Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

TimeRadarTrees: Visualizing Dynamic Compound Digraphs

M. Burch[1] and S. Diehl[1]
[1]University of Trier, Germany

Abstract

The evolution of dependencies in information hierarchies can be modeled by sequences of compound digraphs with edge weights. In this paper we present a novel approach to visualize such sequences of graphs. It uses radial tree layout to draw the hierarchy, and circle sectors to represent the temporal change of edges in the digraphs. We have developed several interaction techniques that allow the users to explore the structural and temporal data. Smooth animations help them to track the transitions between views. The usefulness of the approach is illustrated by examples from very different application domains.

Categories and Subject Descriptors (according to ACM CCS): E.1 [Data Structures]: Graphs and Networks

1. Introduction

Information hierarchies occur in many application domains such as the hierarchical organization of companies, news topics and subtopics, file/directory systems, products and product groups of a department store, or phylogenetic trees in biology. The evolution of dependencies in such information hierarchies can be modeled by sequences of compound digraphs with edge weights. While there has been a lot of work on visualizing information hierarchies [RT81, YFDH01, JS91, AH98, SZ00], only few researchers have developed methods to visualize dependencies between elements in the hierarchy [NSC05, FWD*03, ZMC05, Hol06], see Section 4 for a more detailed discussion.

The TimeRadarTree approach presented in this paper supports the visualization of the evolution of weighted dependencies in information hierarchies in a single diagram. To this end the TimeRadarTree approach integrates three views into one:

Interactive radial tree: It shows the whole hierarchy to an interactively selectable level. By clicking at a node on the circumference it is expanded, by clicking at an intermediate node, the subtree starting at that node is collapsed and the node is put on the circumference. Expanding or collapsing subtrees of the hierarchy can help to detect relations at different levels of abstraction.

Inner Circle (Time Radar): Incoming edges of leaf nodes or collapsed subtrees are shown as colored parts of a circle sector related to that node or subtree. The color of each part encodes the weight of the edge, i.e. the strength of the dependency.

Outer Circles (Thumbnails): The smaller outer circles related with each hierarchy node show the outgoing edges of the related node. The target node of each edge can be inferred from its direction, shape, and color.

We have developed several interaction techniques that allow the users to explore the data. Smooth animations help them to track the transitions between views. Thus, TimeRadarTrees allow the user to detect when and how strong elements of the hierarchy are related.

The rest of this paper is organized as follows: In Section 2 we introduce the different constituents of the TimeRadarTree visualization by means of examples. Then, in Section 3 we illustrate the usefulness of our approach by looking at data sets from various application domains. Related work is discussed in Section 4. Finally, Section 5 gives some conclusions and possible directions for further work.

2. TimeRadarTrees – step by step

We illustrate our visualization technique by starting with the representation of a single graph and then adding features step by step. As first example consider the node-link diagram of a single, directed graph (short: digraph) shown in Figure 1. Nodes are represented by circles, directed edges by arrows from one node to another.

In TimeRadarTrees there is not a single representation of a node or an edge, see Figure 2. But for each node its in-

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

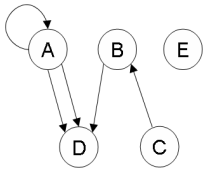

Figure 1: *Node-link diagram of a graph.*

coming edges are represented by sectors of the large circle in the middle, while the smaller circles on the circumference of the inner circle show the outgoing edges. These circles are subdivided into sectors as follows. First, if we have n nodes, the circle is divided into n equally-sized sectors. Each of these sectors is associated with a certain node. In the example, the lower left sector of all circles is associated with node D. Next, each of the sectors is subdivided into a number of smaller sectors depending on the number of incoming edges of the associated node. In the example, the three colored sectors related to the node D in the inner circle represent the three incoming edges of node D, while the one big colored sector related with node A indicates that node A has only a single incoming edge, and finally, the white sectors related to E and C show that these nodes have no incoming edges at all.

