Harnessing the Web Information Ecosystem with Wiki-based Visualization Dashboards

Matt McKeon

Abstract—We describe the design and deployment of Dashiki, a public website where users may collaboratively build visualization dashboards through a combination of a wiki-like syntax and interactive editors. Our goals are to extend existing research on social data analysis into presentation and organization of data from multiple sources, explore new metaphors for these activities, and participate more fully in the web's information ecology by providing tighter integration with real-time data. To support these goals, our design includes novel and low-barrier mechanisms for editing and layout of dashboard pages and visualizations, connection to data sources, and coordinating interaction between visualizations. In addition to describing these technologies, we provide a preliminary report on the public launch of a prototype based on this design, including a description of the activities of our users derived from observation and interviews.

Index Terms—visualization, web, social software, wikis, social data analysis, collaboration, dashboards, visual analytics.

1 INTRODUCTION

Historically, visualization developers worked in the context of a single application. This application was typically architected in a manner similar to the visualization pipeline pattern described by Chi & Riedl [6] and codified by Card et al [3] and later Heer & Agrawala [7][8]: data is loaded and transformed, visual encodings are applied to convert data into symbols, and then views are constructed from those symbols.

The rise of web-based information visualization has brought new considerations to light. No visualization application on the web exists in a vacuum; instead, every visualization is a potential participant in a diverse information ecology. Thanks to the proliferation of standards for data and media, multiple web applications may be combined into visualization mashups with reusable components. A data source may be a table on a web page, an Atom feed, or an online office application such as Google Docs. Services such as Yahoo Pipes and DabbleDB provide users with the ability to combine, sift, and transform data from multiple sources into a single dataset. One may choose from a wealth of online visualization providers (such as Many Eyes, Google Maps, iCharts, or widgenie) to create views of this data. Finally, a number of options exist for the user to aggregate, present, and use these visualizations in a discussion, whether they are social applications such as blogs or Facebook or corporate intranets. Thus we can shift the classic pipeline pattern to the web, and naturally extend it to include interaction and discussion as its final stage [Figure 1].

The true power of this shift from monolithic applications to information ecologies comes when the flow of data, images and rich media between phases of the pipeline is continuous. A change in data or visualization style at one stage of the pipeline can propagate forward to provide new views, new capabilities, and new subjects for discussion. The difficulty is that such applications are frequently challenging for users to create, since they require the learning and use of multiple tools that often present interoperability problems.

To ameliorate this, a single web visualization application can play multiple roles in the pipeline, making it easy for users to quickly assemble social data analysis applications while exposing each stage for reuse in other contexts. Google Spreadsheets, for example, enables users to create a spreadsheet application that can then be decomposed into a data feed or excerpted visualizations.

Our previous work on Many Eyes [15] led to the discovery that web visualization serves as a valuable community component [4] – a common artifact for discussion that can be appropriated and integrated into existing social applications. Having built a tool that

Figure 1: Applying the Visualization Application Reference Model to distributed web applications
provided these visualization capabilities, we wished to explore possibilities further down the web visualization pipeline (in the realm of presentation) while expanding upon Many Eyes’ participation in the information ecosystem and exploring new metaphors for collaborative visualization and presentation.

As with Many Eyes, our audience is intended to be the general web user who may be unfamiliar with sophisticated visualization tools. However, our goal is not to create a “new version” of Many Eyes – rather, it is to tackle a separate set of tasks in a new application. Instead of visualization-driven discussion, we wish to support distributed authorship of visualization-enhanced content and data, enabling users to construct an argument or build a monitoring dashboard using multiple coordinated visualizations.

Wikis, in particular, are flexible and simple tools for collaborative knowledge development [9]. Our own experiences with wikis as an informal space for collaborative editing and layout, coupled with our aforementioned interest in exploring new forms of content presentation, suggested a wiki-based approach. This led us to develop Dashiki, a service for creating wiki-based visualization dashboards that maintain live connections to data and enable content reuse at every stage. With this service, we seek to address some of the difficulties that users have in creating visualization-oriented mashups while preserving the flexibility of a component approach.

In this paper we first discuss the prior work that led to our interest in developing visualization and presentation tools that can serve within the larger context of a distributed application. We then describe the design decisions and technical architecture of Dashiki: its capabilities, features, and some details of its use. In Section 4 we detail the preliminary results from a public deployment of a system based upon Dashiki, including observations of user activity and three detailed profiles of use derived from interviews. Finally, we conclude with a discussion of the implications of our results and their impact on the future of these designs.

