The Effect of Animation, Dual View, Difference Layers and Relative Re-Layout in Hierarchical Diagram Differencing

Loutfouz Zaman
York University, Toronto, Canada

Ashish Kalra
NIT Kurukshetra, India

Wolfgang Stuerzlinger
York University, Toronto, Canada

ABSTRACT

We present a new system for visualizing and merging differences in diagrams that uses animation, dual views, a storyboard, relative re-layout, and layering. We ran two user studies investigating the benefits of the system. The first user study compared pairs of hierarchical diagrams with matching node positions. The results underscore that naïve dual-view visualization is undesirable. On the positive side, participants particularly liked the dual-view with difference layer technique. The second user study focused on diagrams with partially varying node positions and difference visualization and animation. We found evidence that both techniques are beneficial, and that the combination was preferred.

KEYWORDS: Design, Human Factors

INDEX TERMS: H.5.1 Multimedia Info. Systems: [Animation]; H.5.2 User Interfaces: [Graphical user interface (GUI)]

1 INTRODUCTION

Computer-supported version differencing and merging of text documents has been used at least since the introduction of the Unix diff tool [14]. Modern version control tools for text are much more user-friendly by incorporating visual interfaces that facilitate differencing and merging. One example is the use of highlighting. Another, more recent one, is the use of animation [7].

A number of algorithms and interaction techniques have been proposed for effective dynamic graph visualization. Recently, user studies were conducted to evaluate these [3, 4]. However, these user studies focused on generic graphs where attribute values associated with nodes or edges are irrelevant. Only a small fraction of research addresses diagrams where nodes in the graph are identified by name, see Purchase *et al.* [26]. Also, dynamic graph visualization research primarily targets differencing alone, and to our knowledge, no previous quantitative research exists on visualizations that support merging of diagram versions.

To address these shortcomings, we introduce a system for differencing and merging diagrams that makes use of Dual View, Animation, Re-Layout, Layers and a Storyboard, abbreviated DARLS. The system is targeted at diagrams with node and edge attributes. Such diagrams are used frequently in architecture, design, information and concept visualization, and in software engineering, i.e. *software documents* such as UML diagrams. For example, the system can be used to track the evolution of class dependency diagram over releases, a particular course in the prerequisite visualization, or to visualize the evolution of any diagram in general. It also can be used to merge versions of a diagram and to perform selective undo.

2 RELATED WORK

Dynamic graph drawing is a well-researched area within the field of visualization. It deals with the problem of visualizing a graph that evolves over time and, therefore, it is directly related to our work. The concepts of *mental map*, *difference map*, *small multiple* and *animation* are thus related to our work. Also, our works builds on side-by-side views for visual comparison, storyboards for non-linear access, as well as text and UML diagrams versioning.

2.1 Mental Map

Our system uses incremental layout methods for differencing diagrams. Such layouts aim to preserve the user's mental map, which refers to the structural cognitive information a user creates internally when observing the layout of a graph [9]. The mental map facilitates navigation in the graph or comparison of it and other graphs. Purchase *et al.* [26] examined the effect of mental map preservation on dynamic graph readability for directed acyclic graphs drawn in a hierarchical manner. The authors found that the mental map was important for questions that required nodes of the graph to be identified by name, but less important for questions that focus on edges or do not require nodes to be differentiated. Maier and Minas [17] demonstrated that it is meaningful to define incremental layout algorithms for visual languages with both graph-like and non-graph-like features, such as class diagrams. Both these efforts inspired us to make use of relative graph re-layout in our system as we target the same kind of diagrams. For other work on mental maps see e.g. [27, 29].

2.2 Difference Map, Small Multiples and Storyboard

Our proposed layering technique is related to the concept of a difference map in dynamic graph drawing. A difference map presents the union of all nodes and edges in the two graphs for two different timeslices [2, 4]. Using a time slider is a common method for version access. It is used in the Diffamation System for text version differencing [7] and in TimeTree [6] for navigating hierarchies changing over time. However, our system uses a storyboard for this purpose. Su [32, 33] introduced a new interaction metaphor and visualization of the operation history for 2D illustrations. The user has access to the history via graphical depictions at the top of the document. Other approaches to storyboard have been presented, too [16, 19, 21]. In dynamic graph drawing, small multiples display dynamically evolving data via a matrix of images that visualizes the differences between objects. Each image is a timeslice [3]. In our system the storyboard and the dual view can be thought of as small multiples.

2.3 Animation

Today, many visual systems utilize animation to help the user understand transitions. Examples include changes in node-link diagrams and structural relationships [31], perception of statistical data visualizations [13], and dynamically evolving data in graphs, see below. A number of papers support the idea that animation can be beneficial for the purposes of visualization, e.g. [5, 35]. The utility of animation has been questioned by Tversky *et al.* [34], yet it was acknowledged that animation may be an effective way of presenting transitions. Robertson *et al.* [28] compared animation, trace line, and small multiples visualization on multi-dimensional data. Animation was found least effective, whereas small multiples and trace lines were faster than animation, and small multiples were more accurate. Griffen *et al.* [12] suggest that animation can be helpful in discovering space-time clusters.

