Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Improving Stability and Compactness in Street Layout Visualizations

Julian Kratt & Hendrik Strobelt & Oliver Deussen

University of Konstanz, Germany

Abstract

We present and evaluate improvements for Street Layout, a technique that can be used for visualizing evolving hierarchical data such as file structures or software systems. Street Layouts represent data as street networks, where each street represents a branch of the hierarchy and buildings around streets represent leaves. We extended the initial idea in various ways to increase compactness and visual stability. Our approaches are compared against the current methods in a conducted technical evaluation. A prototypic application shows the applicability of our improvements for visualizing a real world data set.

1. Introduction

One of the main goals of information visualization is to use visualization space efficiently in order to convey as much information as possible in the most efficient way. An additional role is to communicate clear information in a way that the users don't get lost. These two requirements are very generic and encompass any kind of visualization design. In this paper, we build on top of the recently proposed *Street Layout* algorithm [Ste10] and propose a series of techniques to make it more space efficient and more stable under data changes.

While space efficiency is broadly accepted as a fundamental requirement for visualization, stability has gained some interest only recently. The stability problem arises as soon as we shift from the static repository paradigm to a dynamic one, where date evolution plays a role. Also compactness and stability are interrelated in an antipodal manner. Intuitively, the more compact a visualization design is, the harder it is to keep the layout stable under data changes.

In this paper we first discuss the limitations of the existing Treemap and Street Layout algorithms (Section 3). This is followed by a set of improvements for compactness and stability (Section 4). The study is supported by an extensive experiment based on synthetic data comparing our solution to existing ones. For each improvement we demonstrate the benefits and discuss the implications. In particular, in Section 5.4 we provide a summary of our findings and suggest how to use combinations of our techniques according to the different task and data scenarios one may encounter. Finally we discuss in Section 6 the use of our prototypic implementation in a real-world scenario.

2. Related Work

From the variety of visualization methods for hierarchies that are available we focus on Treemaps algorithms and City Layout algorithms and refer to loosely related work towards the end of this section.

Treemaps are a commonly-used approach for visualizing hierarchies. Each of them follows the same strategy. The Treemap represents the hierarchy data via recursive subdivision of the given area. The size of one region corresponds to the size of the element in the hierarchy. This space-filling method produces layouts without holes or overlapping regions. Thus, the entire area is used as efficiently as possible. Johnson and Shneiderman [Shn92] introduced the first Treemap algorithm called Slice and Dice. This algorithm recursively traverses the hierarchy and subdivides each area horizontally and vertically alternating between the two. As a result, rectangles with high aspect ratios can emerge. The Squarified Layout [BHvW00] addresses this issue and produces rectangles that have aspect ratios close to one. The Strip Layout algorithm [BSW02] is a modification of the Squarified Treemap algorithm. In comparison to the Squarified algorithm the elements are processed in or-

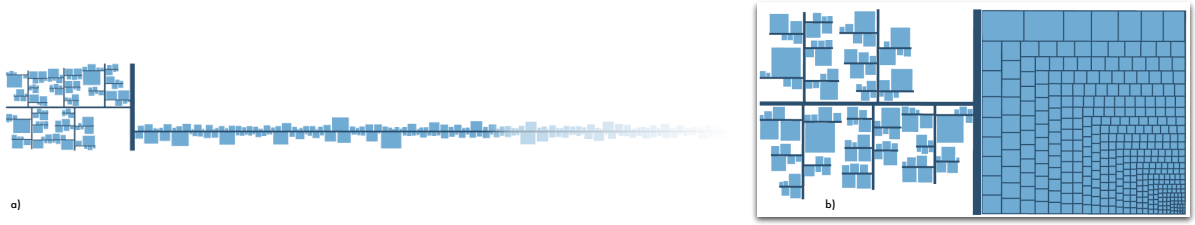

Figure 1: *Exemplified problem with a long street in Street Layout (a) vs. the street folded into a Treemap (b).*

der and laid out in horizontal or vertical strips of varying thicknesses. Quantum Treemaps [BSW02] produces rectangles with widths and heights that are integer multiples of a given elemental size. Thus, a constant aspect ratio is guaranteed and the elements can be laid out in a grid. Spiral Treemaps [TS07] arrange the elements along a spiral to achieve a good balance in stability, continuity, readability and average aspect ratio. However, the readability is not as good as with the Squarified Treemap. Cushion- [VWvdW99] and Cascaded Treemaps [LF08] extend Treemaps by additional shading of the rectangles and by using cascaded rectangles instead of the traditional nested ones, both to increase readability. Mixed Treemaps [VvWvdL06] combine the several standard algorithms. Not all Treemap approaches use rectangular shapes to represent the hierarchy. In [BD05] a polygon-based two-dimensional subdivision is proposed.

