From Research to Product: Integrating Treemaps into Enterprise Software

Joseph H. Goldberg, Jonathan I. Helfman, and John Beresniewicz

Oracle USA, 500 Oracle Parkway, MS 2op2, Redwood Shores, CA USA
{Joe.Goldberg,Jon.Helfman,John.Beresniewicz}@oracle.com

Abstract. The difficult journey of introducing a new treemap visualization into enterprise software products is described, highlighting interactions among a research team, product teams, and management. Successful product integration ultimately required multiple iterations of prototyping, review, and redirection, as products were cancelled and modified. Several lessons learned are provided, including the need to build flexible and generic prototypes, cultivate champions, and be tenacious.

1 Introduction

Introducing new user interface (UI) ideas into products at large enterprise software companies can be challenging. Development teams, product teams, executives, and others often view innovation as risky and therefore sometimes resist proposed changes to successful, money-making products. This resistance usually stems from a lack of understanding about the trade-offs between the risks of innovation and the benefits of increased customer satisfaction with associated sales revenues and profits. However, communication between research and product teams may be poor, and corporate methods or processes for introducing innovation may be undefined.

This paper highlights the journey of a treemap visualization concept into enterprise software products at Oracle. The focus is on the interplay between a corporate UI research group and product teams. Design decisions were made at each step, and several rounds of prototype development were required. Lessons learned from this experience are provided to facilitate streamlining the adoption of innovative visualizations and other technologies.

1.1 Treemap

A treemap is a graph that shows hierarchical datasets as nested rectangles, with areas of rectangles conveying a quantitative (or numerical) dimension. The color of the rectangles may also represent an additional data dimension [2]. Treemaps gained significant popularity following Smart Money's introduction of the Map of the Market (Fig. 1, http://www.smartmoney.com/map-of-the-market/), which enables a user to simultaneously monitor hundreds of stock prices [3].

Oracle has a UI research group that investigates new concepts and interaction techniques. Several Oracle product teams expressed interest in treemaps following an

M.J. Smith and G. Salvendy (Eds.): Human Interface, Part I, HCII 2009, LNCS 5617, pp. 243–252, 2009.

early-2004 UI research group proposal. The research group began collecting treemap requirements for enterprise users and obtained demo treemaps from commercial vendors, to determine which commercial version best satisfied these requirements. In early 2005, the research group conducted a usability evaluation in which 10 network administrators used treemaps and hierarchical tables to monitor transactional data. The administrators completed their tasks more slowly when using a larger dataset than when using a smaller dataset, and this difference was twice as great when the administrators used tables than when they used treemaps. The network administrators also missed more information when it was presented in tables rather than in treemaps. Subjective impressions strongly favored treemaps over tables for completing network administration tasks [1].

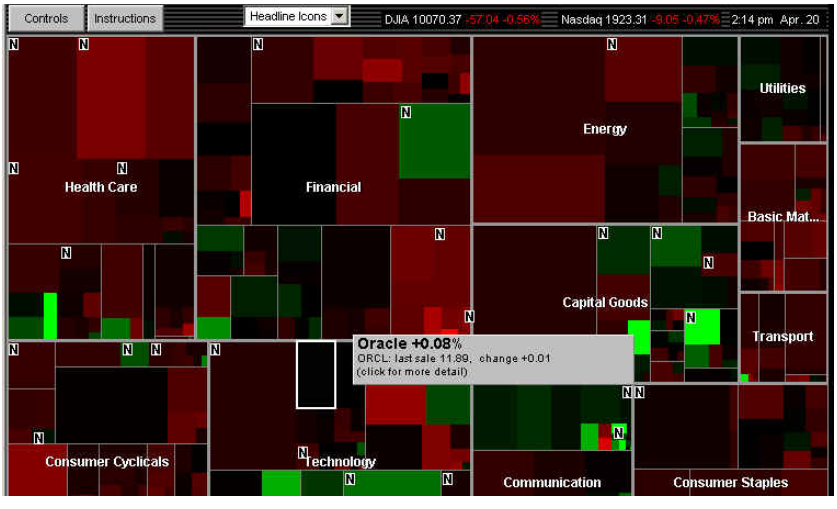

Fig. 1. Smart Money Map of the Market, with area indicating relative market capitalization and color indicating stock price movement for the current day. A two-level hierarchy is shown, with individual stocks grouped by sector. (Used with permission. To license Tree Map software from SmartMoney, email licensinginfo@Smartmoney.com.)