GI '11, May 25-27 2011, St. John's, NL, Canada

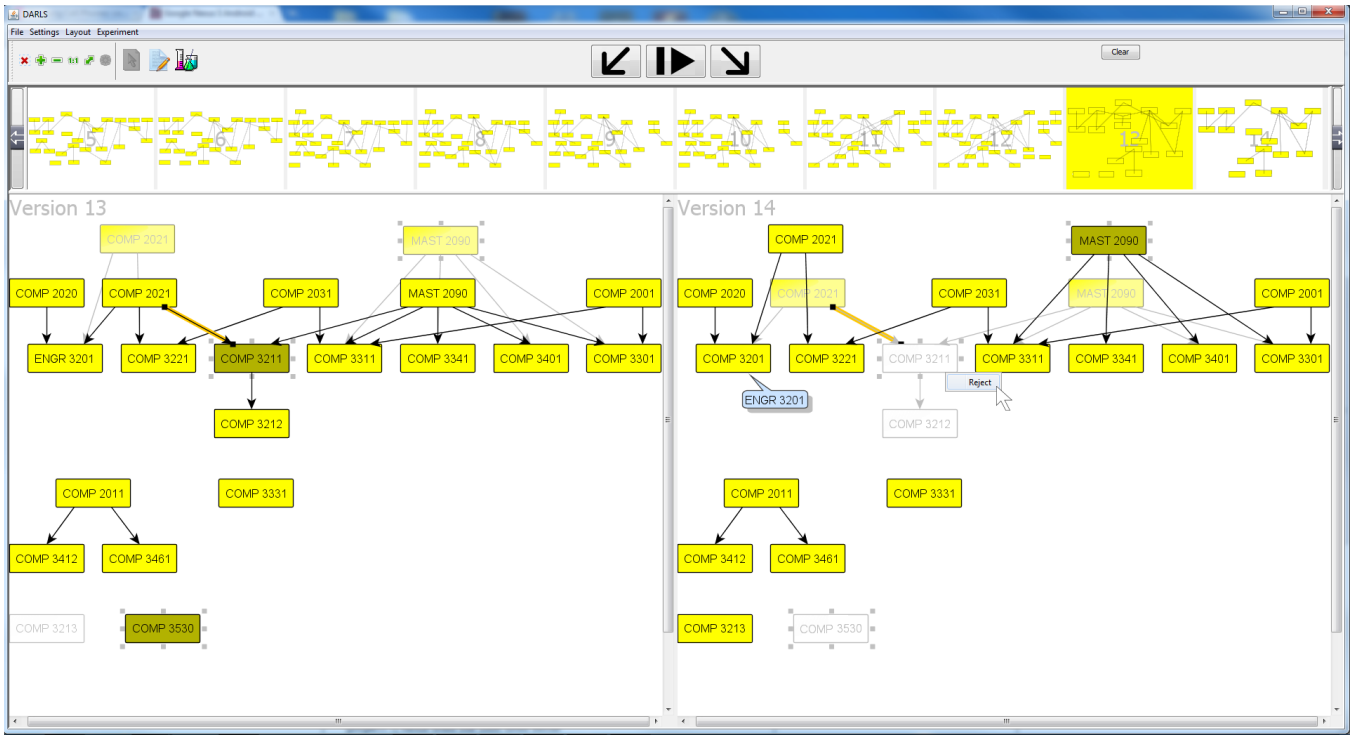

Figure 1. DARLS showing two versions of a diagram, which visualizes course pre-requisites for an undergraduate computer science program. The visualization shows a difference layer and uses the relative optimal re-layout explained in Section 3.5.

Animation has also been used for communicating dynamically evolving data in graphs, which is directly related to our work. We are aware of three user studies that explored the effects of difference maps, small multiples, slide shows, and mental map preservation. Farrugia *et al.* [10] compared animation and small multiples on two dynamic graph series. Small multiples were faster for most tasks. Archambault *et al.* [3] performed a user study where they investigated the effect of animation, small multiples, and mental map preservation for reading graphs that evolve over time. The study found that overall small multiples gave better performance than animation, but animation had fewer errors for some tasks. No effect for preserving the mental map was found, but this study used graphs with unlabeled nodes. The same authors also conducted a study to evaluate the effectiveness of difference maps in comparison to presenting the evolution of a dynamic graph over time in three interfaces (animation, slide show, and small multiples) [4]. Evidence was found that difference maps produced fewer errors when determining the number of edges inserted or removed from a graph as it evolves over time. Also, difference maps were preferred on all tasks.

2.4 Text Versioning and Side-by-Side Views

Side-by-side views have been used for visual comparison of objects long before computers were invented. One popular modern adaptation is a side-by-side view for comparing text documents. There are many publicly available tools, such as: GNU Emacs, Kompare, WinMerge, Araxis Merge, and Scooter Software Beyond Compare. Some of these are not only capable of comparing plain text, but can also deal with XML documents, file directory structures, and even binary files. TreeJuxtaposer [20] is targeted at comparing large trees and uses side-by-side views. However, to our knowledge, there are no publicly available tools for generic diagrams at the moment. A recent study showed that animation facilitates text document comparison [7], and enables users to better identify changes between versions.

2.5 Diagram Differencing and Merging

Förtsch *et al.* [11] presented a survey on differencing and merging of software diagrams and listed requirements for UML diagram versioning tools. One of the main requirements identified is a user-friendly representation. They also point out that it is desirable for diagrams to be displayed side-by-side with differences being marked graphically. If not enough space is available, a unified diagram may be constructed instead. This inspired us to try side-by-side views for diagram versioning. An approach for comparing documents based on a single unified diagram was studied by Dadgari *et al.* [8]. They evaluated multiple graph differencing methods and merging interaction techniques qualitatively with a questionnaire. A translucent view that overlaid the versions was preferred, but no measure of user performance was assessed.

Software engineering research on UML model versioning is extensive, e.g. [1, 15, 22, 36]. However, this work focuses more on theoretical foundations, efficiency, robustness, and correctness. Often, the work is backed up by case studies using evolving software projects. Such work is typically not concerned with user interface issues, a gap that we are trying to address. Visual comparison of UML diagrams is rarely investigated. Ohst *et al.* [23, 24] introduced a unified document approach that highlights common and specific parts of two diagrams. Mehra *et al.* [18] described an approach for visual differentiation. They conducted a user survey and got good feedback on response time, the approach to present changes, the support for incremental changes, merging, and the overall support for diagram-based design activities.

3 The DARLS System

We developed a new system capable of versioning and visualizing differences between diagrams with a number of techniques. Nodes and edges are disambiguated with unique identifiers. The system currently supports differencing and merging of generic and UML

class diagrams. It was implemented in Java using the yFiles Graph Visualization Library, http://yworks.com. To illustrate the user interface, see Figure 1, where we use two versions of a course prerequisite diagram from two subsequent years as an example.