A number of other techniques for visualizing hierarchical data has been developed in the field of software visualization. There, suitable visualizations are needed to represent static and evolutionary aspects of a software system. In [WL07] a modified Treemap algorithm is proposed to present static attributes of a given software system. The visualization uses a city metaphor to ease program comprehension. Thus, each inner node of the hierarchy can be seen as a district and the leaf nodes are considered as buildings that reside in those districts. The algorithm tries to solve a two-dimensional rectangle-packing problem, which takes a set of rectangles and places them into a rectangle of minimum area. This approach is not a space-filling technique at all, consequently freespace can occur.

The Evo-Street Layout approach [Ste10] maps the hierarchical structure on a hierarchical street system. The entire system is represented by a main road, from which the subsystems form branching streets. This is similar to a hv-drawing of a graph [CDBP92].

[ZMC05] proposes a combination of Treemaps and standard node-link diagrams. In contrast, our approach aims to combine traditional Treemaps with the already mentioned Street Layout to gain the advantages of both visualizations. Another combination of existing techniques can be seen in [Fek03], where Treemaps are enhanced with node links to visualize the hierarchical structure.

Kleinberg et al. [KWW] provide a mapping of the hierarchy onto a 3D botanical tree structure where long branches are contracted and replaced by spheres. Using a 3D visualization, typical problems like occlusion, problems with navigation, etc. occur, which is out of our focus of study.

3. Analysis

In this section we analyze Treemaps and the Street Layout approaches in more detail with a focus on their stability and compactness properties.

3.1. Treemap Visualizations

Each Treemap algorithm has its own strengths and weaknesses. The Slice and Dice algorithm comes up with a substantial problem. In every step the area is subdivided only in one dimension. Consequently thin elongated rectangles with a high aspect ratio can emerge. Such rectangles are hard to see, select, compare in size and label ([TJ92], [BHvW00]). In contrast, the Slice and Dice method creates a high stable layout [SW01].

The presentation of square-like rectangles in the Squarified algorithm has several advantages: First, such rectangles use the available space more efficiently than non-square rectangles. Next, square items are easier to detect, point at and it is easier to compare two different squares with approximately the same aspect ratio ([BHvW00]). But there are also drawbacks to this algorithm. If the underlying data structure is a tree with leaf nodes of equal size, then the layout degenerates into a regular grid. The hierarchical structure of the data is hard to see. Another problem is the initial sorting of the elements and the way the elements are laid out. This causes the layout to become very unstable, because each change in the data causes a structural change in the layout.

The Strip Layout is a modification of the Squarified Treemap and processes the elements in a given order, laying them out in horizontal or vertical strips of varying thicknesses. Accordingly, the aspect ratio is not as good as with the Squarified approach. On the other hand, the non-sorting leads to higher stability than the Squarified layout.

	stability	compactness	aspect ratio
Slice Dice	+	+	-
Squarified	-	+	+
Strip Layout	0	+	0
Street Layout	0	-	+
Our Approach	see 4.2	see 4.1	

Table 1: *Comparison of Treemaps with the Street Layout. Our approach improves the default Street Layout in terms of visual stability and compactness. The sections in which these aspects are discussed are listed in the table.*

3.2. Street Layout Visualization

The Street Layout has several benefits. The use of the city metaphor provides a better orientation in the layout than one with an arbitrary mapping. The user is familiar with the metaphor making certain parts of the layout easier to remember and recognize. Furthermore, all leaf nodes and thus all data files of the hierarchy are represented as squares. It is therefore very easy to compare elements in their size. Another advantage is that the hierarchical structure is clearly visible. This is achieved by adapting the width of the streets to the depth within the hierarchy. With increasing depth the streets getting narrower.

The Street Layout concept comes up with some drawbacks. One major problem is that the degree of compactness depends on the underlying data. The hierarchy can be unbalanced, which means that the number of child nodes between the parent nodes in a hierarchical level differs largely. If the hierarchy is unbalanced, then a lot of freespace occurs in the layout. As a result, the length of the street in relation to others appears too long, lowering space efficiency and therefore readability significantly. Figure 1 exemplifies this.

