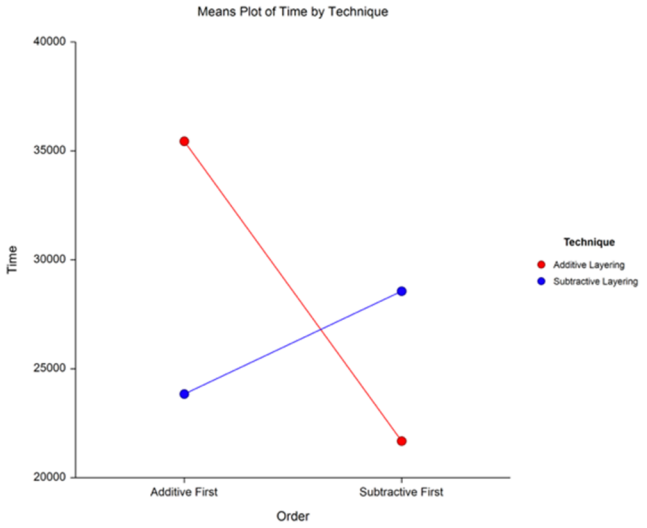

Figure 10. Interaction between order and technique.

4.7.2 Error Rate. The main effect of technique was not significant, $F_{1,10} = 2.25$, $p > .05$. On average, 0.143 errors were found with additive layering, 0.12 – with subtractive layering. The main ordering effect on time was not observed, $F_{1,10} = 0.65$, ns. The interaction between order and technique was significant, $F_{1,10} = 5.37$, $p < .05$ (power = 0.55 at $\alpha = 0.05$). However, a Tukey-Kramer analysis on the interaction between order and technique failed to reveal this difference.

4.7.3 Correlations with Difference. Using the proposed measure for the pairwise difference (Section 3.2.3) we found that the average difference of the set used in this study was $\mu_{D_{rel}} = .59$, $\sigma_{D_{rel}} = 1.06$, $N = 44$. A correlation analysis revealed a very weak correlation between difference and trial completion

time (0.2) and even lower value between difference and error rate (0.1). We then looked at the absolute difference, D_{abs}:

$$D_{abs} = n_{\neq} + n_{+} + n_{-} + e_{+} + e_{-}$$

We found that the correlation between D_{abs} and trial completion time was 0.69, and correlation between D_{abs} and error was rate 0.52.

4.8 Feedback from Participants

Participants were asked to rank each of the two differencing techniques on a Likert scale from 1 to 10 (10 being the best). The results are summarized in Figure 11. The rankings are consistent with our findings. 7 of 12 participants ranked the technique they did last higher than the technique they did first and three ranked them equally. Out of the two participants who ranked the first technique higher, one wrote in the freeform feedback that the subtractive layering technique was easier because he did it second, although he believes the additive technique is better. Two participants who ranked subtractive layering higher stated that they believe there is less clutter with this technique. One participant believed that additive layering is better because it appeared to him that there were fewer connectors to count. Another participant wished that one could toggle between the number of compared alternatives presented at a time and wished connectors also had labels like nodes. Yet another participant expressed that the only thing that appeared different to him was the presence of connectors across the networks in the subtractive layering. One participant left more informative feedback. He said that grey connectors are hard to see and that he would add a squiggle when they overlapped to tell them apart. He also stated that hovering on a node and animating its connectors would help and that having a legend telling how many there are for each node would be better.

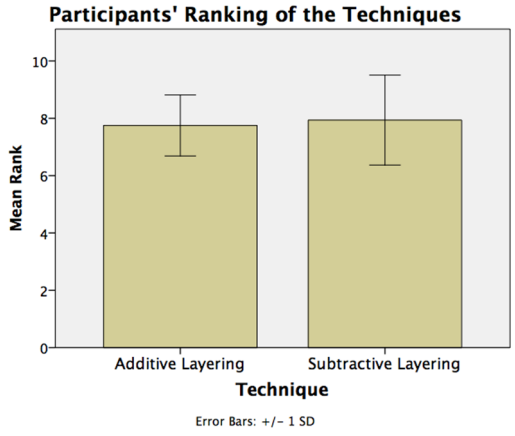

Figure 11. Participants' ranking of the differencing techniques. Error Bars: ±1 SD.