3.1 Accessing Versions and Navigating the Views

The system features side-by-side views of two versions of a diagram. Zooming with the mouse wheel and panning with the scroll bars is synchronized between the two main views. Buttons on the panel allow toggling between the editing and selection modes. Diagram repositories are accessed through the file menu. Both side-by-side graphs can be edited and committed back into the repository. The user can directly access ten versions of the diagram in the scrolling storyboard. Any version can be compared against any other version in a repository. Selecting a version from the storyboard and clicking on the arrow button pointing to the desired view loads a version into that view.

3.2 The Difference Layer

Here, the differences between the two diagrams in the side-by-side views are visualized using a transparent underlay pane in the background of either view, which shows the other diagram. We call this a *difference layer*. This is similar to our previous work on single-view differencing [8], but different from Pounamu [18], which uses also a single merged view, yet where the objects common to both compared diagrams are *not* shown. Our difference layer is also different from difference maps [4], as it displays the common nodes and edges between two versions, even if a node was moved. The rationale is to also enable accept/reject of node movements. A configuration dialog accommodates different color schemes. If the visualization gets too cluttered, the types of objects displayed in the difference layers can be customized, or they can be disabled completely.

By default, all missing nodes and edges for a diagram are shown in neutral transparent gray in the difference layer. See e.g., COMP 3212 in the right view in Figure 1. Nodes that are common to both diagrams but shifted, resized, or morphed are visualized with reduced transparency, e.g. MAST 2090. This implicitly visualizes all differences between the diagrams, as deleted nodes show up semi-transparent in the right diagram and changed nodes are visualized with reduced transparency.

Moreover, if the user selects a node in the right view, the corresponding node in the left view is shown selected as well, with different styles depending if the node exists in the other diagram. The user can customize this, so that either the node on the foreground and the difference layer is selected, or only the node on the foreground of the left view is selected. Nodes in the difference layer in the right view also can be selected by clicking. This is used for version merging see the next section. Also, everything described applies to edges as well.

3.3 Version Merging using Selection

The ability to accept and reject graph edits was previously presented in Pounamu [18]. In our system, a context-sensitive right-click menu provides easier access to this functionality. See the popup next to COMP 3211 in the right view in Figure 1. In our system, a reject operation can undo the creation or deletion of nodes and/or edges, shifting, and morph/resize operations on nodes. For example, if the user "rejects" the change in Figure 1, node COMP 3211 and its adjacent edge connecting to node COMP 2021 will be re-instantiated in version 14. As other nodes are also selected, COMP 3530 will be re-instantiated and MAST 2090 will be shifted down to the same location as in version 13. Figure 5 shows the state of the diagrams after the reject operation.

3.4 Animation and Other Techniques

When the play/pause button is pressed in the top panel the differences between the diagrams in the two views are animated in three phases. First, removed objects fade out, then moved objects are shifted from the old to their new locations with morphed changes in shape and color, and finally new nodes and edges fade in. The sequence and concurrency of these events can be customized. Also the system can highlight new nodes and edges with another distinct color (blue by default), once all animations end, to assist the user in identifying changes. Nodes that changed labels, such as COMP 3201/ENGR 3201, have a call-out added for the change. An additional option gives access to an animation where new nodes and edges blink in a distinct color (red by default), once the first animation ends. Previously, Plaisant *et al.* [25] proposed decomposing animation in their SpaceTree system into three steps: trim, translate, and grow, which is similar to our method. Heer *et al.* [13] demonstrated that staged animations are favored over "all at once" animations for statistical visualizations.

3.5 Relative Graph Re-Layout

As more nodes and edges are added to later versions of a diagram it may get difficult to differentiate and merge different versions, even if the user has access to all provided features, as the layout of the graph may have changed (too) much. Therefore, we added an option to interactively re-layout a diagram relative to another to minimize visual differences between them. We implemented two relative re-layout algorithms: *incremental*, which preserves the locations of nodes, and *optimal*, which rearranges nodes to better use the screen space. Both layout methods keep the positions of nodes and edges common to both diagrams stable and thus preserve the mental map. We based our implementation on the Hierarchic and Incremental Hierarchic Layouters in yFiles and adapted these to our diagram differencing task as follows.

By default, we re-layout the left diagram relative to the right because we assume the diagram in the right is the latest version. The incremental re-layout algorithm first adds all nodes from the left graph that are missing in the right graph, to that right graph to generate a composite graph. It then partitions space into horizontal lanes and fixes the positions of the common and newly added nodes. The remaining nodes are assigned to these lanes so that the number of edges pointing upward is minimized, while keeping the edges short. Then these nodes are arranged within their lanes so that the number of edge crossing is also minimized, and finally, they are arranged to minimize edge bends. Then the layout of the composite graph is copied to the left and right graphs, but only for those nodes and edges that "belong" to the respective graph.

The optimal re-layout algorithm is similar to the one described above but with two differences: nodes are not fixed in place and node and edge placement heuristics can be specified through a menu. Figure 1 demonstrates two diagrams where optimal re-layout was performed. Please note that COMP 2021 and MAST 2090 were manually raised higher after the re-layout.

Currently, there is no propagation effect to keep the layout consistent across versions if merging or other editing occurs.

4 User Studies

We ran two user studies on the new system. Both revolved around diagram differencing using the techniques described above. There were a number of goals for the studies. The primary goal was to investigate the fitness of difference layers for diagram merging. In contrast to previous work [8], we also wanted to *quantify* the user performance of diagram differencing techniques. We also wanted to investigate the incremental and optimal re-layout techniques. Finally, we wanted to confirm the validity of the proposed

requirement by Förtsch *et al*. [11], which states that diagrams should be displayed side-by-side with marked differences.

A secondary consideration was that the study by Archambault *et al.* [3] was performed on graphs with no node or edge titles. Moreover, participants had to answer multiple-choice questions, instead of asking participants to select nodes and edges directly. This effectively removed any visual search time. The authors argued that such questions are preferable as animated nodes may move rapidly, which would disadvantage some layouts. Therefore, and as confirmed by the authors, their results cannot be generalized to tasks that involve named nodes. We wanted to address this limitation, as named nodes are important in many applications. The unenhanced dual view technique in our first user study targets this visual search time issue.