1.2 Interactivity and Enterprise Technology Decisions

User interface innovation, such as a new treemap, carries many potential design tradeoffs in areas such as interaction capabilities and choice of technology platforms. Product requirements and results from user performance testing should ultimately determine these design decisions. Treemap technology decisions include issues such as thick versus thin client, and client- versus server-based data aggregation and rendering. In addition, enterprise software must support large and complex queries on both relational and OLAP databases, providing challenges for applications with user interfaces that include interactive data visualizations.

User experience can differ markedly among different treemap designs with different levels of interactivity. At one extreme, a treemap may be a static image, with or without (1) tooltip feedback, (2) user-defined configuration of data to color and area,

and (3) clickable master/detail drilling. Static treemaps are intended for more casual or occasional users, who primarily consume information. At the other extreme, a treemap may be a dynamic view with or without (1) in-place drilling, (2) hierarchical, area and/or color filtering, and (3) automated updating. Dynamic treemaps are intended for more advanced enterprise users such as analysts or administrators.

A visualization such as an enterprise treemap must be both vertically and horizontally scalable. Vertical scalability refers to the ability to investigate deep data hierarchies that may pivot about multiple dimensions. Horizontal scalability refers to the ability to see and compare many objects (e.g., databases) within a single view. Treemaps can be used to navigate, filter, and show deep, multidimensional data. Their space-filling property enables a user to simultaneously view, prioritize, and interact with hundreds or even thousands of objects at typical screen sizes.

Enterprise applications are developed using a combination of standard components and custom code. A standard visualization component is always a compromise that attempts to implement a majority of the highest-priority requirements from the product teams that will use it. Ultimately, the design of an enterprise visualization component such as a treemap must address significant design trade-offs that are specific to applications. Dynamic updates and interactive features may be sacrificed in support of faster performance. The need for real-time filtering may supplant certain performance requirements in some applications. Two versions of a component may be required in some cases: one that implements basic functionality very efficiently and another that implements sophisticated features but requires more memory. These decisions can also be complicated by difficulties in obtaining useful and accurate information from multiple product teams about how they would prioritize requirements for a visualization component with which they may have little experience.

The treemap discussed here is not yet a standard component in any of Oracle's development toolkits. The entry path into product was through custom-coded components. While product integration is easier using standard components, product teams may be willing to include nonstandard components because their end users can benefit from customized functionality.

2 Oracle Business Intelligence: First Treemap Attempt

In early 2005, the Oracle team that developed a product to create ad hoc queries and business intelligence reports was interested in providing treemaps as data visualizations to their business analyst end users. The team wanted the treemap in their next release, which was to include both a thick-client product for creating and editing reports, Workbook Builder, and a thin-client product for reading reports, Report Center. Intended users of the thick-client product had different interaction requirements and were more technically oriented than intended users of the thin-client product. Workbook Builder users were expected to choose and configure graphs for reports, while Report Center users were expected to mostly read reports that had been prepared for them. Report Center requirements included minimal interactive capabilities, while Workbook Builder had extensive and dynamic interaction, including in-place drilling and interactive filtering.

Differences in data storage between the thick- and thin-client treemaps dictate important architectural differences in treemap layout and rendering. As an example, layout and cell color assignments should logically run on the server because client-side data storage is discouraged in thin-client treemaps. From a usability perspective, this can cause a frustrating and slow visual refresh for interactive tasks such as filtering data.

The UI research group worked with both product teams to design a flexible treemap component with Java classes to support a rich API that could be used to load hierarchical data into treemaps, compute layouts, and associate treemap cells with colors. The component was designed to render to Java (Workbook Builder) or HTML/JavaScript (Report Center). The product teams and their management approved the design.