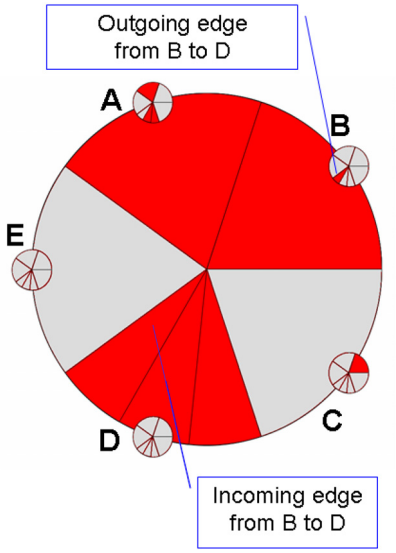

Figure 2: *TimeRadarTree of a single graph.*

Note, that by looking only at the inner circle, we can not identify the nodes where the incoming edges start from. This information can be grasped by looking at the outer circles. For example, by looking at the outer circle related to node B, we see that there is only one outgoing edge, and this outgoing edge is drawn in the part of the circle associated with the node D. In comparison with node-link diagrams, an important advantage of the TimeRadarTree visualization is that there are no edge crossings leading to visual clutter.

While this sector-based representation might seem awkward at first and it needs some training to read this representation, it will turn out useful, once we add more features. Let's start by trying to visualize a sequence of graphs instead of a single graph. As an example, we use the sequence of graphs shown in Figure 3. Each graph of the sequence is shown by a separate node-link diagram.

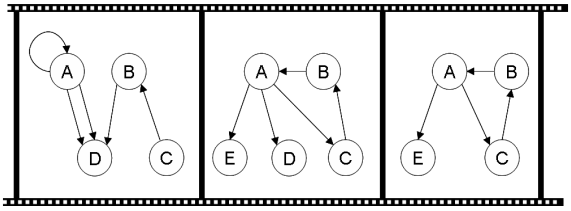

Figure 3: *Node-link diagrams of a sequence of graphs.*

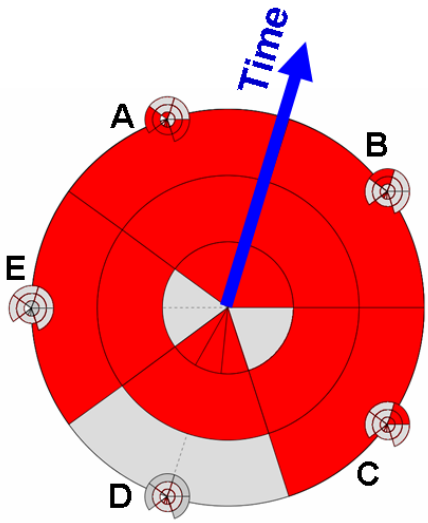

Figure 4: *TimeRadarTree of a sequence of graphs.*

Figure 4 shows a TimeRadarTree representation of the same sequence of graphs. Here the edges of each graph are represented by sectors of the same ring. Actually, in this example, the innermost circle corresponds to the first graph, that we have seen before, the next graph of the sequence is represented by the inner ring, and the third graph by the outer ring. Looking at the lower left sector of the first graph, we see for example, that there have been three incoming edges for node D in the first graph, one in the second, and none in the third. By looking at the lower left sectors of the small outer circles, we see that two of the incoming edges of D start from node A, and its third incoming edge starts from node B. Furthermore, we see that the single incoming edge in the second graph starts also from A. In comparison with animated node-link diagrams, the integration of all graphs in the sequence into a single diagram helps the user to preserve the mental map.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

In a compound digraph, like the one shown in Figure 5, the nodes of the graph are additionally related to leaves of a hierarchy. In the example, the graph and the hierarchy are both shown as node-link diagrams with additional edges connecting the nodes of the graph with the leaf nodes of the hierarchy.

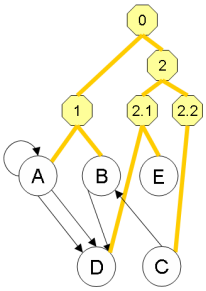

Figure 5: *Node-link diagram of a compound graph.*

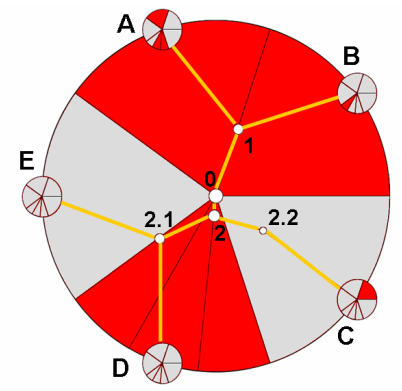

Figure 6: *TimeRadarTree of a compound graph.*

In Figure 6 the same compound graph is shown as a TimeRadarTree using radial layout to embed the node-link diagram of the hierarchy in the sector-based visualization of the graph.