In summary, both Treemaps and the Street Layout approach have their strengths and weaknesses. Table 1 gives an overview over the mentioned aspects. Our approaches address the drawbacks. As part of our work we evaluate the benefits of combining both ideas.

4. Techniques

4.1. Improvements for Compactness

We propose three techniques to cope with freespace in Street Layouts (see 3.2). These extensions can either be used individually or combined. The user is able to apply all extensions interactively.

4.1.1. Combination of Street Layout and Treemaps

The first way of dealing with this problem is to substitute long streets with a Treemap representation. This Treemap consists of all the elements that are aligned on the street. For this purpose any of the Treemap approaches can be used. It is therefore possible to choose the one that produces the best results in terms of compactness and readability. Because Treemaps are space-filling techniques the highest compactness within their area is guaranteed. Substitution is possible on each level of the hierarchy; an example is given in Figure 1 b).

4.1.2. Turning the Street

Another possibility to get a more compact representation is to make additional turns on the street. For this purpose the whole street is subdivided into side streets of the same length. In the next step all neighboring side streets are connected.

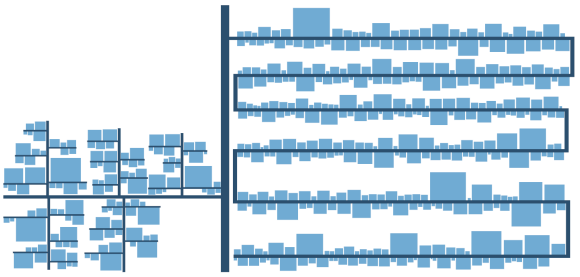

Figure 2: *Street Layout Visualization. The right side street branching off the main road is folded to increase compactness.*

The number of necessary turns to produce an optimal result is determined by the aspect ratio of the bounding box enclosing the turned street. This aspect ratio should be close to one. To avoid overlaps in the visualization the distance between two side streets depends on the size of the biggest opposite elements. This can lead to increasing freespace if two huge elements oppose each other. Figure 2 gives an example of the layout and shows the sparse usage of space between turns.

4.1.3. Turning the Street with Additional Sorting

To increase density when turning the streets, we propose sorting the elements by size. This lowers the probability that two elements with a significant difference in size are opposite to each other (Figure 3).

4.2. Improvements for Stability

We present two approaches to gain a more stable Street Layout under data changes.

4.2.1. Local Recalculation of the Layout

The basis of a Street Layout visualization is the underlying data, which is a hierarchical tree. In this tree only the leaf nodes represent the real data files. The inner nodes for example are the packages in a software system or the directories in a file system. If there is one change in the data, then in most cases only a particular part of the tree is affected and

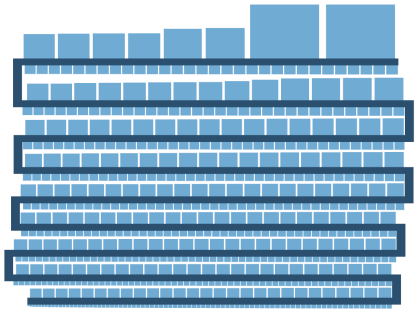

Figure 3: *Folded Street Layout with additional sorting.*

not the whole tree. This can be used to reach a higher degree of stability.

Instead of recalculating the entire layout, only the part of the layout where the changes occur is reconfigured. Afterwards, the visualization has to be checked for overlapping regions. If there are any the next upper level in the hierarchy has to be recalculated. In a best-case scenario, the layout is only rearranged locally. But in the worst-case, the change in the data is propagated to the root node and the whole layout is recalculated. Mostly, the change affects only a few levels in the hierarchy, with the effect that the visualization becomes more stable especially in combination with the next approach.

4.2.2. Increasing Freespace

As described, what sometimes happens is that a change in the hierarchy is propagated further and further up so that the entire layout has to be designed from scratch. In order to minimize this worst-case, each data element is assigned more freespace. The freespace is assigned around the element with a given percentage of the element's occupied space. By doing this, the probability of only having to perform a local recalculation increases, the more additional freespace is assigned to the elements.

5. Experimental Design and Results

5.1. Measuring stability and space-efficiency

To measure the stability of a layout we make use of the *distance change function* (DCF) [SW01]. This function is a metric on the space of Treemap Layouts and can be extended to the space of Street Layouts, because both layouts are based on the same graphical primitives - rectangles. Thus, it is possible to compare the different layouts with respect to the stability property. The DCF requires the definition of a metric on the space of all rectangles. In our case, the euclidean distance is selected. Therefore, the DCF is the average euclidean distance between each pair of corresponding rectangles in a layout. The higher the value of this function, the more unstable the layout is considered. Consequently, a low value is desirable.