The research group subsequently helped to integrate the interactive Java applet treemap and the HTML/JavaScript treemap into Workbook Builder (Fig. 2) and Report Center, respectively. The Java applet implemented business intelligence requirements to control the visibility, coloring, and text labeling of the treemap cells.

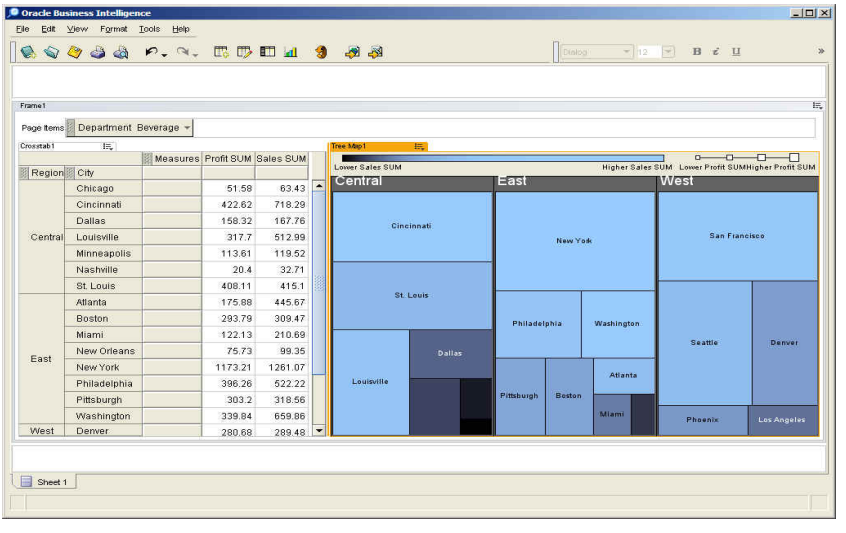

Fig. 2. Workbook Builder screen, merging a flexible hierarchical table with a treemap separated by a vertical splitter bar. Selecting a cell on either component highlights the same data on the other component. Dragging and dropping the table's row and column headers pivots and reaggregates the data, causing the treemap to recalculate its layout and re-render.

It had deep interaction, enabling end users to select dimensions, filter dimensions based upon color and area, customize graphical notifications, and trim the depth of the visible hierarchy. The Report Center treemap cells were implemented as HTML elements with cell borders, background colors, and textual content specified by using cascading stylesheets. JavaScript code was developed to provide a tooltip and to handle selection events on cells.

In late 2005, Oracle purchased Siebel Systems, which had a competing ad hoc reporting product, and both Workbook Builder and Report Center were cancelled. In

response, the UI research group worked with both product teams to amend and reprioritize the treemap requirements. In the first half of 2006, the research group developed a JavaScript treemap that used animation while zooming and helped define dialogs for configuring treemaps. The product teams were then reorganized and development priorities changed. The thin-client code was to be integrated into a Siebel product, but later fell out of scope.

Although the treemap was not successfully integrated into a product at this point, the extensively featured thick-client prototype completed by the UI research group provided a flexible testbed for other teams exploring the use of treemaps in their products. One of these teams was Oracle Enterprise Manager.

3 Enterprise Manager: Second Attempt

Oracle's Enterprise Manager (EM) is a software tool through which IT administrators can monitor and manage the availability and performance of IT infrastructure, applications, and databases. EM is a complex, three-tier Java application comprising thousands of pages and hundreds of use cases.