We can now combine both features, i.e. visualize sequences of compound graphs. Figure 7 shows them as a sequence of node-link diagrams, while Figure 8 combines the two approaches discussed above to integrate all information into a single TimeRadarTree diagram.

Another way to extend the visualization is to consider directed graphs with edge weights as the one shown in the node-link diagram in Figure 9. Instead of using numbers, we can encode weights by colors – both in the node-link diagram as well as in the TimeRadarTree, see Figure 10.

Finally, by combining all the features that we discussed above, we are able to visualize sequences of compound digraphs with edge weights in a single TimeRadarTree diagram. Our TimeRadarTree visualization tool provides many

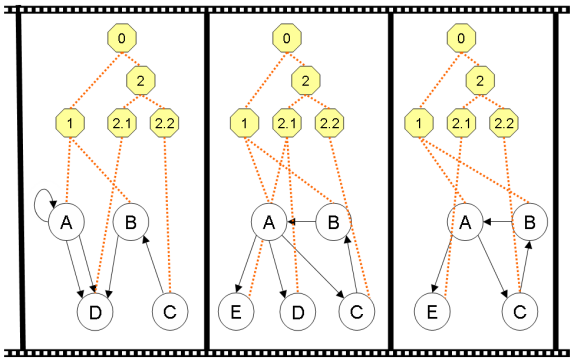

Figure 7: *Node-link diagrams of a sequence of compound graphs.*

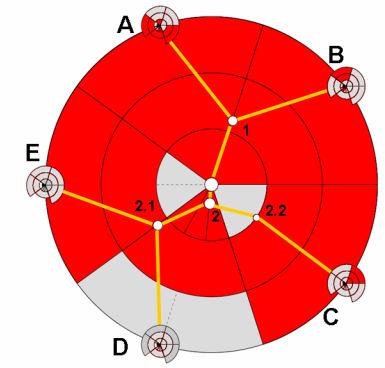

Figure 8: *TimeRadarTree of a sequence of compound graphs.*

additional interactive features such as brushing, tool tips, and a selection of predefined color scales to choose from. The most important of these features is that the user can interactively expand or collapse subtrees of the hierarchy. Smooth animations help the user to keep track of the resulting changes of the visualization. Furthermore, to accomodate for the low resolution of sectors in the center of the circle, the user can perform a circular shift of the time axis, thus moving graphs from the inner rings to the outer rings and vice versa.

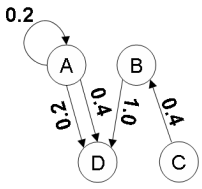

Figure 9: *Node-link diagram of a weighted graph.*

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

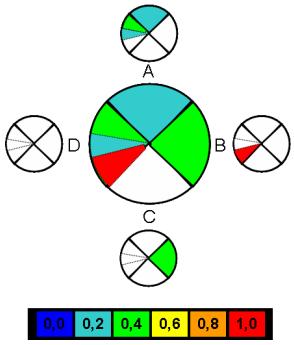

Figure 10: *TimeRadarTree of a weighted graph.*

3. Applications of TimeRadarTrees

To illustrate the usefulness of our visualization system, we apply the TimeRadarTrees system to data sets of very different domains.

3.1. Soccer Match Results

The world's national soccer teams can be hierarchically organized by first subdividing them by continents, and then further subdividing them by regions of these continents, for example north, south, east, or west. The results of soccer matches that took place within a given time interval can be used to generate a graph in the following way: The number of goals $g_{A\to B}$ of team A against team B are represented by a directed edge between A and B with the weight $g_{A\to B}$ and an edge leading from B to A with the weight $g_{B\to A}$, i.e. the number of goals that team B has scored against A in this match. It is important to differ between edges with weight 0 and non-existing edges. Looking at many of these subsequent graphs in a single view can provide important insights of the national teams soccer playing quality. Figure 11 shows a sequence of 14 compound digraphs generated for soccer matches between national teams of Central Europe and South America from 1992 to 2005. The first observation that one can make is that there have only been a few matches between teams of Central Europe and South America. This fact can be found out by having a closer look on the thumbnail circles. The thumbnail circles in the lower part only have a few colored edges in their upper part, and the thumbnail circles in the upper part have only a few colored edges in their lower part. Only the teams of Germany and Brazil have played against each other a bit more frequently. A second observation is that the teams of South America played more matches against each other than the teams of Central Europe in the same period of time.