Previous studies also investigated effects globally across *multiple* versions of a graph by displaying everything simultaneously. Our new difference layering technique compares only *two* versions of a diagram. With our incremental layout the locations of nodes remain stable across different versions. Hence, we investigate the effect of presenting the version pairs sequentially in this condition, to see if participants can better trace the evolution of the graph. In the first user study, we also ask participants to select nodes and edges, as we want to observe how the techniques affect the understanding of the variations in layout. Finally, unenhanced side-by-side text differencing is tedious and we wanted to investigate if this is also true for diagrams.

4.1 The Diagrams

In both studies we used versions of a diagram, which depicts the evolution of an anonymized subset of course prerequisites in our Computer Science program over the past two decades. This is a classic real-world application for diagram evolution. We excluded almost all instances without a change in prerequisites, but kept one to serve as a control. In total, we ended up with 12 versions of the diagram. From among these we selected a set of six version pairs, which cover all qualities, such as the magnitude of change in the number of nodes and edges: 1→3, 3→4, 4→5, 5→8, 8→10, 10→12. For brevity, we will refer to them as pairs 1 to 6 from here on. The second pair did not have any changes and is designed as a control. The diagrams appearing in the left view had on average 26.5 nodes (δ=1.52) and 23.66 edges (δ=1.21), and 27.33 nodes (δ=1.21) and 24.17 edges (δ=1.47) in the right. Details of the diagrams along with figures and videos of tasks are available at the paper website *http://sites.google.com/site/thedarlssystem*. As we wanted to focus on the understanding of graph structure, we did not include changes in node titles in the studies.

4.2 Participants

16 paid participants (5 females) were recruited for both studies. The participants were between 18 and 35 years old with an average of 23.82. 4 participants were left handed but all chose to use their right hand for the experiments. 7 participants indicated that they were aware or had previous experience with text, diagram, source code differencing, or versioning tools. One participant used them regularly. None of the participants had previous experience with DARLS. None of the participants were colorblind or had difficulty reading small text. Participants reported an average of 6.1 hours (δ=2.8) of daily mouse use.

All participants performed both studies in counterbalanced order, but due to an implementation issue, the data for the first 4 participants in the second study had to be discarded.

4.3 Apparatus

The user study was conducted using a high-end laptop with a USB wheel mouse and a 22" external display at 1920×1080 in full-screen mode. All events, timings and responses were logged.

4.4 Pilot Study

In a pilot study we asked 4 unpaid participants to select objects that were added in a newer version of the diagram, similar to study I below. The results indicated that the dual-view condition without layering was the slowest overall. Direct change highlighting was the best, but here the task degenerated to selecting all highlighted targets, without requiring any understanding of the diagram evolution. Hence, we removed this condition from User Study I. This may limit ecological validity, as one wants the system to highlight differences. But we are unaware of a good way to avoid this degeneration issue in experiments.

We also observed that when participants didn't read the node labels, certain tasks became unsolvable. A good example is the unenhanced pair 4 in the optimal layout, where the added MAST 1090 node was often confused with the deleted MAST 2090 node due to both nodes appearing at the same level next to each other and having the same number of edges. As the result, some participants could not identify the change. Hence, we instructed participant to carefully pay attention to node labels in the studies.

4.5 User Study I

This user study investigated how different visualization techniques help in understanding the evolution of diagrams with matching layout. At any time two versions of a diagram were shown and participants were asked to select all nodes and edges that were added to the newer version relative to the older one.

4.5.1 Experimental Design

We used a 4x2x6 repeated measures design (4 differencing techniques, 2 layouts, 6 version pairs). The four tested differencing techniques were: single view with animation, single view with toggling between versions, dual view with difference layer, dual view without difference layer. The layouts used were incremental and optimal. In the incremental condition we used the layout as created by the original designer of the diagrams and only re-arranged changes incrementally while keeping the original node positions. As the original layouts were created manually in an incremental fashion, the node positions for any pair matched in sequence. The optimal method re-arranged the whole layout and the settings for that method are summarized on the paper website.

The intent was to compare four distinct differencing techniques in a use case with matching node positions, while also investigating the effect of layout techniques. Especially in the dual view technique with no difference layer, no visual aids are available to participants, and any effect of layout should thus be most prominent in this condition.

4.5.2 Procedure

When the experiment started all layouts for all version pairs were calculated and the zoom level was set so that zooming and/or panning was not necessary. In fact it was disabled to remove a potential confound. This also guaranteed consistent size of nodes and edges across all layouts and diagrams. In the incremental layout condition we presented pairs sequentially to allow participants to trace the evolution since this is complimentary to fixing the node positions. Otherwise, version pairs were presented randomly without replacement to combat learning effects. Technique and layout were also counterbalanced, but we blocked the order of techniques to reduce participant confusion.

In the *single-view animation* condition participants were asked to click on the nodes and edges that were new to the latest version of the two diagrams displayed. Participants could click on an object again to toggle selection. Moreover, a "deselect all" button was available in the top panel. Rectangle and lasso selection methods were not available to limit the effect of different experience and/or selection strategies. For the animation condition new nodes and edges were faded in and the deleted ones faded out automatically upon first display. A re-play of the animation happened whenever the users pressed the <LEFT ARROW> key. Pausing was not provided due to the short animation duration. Selection of nodes and edges was enabled even during the animation. During the pilot study all animations lasted about 2 s, and users found this duration appropriate. In all single-view conditions only the right view was used and nothing was displayed on the left side. At any given time, no more than 12 objects were animated, see the paper website for details.

In the *single-view toggling* condition, holding the <LEFT ARROW> key down switched the right view to display the previous version of the diagram. Just like in the animation condition participants were allowed to toggle between versions as many times as needed. Participants were instructed to select new nodes and edges when the newer version was displayed. The *dual-view without difference layer* condition was basically the dual-view equivalent of single-view toggling. The two versions of the diagram were displayed side-by-side and participants had to select the new nodes in the right view. The *dual-view with difference layer* condition featured a difference layer in both views, which illustrated all differences. Since the positions of common nodes matched due to the relative re-layout, only added and deleted objects were displayed on the difference layer. Moved, i.e. shifted nodes, were excluded. When the participant clicked on any object in the right view, which was visible in the difference layer in the left view, the selection state is also shown on that left view.