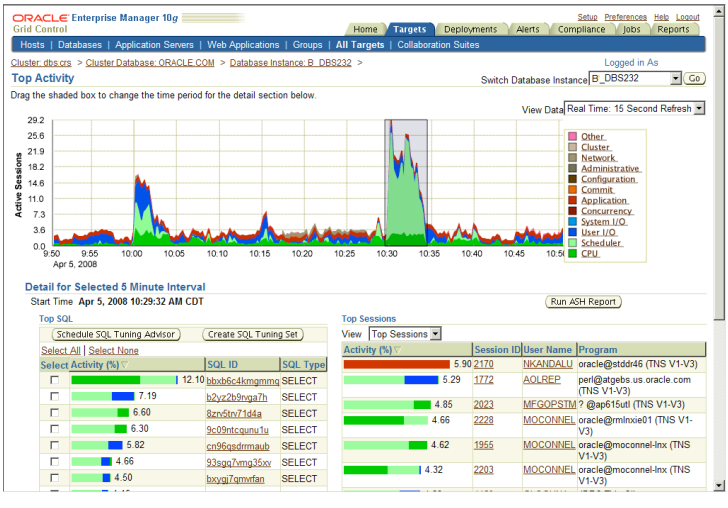

Fig. 3. Enterprise Manager 10g Database Top Activity page. The top chart graphs DB Time over time as a set of stacked line charts by wait class, with preassigned colors for each class. The bars below show the highest accumulators of DB Time by other dimensions within the user-draggable highlighted area in the upper chart.

The end users are highly technical and frequently select and filter data dimensions as they diagnose system issues. Release 10g of the Oracle database introduced the concept and instrumentation for measuring Database Time (DB Time) as the primary internal measure of database activity and performance. Stripchart visualizations of DB Time were included in the user interface to support database tuning and performance diagnosis, where an end user identifies and reduces the largest sources of DB

Time accumulation. In these screens, larger amounts of DB Time display with a larger visual footprint, attracting user attention for subsequent drill-down analysis. This usage metaphor ("click on the big stuff") is easily explained to and adopted by users. Two important visualization properties of DB Time are its inherent multidimensionality (some 30 dimensions of potential interest are instrumented and captured) and scalability within dimension (dimensions could potentially include thousands of values).

The EM Top Activity page (Fig. 3) has been both commercially successful and popular with users. The page facilitates interactive diagnosis, enabling the user to see unusual accumulations of DB Time then isolate specific time periods for details. The details show ranked contributors to DB Time during the selected time period as color-coded stacked bars for correlation with the main chart. User-selectable dimensions and navigation drill-down to dimension-specific tuning tools are also available.

3.1 Prototyping DB Time Treemaps

EM product architects started exploring treemaps for visualizing DB Time after reading the 2004 UI Research group report. The Top Activity page displays two independent dimensions as lists, but requires scrolling to see all elements. The ability of a treemap to integrate different dimensions and quickly identify large DB Time accumulations would combine these lists into a single nonscrolling view. The research group's Java treemap provided a testbed to conduct visualization experiments of production DB Time data (Fig. 4).

The Java applet enabled users to explore different dimensional hierarchies, coloring schemes, and simulated real-time updates. Several diagnostically useful treemaps

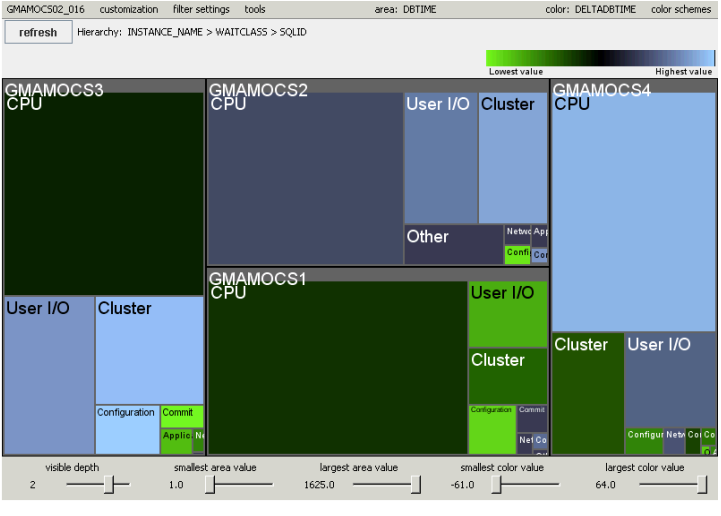

Fig. 4. Java treemap experiment showing breakdown of DB Time spent in an Oracle database using the INSTANCE_NAME>WAITCLASS>SQLID hierarchy. The visible depth level is set to show the instance name and wait class but not SQLID. Sliders enable filtering by low/high color values, large/small areas, and visible depth level.

using DB Time for cell size were generated and a presentation of results was circulated, resulting in significant interest.

