DARLS: Differencing and Merging Diagrams Using Dual View, Animation, Re-Layout, Layers and a Storyboard

Loutfouz Zaman
Department of Computer Science and Engineering
York University
Toronto, Ontario M3J 1P3 Canada
zaman@cse.yorku.ca

Ashish Kalra
Information Technology Department
NIT Kurukshetra, India
ashishorkalra@gmail.com

Wolfgang Stuerzlinger
Department of Computer Science and Engineering
York University
Toronto, Ontario M3J 1P3 Canada
wolfgang@cse.yorku.ca

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
We present a new system for visualizing and merging differences in diagrams. It uses animation, dual views, a storyboard, relative re-layout, and layering to visualize differences. The system is also capable of differencing UML class diagrams. An evaluation produced positive results for animation and dual views with difference layer.

Keywords
Versioning, Differencing, Diagrams, Animation, UML

ACM Classification Keywords
H.5.2 User Interfaces [Graphical user interfaces (GUI)]

General Terms
Design, Human Factors

Introduction
Computer-supported version differencing and merging of text documents has been used at least since the introduction of the Unix `diff` tool. Modern version control tools for text have become much more user-friendly by incorporating visual interfaces that facilitate differencing and merging. Examples are the use of text highlighting and context sensitive menus. Another, more recent example is the use of animation [4].

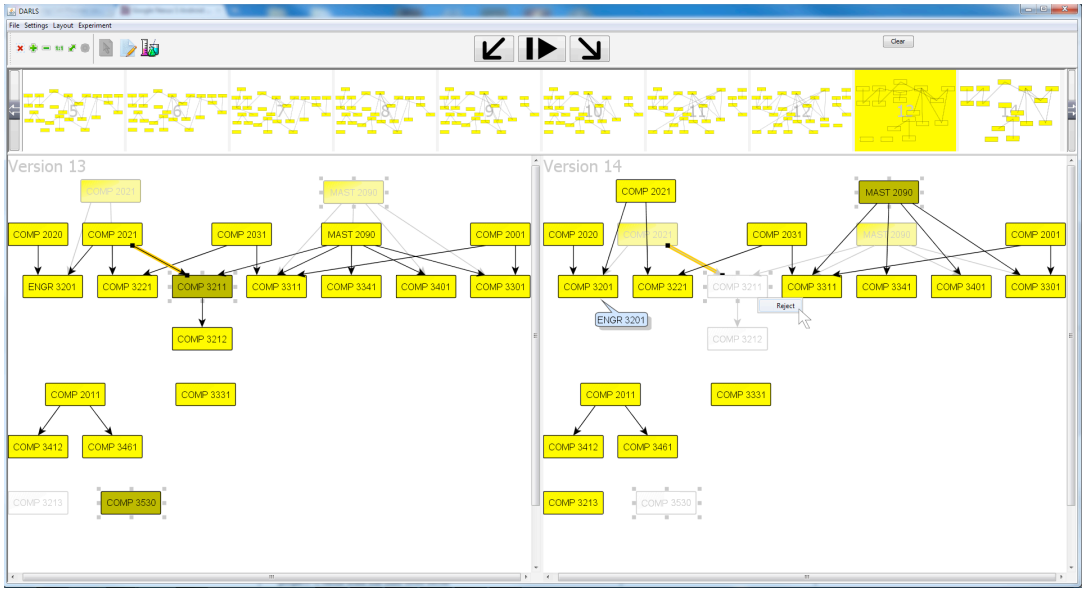

figure 1. DARLS showing two versions of a diagram, which visualizes course pre-requisites. The visualization shows a difference layer and uses the relative optimal re-layout.

Many interaction techniques have been proposed for dynamic graph visualization. Recently user studies evaluated some of them [1, 2]. The studies focused on generic graphs where node and edge attributes are irrelevant. Very little work deals with diagrams where nodes in the graph are identified by name [10]. Also, this research primarily targets differencing, and to our knowledge, no quantitative research exists on visualizations that support diagram version merging.

To address these shortcomings, we introduce a new system for differencing and merging diagrams that makes use of various user interface techniques, such as Dual View, Animation, Re-Layout, Layers and a Storyboard, short "DARLS". It is targeted at diagrams where node and edge attributes are as important as the graph structure itself. One example is vector graphics diagrams in Wikipedia. Diagrams are used frequently in architecture, design, information and concept visualization, and in software engineering as *software documents*, e.g. UML. Also, more and more diagrams that change over time appear today in electronic form. Diagram versioning also greatly facilitates collaboration around diagrams.

Related Work

Our work builds on difference maps, mental maps, animation and small multiples in the context of dynamic graph drawing, side-by-side views for visual comparison, the use of storyboards for non-linear access, and text and UML diagrams versioning.