Participants were asked to press the <RIGHT ARROW> key when they thought that they were done with the task. If the current state did not match the expected result, the window blinked red and a sound was played. Participants would then have to modify their selection and submit the result again. We logged every such attempt. The submit key was disabled unless at least one change to the selection was made to prevent abuse of this feature. Based on observations from the pilot and if a participant was not able to complete the task within 2 minutes, the right side view would blink in yellow, a different sound would play and the next task would start. If a participant selected everything correctly, the right view would blink in green upon the key press and the next task would start. Whenever there was a change in the differencing technique an appropriate message box would pop up with instructions. Participants were allowed to take a break during that time. Logging only resumed once they clicked the <OK> button.

In the pilot study we found that it is important to inform the participants before the study that there be could a situation when there is no change in the diagram, such as version pair 2, and we informed participants accordingly. We also stressed in the initial training that common nodes always have matching positions.

4.5.3 Results

No ordering effects were observed. For brevity, insignificant results or groupings are reported only selectively below. The main effects of differencing technique, $F_{3,45} = 104.06$, $p < .0001$, and version pairs, $F_{5,75} = 53.15$, $p < .0001$ on task completion time were significant. The layout factor was insignificant, $F_{1,15} = 0.03$, ns. There was also a significant interaction between differencing technique and version pair, $F_{15,225} = 6.93$, $p < .0001$ and layout technique and version pair, $F_{5,75} = 14.90$, $p < .0001$. The interaction between differencing technique and layout was not significant, $F_{3,45} = 1.18$, $p > .05$, thus any hypothesis about the potential effect of layouts was not confirmed.

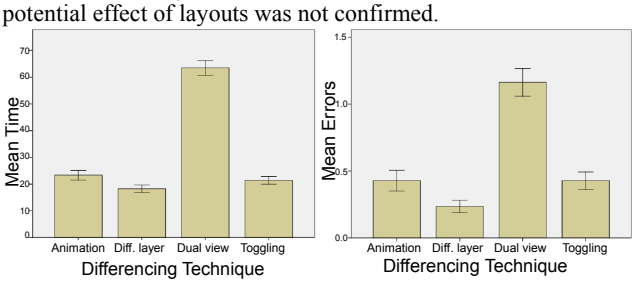

Figure 2. Mean time and errors for the techniques in User Study I. Error bars: ± 1 SE.

A Tukey-Kramer analysis revealed that dual view with no difference layer was significantly slower (average 65.5s) than any of dual view with difference layer (18.3s), toggling (21.4s) and single view with animation (23.3s), see Figure 2 (left). Another Tukey-Kramer analysis was performed to detect version pair groupings. Pair 2 (10.7s), i.e. the unchanged one, was the fastest and different from the group of pairs 3 (21.4s) and 5 (30.3s), which again was different from the group with pairs 1 (50.6s), 4 (50.7s) and 6 (43.7s). Analysis on the interaction between differencing technique and version pairs revealed that the dual view without differences was slowest overall, except for pair 2.

A Tukey-Kramer analysis on the interaction between layout technique and version pair revealed no difference between incremental (6.1s) and optimal (7.2s) layouts on pair 2, the one without differences. However, pair 6 showed a marked interaction effect. Here, incremental layout was significantly slower (40.2s) than the optimal one (30.9s). To investigate this in more detail, we analysed the frequency of false negatives for each of the two layouts. We found that for the incremental layout of pair 6, the top-ranked false negative nodes were: COMP 3403 (64 counts), COMP 3481 (47) and COMP 3214 (45). The same nodes in the optimal layout were also ranked at the top, but with exactly 12 counts each. The top-ranked false negative edges in the incremental layout were: COMP 2031→3215 (40 counts), COMP 3213→ 3481 (32). The same edges in the optimal layout were also top at the top and had counts of 21 and 28 respectively. We did not perform the same analysis on false positive nodes and edges due to insufficient sample size.

We used the number of "submit" attempts as a measure of error rate. For these, the main effect of differencing technique, $F_{3,45} = 25$, $p < .0001$ was significant, but layout was not, $F_{1,15} = 0.06$, ns. Tukey-Kramer revealed that dual-view with no difference layer had the most attempts (1.16) on average, which was different from the group of dual view with difference layer (0.23), toggling (0.43), and animation (0.43), see Figure 2 (right).

4.5.4 Feedback from Participants

Participants were asked to rank each of the four diagram differencing techniques on a Likert scale from 1 to 7 (7 being the best). The results are summarized in Figure 4 (left).

In freeform feedback we received several interesting comments. One participant pointed out that when the conditions changed, it took a few trials to get used to the new one, despite the explicit instructions on each condition change. Another stated that the greyed objects in the difference layer "caught his eye", but that he found animation confusing. Yet another identified the dual-view condition without a difference layer as particularly hard, but got only gradually used to the difference layer visualization. One

participant pointed out that toggling was somewhat confusing. Another participant said that the dual view with the difference layer was the easiest to use, as it was easier to see what was missing. The same participant also stated that toggling made it easier to identify missing parts and animation was sometimes confusing. Another found animation more difficult as he kept choosing the nodes that were removed instead of the new nodes.

4.5.5 Discussion

Dual view without difference layer was the slowest technique, which is not surprising. Similar to text differencing, showing two versions side-by-side does not make it easy to spot differences. On the other hand and as underscored by the pilot, highlighting differences makes identifying them easy, but helps little for understanding changes. Overall, the results illustrate that toggling and animation are good techniques which are well liked, but not by everybody. For example, we noticed that some participants got confused about which state permitted selection in the toggling condition. This was also reported in the feedback.