2 BACKGROUND

Our work draws together several threads from web-based information dashboards, visualization research, wikis, and collaborative analytics.

A web-based visualization dashboard is a web application that provides multiple views of a large dataset. As the dataset is updated, the visualizations in the dashboard change to reflect the new data. Typically, dashboards are customized by developers for specific business intelligence and analytic applications within organizations, such as SAP Xcelsius [19] or Spotfire Posters [20]. Researchers have also built dashboards for a variety of applications, including tracking persistent conversations in an online community [13] and coordinating software development between remote teams [2]. These applications seldom permit end-users to customize the dashboard or integrate additional outside data, and then only in a limited fashion.

Several researchers have explored the collaborative development of web-based information presentation within the field of visual analytics. The Scalable Reasoning System [12] defined a set of interoperable services and modules that enabled mashups of data and visualization in a web interface. SRS also provides tools for users to record questions, hypotheses, evidence, and other components of their analytical process in a structured manner. While SRS offers a wide variety of modules and has demonstrated interoperability with several third-party services, it is still primarily assembled and customized for the end-user by developers.

End-user composing of separate visualization components into a unified application was first popularized by the Snap-Together Visualization approach [11]. In this system, the user connects visualizations with a drag-and-drop gesture, and then specifies relationships between the datasets of the two visualizations in a popup dialog. This may result in a range of coordination behaviors (such as brushing, overview-and-detail, and drill-down) that are automatically constructed based on the data relationship. While this is not a collaborative system, it highlights the importance of a data-driven approach to the end-user construction of visualization applications.

VisGets [5] sought to close the loop of information selection, preparation and visualization by assembling several visualizations of time, location and tags associated with web content and coordinating them in the browser. Both selection and query parameters are shared between widgets on the page, and each visualization serves as a dynamic query interface to filter information. VisGets provide strongly linked and flexible views of dynamic data, however they limit themselves to the domains of data commonly available on the web (time, word frequency and location via RSS feeds) and do not provide great flexibility in layout or visualization options.

Marchese and Brajkovska [10] combined a scriptable molecular visualization applet with a wiki; this enabled scientists to collaboratively edit web documents with embedded scripts that drove interactive visualizations on the same page. Their prototype displayed a rich degree of coordination between document and visualization; however the datasets were highly specialized and domain-specific, and the scripting language necessary to create a visualization (JMOL) is beyond the reach of most web users.

Some wiki systems, such as TWiki [21], included limited visualization capability. Typically these include a special wiki syntax that enables the user to precisely define business graphics such as bar charts or line plots. Data is retrieved either from a table on the wiki page or from a file on the server. These systems typically require one to learn complex wiki syntax to declaratively specify every aspect of the visualization, and do not enable the user to visualize data from sources outside of the wiki.

3 DESIGN AND ARCHITECTURE

Based on this research, our designs evolved over the course of several months of prototypes and investigation into collaborative dashboard systems. The final pre-release design, which we dubbed Dashiki, combines several features from visualization dashboards and wiki systems into a single web application. Our decisions as to this application’s design and architecture can be parcelled into four major categories: document and visualization editing, visualization relationships, content import and export, and collaborative features.

3.1 Document and Visualization editing

Dashiki is a wiki that is structured into pages and namespaces (called dashboards). Pages may contain a combination of formatted text, links to other pages or websites, data, and embedded interactive visualizations of various sizes.

Every page in Dashiki is constructed from user-editable wiki markup based upon the WikiCreole 1.0 standard [22]. This standard combines elements of the most common wiki markup languages for the most common HTML formatting tasks, such as links, lists, and bold text; it was constructed specifically to be easy to learn, especially for users who are already familiar with another wiki system. This fit with our goal of making Dashiki as simple as possible for the general web user.
Any user may edit a Dashiki page directly in their browser [Figure 2]. The Dashiki page editor is a simple text box containing the page markup, but also includes a toolbar that automatically inserts markup and formats selected text for a variety of styles and HTML entities. Like many wikis, a complete revision history is stored for every page in Dashiki; users may easily compare revisions or revert a page to a previous revision.

