Adaptive Treemap Based Navigation Through Web Portals

Andreas Nauerz  
IBM Research and Development  
71032 Böblingen, Germany  
andreas.nauerz@de.ibm.com

Sirko Schindler  
University Jena  
07743 Jena, Germany  
sirko.schindler@informatik.uni-jena.de

Fedor Bakalov  
University Jena  
07743 Jena, Germany  
fedor.bakalov@informatik.uni-jena.de

Abstract

Today, Portals most often present the navigation in form of tree-like structures to users. These trees follow a representation of data with a natural hierarchy. Single nodes are either parents or children. This representation style has several drawbacks: for reaching the leaves users have to traverse rather long paths. During the traversal of the tree users get often lost and, finally, the content relations remain unclear. It is not obvious where the "most" content is located, too. We regard treemapping (a method for displaying tree-structured data using nested polygons) as a promising technology to overcome these drawbacks. We especially focus on adaptive treemaps displaying different treemaps to different users - always trying to satisfy each single users' individual needs. Our main concepts have been prototypically embedded within IBM's WebSphere Portal.

1 Introduction

Portals provide users with a central point of access to companywide information. Initially they focused on presenting the most valuable and widely used information to users providing them with quick and efficient information access. But the amount of information accessible quickly grew and finding the right information became more and more complex and time consuming. As the amount of content stored in such systems grew, the navigation structures providing access to this content became more and more complex, too. They are most often presented in form of tree-like structures to users. Here, trees are graphs with a specific structure (connected graph with $n - 1$ edges); they follow a representation of data with a natural hierarchy. Single nodes are either parents or children. The result is a complex hierarchical navigation through which users have to navigate over and over again. Administrators are usually concerned with creating structures that satisfy the majority of all users. They are normally not interested in (i.e. do either not have the relevant knowledge or the necessary time) creating the perfect arrangement for each single user.

Tree-like navigation structures have several drawbacks:

- For reaching the leaves people have to traverse rather long paths. Additionally they have to know which places to look at, which can be rather difficult if the administrators view is completely different.
- Users get often lost in these structures ("lost in hyperspace phenomenon").

- It is also not immediately clear how content is related and where the "most" content is located (which node has more children than another) which makes overview browsing difficult.
- Additions and changes that lie deep in the structure of a webpage may not even be noticed though they are of interest to a user.
- The area used to display conventional navigation menus is not utilized to its full capacity leaving vast spaces blank.

Thus, people are seeking new ways for navigating through large information spaces such as Portals. Treemaps, among other representations, seem to be very promising in overcoming these drawbacks. As said, treemapping is a method for displaying tree-structured data using nested polygons. Treemaps display rows of data as groups of polygons that can be arranged, sized and colored to graphically reveal underlying data patterns.

Brian Johnson & Ben Shneiderman in their widely cited paper [Johnson and Shneiderman, 1991] consider the algorithm "best suited to hierarchies in which the content of the leaf nodes and the structure of the hierarchy are of primary importance, and the content information associated with internal nodes is largely derived from their children". The parallels to Portals are obvious: The content of a Portal is served via small artifacts, so called portlets. One or more portlets can be aggregated to a Portal page. One may view Portal pages, navigation pages and portlets as nodes of navigation graphs and thereby meet the definition of Johnson and Shneiderman.

This visualization technique allows end users to easily recognize complicated data relationships that are otherwise non-obvious. It especially gives them a feeling how things are related, prevents getting lost, and shows them where "more or less" content is available at once without the need for much traversal.

Treemaps make 100% use of the available display space and efficiently display large trees in limited space. All the leaf nodes can be viewed. Area-size and color coded attributes of a node can be recognized very easily. So, the advantage of this technique for users is that vast amounts of data can be seen in one fell swoop - and simply. The fundamental data can then be accessed through zooms, sliders, hierarchical views or other ways of drilling down into the information.

This graphical approach also takes advantage of the human ability to extract information from pictures and note even subtle changes.

Our novel interesting extension is the development of adaptive Treemaps: Extracting users interests and pref-
ferences from user models (which we create via web usage mining, tagging behavior analysis etc. as described in [Nauerz and Welsch, 2007; Nauerz et al., 2008b; Nauerz et al., 2008a]) allows us to derive a weight function which assigns a level of importance to each node.

We refer the interested reader to our previous work [Nauerz and Welsch, 2007; Nauerz et al., 2008b; Nauerz et al., 2008a] to learn more about our approaches for constructing user- and context models.