5 DISCUSSION

The problem of showing differences is important for generative design because designers typically create several alternatives based on a single idea, which they use as a reference. Designers also tend to work on multiple design alternatives concurrently [23]. Then management of these alternatives becomes an issue, which requires also the ability to visualize differences between alternatives. We presented a solution to difference visualization for alternative graphs: added, deleted, (un)changed nodes in both the reference and all compared graphs as well as added and deleted connections.

Pairwise difference visualization for directed graphs with nodes identifiable by name is not new, e.g., [35]. Comparing more than two such graphs of the evolution of a *single* data set can be done with animation and/or time slices, e.g., [29]. Yet, in generative design, where parallel and non-linear creation and editing of alternatives is the norm, we deal with situations well beyond the evolution of a single graph. Animation is therefore not applicable. Agglomeration is also not applicable because we are dealing with DAGs, not trees. Thus, we proposed using subtractive and additive encoding. Subtractive encoding is a new form of explicit encoding where members of the intersection are removed from the compared graph. This extends Gleicher et al.'s taxonomy work [11]. Subtractive layering extends a) the layering approach used in several instances of previous work, such as [2, 8, 35], by hiding unchanged nodes to reduce clutter and drawing connectors to the reference to enhance juxtaposition; b) side-by-side views [14, 26, 27] by going beyond pair-wise comparisons; c) existing work on subjunctive interfaces [30], *TreeVersity* [15–17, 19] and *TreeVersity2* [18, 19] to show a larger variety of difference visualizations. Both of our approaches also show difference visualizations of multiple group nodes. This supports scalability to generative networks with many nodes, where node grouping becomes a necessity as otherwise the graph becomes much too large for a single screen.

Subtractive encoding is as an extension to Shireen et al.'s [30] concept for parallel creation and editing of alternatives. In their prototype Shireen et al. included an option to create an empty design model. The user then selects a node in the original design and inserts it in the subjunctive graph at which point a new instance of the design is created for editing. In contrast, our interface requires that a user creates a clone (a branch from an earlier state) to start editing a new alternative. Subjunctive nodes are revealed as common nodes with the original. We did this to make the interface for editing in difference visualization mode transparent and minimally intrusive within *GEM-NI*. In contrast to Shireen et al.'s work [30], *GEM-NI* pushes changes always to all the non-idle alternatives. *GEM-NI* does not distinguish between the original or an alternative when pushing changes. Any design can change the role from the original to an alternative by pressing a GUI button. In Shireen et al.'s conceptual prototype [30] users can substitute or replace certain features of a design by creating new structures in the relevant subjunctive graphs, connecting them to the prototype, and removing the substituted nodes from the alternative. Our interface enhances this further by also displaying nodes and connectors that were removed from the original as we designed the interface to serve as a difference visualization interface as well. This way all the differences between the original and the alternatives can be visualized at once. To reduce visual clutter, Shireen et al.'s conceptual prototype [30] emphasizes nodes and

connectors related to the currently selected alternative and de-emphasizes everything else. We implemented a variant of this in *MACE*. If the user sets the active document to any document other than the original, all the links connecting other alternatives to the original are removed, but when the user sets the original alternative as active then everything is shown. Furthermore, selecting the active node hides other connectors in that alternative. If needed, the user can disable difference visualizations in *GEM-NI* by disabling *MACE*. This effectively enables the user to focus on a single alternative, similar to the design of Shireen et al. subjunctive dependency graphs [30].

MACE also introduces "reveal-to-edit", a new way to interact with hidden, common nodes. The "dragging" solution proposed by Shireen et al. [30] is less user friendly and does not match standard GUI conventions.