Color coding can be used to find out the teams which scored many goals against other teams. Here we use the following color coding scheme: black indicates 0 goals, blue indicates $1-2$, green indicates $3-4$, yellow indicates $5-8$ goals, and red indicates more than 8 goals. Looking at Figure 11 again one can easily find out that Germany has a red outgoing edge to the national team of Liechtenstein. A detail-on-demand request for that edge provides the information, that Germany won the match 9:1 on June 4th 1996. Moreover it can be seen that the team of Brazil scored very often and not surprisingly the team of Liechtenstein scored very seldom.

TimeRadarTrees can also be used to just get an overview of the evolution of incoming edges. Especially for this dataset this means which teams have a good and which have a bad defense. In Figure 11 it can be seen that the team of Liechtenstein has many against goals. For the team of Austria we see that the situation of against goals worsens after the year 2002. The outermost thin ring shows the average weight for each of the teams which means the average number of against goals in the years from 1992 to 2005. Here we can see that Liechtenstein has the biggest value (green) followed by Venezuela. All other national teams have much lower values (blue).

In Figure 12 we are not interested in the hierarchical structure but more in the graphs. We selected the Central Europe part of the whole hierarchy and got a TimeRadarTree which shows 9 national teams and their match results of the time period 1992-2005. The inner big circle visualizes the goals against, the outer smaller circles the scored goals against the corresponding teams. We can immediately recognize that the team of Liechtenstein is unsurprisingly the worst team with respect to its soccer skills. Only a few black or dark colored arcs show that the opponent teams was not able to score. A closer look at the thumbnail view gives an insight that these opponent teams were very weak ones by themselves.