Extending from this initial enthusiasm, a prototype treemap viewer was developed in early 2007. Preconfigured sets of DB Time dimensions were loaded into the treemap using predefined queries in order to demonstrate its usefulness for performance analysis on production databases. After starting the prototype, further work would require formal project staffing and resources. Despite substantial interest in the treemap, the existing EM Top Activity page was considered more than adequate for users. In addition, the availability of resources to integrate and maintain the treemap was unclear. The potential benefits of a new treemap view didn't appear to outweigh its potential costs. At this time, treemaps were considered an interesting research area, but not cost-justified for production development within EM.

3.2 DBA Console: A New Opportunity

A decision was made in early 2008 to have administrator-specific login pages for the next version of EM, including a homepage console for database administrators (DBAs) that would summarize database performance across the enterprise. A proposal was developed that showed each database as a treemap cell, sized by DB Time accumulation. This treemap ("Enterprise Loadmap") would be the centerpiece of the DBA console page, which would also show a list of databases that are "down" (not bearing load and thus not appearing in the treemap) as well as a list of recent database incidents. The treemap would be implemented as an Adobe Flex component to ease the integration process with the existing product, which already included other Flex components.

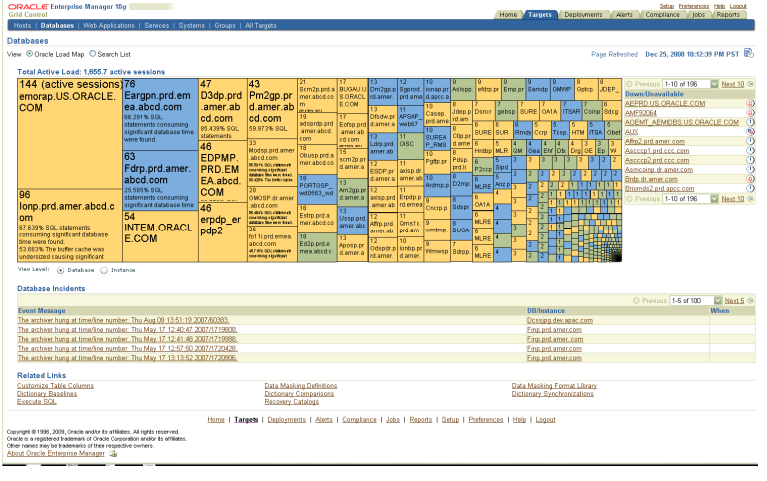

Fig. 5. Enterprise Database Loadmap, a treemap showing DB Time accumulation across an enterprise. Cell size is proportional to database load, and color indicates dominant time component: CPU, I/O, or wait. Cell text shows recent diagnostic findings, and cell click-through navigates to the Top Activity page (Fig. 3) for that cell.

Whereas the earlier treemap DB Time experiments focused on analyzing how time accumulates within a single database, the console Enterprise DB Loadmap (Fig. 5) shows where DB Time is accumulating across an entire enterprise of databases. Databases accumulating more DB Time have visually larger cells and this real estate is used to display database-specific diagnostic information. Busier databases are usually more important to watch, so the visualization favors this natural prioritization. Cell color indicates the dominant component of DB Time (CPU, I/O, or wait). The treemap's efficient use of space scales well to the 10 to 500 (or more) databases characteristic of larger data centers.

The treemap solved the problem of how to show many database targets in a single view, organized so the most important are most accessible and information-rich. Decision makers clearly understood these advantages over list-and-search oriented solutions, and approved the design for product inclusion. The DB Time treemap was once again scheduled to be included in a product.

A final challenge for the treemap surfaced in mid-2008 when the scheduled EM release (including the DBA Console) was suspended in favor of developing the next incremental version of the prior release, which did not include the console. The team quickly developed a proposal for including the treemap into this incremental release as a user-selectable alternative to an existing screen listing all databases in the enterprise. The proposal for inclusion was accepted, receiving very positive reception from an audience of EM executives. The version of EM including the Enterprise Loadmap is currently scheduled for release. Since then, other EM groups have also expressed a desire to explore similar visualizations over other datasets.