### 3.1.1 Data pages

In addition to basic HTML markup, Dashiki introduces the notion of embedding data in a page. By wrapping a block of text in special markup, Dashiki will treat that block as a dataset. An example of this data embedding markup can be seen in Figure 3.

Our previous experience with Many Eyes has shown that general web users benefit when data formats are kept simple and human-readable; thus a dataset in Dashiki may consist of a single table of data (formatted as comma-delimited or tab-delimited text for easy cut-and-paste from spreadsheet applications) or unstructured text.

A dataset in Dashiki has a schema that describes its general format (tabular or free text) and, if tabular, its column types. The schema of a given revision of a page is stored along with its markup source. This schema is automatically detected when a page is first created; it may be manually corrected when the dataset is visualized.

When a page with a dataset is rendered, a preview of the data is constructed and shown to the user [Figure 3]. In addition, the dataset of any page in Dashiki is made available in excerpted, raw text form at a special URL. Thus the dataset is made available both in Dashiki and to external applications. If a page does not define a specific dataset in its markup, it is implied that the “data” provided by a page is the page’s raw text.

### 3.1.2 Visualizations

Dashiki provides special editors for wiki pages depending on the formatting of the page name. Any page with a colon (“;”) in its name is treated as a visualization.

Dashiki interprets a visualization name as follows:

\[
\text{[data page name]}:\text{[visualization title]}
\]

**example:**

**Stock Data:Change in S&P 500 Over Time**

Everything before the colon is the name of the data page for the visualization; this page contains the dataset that is used by the visualization. The visualization title is a unique name that distinguishes this visualization from others of the same dataset. In the above example, the data for this visualization comes from a page named “Stock Data”; the fully qualified name of the visualization is “Stock Data:Change in S&P 500 Over Time.”

Thus pages and visualizations are each managed as unique wiki entities with full revision histories and consistent URLs. Creating consistent and human-readable identifiers for visualizations, and making the origins of their data clear and easy to find, is an important part of our integration with a wiki model.

When a user edits a visualization, they are taken to a special WYSIWYG editor [Figure 4] where they may adjust the visualization style and appearance. Users familiar with Many Eyes
will note that Dashiki offers seventeen of the visualization types provided by that application, from simple bar charts and line plots to treemaps, bubble charts, network diagrams and tag clouds. As in Many Eyes, users may interactively navigate through these visualizations, click on symbols such as bars and bubbles to highlight them, and change other view parameters. In addition, Dashiki’s visualizations have been enhanced with features that enable selection coordination between visualizations on the same page, as well as Dashiki’s iterative editing process.

Note that it is only in the visualization editor that additional schema information (such as column types and data mapping) is imposed on the dataset. This decision was made to simplify the data editing process, offloading schema editing until the point at which the user can see the effects of this mapping on the visual presentation of their data. When the visualization is saved, any changes to the data page’s schema are also stored as a new revision of the data page, appearing in the data page’s revision history. This schema is then propagated to other visualizations of the same data.

Upon saving, the current visual configuration (such as visualization type, axis mappings, scale, and highlighted symbols) is saved as a new revision; like a normal Dashiki page, a complete revision history of the visualization is stored and can be explored and reverted via a visual thumbnail history [Figure 5]. When the visualization is displayed, the configuration of the desired revision is loaded and the interactive visualization restored to this state.

3.1.3 Embedding and laying out visualizations

A visualization is embedded in a page by wrapping its name in curly braces (e.g. {{SampleData:MyHistogram}}). This page always shows the most recent revision of the visualization. If the visualization is updated then its appearance is changed wherever it is embedded.

A user may increase the display size of an embedded visualization by wrapping its markup in additional curly braces. The previous example will display in a minimal square on the page, while entering {{{SampleData:MyHistogram}}}} will insert a visualization that will stretch to fill 100% of the page width. Visualizations in Dashiki dynamically expand or contract their UI depending on their display size. Users may also lay out visualizations and text side-by-side using a simple table markup, e.g.

| {{Data:Viz1}} | {{Data:Viz2}} |

Users can thereby create complex matrices of visualizations or flow visualizations and text together seamlessly using these simple layout components.

3.2 Visualization relationships

Visualizations that are placed on the same page in Dashiki are automatically associated in certain ways based on their datasets. Additionally, users may specify focus and context relationships between visualizations using a special syntax.