No significance was found for the layout factor. Thus, presenting pairs sequentially with incremental layout either did not help or had only an insignificant effect. The effect of differencing techniques on layout was also insignificant. The significant interaction with the version pairs indicates that rigourously preserving node positions may even be detrimental to understanding diagram evolution. One issue is that this can cause node overlap, which leads to participant complaints. However, diagram creators may need this option, so it cannot be discounted completely. To investigate the interaction of layout and version pair we took a closer look at pair 6 diagrams and found that participants missed the same nodes and edges more often than any other pair in both layouts. However, the frequency of misses was much higher with the incremental layout. The two most frequently missed nodes have no edges attached. In the optimal layout these two nodes were placed in a very conspicuous cluster at the bottom. For edges the same pattern was observed. We speculate that the longer average edge length in the incremental layout and/or more edge crossings and/or the absence of edge bridge connectors resulted in higher miss rates. Yet, this may also point to fundamental limitations of incremental layout techniques.

In hindsight, we should have considered shorter animations. Transition intervals of 0.25 to 1 s have been found insignificant in zooming interfaces [30]. This finding may apply to our tasks, too.

4.6 User Study II

This user study compared two visualization techniques to identify shifted nodes in two versions of a diagram with *non-matching* layout. Participants were asked to select nodes that moved in the newer version of the diagram relative to the older version. Here, selecting edges was not investigated.

4.6.1 Experimental Design

We used a 3x2x6 repeated measures design (3 techniques, 2 node randomization levels, 6 version sets). The 3 tested techniques were: single-view animation, dual view with difference layer, and the combination of dual view with difference layer with animation. The motivation for including the difference layer is the reject/accept technique in DARLS. Unlike the first user study we did not include view toggling or unenhanced dual views as initial evaluations showed that those conditions take too much time to be used in our experiment. For this study we first laid out each diagram with the optimal hierarchical re-layout algorithm with the same node placement heuristics as in the first user study. Then we used a graph randomization algorithm, which shifts a percentage of random nodes in random directions while retaining a minimal distance constraint between nodes to prevent overlap. For simplicity, our algorithm does not employ any node placement heuristics and does not optimize for edge crossings or bends. However, we do not believe this is a major issue since our goal was to simulate scenarios when a user rearranges nodes in a diagram, which may generate substantial edge crossings. We shifted 22% or 44% of all nodes. We used the same six version sets as in the first study. We also used the same randomization seeds for all layouts to keep them consistent across participants. Counterbalancing was done similarly as in the first user study.

The intent of the design was to compare our difference layer method with animation and to see if participants would use our method if given the choice between the two. Initially, we intended to include more randomization levels but in the interest of keeping the experiment length reasonable we selected to use only two.

4.6.2 Procedure

Similar to the first user study, participants were allowed to select nodes while they were being animated. The <LEFT ARROW> key was also available for (re-)playing the animation in the single and dual-view animation conditions. The dual-view with difference layer condition was the same as in the first study. We informed participants during the training session that one possible strategy is to use the difference layer to match the selection of the nodes on the foreground in the right view with the reduced yellow color nodes on the background in the left view. Pair 2 was the one with the no structural changes and has 28 common nodes. This means that at any given time no more than 28×44%=12 nodes were animated. In the combined condition, animation plays automatically when new diagrams are loaded to remind participants that they can use animation. All the remaining aspects of the procedure were identical to the first user study.

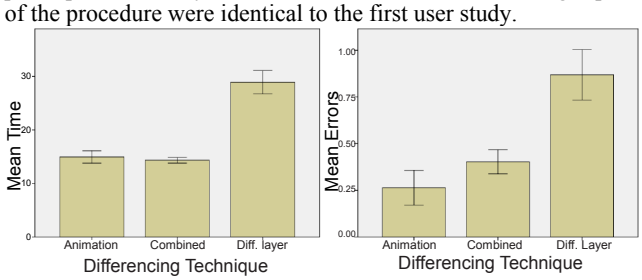

Figure 3. Mean time and errors for the techniques in User Study II. Error bars: ± 1 SE

4.6.3 Results

No ordering effects were observed. The main effects of technique $F_{2,22} = 11.08$, $p < .0005$, randomization level, $F_{1,11} = 11.57$, $p < .001$, and version pair, $F_{5,55} = 4.15$, $p < .005$ on task completion time were significant. The interaction between randomization level and technique was also significant, $F_{5,55} = 4.94$, $p < .001$. A Tukey-Kramer analysis revealed that the difference layer alone (average 28.9s) was slower than both animation (15s) and animation with difference layer (14.3s). The 22% node randomization level (14.5s) was different from the 44% level (24.3s), see Figure 3 (left). The error rate data in this experiment was not normally distributed, therefore we decided to just report the averages, see Figure 3 (right).

We used a Kruskal-Wallis test on the number of re-play key presses for both animated conditions, as this data was not normally distributed. This identified a significant difference, $H_1 = 38.09$, $p < .0001$. Animation alone had an average of 174 button presses while the animation with difference layer had 114.

4.6.4 Feedback from Participants

Participants were asked to rank the techniques similar to User Study I. The results are summarized in Figure 4 (right).

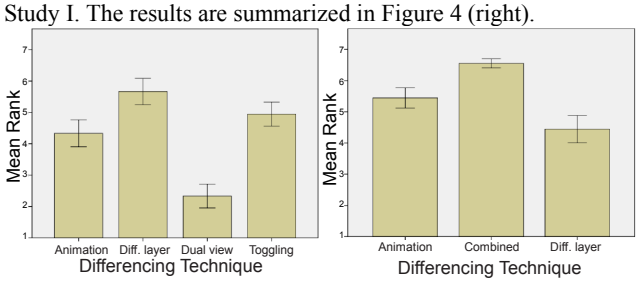

Figure 4. Participants' ranking of the differencing techniques in User Study I (left), User Study II (right). Error bars: ± 1 SE

Here are some of the most mentionworthy comments from the freeform feedback. One participant stated that animation helped to see the changes and moving objects could be identified even when he was not directly looking at them. Another participant found it difficult in the difference layer condition to find a node shifted slightly because the background node would be hidden under the foreground node due to overlap. Others pointed out that nodes that move little in the animation condition are harder to identify as well. Several participants said that for selecting the moved nodes it was easiest to use animation. They also clicked on nodes as they moved. Relative to that they pointed out that the difference layer method was harder to use, but still allowed one to check the "results" of the animation. One participant found the animation speed a bit slow. Another participant said that it would be nice to have a "shadowed" mouse cursor in the "dual-view" panel.