For the calculation of compactness of a layout we determine the effective use of space by measuring the ratio between the used and the total area. This ratio shows the percentage of the entire space, that is used effectively. To ensure that two distinct layouts can be compared, each layout is bounded by a 1x1 square. The metric that is related to space-efficiency is

$$A = \sum_{k\ node} a(k)\ ,\quad 0 \leq A \leq 1 \text{ (total area [MR10])}$$

Let $a(k)$ be the area, in which node k occupies in the layout. Furthermore, it must be ensured that no area is counted twice. This is particularly relevant if there is a Treemap visualization. Thus, all overlapping regions have to be subtracted. Speaking of *compactness* in the subsequent text, we refer to this measure.

5.2. Data generation and experimental design

For generating the data and designing the experiment we followed the method described in [SW01]. We have performed two different experiments. For each one the size of the data elements were drawn from a log-normal distribution with mean 0 and variance 1.
The first experiment was a sequence of Monte Carlo trials to measure the stability for a given layout. Therefore, we ran 100 trials of 100 steps each. In every step we updated the underlying data by a random variable x, where x is also distributed by log-normal distribution. Finally we took the average distance over all 100 trails. In the second experiment we measured the compactness for a static layout. Here we also ran a total of 100 trials, but without updating the layout.

To get meaningful results the experiments were performed on several hierarchies. Once the hierarchies used in [SW01] were taken (20x1, 8x3). Additionally a 5x4 and a number of flat hierarchies with increasing number of elements were investigated. The 8x3 hierarchy for instance consists of a total of three levels and eight items at each level.

5.3. Results

† The result of a general stability comparison between Street Layout and Treemap is given in Figure 4. The special case of flat hierarchies of different sizes reflects the behavior of subtrees with varying width. It can be seen, that Slice and Dice outperforms all other approaches when faced with very broad subtrees, but especially up to width 200 the Street Layout is slightly below Slice and Dice. Our proposed improvements are now discussed in detail considering stability and compactness in parallel.

† We exemplify insights here. An additional list of experimental results are given as supplementary material.

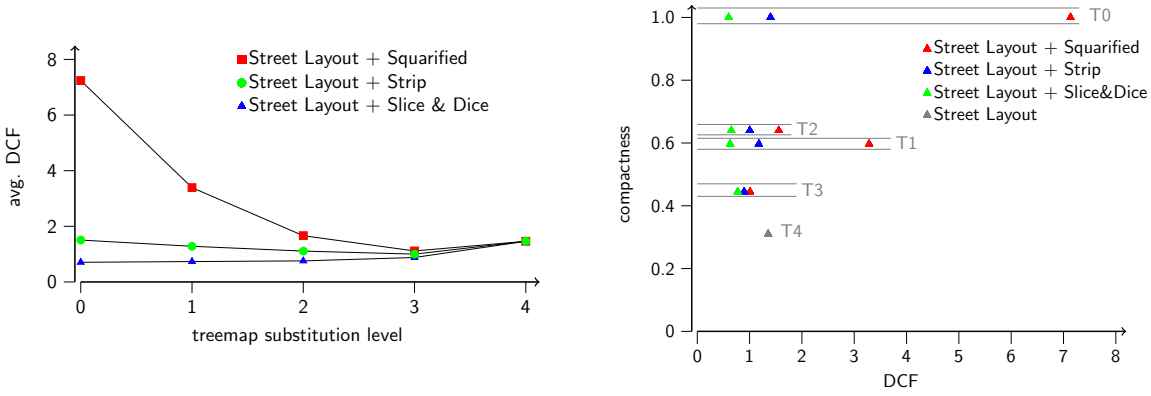

Figure 5: *Stability (on the left) and compactness (both together on the right) analysis of Street Layout in combination with different Treemap approaches based on 5x4 hierarchies. The substitution level indicates the level in the hierarchy from which the elements are represented through Treemaps. Level 0 corresponds to a complete Treemap and level 4 to an unmodified Street Layout.*