3.2.1 Coordinated Selection

Coordinated selection or brushing is a commonly desirable feature in multi-visualization systems, and has been implemented in a variety of ways in the past [1][7][11]. When views display data from disparate data tables, a common approach (popularized by North & Shneiderman [11]) uses explicitly defined relations between data tables to drive highlighting semantics across visualizations.

In Dashiki, selection is automatically coordinated between visualizations on the same page that display data from the same dataset; in this respect we take an approach that is essentially identical to Becker & Cleveland’s brushing model [1]. When a user highlights a symbol in one visualization, any symbols corresponding to the same rows or text elements in the dataset are highlighted in other visualizations on the same page [Figure 6]. Javascript in the page detects the selection event and propagates it to each visualization element as a string of JSON text. For example, the
The JSON below represents a bar selected in a bar chart for the “weight” series, representing 4 rows in the table identified by their indices:

```
{ "weight": [5, 8, 10, 13] }
```

In order to coordinate selection between visualizations that do not show the same data, Dashiki takes a simpler and less accurate approach than North & Shneiderman’s snap-together visualization [11]. Rather than requiring the user to define explicit relationships between datasets, Dashiki interprets a selection gesture as the creation of a user-defined subspace in the data that varies based on the nature and configuration of the visualization. This subspace is reduced to a list of series and attribute-value pairs, which is propagated to other visualizations on the page as formatted JSON.

For example, selecting the symbol representing Canada’s GDP on a map results in the following:

```
{ "*": [ { "country": "Canada" } ] }
```

Instead of the row indices from the previous example, this representation provides an array of attribute/value tuples corresponding to the user-defined space of values. In addition, because this map visualization lacks the notion of series in its configuration, the series identifier from the previous example is replaced with a wildcard (“*”) that matches all series in a visualization.

Other visualizations on the page then attempt to select symbols based on these attribute-value pairs and data series. For example, when receiving the above JSON text, a scatterplot showing per-capita health spending vs. infant mortality for various countries will highlight all dots corresponding to rows with the value “Canada” for the column “country.”

This is an imprecise technique for coordinating selection between visualizations with different datasets – it maximizes flexibility and ease of use without impacting accuracy for most visualizations that are displayed on the same page. In cases where false implicature does arise, users can easily revise the datasets for these visualizations, changing label values or attribute names to prevent selection from being coordinated between conceptually unrelated entities.

3.2.2 Overview and details

Users may also specify a simple content relationship between visualizations on the same page.

```
{{SampleData:MyHistogram}}
{{SampleData.selected:MyScatterplot}}
```

By appending “.selected” to the dataset name in the visualization title, “MyScatterplot” acts as a detail visualization for “MyHistogram”. When a user selects one or more symbols in “MyHistogram,” Javascript on the Dashiki page is notified of the event, pulls the selected data from “MyHistogram,” as an array of JSON objects, and then sets it as the content of MyScatterplot. When the user edits MyScatterplot, it is populated with a sampling of data from the SampleData dataset so that it may be properly configured.

3.3 Content import and export

One of our principal goals with Dashiki is to create an application that participates fully in the web’s information ecosystem – to dynamically import and export content and support live connections between web applications. To that end we support three principal means for users to import and export content from Dashiki: live data, data export, and page embedding.

3.3.1 Live data

Datasets in Dashiki may be pasted directly into a page’s markup, or they may be dynamically pulled from a URL. To specify that a page’s dataset should come from a URL, a user wraps the URL in the same markup as pasted data. For example:
This is referred to as live data. Internally, Dashiki asynchronously fetches this data in the background, strips it of any formatting tags or entities, caches it on the server and inlines its contents into the page’s markup. The data preview shows the original link, the previewed data and the last time the data was refreshed.

Any time a page containing a live dataset is accessed, Dashiki checks to see the last time the dataset was fetched from its URL. If the interval is past the refresh time of the dataset (now fixed at 30 seconds), Dashiki asynchronously refetches and downloads the data.

### 3.3.2 Data export and page embedding

While Dashiki includes several features that support collaboration, we expect that it will see the majority of its use as a community component – a common artifact for presentation and discussion in other web applications. We support content reuse from Dashiki at multiple stages: data, visualization and presentation.