Given that alternatives for a design problem will likely be similar due to the shared goal, we expect fewer differences among the data-flow networks of alternatives compared to the number of similarities. This assumption underlies the design of subtractive layering. We performed an analysis on the alternatives obtained through a user study to test this assumption. Using the introduced difference measure, based on measuring the visual similarity of the alternative networks, we computed a low degree of difference, which confirmed the appropriateness of the design choice of the technique. The correlation analysis between relative difference, trial completion time and error rate did not reveal a strong correlation. This is not too surprising. The following example clarifies this. The largest value for D_{rel} in the study set was 6. The corresponding pair is shown in Figure 12. Arguably, despite the large D_{rel}, identifying these differences should not be difficult since there are not that many nodes and edges in total. While this measure helped us to motivate our work, it's D_{abs} that relates to the actual task measures.

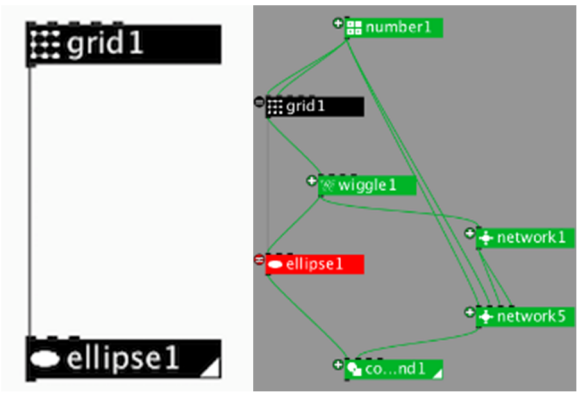

Figure 12. An example where $D_{rel} = 6$.

Our hypothesis was that subtractive layering improves the readability of our difference encoding by keeping visual clutter low because the designs share a common goal. Contrary to our intuition, the findings and participants' feedback reject this hypothesis. We showed that this is true for networks with an average number of 10 nodes and connectors. Still, these findings are not that surprising. Previous work [35] demonstrated that there is no significant difference between difference visualization techniques for static diagrams and our results complement those findings. Although not significantly so, the layering technique in this previous work was found most efficient. Thus, in our current user study, we were essentially comparing subtractive layering to the best-known technique.

Nevertheless, the interaction between order and technique is an interesting finding. Additive layering seems initially far more challenging to learn than subtractive layering – when additive layering was presented first it took 24% time longer to complete the task. One possible explanation could be that the larger amount of node clutter with additive layering forced participants to check for changes in connections between unchanged nodes. We observed this issue during the user study on multiple occasions, yet did not have a mechanism in place to log this. Given this finding, subtractive encoding could be a better choice for difference visualizations for novice users.

Our evaluation has limitations. We do not account for situations when a participant falsely identifies a change of one type and fails to identify another change of the same type. Then, the total number of changes will add up to the correct number but the identified changes will be incorrect. Also, participants cannot be prevented from guessing the number of changes. Another issue is that the task also involves dealing with inputting the responses through the interface of the dialog, rather than purely identifying changes, which means that there is also a learning curve for the dialog, which may not relate to the rate at which participants identify the changes. We were not able to directly adopt the approach used in *DARLS* [35], due to the fact that in *NodeBox* connectors cannot be selected directly.

Finally, we believe that the ideas behind our difference visualizations generalize to other visual programming environments, including 3D modeling (e.g., *Grasshopper 3D*, *Houdini*). For other types of media, such as video or audio (e.g., *Max/MSP*), a different approach to illustrating differences may be necessary.

6 CONCLUSION AND FUTURE WORK

We presented a new interface for interactive difference visualizations for generative design alternatives. The new techniques enable comparison of more than two alternatives at a time and enable differencing for the output, parameter and network views. The techniques are interactive and enable creating and editing alternatives through cloning of the original design and using the "reveal-to-edit" feature. The interface also allows post-hoc merging of the state of a parameter across a set of alternatives. For the network view, we investigated two difference visualization approaches: subtractive and additive layering. We introduced new ways to emphasize added, deleted, (un)changed nodes, and connectors. Subtractive layering is based on the idea of subtractive encoding, which we hypothesized as being useful in data that has more similarities than the differences, such as the alternatives we are dealing with. We used a difference measure to confirm the appropriateness of the approach for at least one alternatives dataset produced by designers. We evaluated subtractive against additive layering in

a repeated measures experiment to investigate the hypothesis. The results were not confirmatory, but our findings suggest that subtractive layering may be better suited for novice users.