Leveraging information from user- and context models allows us to derive a weight function which assigns a level of importance to each node.

Once the tree is generated we choose between several algorithms to create the actual Treemap. The algorithms can be categorized by the following criteria: the ordering of the siblings may or may not be preserved. When associated weights change in time the Treemap-polygon attached to a navigation node may change its position as well as its size. The rate of changing positions of polygons is called stability of the algorithm and can be measured with certain metrics [Shneiderman and Wattenberg, 2001]. Another point is the aspect ratio of the shown polygons. There may be some rather long but slim polygons. This is of great concern to us, because in real world applications it would be quite difficult to spot or use such polygons effectively.

To date, six primary treemap algorithms have been developed to create the Treemap:

- **BinaryTree** - Partially ordered, high aspect ratios, stable
- **SliceAndDice** [Johnson and Shneiderman, 1991] - Ordered, very high aspect ratios, stable
- **Squarified** [Bruls et al., 1999] - Unordered, lowest aspect ratios, medium stability
- **Ordered/Pivot** [Shneiderman and Wattenberg, 2001] - Partially ordered, medium aspect ratios, medium stability
- **Voronoi** [Balzer and Deussen, 2005] - low aspect ratios, good stability, ordering varying
- **Strip** [Bederson et al., 2002] - Ordered, medium aspect ratios, medium stability

While applying these algorithms we included some extensions to encode additional information:

The basic algorithms always split the whole area of the parent node between its children. This way one can only see leaf nodes. All internal nodes are hidden, although they might contain some useful content which should be accessible, too. To solve this problem we made use of a technique called nesting as already proposed in [Johnson and Shneiderman, 1991]: the algorithm reserves only a part of the parents’ area for their children. This leaves a border area of the parent node visible and thereby accessible.

Since a Treemap is a graphical representation of the hierarchy there are plenty of ways to encode additional information into it:

- **Filling color for each polygon** (hue, saturation, intensity)
- **Style of borders** (solid, dotted, ...)
- **Opacity**
- **Blinking effects**

The additional information encoded can be of various types:

- **Show the semantical relationship between content** (portlets or pages). Relationships could be derived for example from users’ tagging behavior.
Highlight recently updated content, especially w.r.t. content from which we know that a particular user might be very interested in.

Highlight new content that made it into the Portal.

Highlight content that might be of interest a user w.r.t. to a certain task (“context-sensitivity”).

Adapt not only to single users, but also to entire groups, communities, etc.

Visualize the overall interests of all users or certain groups.

Both lists are not complete, but should give an idea of what is possible. Of course one should avoid information overload. If confronted with too much information at once, users can get confused. Then all advantages might get nulled.

Some of the information is better shown in an info box aside. When a user points at a certain area, he sees a box containing additional informations. This way he has more influence on what he sees and at which time. Thereby the Treemap itself has fewer tasks to solve. The information shown in an info box is not limited to metrics of any kind like the ones mentioned above. Here everything can be displayed reaching from a tag cloud to a summary of the page or even a small thumbnail-alike preview.

4 Implementation

To test several algorithms we have developed a prototype which we embedded into IBM’s WebSphere Portal. In addition to the Portal’s ordinary navigation menu, the prototype presents an adaptive Treemap to its users.

First, based on the hierarchical organization of pages and portlets, the prototype constructs a navigation tree. Each node (page or portlet) as a part of this tree is assigned a weight which reflects the assumed importance it has to the user. The weights being assigned are based on the information we gain from the previously mentioned user- and content models. Once the (weighted) tree is generated it is passed to a renderer. So far we have implemented renderers for SliceAndDice, Ordered Treemap and Voronoi Treemaps. The renderer generates a SVG-file which is finally presented to the user.

The SVG (cp. fig 2) enables users to zoom in and out of the Treemap to reveal further details and nodes that were not shown in the overall display. Other features allow limiting the depth of the shown nodes so that only nodes with a certain distance to the root node are displayed.

5 Conclusion and Future Work

The results we got so far from our prototype implementation are quite promising. We are able to display adaptive graphical representations of the entire content stored in the Portal. This allows users to understand the Portal’s structure and to reach content particularly of interest to them more quickly.

In the future we focus on the integration of more sophisticated algorithms for weight calculations. Additionally we plan to include more information, like the ones mentioned in section 3, into the Treemap. One important question here is how much information we can integrate before users get confused. Another point of research is which other Treemap-like algorithms might be applicable.

IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries or both. Other company, product and service names may be trademarks or service marks of others.

References