Users may access the raw text dataset from any Dashiki page by appending /data to the page URL. This consistent mapping enables anyone to easily reuse Dashiki data in other applications. A Dashiki visualization can be embedded as a live applet in any web page. Clicking on the “embed” link on any visualization provides a scrap of HTML that can be copied and pasted into any page. Users have a choice of embedding a specific revision of a visualization or the most current revision; this allows them to control whether the content they embed in another application will update when it is revised in Dashiki.

Similarly, a user may embed any Dashiki page in another application. This enables them to quickly assemble a collection of visualizations and analysis using Dashiki markup, then embed the results in any other web application. The embed code for pages uses JavaScript to dynamically insert content, enabling external applications to maintain live connections to the content they embed or to restrict the content to a specific revision.

### 3.4 Collaboration Features

Dashiki was envisioned primarily as a tool for visualization and presentation that is reused in other contexts. However, there are significant benefits to opening visualization content to collaborative revision [16]. We provide a basic discussion feature – every page in Dashiki has a corresponding discussion page where users can easily post comments and ask questions using an inline form.

In addition, to make it easier for editors to reuse each others’ work, we provide a deep copy and reuse feature – any user may easily duplicate a page within or between dashboards with a single click. This duplication includes all of the visualizations embedded in the page as well as their datasets. This enables users to easily “learn by copying” and customizing an existing solution.

### 4 Deployment and Observations of Use

The public version of Dashiki was built as a Ruby on Rails application and made available on the Internet under the name “Many Eyes Wikified” [23]. Some changes were made to the public offering from our reference design – the name was altered\(^1\) for legal reasons, while the coordinated selection features described in Section 3.2 were temporarily omitted due to bureaucratic delays regarding code approval. We invited some users of Many Eyes to a limited-access beta starting in November of 2008; in late February of 2009 we opened the site to the public.

#### 4.1 Preliminary user activity

We have had (as of mid-March 2009) a total of 28,971 page impressions since the beginning of the private beta; 19,852 of those impressions have occurred since the launch of the site. Editing-related activities (such as creating a new page or editing a visualization) make up approximately 3.5% of our traffic since the public launch, compared to approximately 6% before the launch. This fits our expectations that editing activity would decrease as a percentage of total traffic when members of the public are permitted to view the pages. 118 unique users have registered for Many Eyes Wikified; of these, 37 users have revised or created a total of 349 distinct pages and visualizations on the site.

Users created live data links to 66 different URLs; of these, 23% were blogs and other text-oriented sites as opposed to structured data URLs. Among the URLs to structured data, 64% referenced Google Spreadsheets, while the rest were a combination of mashup services such as Yahoo Pipes and DabbleDB along with a few user-hosted data files. Much as we found with Many Eyes, simple data tools such as spreadsheets formed the majority of our users’ activity.

Our participants have used Many Eyes Wikified’s live data features to connect to a wide variety of data. Continually updated sites (such as blogs, news articles, or collaboratively edited reviews) were the most popular sources of text for Wikified’s tag cloud and word tree [18] visualizations. Meanwhile, the most popular kinds of quantitative datasets were organizational statistics (such as site traffic) followed by mashups with data from organizations that provide open data APIs (such as the Guardian newspaper in the UK) [24].

#### 4.2 Styles of pages created

It is difficult, at this preliminary stage, to draw firm conclusions about users’ editing habits. However casual observation of our users’ activity allows us to tentatively group the Many Eyes Wikified pages they have created into three categories. Many users create a combination of these styles within a single dashboard as they explore the capabilities of the system.

First, a small minority of users treat the wiki as a freeform notebook. They record observations and hypotheses, mix and match visualizations that help them explore their ideas, and frequently reorganize the structure of their dashboard. These users seldom made use of live data; instead they preferred to keep their data within Dashiki, presumably so that they did not have to continually switch between applications when revising their data.

Second, a larger minority of users create well-structured dashboards, with formal titles and explanations for each visualization, along with introductions or discussions of their content. These are frequently large, multi-screen pages that set their visualizations to large display sizes.

Finally, the majority of users create simple pages with two or three visualizations and little in the way of titles or explanatory text. Interestingly, these pages tend to take fullest advantage of Wikified’s live data features. In addition, it is this style of page that users seem to most frequently embed or reference in a blog post.