In the future, we will investigate the scalability of our techniques for even larger numbers of alternatives and investigate other approaches to difference visualizations. In our work we only focused on the difference visualizations and not editing. Our future user studies will evaluate the editing technique in *MACE*. We are also considering a direct comparison with *DARLS*. Additionally, we will explore difference visualizations for visual programming environments in other domains.

ACKNOWLEDGMENTS

We gratefully acknowledge financial support from the NSERC Discovery program and the GRAND NCE. We would also like to thank Robert Woodbury, Maher Elkhaldi and Naghmi Shireen for their feedback on an early version of the difference visualizations in *MACE*.

REFERENCES

[1] Alper, B., Bach, B., Henry Riche, N., Isenberg, T. and Fekete, J.-D. 2013. Weighted Graph Comparison Techniques for Brain Connectivity Analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), 483–492.

[2] Andrews, K., Wohlfahrt, M. and Wurzinger, G. 2009. Visual Graph Comparison. *Information Visualisation, 2009 13th International Conference* (Jul. 2009), 62–67.

[3] Archambault, D. 2009. Structural differences between two graphs through hierarchies. *Graphics Interface 2009* (Kelowna, British Columbia, Canada, 2009), 87–94.

[4] Archambault, D., Purchase, H. and Pinaud, B. 2011. Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics*. 17, 4 (Apr. 2011), 539–552.

[5] Archambault, D., Purchase, H.C. and Pinaud, B. 2011. Difference Map Readability for Dynamic Graphs. *Graph Drawing*. U. Brandes and S. Cornelsen, eds. Springer Berlin Heidelberg. 50–61.

[6] Bach, B., Pietriga, E. and Fekete, J.-D. 2014. GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics*. 20, 5 (May 2014), 740–754.

[7] Bohnacker, H. 2012. *Generative Design: Visualize, Program, and Create with Processing*. Princeton Architectural Press.

[8] Dadgari, D. and Stuerzlinger, W. 2010. Novel User Interfaces for Diagram Versioning and Differencing. *British HCI* (2010).

[9] Förtsch, S. and Westfechtel, B. 2007. Differencing and Merging of Software Diagrams - State of the Art and Challenges. *ICSOFT (SE)* (2007), 90–99.

[10] Girschick, M. 2006. *Difference Detection and Visualization in UML Class Diagrams*. TU Darmstadt.

[11] Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C.D. and Roberts, J.C. 2011. Visual comparison for information visualization. *Information Visualization*. 10, 4 (Oct. 2011), 289–309.

[12] Graham, M. and Kennedy, J. 2010. A survey of multiple tree visualisation. *Information Visualization*. 9, 4 (Dec. 2010), 235–252.

[13] Graham, M. and Kennedy, J. 2007. Exploring multiple trees through DAG representations. *IEEE transactions on visualization and computer graphics*. 13, 6 (Dec. 2007), 1294–1301.

[14] Guerra Gómez, J.A. *Exploring Differences in Multivariate Datasets Using Hierarchies: An Interactive Information Visualization Approach*. University of Maryland.

[15] Guerra-Gómez, J.A., Buck-coleman, A., Pack, M.L., Plaisant, C. and Shneiderman, B. 2013. TreeVersity: Interactive Visualizations for Comparing Hierarchical Datasets. *Transportation Research Record (TRR), Journal of the Transportation Research Board (2013)*. (2013), 21.

[16] Guerra-Gómez, J.A., Buck-Coleman, A., Plaisant, C. and Shneiderman, B. 2011. TreeVersity: Comparing tree structures by topology and node's attributes differences. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct. 2011), 275–276.

[17] Guerra-Gómez, J.A., Buck-coleman, A., Plaisant, C. and Shneiderman, B. 2012. TreeVersity: Visualizing Hierarchal Data for Value with Topology Changes. *Proceedings of the Digital Research Society 2012*. 2, (Jul. 2012), 640–653.

[18] Guerra-Gómez, J.A., Pack, M.L., Plaisant, C. and Shneiderman, B. 2013. Discovering temporal changes in hierarchical transportation data: Visual analytic & text reporting tools. *Transportation Research Part C: Emerging Technologies*. 51, (2013), 167–179.