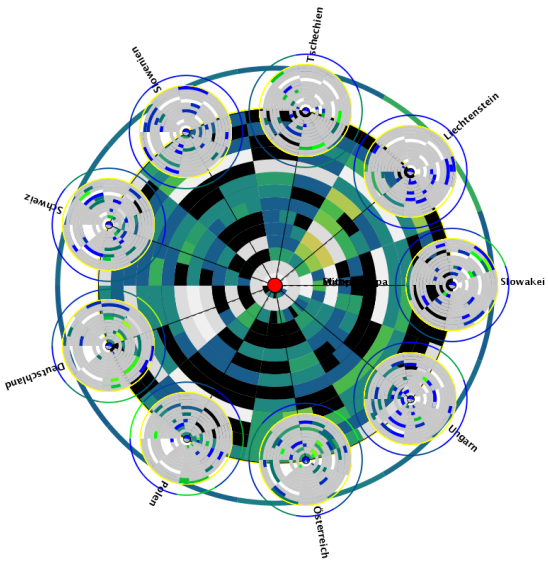

Figure 12: *A Comparison: Soccer matches of national teams in Central Europe.*

Figure 13 shows all against goals (incoming edges) of soc-

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

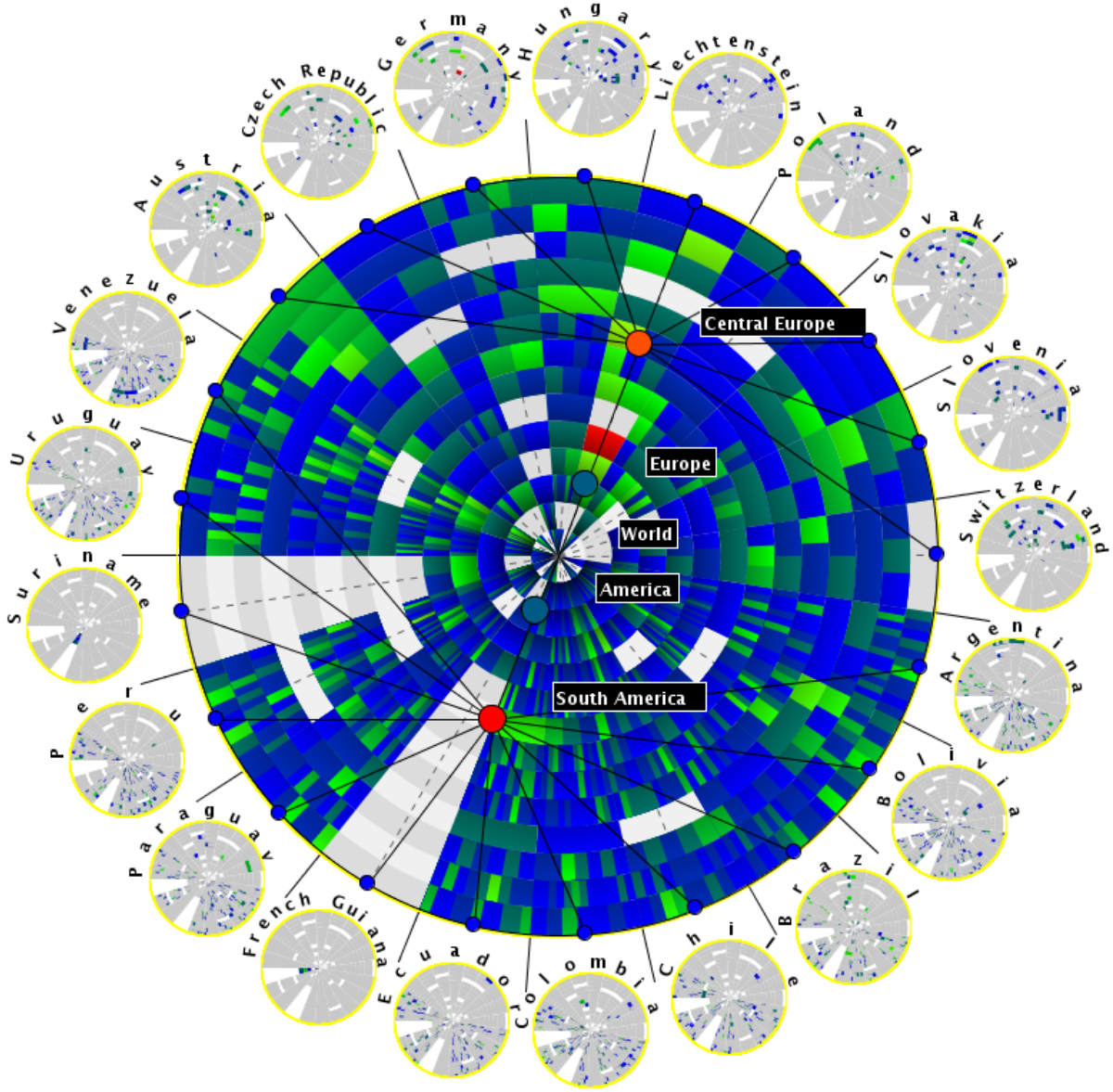

Figure 11: *A Comparison: Soccer matches between national teams of Central Europe and South America.*

cer matches of all national teams in the world from 1992 to 2005. Here we can see that the teams from the Caribbean and the teams from Oceania have the most against goals. This is an indicator that these teams can not keep up with the world's best national teams during that 14 years period. In the tool, the user can also interactively inspect the scored goals (outgoing edges) of each national team by selecting a leaf node on the circumference.

3.2. Triplet Codes in Gene Sequences

A genetic code consists of triplet codes called codons. Each codon consists of three nucleotides namely Adenine(A), Cytosine(C), Guanine(G), and/or Thymine(T), representing a single amino acid. Thus, codons are a three letter code over a four letter alphabet, and there are $4^3 = 64$ different possibilities. For the visualization we use the prefix tree of the codons as a hierarchy and the position of a codon in the gene sequence forms a natural order. Furthermore, two codons are

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

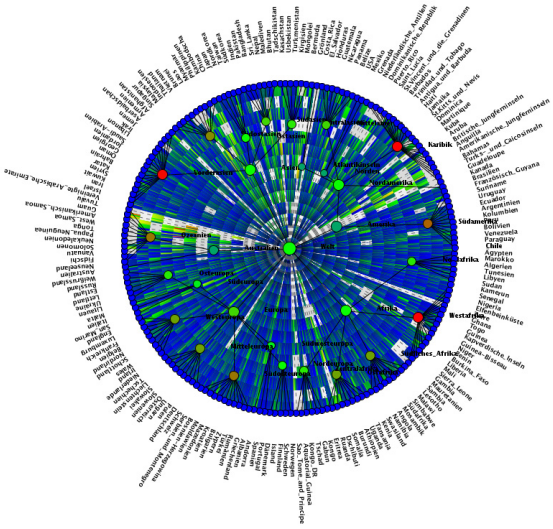

Figure 13: *Soccer matches of national teams in the whole world between 1992 and 2005.*