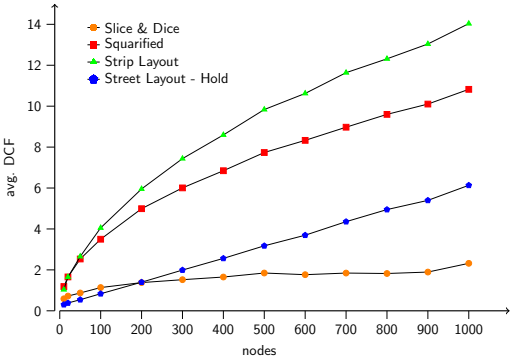

Figure 4: *Stability comparison of Street Layout against Treemap approaches based on a flat hierarchy by increasing number of nodes.*

5.3.1. Street Layout and Treemaps

Figure 5 (left) shows the stability results of combining the Street Layout with Treemaps. For each level of depth in the tree, the branches were substituted by Treemaps. Substitutions on the root level (level 0 - highest) result in a complete Treemap, whilst substitutions on leave level (level 4 - lowest) represent an unmodified Street layout. It can be seen, that the behavior of each Treemap becomes more dominant with a higher level of substitution.

Although Street Layout and Squarified Treemaps are similar in that they favor an aspect ratio (of one) to ease readability, they are contradictory in terms of stability. When combined, a lower level of substitution is recommendable. Using the Strip Layout, no major deterioration in stability can be found. The application of Slice and Dice leads to an improvement of stability when substituting in higher levels. This comes with the drawback of less readability (see 3.1). Substituting lower levels, the DCF can undergo the DCF of a plain Street Layout (level 4). This originates in the difference of local movements within a Treemap (only within a Treemap) vs. within a street (the whole street and neighboring streets have to move).

Figure 5 (right) shows the results for compactness in combination with stability measures for 5x4 hierarchies. Since each Treemap has a compactness value of one, which approach is used for the combination is not crucial for the measurement. Therefore, the results of the Treemap approaches are slotted along the compactness axis referring to the substitution level T0 ... T4. As a global trend, a higher substitution level corresponds to a more compact layout. The trend is interrupted on level one (T1), where compactness decreases extremely. This corresponds to the scenario, that all substitutional Treemaps are along one street. This does not allow aspect ratio optimizations.

Finally, Figure 5 (right) shows that a pure Slice and Dice Treemap is the most stable visualization with the highest compactness (top left triangle). Addressing readability and orientation within a visualization, an alternative could be to use Squarified Treemaps at a midrange level. In general, the Street Layout can clearly benefit from the combination with Treemaps when aiming at higher compactness and/or higher stability.

5.3.2. Turning the Street

The comparison of the Street Layout, Street Layout with turning substreets and additional sorting considering visual stability is illustrated in Figure 6 (left). The comparison is based on flat hierarchies up to 30 elements. From this we can conclude that if streets get folded to gain a more compact layout, the stability is slightly increased. This is caused by the fact that the impact of local refinements becomes greater

if additional turns are used. That means, the number of elements to be moved is only the fraction of elements which reside on the folded part of the side street (up to a full packing).

Figure 6 (right) presents the benefits for compactness when turning a street. It is apparent that a Street Layout without any extensions produces less compactness with an increasing number of elements. Turning the street behaves in a contradictory manner and becomes more effective with increasing number of elements. Without sorting, street turning increases the use of space up to 50 percent. With sorting, up to 80 percent of the area is used.

However, we have observed that turning a street only has a positive impact on the compactness, if there are more than 15 elements aligned on the street. Therefore the impact on the 8x3 or the 5x4 hierarchy cannot be shown.

5.3.3. Increasing Freespace

Figure 7 exemplifies that more freespace leads to more stability. For comparison, the average stability value of the Slice and Dice Layout is also displayed. It is shown that with an amount of 20 percent we reach nearly the same stability as the Slice and Dice Layout.

Considering both attributes leads to the results presented in Figure 8. It allows us to estimate a good tradeoff between stability and compactness using freespace. Taking 20% of freespace, compactness lowers to 30 % and the stability value is approximately 0.8. This corresponds to the stability of the Slice and Dice algorithm with significantly less compactness, but with the benefits of the Street Layout (see 3.2).