[19] Guerra-Gómez, J.A., Pack, M.L., Plaisant, C. and Shneiderman, B. 2013. Visualizing Change over Time Using Dynamic Hierarchies: TreeVersity2 and the StemView. *IEEE Transactions on Visualization and Computer Graphics*. 19, 12 (2013), 2566–2575.

[20] Hong, J.Y., D'Andries, J., Richman, M. and Westfall, M. 2003. Zoomology: ComparingTwo Large Hierarchical Trees. *Poster at Compendium of InfoVis 2003*. (2003), 120–121.

[21] Isenberg, P. and Carpendale, S. 2007. Interactive Tree Comparison for Co-located Collaborative Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 13, 6 (Nov. 2007), 1232–1239.

[22] Lee, B., Robertson, G.G., Czerwinski, M. and Parr, C.S. 2007. CandidTree: Visualizing Structural Uncertainty in Similar Hierarchies. *Human-Computer Interaction – INTERACT 2007*. C. Baranauskas, P. Palanque, J. Abascal, and S.D.J. Barbosa, eds. Springer Berlin Heidelberg. 250–263.

[23] Lunzer, A. and Hornbæk, K. 2008. Subjunctive Interfaces: Extending Applications to Support Parallel Setup, Viewing and Control of Alternative Scenarios. *ACM TOCHI*. 14, 4 (Jan. 2008), 17:1–17:44.

[24] 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. and Shieber, S. 1997. Design galleries: a general approach to setting parameters for computer graphics and animation. *SIGGRAPH '97* (New York, NY, USA, 1997), 389–400.

[25] Mehra, A., Grundy, J. and Hosking, J. 2005. A generic approach to supporting diagram differencing and merging for collaborative design. *ASE 2005* (Long Beach, CA, USA, 2005), 204–213.

[26] Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L. and Zhou, Y. 2003. TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. *SIGGRAPH 2003*. 22, 3 (2003), 453–462.

[27] Namata, G.M., Staats, B., Getoor, L. and Shneiderman, B. 2007. A dual-view approach to interactive network visualization. *CIKM 2007* (Lisbon, Portugal, 2007), 939–942.

[28] Ohst, D., Welle, M. and Kelter, U. 2003. Differences between versions of UML diagrams. *ACM SIGSOFT Software Engineering Notes*. 28, 5 (Sep. 2003), 227–236.

[29] Rufiange, S. and McGuffin, M.J. 2013. DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation. *IEEE Transactions on Visualization and Computer Graphics*. 19, 12 (Dec. 2013), 2556–2565.

[30] Shireen, N., Erhan, H., Botta, D. and Woodbury, R. 2012. Parallel development of parametric design models using subjunctive dependency graphs. *ACADIA 2012* (San Francisco, CA, USA, Oct. 2012), 57–66.

[31] Terry, M., Mynatt, E.D., Nakakoji, K. and Yamamoto, Y. 2004. Variation in element and action: supporting simultaneous development of alternative solutions. *CHI 2004* (New York, NY, USA, 2004), 711–718.

[32] Woodbury, R. 2010. *Elements of Parametric Design*. Routledge.

[33] Zaman, L. 2015. *User Interfaces and Difference Visualizations for Alternatives*. York University.

[34] Zaman, L., Kalra, A. and Stuerzlinger, W. 2011. DARLS: differencing and merging diagrams using dual view, animation, re-layout, layers and a storyboard. *CHI 2011 Extended Abstracts* (Vancouver, BC, Canada, 2011), 1657–1662.

[35] Zaman, L., Kalra, A. and Stuerzlinger, W. 2011. The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing. *Graphics Interface 2011* (St. John's, Newfoundland, Canada, 2011), 183–190.

[36] Zaman, L., Stuerzlinger, W., Neugebauer, C., Woodbury, R., Maher, E., Shireen, N. and Terry, M. 2015. GEM-NI: A System For Creating and Managing Alternatives In Generative Design. *CHI 2015* (Seoul, Korea, 2015).