4 Discussion

The preceding examples used a variety of approaches for showcasing treemaps in enterprise products. Use cases include monitoring databases, responding to slow database instances, querying sales data, and reporting sales data. Technologies include Java, HTML/JavaScript, and Flex. Each of the treemaps had tooltip feedback and selection capability to expose details about a cell. Otherwise, interaction levels differed in the various prototypes. The business intelligence thick-client treemap supported interactive filtering, in-place drilling, and coordinated highlighting with a hierarchical table. The Enterprise Loadmap enabled users to control depth level, but didn't support filtering to choose a subgroup of the results.

While the level of interaction needs to be matched to the expected user tasks, this matching becomes quite a challenge for a general component that is to be used across multiple applications with multiple user roles and different tasks. We found that having at least one prototype that implements a very large and representative set of the requirements is invaluable for allowing product teams to experiment with the features. Product teams need to be able to configure the prototype with representative customer datasets to see how the features apply to actual customer tasks. The more product teams that prioritize the requirements, the more likely the design of a reusable component will meet the interaction requirements for most of the products.

Why did the Enterprise Loadmap get into product, but not the earlier DB Time treemap prototype? The difference was that the Enterprise Loadmap addressed an unresolved issue: it provides the DBA with a comprehensive overview of the enterprise, with easy access to details or drill-down into specific databases. It is intuitive and easily readable by anyone familiar with the problem domain because databases are what matters most to a DBA, and each database can be seen as a treemap cell, the most important databases having the largest cells.

Product innovation is essential for companies to compete and grow; yet the path for incorporating new ideas into products is not always clear or efficient. The present treemap case studies illustrate how an established concept can be included in different enterprise products, while adapting and morphing to suit individual product and user requirements. Successfully growing organizations should strive toward eliminating the technical and social barriers to include new ideas.

The present exercise showed that both great ideas and attention to interpersonal relationships are needed to integrate a new, innovative concept into an enterprise product. While great ideas are timeless, significant effort must also be spent convincing decision makers that there is an added value from the concept. Though several ups and downs were encountered in the present case, a product team eventually adopted the ideas.

Lessons learned from this project can be divided into those related to developing realistic concepts that provide solutions to real problems, and those related to gaining uptake from product teams.

Concept and Prototype Development

- Ensure that the new concept provides a solution to a well-articulated problem.
- Remain flexible on technology details, as the version that makes it into product will likely require a different implementation.
- Separate ideas from underlying technology. Changing technology may make previously impractical ideas feasible.
- Create a prototype implementing as many requirements as possible so different teams can use it to understand which capabilities they find most useful.

Gaining Product Uptake

- Large companies may move slowly when implementing new solutions, due in part to the complexity of integrating and communicating across many diverse teams.
- Even though potential benefits of new ideas can outweigh their cost of development, constant juggling of project leadership and release schedule priorities may prematurely terminate an innovative project.
- If one implementation avenue fails, try another. Once one team uptakes an idea, other teams may follow.
- Integration of great ideas into existing products requires great timing, in addition to good usability and marketability. This requires management of relationships, not just management of ideas.
- Find and cultivate champions that can shepherd the new concept through the ups and downs of product adoption.

- Product and development teams are most likely to understand a complex visualization if it shows their own data, with demonstrations that solve their problems. Many need to be able to see meaning in their own data to recognize the value of an unfamiliar visualization technique.

References

1. Goldberg, J., Helfman, J.I.: Enterprise Network Monitoring Using Treemaps. In: Proc. 49th Ann. Hum. Factors & Ergo. Soc., pp. 671–675. Santa Monica, HFES (2005)
2. Shneiderman, B.: Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. ACM Trans. on Graphics 11(1), 92–99 (1992)
3. Wattenberg, M.: Visualizing the Stock Market. In: Proc. CHI 1999, ACM/CHI, pp. 188–189 (1999)