4.6.5 Discussion

Although the difference layer method itself was slowest, it still contributed positively overall, as the combined method was both fastest overall, although not significantly so, and most preferred. Another indication for the benefits of the difference layer is the result on the re-play key presses. Also, and as revealed by the rankings, some participants found this the easiest technique.

During the experiment we noted that some participants were confused about how to use the difference layer and were either trying to select objects on the background of the right view or objects that were not different. This indicates that this method might not work well if both views are fully interactive. On the other hand, we also saw that animation alone is not a perfect solution to easily identify nodes that moved just a little. To investigate this, we analyzed all instances where participants failed to identify a moved node. For all 679 false negatives, the average movement distance was 83 pixels while the median was only 69 pixels. A node in the experiment was 30x80 pixels large. Since the median is smaller than the mean we can argue that participants had more trouble with nodes that moved less. This indicates that none of the techniques are perfect in isolation.

4.6.6 Overall Discussion

In our first experiment where node positions matched, the dual-view technique with difference visualization was the fastest technique overall and had the least amount of errors. It was also ranked highest in terms of user preferences. In our second study with partially non-matching node positions, animation and the combination with the difference visualization technique was best.

It is not easy to compare our work directly with Archambault *et al.* [3, 4], as many details are different. But we do not believe that our findings contradict their work. Our participants, for example, also preferred the difference layers in the first study and made less errors with them. However, the difference layer was not preferred in our second experiment. Archambault *et al.* also stated in earlier work that a small number of timeslices (such as two) are not enough to represent the evolution of a graph adequately. Our findings generalize this insight to diagrams with named nodes.

Both of our layout methods were designed to preserve the mental map. The optimal layout is based on aesthetics, but the incremental layout preserves the mental map more faithfully. Hence, the incremental condition in the first study would show the effect of mental map preservation best in our context. However, and as we did not find significant effects of layout, this suggests that mental map preservation is not effective, similar to [3].

Heer and Robertson [13] found that animated transitions between statistical visualizations work well and that staged transitions are preferred. Similarly, our first study suggests that animated transitions in diagram differencing tasks are preferred.

Naturally, the results of the first user study are only directly valid for hierarchical diagrams laid out top to bottom. Our results may not generalize to other types of diagrams. Many other factors can influence the results and we even observed an interaction between layout and the version pair. On the other hand, we believe the results of the second study are more generalizable, because they depend less on the layout technique.

5 Conclusion and Future Work

We presented a new system for diagram difference visualization and merging. It uses animation, dual views, a storyboard, relative re-layout, and difference layers. We ran two user studies to investigate the benefits of the system and found that naïve dual-view visualization is not desirable. The dual-view option with a difference layer was most preferred for comparing diagrams with matching node positions. For diagrams with non-matching positions we found evidence that animation is beneficial, but that the combination with a difference layer was liked best. In summary, we can say that our difference layer technique is useful and is a good complement to animation. This has positive implications for the diagram merging method introduced above.

Some of our findings indicate that the interaction between differencing techniques and layouts is a rich area for future work. In other words, a closer look is needed at the combined effect of layout techniques, such as our optimal layout method, and specific visualization features. Another direction is the generalization of the work to UML diagrams, with their information-rich nodes and edges. In the future we also plan to investigate the storyboard in more details, e.g. if it can be directly used for difference visualization, similar to small multiples. One idea is to use highlighting on the small views *in combination with* difference layers in the large ones. Yet another direction is to investigate merging. Our findings are positive for animation, but more can be done. With larger sets of changes, animating all changes at once may be counterproductive. In the future, we want to investigate whether it makes general sense to break change visualizations into smaller sets for easier comprehension. Currently the system is targeted at diagrams with up to 50 nodes. However, we do not see any major obstacles to enhancing the system to deal with larger diagrams, such as the hierarchies described in [6, 20].

References

[1] M. Alanen and I. Porres, Difference and union of models. *UML 2003*, 2-17.

[2] D. Archambault, Structural differences between two graphs through hierarchies. *Graphics Interface 2009*, 87-94.

[3] D. Archambault, H. Purchase and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4), 539-552, 2011.

[4] D. Archambault, H. Purchase, B. Pinaud. Difference map readability for dynamic graphs. *Graph drawing*, Springer 2011, 50-61.

[5] L. Bartram and C. Ware Filtering and brushing with motion. *Information Visualization*, *1* (1), 66-79, 2002.

[6] S. K. Card, B. Sun, B. A. Pendleton, J. Heer and J. W. Bodnar, Time tree: Exploring time changing hierarchies. *Symposium On Visual Analytics Science And Technology 2006*, 3 -10.

[7] F. Chevalier, P. Dragicevic, A. Bezerianos and J.-D. Fekete, Using text animated transitions to support navigation in document histories. *CHI 2010*, 683-692.

[8] D. Dadgari and W. Stuerzlinger, Novel user interfaces for diagram versioning and differencing. British HCI 2010.

[9] S. Diehl and C. Görg, Graphs, they are changing. Graph Drawing 2002, 23-30.

[10] M. Farrugia and A. Quigley Effective temporal graph layout: A comparative study of animation versus static display methods. Journal of Information Visualization, (to appear), 2011.

[11] S. Förtsch and B. Westfechtel, Differencing and merging of software diagrams: State of the art and challenges Workshop on Comparison and Versioning of Software Models 2008, 7-12.

[12] A. L. Griffin, A. M. MacEachren, F. Hardisty, E. Steiner and B. Li A comparison of animated maps with static small-multiple maps for visually identifying space-time clusters. Annals of the Association of American Geographers, 96 (4), 740 - 753, 2006.