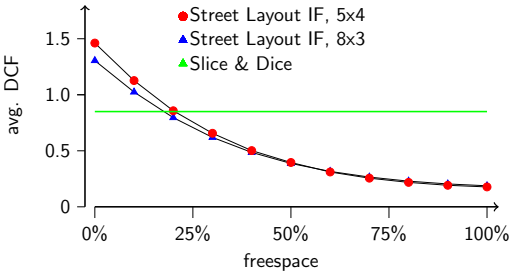

Figure 7: *Increasing freespace analysis of Street Layout based on 5x4 and 8x3 hierarchy compared to the average DCF value of Slice & Dice.*

5.4. Discussion

From the previous section we can conclude several recommendations for using the Street Layout with the developed extensions. To achieve the best results, the context in which the visualization is applied has to be taken into consideration. For a static visualization all the improvements to reach a higher compactness can be used. Long streets can be collapsed into Treemaps or streets can be folded. When folding

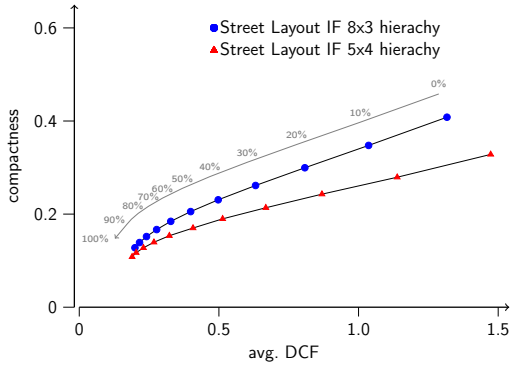

Figure 8: *Compactness and stability analysis of Street Layout with increasing freespace. The amount of additional space is shown with the gray line.*

a street the number of elements has to be large enough, otherwise the folding causes the opposite and the space is used less efficiently.

In a dynamic context, there are additional aspects that need to be considered to get the best results. If the combination with Treemaps is performed on the lowest level in the hierarchy all Treemaps can be used for the substitution. The higher the level of substitution, the more preference should be given to the Slice and Dice or Strip Layout. Assigning additional freespace to the elements also has a positive effect on stability. It was shown that a good tradeoff between stability and compactness can be achieved within a range of 10-20 % additional freespace.

6. Application Scenario

So far, we have seen several extensions for the Street Layout only applied to synthetic data. This was done to provide a better understanding of the way each extension works. In this section the techniques for increasing compactness are applied to a real-world data set to demonstrate the necessity of our work. The underlying data of the layout is taken from the OpenSceneGraph project. OpenSceneGraph is an open-source high performance 3D graphics toolkit; all revisions of the development process are available. We took the recent 500 revisions. From this it is possible to determine the amount of modification of each data element over the last 500 development stages. Therefore we calculate the standard deviation over all sizes and map it on a color interval. The area of one node corresponds to the average size over all revisions. Figure 5.4 a) shows the entire project using the Street Layout without any improvements. It can be seen that a lot of space is unused due to long streets. To provide better compactness and to ease the comparison in color between branches, all the elements on the branch B2 and B3 are collapsed to Treemaps. The result is presented in figure 5.4 b). However, there are also long streets in the layout. To

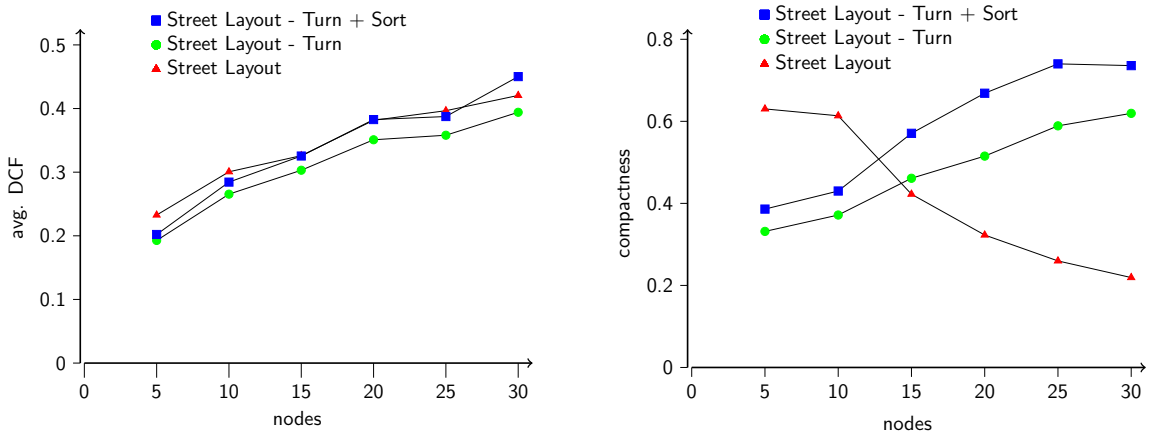

Figure 6: *Comparison of stability (left) and compactness (right) between Street Layout, Street Layout with turning side streets and additional sorting based on flat hierarchies with increasing width.*

increase compactness further, these branches (B1, B2) can be folded (figure 5.4 c)).