The three categories of page style can be generally thought of as fitting along the spectrum of social uses of this design. The “notebook” pages are personal tools not intended for extensive sharing with others; they function primarily as a semi-private and informal space for the user to organize their thoughts and notes. The “dashboard” style of page makes for a big and impressive presentation – a useful component of a demo in a corporate or academic context that illustrates a particular point. Meanwhile the “compact” page style is most in keeping with Dashiki’s mission as a community component – these pages are easily linked from or embedded in other communities, serving as a common artifact for discussion and illustration.

\(^1\) In this paper, we use “Dashiki” to refer to our reference design, and “Wikified” to refer to the public web application that realizes this design.
4.3 User interviews

We conducted three semi-structured voice interviews with three Many Eyes Wikified users, identified through observation of their activity and selected because they were particularly prolific in their use of the system. In conducting these interviews, we sought to identify what it was that drew them to the system, develop a better understanding of the value that it had for them, and provide a context in which they could discuss problems and opportunities for improvement.

By choosing to interview only prolific users, there is of course an inherent selection bias in these narratives that makes it difficult to generalize any conclusions made across a broad population of users. However, we believe that the resulting discussions are still valuable both to inform development of similar systems and to suggest future directions for this research.

4.3.1 First Interview

Our first interview was with a learning technology advisor for a major university. An experienced wiki editor, her initial attraction to Wikified was its combination of a collaborative platform with which she was familiar and comfortable together with its visualization capabilities. This user did not make extensive use of Wikified’s live data; instead, the pages she created were a free-form combination of notes, data, and experiments on a variety of topics. Her explorations of the 2008-2009 world financial crisis included large multi-visualization pages and a complex tree of datasets on equities, commodities, and currencies. This content was then used as part of a lecture on Real Estate Investment trusts and was a hit with students; according to her:

It was really interesting to observe the students’ reactions because on things like a bar chart where we could demonstrate multiple rate changes taking place on one day and the related news also being published that day - they asked really interesting questions that they might not have asked if we had been using something like a typical chart tool. We also could not have compared the news and market rates in this way either.

This user, like the others we interviewed, had some initial difficulties grasping how we modelled pages and visualizations – initially she believed that visualizations and data must be co-present on the same page, and that one could create multiple datasets in a single page. However, she claimed that once she “got it,” it was relatively simple to employ.

She was excited by the coordinated selection design we proposed, indicating that (in the case of her financial dataset) it would be most useful to discover correlations in phenomena that occurred at the same time across several different datasets and visualizations.

4.3.2 Second Interview

Our second interview was conducted with an engineer for a major open source project. He is responsible for tracking, measuring, and optimizing the use of IT facilities related to this project.

Like our first user, he has extensive experience with wikis – they are used intensively to coordinate work within his project. His initial interest in Wikified was due to both its wiki characteristics and its live data facilities. His project is working to provide more open access to statistical information about their primary product -- including the download statistics for builds of that product that correspond to various versions and platforms.

This user was frustrated by the limitations of live data within Wikified. The dataset he wished to visualize consists of over 4 billion records stored in a high-dimensional, column-oriented data warehouse. While he did not expect that we would be able to visualize this entire dataset at once, he had hoped that Wikified would include facilities that would enable users to dynamically query this datastore and extract certain subsets of data. This user worked with some third-party mashup services to try to put this data into an intermediate form that could be exposed in Wikified. However, none of the services that he investigated could handle data on this scale.

He stated that he was “very comfortable” with the wiki paradigm presented in Wikified. He liked the simple and minimal layout features it presented – he could quickly create a set of views that he liked in Wikified, and if he wanted something “pretty” he would embed a Wikified page in a blog post or on his own site. However, he suggested that the means by which Wikified modelled datasets and visualizations was less intuitive than it might be. He would prefer to have a specialized editor for datasets, as opposed to the current syntax that simply wraps data in special wiki markup.

This user also stated that the ability to lay out multiple visualizations side-by-side was extremely important. Particularly in his high-dimensional dataset, being able to present small multiples [14] that vary along a particular dimension was very compelling to both him and his colleagues. Although he also found coordinated selection useful, he suggested that a more valuable way to link visualizations would be a drill-down relationship – selecting an item in one presents a detail view in another.