[13] J. Heer and G. Robertson Animated transitions in statistical data graphics. IEEE Transactions on Visualization and Computer Graphics, 13 (6), 1240-1247, 2007.

[14] J. W. Hunt and D. McIlroy, An algorithm for differential file comparison. Technical Report, AT&T Bell Laboratories, 41, 1976.

[15] U. Kelter and M. Schmidt, Comparing state machines. Workshop on Comparison and versioning of software models 2008, 1-6.

[16] D. Kurlander and S. Feiner A visual language for browsing, undoing, and redoing graphical interface commands. Visual Languages and Visual Programming, 257-275, 1990.

[17] S. Maier, M. Minas, Interact. diagram layout. CHI 2010, 4111-4116.

[18] A. Mehra, J. Grundy and J. Hosking, A generic approach to supporting diagram differencing and merging for collaborative design. Automated Software Engineering 2005, 204-213.

[19] C. Meng, M. Yasue, A. Imamiya and X. Mao, Visualizing histories for selective undo and redo. Proceedings of the Third Asian Pacific Computer and Human Interaction 1998, 459.

[20] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang and Y. Zhou Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. ACM Trans. Graph., 22 (3), 453-462, 2003.

[21] T. Nakamura and T. Igarashi, An application-independent system for visualizing user operation history. UIST 2008, 23-32.

[22] E. Ogasawara, P. Rangel, L. Murta, C. Werner and M. Mattoso, Comparison and versioning of scientific workflows. ICSE Workshop on Comparison and Versioning of Software Models 2009, 25-30.

[23] D. Ohst, M. Welle and U. Kelter, Difference tools for analysis and design documents. Software Maintenance 2003, 13.

[24] D. Ohst, M. Welle and U. Kelter Differences between versions of uml diagrams. SIGSOFT Softw. Eng. Notes, 28 (5), 227-236, 2003.

[25] C. Plaisant, J. Grosjean and B. B. Bederson, Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. Information Visualization 2002, 57.

[26] H. C. Purchase, E. Hoggan and C. Gorg, How important is the "mental map"?: An empirical investigation of a dynamic graph layout algorithm. Graph drawing 2007, 184-195.

[27] H. C. Purchase and A. Samra, Extremes are better: Investigating mental map preservation in dynamic graphs. Diagrammatic Representation and Inference 2008, 60-73.

[28] G. Robertson, R. Fernandez, D. Fisher, B. Lee and J. Stasko Effectiveness of animation in trend visualization. IEEE Transactions on Visualization and Computer Graphics, 14 (6), 1325-1332, 2008.

[29] P. Saffrey and H. Purchase, The "mental map" versus "static aesthetic" compromise in dynamic graphs: A user study. Australasian User Interface 2008, 85-93.

[30] M. Shanmugasundaram and P. Irani, The effect of animated transitions in zooming interfaces. Proceedings of the working conference on Advanced visual interfaces 2008, 396-399.

[31] M. Shanmugasundaram, P. Irani and C. Gutwin, Can smooth view transitions facilitate perceptual constancy in node-link diagrams? Proceedings of Graphics Interface 2007 2007, 71-78.

[32] S. L. Su, Visualizing, editing, and inferring structure in 2D graphics. UIST Adjunct Proceedings 2007, 29-32.

[33] S. L. Su, S. Paris, F. Aliaga, C. Scull, S. Johnson and F. Durand, Interactive visual histories for vector graphics. Technical Report, MIT-CSAIL-TR-2009-031.

[34] B. Tversky, J. B. Morrison and M. Betrancourt, Animation: Can it facilitate? Int. J of Human-Comp. Studies, 57(4), 247-262, 2002.

[35] C. Ware and R. Bobrow Motion to support rapid interactive queries on node-link diagrams. ACM Trans. Appl. Percept, 1(1), 3-18, 2004.

[36] Z. Xing and E. Stroulia, Differencing logical UML models. Automated Software Eng., 14 (2), 215-259, 2007.

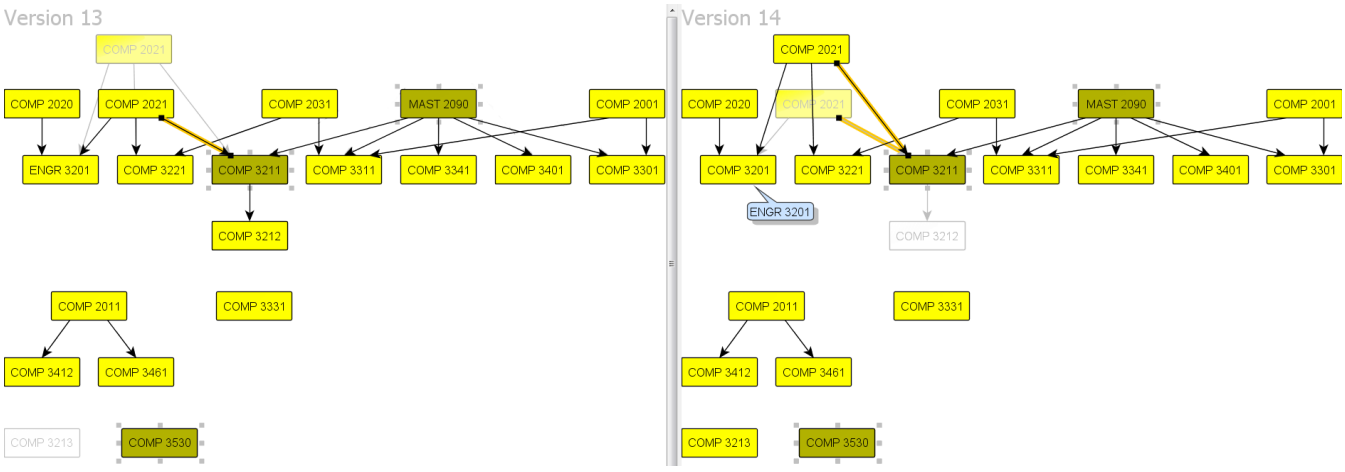

Figure 5. The state of the diagrams after the reject operation in Figure 1 is invoked.