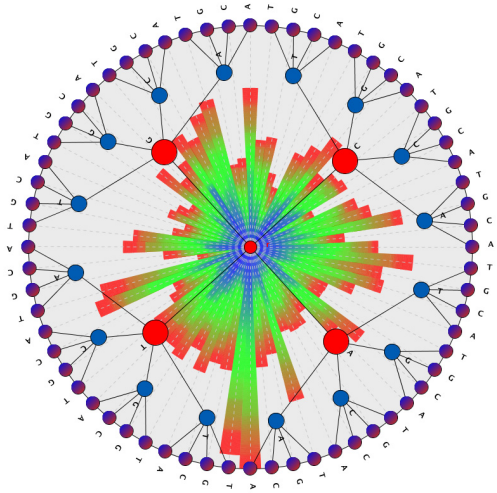

Figure 14: *The first 10000 triplets of the gene sequence of a salmonella in a stacked view.*

related if they are neighbors in the genetic code. To understand the distribution of the triplets over a longer part of the gene sequence TimeRadarTrees can represent the data set in a stacked view which can not keep the edge order any longer but on the other hand it can visualize the absolute number of incoming edges. This representation is similar to a bar chart but in a radial style.

Figure 14 shows the first 10000 triplet codes of the DNA of a salmonella in a stacked view. On the one hand we immediately see that *AAA*(Lysine) and *TTT*(Phenylalanine) are the most frequent codons. On the other hand, all codons starting with the prefix *AC* occur less frequently. The radial tree layout of the codon hierarchy is very similar to the codon wheel [Swa84] which is a widely used visualization in genetic biology. It shows the correspondence of codons and the amino acids they encode. In addition TimeRadarTrees can show the frequency and the distribution of the codons throughout the gene sequence.

3.3. Software Evolution

Next we look at the evolutionary coupling of software artifacts like modules, files, classes, and methods. The strength of the evolutionary coupling of two artifacts is the number of times they have been changed together. With TimeRadarTrees we can show in which time intervals two software components have changed together very frequently and to what extent. The number of changed lines can be color coded and show which software artifacts have been changed together very frequently and how many lines have been involved in this change. Such a coupling between many hierarchy levels can be a hint for a bad software system design [ZDZ03]. To completely understand these phenomena we have to inspect the source code of the corresponding software artifacts.

The TimeRadarTree in Figure 15 visualizes the co-change of files in the `browser` and the `doc` subdirectories of the JEDIT software system (`jedit.org`). The color of an edge indicates the size of the change in terms of the number of changed lines of code. Small changes are indicated by blue, bigger ones by green, and finally red indicates very large numbers of changed code lines. In the example here we can see that the files `TODO.txt` and `CHANGES.txt` have been changed together very frequently but only small changes have been done. Many of the files in the `browser` subdirectory have been changed together and the green color indicates that there are involved several lines of code. The thumbnail can be used to find out that these files have been mainly changed in the same transactions.

3.4. Export/Import Behaviour

Another interesting application area is the export/import behaviour of the world's countries. Figure 16 gives an overview about the exported fuel of the world's countries. A part of the worlds hierarchy is expanded so that the user can see all the countries belonging to a part of Asia. Here red indicates a large amount of exported fuel, while green indicates only a small amount; yellow indicates the values in between. It catches ones eye that Saudi Arabia is one of the world's biggest fuel exporter followed by the United Arab Emirates. Jordan and Israel have not exported as much fuel as the ones mentioned above.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

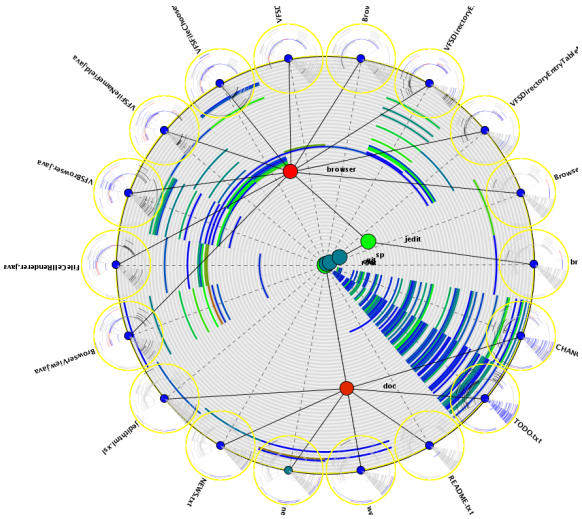

Figure 15: *Commonly checked-in files of a part of the JEDIT Open Source Project.*