4.3.3 Third Interview

Our third interview subject was a university lecturer and prominent blogger. He frequently blogs about mashups and open data initiatives, and saw Wikified as a potentially useful environment to create mashups with a variety of external data sources. His blog posts about Wikified included extensive tutorials with screenshots, explaining to his readers how to use Wikified to visualize data from both Yahoo Pipes and the Guardian newspaper’s Open Platform.

Unlike our other interview participants, this user had no particular attraction to wikis. Indeed he hardly used the layout features of Wikified, although he “intends to explore them.” Most of his interest in the system was oriented towards its potential as a mashup platform for open data. As such, he found our choice of CSV format and means of integration to be “nice and simple – a good lowest common denominator.”

He did however see a potential stumbling block, in that oftentimes he wanted to visualize only a portion of data within a dataset. As it is, he had to manually fiddle with the data URL in order to extract the subset of data that he wanted from the Guardian platform. He also believed that we needed better mechanisms for filtering individual columns and rows from datasets; he desired that each visualization of a dataset should apply its own filter schema, to permit maximum flexibility in a context where a user has little control over the overall contents of the dataset. Furthermore, he thought that it would be important for future use to include some means of access control for live data – such as the ability to securely specify an API key or username/password combination for HTTP basic authentication at a given URL.

This user was most critical of the wiki features of our system – he did not believe that the way in which we modelled visualizations and data was particularly intuitive, as it did not present a clear picture of where a visualization or dataset existed. He did appreciate the potential for layout and coordination afforded by our scheme, but suggested that we might choose a different model that presented visualizations, datasets and wiki pages as wholly separate entities.

5 Conclusion and Future Work

In this paper we have described the rationale, design, and deployment of Dashiki, a web application that enables users to collaboratively construct visualization dashboards. Our goal was to assemble visualization construction, knowledge organization, and multi-visualization layout into a single application that is both well-integrated with general web applications and open to use in a variety of contexts. Our users have successfully built several multi-visualization tools that integrate with both static and live data, in contexts ranging from literary analysis to website traffic statistics to personal explorations of the world financial crisis.
Based on our observations and user feedback, there are four principal areas that we wish to emphasize for further enquiry.

First, while a plethora of standards exist for publishing structured data on the web, there are few available that enable web-based visualization and analysis applications to “close the loop”: to query and transform results from a large repository of structured data without bulk importing the entire set. As the size of available datasets grows, and as more and more organizations and governmental bodies embrace the concept of open data, such standards are critical if we are to realize the full potential of distributed analysis on the web. The Google Visualization Data API [25] is a good starting point for relational data; however the need still exists for standards that can address high-dimensional data (e.g., cubes) and provide statistics from large corpuses of text.

To encourage the propagation of such standards, we believe it is a natural progression to build support into Dashiki for “filtered” datasets: live datasets that treat their URL as an endpoint for a standard query API, along with controls that enable users to navigate these datasets within their dashboard. Exploring these possibilities is an important next step in developing web visualization tools that enable users to create sophisticated analysis applications.

Second, the integration of a versioned wiki system with live data suggests useful techniques for capturing and presenting history. One might imagine a “time machine” – periodic snapshots of a dashboard page that preserve content, layout, and the state of live data at specific intervals. This would enable users to review the historical progress of an event and its analysis across several views.

Third, while we believe that our preliminary results might indicate that a wiki model is a good choice for collaborative content development and dashboard layout, we may need to revisit our choices of how we integrate this model with dataset and visualization editing. A system in which datasets are separate entities from pages, with a specialized editor that enables tasks such as filtering and secure access, would be easier for our users to understand and enable us to provide a wider range of capabilities.

Finally, while we have made an effort to expose as much Dashiki content as possible for reuse in other applications, there is still valuable work to be done. We see a clear need to explore how users are reusing Dashiki content, and what additional steps might be taken to improve its utility, compatibility, and communicative power in these contexts. Some possibilities include open Javascript APIs for manipulating visualizations, graceful scaling of content to smaller display sizes within a larger page, and public APIs for the creation of Dashiki content from other applications. We believe that such research will further illustrate the benefits of the distributed visualization application model represented by web applications like Dashiki, and the tremendous variety and scope of analytical tools this model promises for the future.

ACKNOWLEDGMENTS
The author would like to thank Martin Wattenberg, Fernanda B. Viégas, Jesse Kriss, Frank van Ham, Irene Ros, and Stephen Levy for their support and contributions to this project.

REFERENCES