Su [11] introduced a new interaction metaphor and visualization for the *operation history* of 2D illustrations. The user accesses history via graphical depictions on top of the document. "S*mall multiples"* is a related concept in dynamic graph drawing, which displays dynamically evolving data in a matrix of images. Each image is a timeslice [1]. We chose to use a storyboard in our system, because it provides a good overview of all versions in a graphical form that facilities access, yet uses only limited space. A list of version names is less user friendly in comparison.

As it's hard to trace diagram evolution in unenhanced visualizations we employ a number of techniques that aid the user. Our new layering technique is related to the concept of a *difference map* in dynamic graph drawing. It presents the union of nodes and edges in

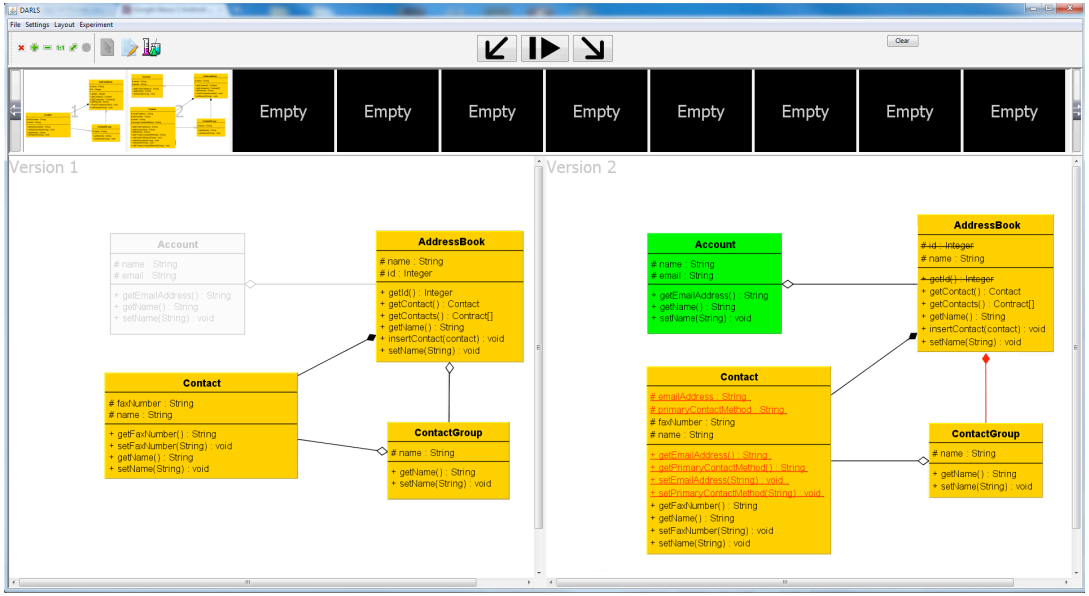

figure 2. Differencing UML Class Diagrams.

the two graphs for two different timeslices [1, 2]. Unified views were used by Ohst *et al.* [9] in a unified document, which highlights the common and specific parts of two UML diagrams. Mehra *et al.* [8] described another set of difference visualizations, which were liked by users. No quantitative user studies were conducted. Generic *diagram unification* for comparing documents was studied by Dadgari and Stuerzlinger [5]. They compared multiple graph differencing methods and merging techniques in a qualitative user study. Participants preferred a translucent view that overlaid the versions. Our unified view shows diagram objects that do not belong to current version in the background to reduce visual clutter. Combined with *side-by-side views* this yields a better visualization.

Side-by-side views have been used for visual comparisons long before computers were invented. One modern adaptation is a side-by-side view for comparing texts and other text-based documents and there are many publicly available tools. However, and to our knowledge, there are no publicly available tools for diagram comparison. Förtsch *et al.* [7] presented a survey on differencing and merging of software diagrams and listed requirements for UML diagram versioning tools. One of the main requirements is a user-friendly representation. The authors also point out that it is desirable for diagrams to be displayed side-by-side with differences marked graphically.

Animation has been used for visualizing dynamically evolving data in graphs, e.g. [3, 12]. A number of user studies explored the effects of difference maps, small multiples, slide shows and mental map preservation, e.g. [1, 2], with positive results for small multiples and animation. A recent study [4] showed that animation facilitates the comparison of texts, and permits users to better identify changes that occurred between versions. We also use animation to show the evolution of changes in the diagrams more clearly.

Incremental 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 the graph [6]. It facilitates navigation in the graph or comparison of it and other graphs. The benefits of mental map preservation have been studied in the past by Purchase *et al.* [10]. In our system we also manipulate the diagram layout in the side-by-side views to facilitate comprehension of diagram evolution.