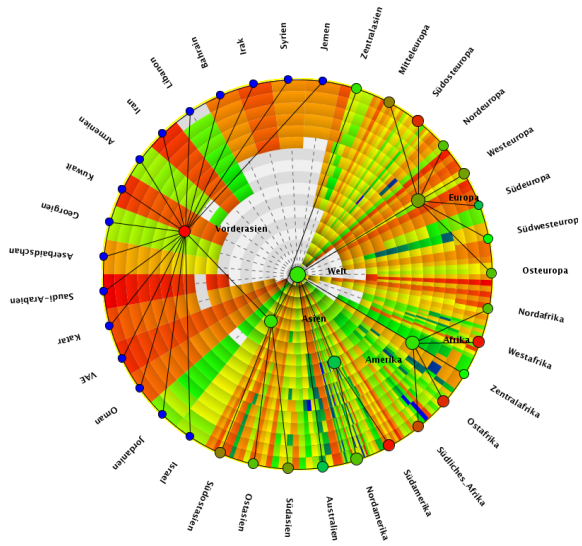

Figure 16: *Exported fuel of the worlds countries in thousand dollars.*

4. Related Work

In this section we discuss some of the related work with respect to the visualization of hierarchies, of dependencies in hierarchies, and of chronological data.

Visualization of Hierarchies Information hierarchies can be seen as a special kind of graphs, namely trees. As a result, information hierarchies can be visualized as node-edge diagrams using specialized graph layout algorithms [RT81]. Radial layout does not only improve the space-efficiency of these diagrams, but also enables new navigation techniques [YFDH01]. Space-filling techniques to visualize hierarchies include Treemaps [JS91], Information Slices [AH98], or Sunburst [SZ00]. Note, that the latter two approaches use circle sectors to represent nodes of the tree, while we use circle sectors to represent edges, i.e. dependencies.

Visualization of Dependencies in Hierarchies Many approaches try to encode dependencies between objects as directed or undirected edges in node-link diagrams. In particular, specialized graph-drawing algorithms for compound digraphs have been developed [FT04, SM91, BM99]. The appearance of edges, for example, their color, shape, orientation, thickness or connection, can represent the strength of the dependencies. There are several approaches that extend Treemaps [JS91] to also show different kinds of relations [FWD*03, ZMC05, BD06]. For example, Arc-Trees [NSC05] is an interactive visualization tool for hierarchical and non-hierarchical relations. It extends the hierarchical view of the Treemap approach with arc diagrams to present relations. Hierarchical Edge Bundles [Hol06] show relations by bundled edges between the nodes of a radial icicle view, treemap or baloon layout.

Visualization of Chronological Data The ThemeRiver [HHWN02] visualization shows the thematic changes of a collection of documents as a set of "rivers" along a time line from left to right. Each river represents a theme and the strength of the theme at a certain point in time is depicted by the width of the river. Growing Polygons [ET03] show unweighted relations by colored sides of nested polygons. Inner polygons represent older relation, outer polygons newer relations. TimeRadarTrees are very similar to this approach, but use circles instead of polygons, show directed instead of undirected relations, and include the information hierarchy.

Discussion While one can certainly think of possible extensions of the approaches discussed above, to the best of our knowledge none of the existing approaches has so far been extended and applied to visualize dynamic, weighted compound digraphs in a single, static image. In Table 1 we compare several relevant approaches. Only Timeline Trees [BBD08] and TimeRadarTrees provide a compact, crossing free representation of dynamic graphs.

5. Conclusions and Future Work

We have introduced the TimeRadarTree visualization as a technique for exploring the evolution of dependencies in information hierarchies. We discussed the various features of the visualization technique, and illustrated their usefulness by applying it to data sets from very different domains.