From this layout we can extract quite a few interesting facts. It is evident that B4 is a highly modified part of the system in contrast to B5, which was never really changed. This can be explained by taking a closer look at the project. B3 represents the *osgWrappers* subsystem. This subsystem has been restructured in the last few revisions. The original part was labeled as deprecated and a revised version of this part moved into a subfolder. This subfolder is B5. The whole branch is gray, because it is new and the amount of modification is low. B4, instead corresponds to the old, deprecated part, which has been highly modified over the time. Furthermore, B2 represents the *osgPlugin* system, which is one of the main parts of the library. Many of the elements aligned on B2 are gray-colored. This means that there have been no major changes over time. Only some of the elements have been updated.

7. Conclusion and future work

In this paper we present several extensions to the Street Layout algorithm to improve compactness and visual stability. Each of the improvements has been investigated in more detail to show the real impact for the layout. We have seen that the default Street Layout might have a problem with space-efficiency. Using our improvements results in a significant enhancement of compactness. We also showed that under specific conditions we are able to improve stability. Motivated by the results of our technical evaluation, we can provide recommendations about how to apply our extensions in an effective way (Section 5.4). Finally, we visualized a real world data set with the optimized Street Layout and show the benefits of our contribution in comparison with the default Street Layout.

A topic for the future is the evaluation of readability. By folding all the streets into a layout it is possible to achieve a very high level of compactness, but the hierarchical structure is hard to see. It will be interesting to find out more about a tradeoff between compactness and a good understanding of the structure of the hierarchy. Furthermore, it would be interesting how fast users can recognize changes in the visualizations using our approach.

8. Acknowledgments

We thank Enrico Bertini for his very helpful support and constructive feedback.

References

[BD05] BALZER M., DEUSSEN O.: Voronoi treemaps. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, p. 7. 2

[BHvW00] BRULS M., HUIZING K., VAN WIJK J. J.: Squarified treemaps. *In Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000)* (2000), 33–42. 1, 2

[BSW02] BEDERSON B. B., SHNEIDERMAN B., WATTENBERG M.: Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph. 21*, 4 (2002), 833–854. 1, 2

[CDBP92] CRESCENZI P., DI BATTISTA G., PIPERNO A.: A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom. Theory Appl. 2* (December 1992), 187–200. 2

[Fek03] FEKETE J.D D. W.-N. D. A. A. C. P.: Overlaying graph links on treemaps. *Information Visualization 2003 Symposium Poster Compendium* (2003), 82–83. 2

[KWW] KLEIBERG E., WETERING H. V. D., WIJK J. J. V.: Botanical visualization of huge hierarchies. In *In INFOVIS 01: Proceedings of the IEEE Symposium on Information Visualization (2001)*, IEEE Computer Society, pp. 87–94. 2