The DARLS System

We developed a new system for visualizing differences between diagrams and versioning of diagrams. Nodes and edges are disambiguated with unique identifiers. The system supports differencing and merging of generic and UML class diagrams. The system is based on the yFiles Graph Visualization Library[1]. To illustrate the system, we use two versions of a course prerequisite diagram from two years. See Figure 1.

The system features *side-by-side views* of two versions of a diagram, with synchronized zooming with the mouse wheel and panning with the scroll bars. Buttons allow toggling between editing and selection modes. Diagram repositories are accessed through the file menu. Both graphs can be edited and committed back into the repository. The user can directly access ten versions of the diagram in the scrolling storyboard.

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 contains the other diagram. We call this a *difference layer*. This is different from Pounamu [8], where only a single merged view is used and the objects common to both compared diagrams are not shown. It is also different from difference maps, as it displays the common nodes and edges between two versions even if a node was moved. The rationale is to enable accept/reject of node movements. A configuration dialog accommodates different color schemes.

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

If the user selects, for example, a node in the right view, the corresponding node in the left view is highlighted with a selection box as well (with different styles, depending if the node exists in the other diagram) and vice versa. The user can customize this, so that either the node on the foreground and the difference layer are 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, all this applies to edges as well.

The ability to accept and reject graph edits was previously presented [8]. In our system, a context-sensitive right-click menu provides easy access to this functionality. See the popup next to COMP 3211 in Figure 1. 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 3213 will be deleted and MAST 2090 will be shifted down.

When the play/pause button in the top panel is pressed the differences between the diagrams in the two views

[1] http://www.yworks.com

are animated in three phases. First, removed objects fade out, then moved objects are shifted from the old to their new locations and changes in shape and color are morphed, and finally new nodes and edges fade in. The sequence and concurrency of these animations can be customized. 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. 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, use a call-out visualization.

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 features that we provide, as the layout of the graph has changed (too) much. Therefore, we added functionality to interactively re-layout one diagram relative to another minimizing visual differences. We implemented two relative re-layout algorithms: *incremental*, which preserves the locations of nodes, and, *optimal*, which rearranges nodes to better use screen space. Both layout methods keep the positions of nodes and edges common to both diagrams stable and thus preserve a mental map. We based our implementation on the yFiles Hierarchic and Incremental Hierarchic Layouters.

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 downward is minimized, while keeping the edge length short. Then these nodes are arranged within their lanes so that the number of edge crossing is 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 2012 and MAST 2090 were manually raised higher after the re-layout.

Our system also allows *UML class diagrams* differencing. Here we used text differencing techniques (strikethrough and underline) as well. Changes in association type, such as from aggregation to composition, are visualized by highlighting edges in a different color. Classes common to two diagrams are shown in one color, newly added ones – in another. Colors can be customized and changes can again be animated. Deleted classes are displayed in the difference layer with reduced transparency. New attributes and methods are highlighted in red and underlined. Deleted ones are crossed out. The user can customize the differences to be displayed in either view or both. Figure 2 demonstrates differences displayed in the right view. Currently our system can visualize class diagrams generated from any Java application, with the help of a freely available UML diagram extractor.

User Studies

We ran two user studies investigating the benefits of the system for generic (non-UML) diagrams[2]. The first user study compared single view with animation or toggling between versions, as well as dual view with and without a difference layer on diagrams with matching node positions. We found the unenhanced dual view to be a bad choice. The difference layer had the least amount of errors and participants liked it best. The second study compared the difference layer, animation, and their combination for diagrams with non-matching node positions. Although the difference layer had the longest completion times compared to both animation and the combined technique, it was still found useful. Participant ranked the combined technique best.

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. The system is also capable of differencing UML class diagrams. We are planning to run more studies, which target the layout techniques and animation in detail. We also did not pay attention to the storyboard and plan to investigate it in the future, 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. The UML aspects of our system have also not been evaluated and we plan to run a study on the benefits for refactoring UML diagrams and source code.

[2] A paper describing the user studies is currently under review.

References

[1] Archambault, D., Purchase, H., Pinaud, B. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE TVCG* (99). 1-1.

[2] Archambault, D., Purchase, H., Pinaud, B. Difference map readability for dynamic graphs, *Symposium on Graph Drawing 2011*, To appear

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

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

[5] Dadgari, D., Stuerzlinger, W., A novel user interface for diagram versioning and differencing. *British HCI 2010*.

[6] Diehl, S., Görg, C. Graphs, they are changing - dynamic graph drawing for a sequence of graphs, *Graph Drawing 2002*, 23-30.

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

[8] Mehra, A., Grundy, J., Hosking, J., A generic approach to supporting diagram differencing and merging for collaborative design. *Conference on Automated software engineering 2005,* 204-213.

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

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

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

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