We performed an formative evaluation of an early prototype of the tool in form of a small user study with five

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

Visualization technique	Free of crossings	Compactness	Following a path	Dynamic
Node-Link	no	no	yes	difficult
Matrix	yes	high	difficult	no
Timeline Trees	yes	high	difficult	yes
TimeRadarTrees	yes	high	difficult	yes
Hierarchical Edge Bundles	no	high	difficult	no
ArcTrees	no	medium	difficult	no
Graph Links on Treemaps	no	high	difficult	no

Table 1: *Comparison of visualization techniques*

PhD students not involved in the development of the tool. We found that the participants learned quickly how to use the tool and were able to explore the data set and make interesting observations. Motivated by their feedback we plan the following improvements of the system in the near future: better user control of the animation, a zoom function (similar to the inner-detail technique of Sunburst) to better exploit radial distortion of the time radar as a focus+context technique, and a prune function to remove subtrees and their related data completely. We are currently performing a task-oriented evaluation of the approach comparing it with a similar non-radial visualization We hope to publish the results in the near future.

References

[AH98] ANDREWS K., HEIDEGGER H.: Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs (late breaking hot topic paper). In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS'98)* (Research Triangle Park, NC, 1998), pp. 9–12.

[BBD08] BURCH M., BECK F., DIEHL S.: Timeline trees: Visualizing sequences of transactions in information hierarchies. In *Proc. of 9th International Working Conference on Advanced Visual Interfaces (AVI2008)* (Naples, Italy, 2008).

[BD06] BURCH M., DIEHL S.: Trees in a treemap. In *Proc. of 13th Conference on Visualization and Data Analysis (VDA 2006)* (San Jose, California, 2006).

[BM99] BERTAULT F., MILLER M.: An Algorithm for Drawing Compound Graphs. In *Proc. of 7th International Symposium on Graphdrawing, GD* (1999), vol. 1731 of *LNCS*, Springer, pp. 197–204.

[ET03] ELMQVIST N., TSIGAS P.: Causality visualization using animated growing polygons. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS'03), Los Alamitos, CA, USA* (2003), IEEE.

[FT04] FRISHMAN Y., TAL A.: Dynamic Drawing of Clustered Graphs. In *Proc. of 10th IEEE Symposium on Information Visualization, INFOVIS* (2004), IEEE Computer Society, pp. 191–198.

[FWD*03] FEKETE J.-D., WAND D., DANG N., ARIS A., PLAISANT C.: Overlaying graph links on treemaps. In *Poster Compendium of the IEEE Symposium on Information Visualization (INFOVIS'03), Los Alamitos, CA, USA* (2003), IEEE.

[HHWN02] HAVRE S., HETZLER E., WHITNEY P., NOWELL L.: Themeriver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 9–20.

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualizing of adjacency relations in hierarchical data. In *Proc. of the IEEE Transactions on Visualization and Computer Graphics, 12, 741-748* (2006), IEEE.

[JS91] JOHNSON B., SHNEIDERMAN B.: Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proc. of IEEE Visualization Conference* (San Diego, CA, 1991), pp. 284–291.

[NSC05] NEUMANN P., SCHLECHTWEG S., CARPENDALE S.: Arctrees: Visualizing relations in hierarchical data. In *Data Visualization, Eurographics/IEEE VGTC Symposium on Visualization* (Aire-la-Ville, Switzerland, 2005), Brodlie K. W., Duke D. J., Joy K. I., (Eds.).

[RT81] REINGOLD E. M., TILFORD J. S.: Tidier drawing of trees. *IEEE Transactions on Software Engineering 7*, 2 (1981), 223–228.

[SM91] SUGIYAMA K., MISUE K.: Visualization of Structural Information: Automatic Drawing of Compound Digraphs. *IEEE Transactions on Systems, Man and Cybernetics 21*, 4 (1991), 876–892.

[Swa84] SWANSON R.: A unifying concept for the amino acid code. *Bulletin of Mathematical Biology 46* (1984), 187–207.

[SZ00] STASKO J., ZHANG E.: Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proc. of the Symposium on Information Visualization (InfoVis'00), Salt Lake City, UT* (2000), IEEE Computer Society Press, pp. 57–65.

[YFDH01] YEE K.-P., FISHER D., DHAMIJA R., HEARST M.: Animated exploration of dynamic graphs with radial layout. In *Proc. of the IEEE Symposium on Information Visualization, San Diego, CA, USA* (2001).

[ZDZ03] ZIMMERMANN T., DIEHL S., ZELLER A.: How History Justifies System Architecture (or not). In *Proc. of International Workshop on Principles of Software Evolution IWPSE 2003, Helsinki, Finland, September* (2003).

[ZMC05] ZHAO S., MCGUFFIN M. J., CHIGNELL M. H.: Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS'05), Minneapolis, MN, USA* (2005), IEEE.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.