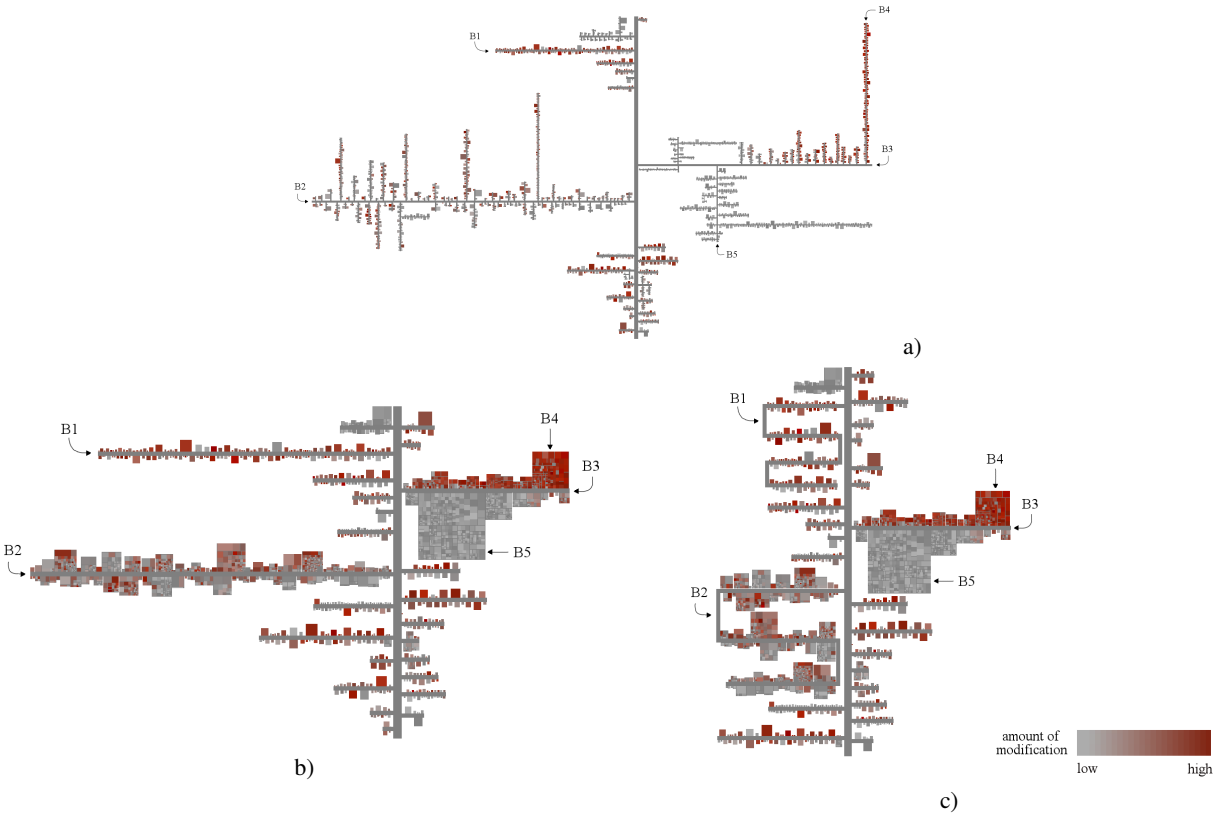

Figure 9: *Visualization of the OpenSceneGraph project using the Street Layout. a) visualizes the system with the default Street Layout. b) and c) show the Street Layout with the developed improvements such as substitutions with Treemaps and additional turning of side streets. Color indicates the amount of modification over the recent 500 revisions. The visualizations are scaled to use nearly same space.*

[LF08] LÜ H., FOGARTY J.: Cascaded treemaps: examining the visibility and stability of structure in treemaps. In *Proceedings of graphics interface 2008* (Toronto, Ont., Canada, 2008), GI '08, Canadian Information Processing Society, pp. 259–266. 2

[MR10] MCGUFFIN M. J., ROBERT J.-M.: Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization 9*, 2 (2010), 115–140. 4

[Shn92] SHNEIDERMAN B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph. 11*, 1 (1992), 92–99. 1

[Ste10] Representing development history in software cities. In *Proceedings of ACM SOFTVIS 2010* (Salt Lake City, Utah, USA, 2010), ACM, p. 10. 1, 2

[SW01] SHNEIDERMAN B., WATTENBERG M.: Ordered treemap layouts. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 73 – 80. 2, 4

[TJ92] TURO D., JOHNSON B.: Improving the visualization of hierarchies with treemaps: design issues and experimentation. In *Proceedings of the 3rd conference on Visualization '92* (Los Alamitos, CA, USA, 1992), VIS '92, IEEE Computer Society, pp. 124–131. 2

[TS07] TU Y., SHEN H.-W.: Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics 13* (November 2007), 1286–1293. 2

[VvWvdL06] VLIEGEN R., VAN WIJK J. J., VAN DER LINDEN E.-J.: Visualizing business data with generalized treemaps. *IEEE Transactions on Visualization and Computer Graphics 12* (September 2006), 789–796. 2

[VWvdW99] VAN WIJK J. J., VAN DE WETERING H.: Cushion treemaps: Visualization of hierarchical information. In *Proceedings of the 1999 IEEE Symp. on Information Visualization* (Washington, DC, USA, 1999), IEEE Computer Society, pp. 73–80. 2

[WL07] WETTEL R., LANZA M.: Visualizing software systems as cities. In *Visualizing Software for Understanding and Analysis, 2007. (VISSOFT 2007)* (2007), pp. 92–99. 2

[ZMC05] ZHAO S., MCGUFFIN M. J., CHIGNELL M. H.: Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 8–. 2