Aperture: An Open Web 2.0 Visualization Framework

David Jonker
Oculus Info Inc.
djonker@oculusinfo.com

Scott Langevin
Oculus Info Inc.
slangevin@oculusinfo.com

Neil Bozowsky
Oculus Info Inc.
nbozowsky@oculusinfo.com

William Wright
Oculus Info Inc.
bwright@oculusinfo.com

Abstract

Aperture is an open, adaptable and extensible Web 2.0 visualization framework, designed to produce visualizations for analysts and decision makers in any common web browser. Aperture utilizes a novel layer based approach to visualization assembly, and a data mapping API that simplifies the process of adaptable transformation of data and analytic results into visual forms and properties. This common visual layer and data mapping API, combined with core elements such as contextually derivable color palettes, layout and symbol ontology services is designed to enable highly creative and expressive visual analytics, rapidly and with less effort. This paper introduces the Aperture framework, describing key features of the programming API and reference implementation, presents example use cases, and proposes an approach for measuring technical performance metrics for software development, and operational performance metrics for visualization support of analysis and decision making.

1. Introduction

To achieve optimal information visualization and visual analytic solutions [1], [2], rapidly and with less effort, we have designed and developed an open, adaptable and extensible software development framework, to produce visualization applications for analysts and decision makers in any common web browser. This framework—its design, API and implementations—is called Aperture.

Key to Aperture is a new layer based approach to visualization assembly, and a data mapping API that simplifies the process of adaptable transformation of data and analytic results into visual forms and properties. The Aperture framework and API are designed for ease of extension, allowing a broad community to leverage and extend capabilities and even to invent new paradigms, and interoperability with Web 2.0, Service Oriented Architecture (SOA) and geographic information system (GIS) standards.

Our primary goal in designing Aperture was to support "limitless extensibility" (layering visualizations in unpredictable ways) for complex data problems, using a simple and intuitive programming grammar. Key constraints were that it be optimized for efficient dynamic updates, and runtime adaptive to varying levels of browser support for graphics APIs such as svg, vml and canvas.

In the next section we provide brief background on the challenges in creating effective information visualizations and visual analytics. In Section 3 we present our objectives in designing and implementing the Aperture visualization framework. Section 4 discusses some examples of related work. Section 5 describes Aperture as both an architectural framework and reference implementation, with use case examples to illustrate the layer based visualization and data mapping APIs. An example application where Aperture was used to rapidly develop a tailored cyber situation awareness and analysis "big data" application (8 GB, 158M rows) is discussed in Section 6, and in Section 7 we conclude and suggest future work.

2. Background

Visualization is an external mental aid that enhances cognitive abilities. When information is presented visually, efficient innate human capabilities can be used to perceive and process data. Information visualization techniques amplify cognition by increasing human mental resources, reducing search times, improving recognition of patterns, increasing inference making, and increasing monitoring scope [1], [3]. These benefits can translate into significant system and task related performance gains.

Recently a new field has emerged called visual analytics which builds on information visualization and computational analytics. Visual analytics support analytical reasoning facilitated by interactive visual interfaces and integration with computational analytics. People, data and analytics work together in a visual system of systems to harness the respective strengths of each component. People use visual analytics tools and techniques to synthesize information and derive

Accepted HICSS, 2013
© 2012 Oculus Info Inc.

insight from massive, dynamic, ambiguous, and often conflicting data; detect the expected and discover the unexpected; provide timely, defensible, and understandable assessments; and communicate assessments effectively for action [2]. Visualization and visual analytics produce superior information and knowledge products to support human situation awareness and decision-making.

However, visual analytics systems can be difficult to design and build effectively, often requiring very experienced researchers and designers for success. A basic vocabulary of standard chart widgets often falls short of the mark. Mapping complex data to appropriately informative visual forms and devising a means and structure for navigating it requires knowledge and application of graphic design principles, user interface design, the task domain, human factors, visualization techniques and creativity. A misstep in any of these areas will result in poor performance, obscuring the data or confusing its interpretation. To make the art of crafting visual analytics more accessible and robust for the average developer, new framework models are required.

3. Objectives

Application developers often fail to consider visualization as a holistic system, inserting any number of independent, generic charts into their user interface (UI). But cognitive interpretation of basic graphical elements such as color and scale are highly sensitive to contextual factors and the nature of the data [3]. Systemic approaches to visualization have often been either drag and drop dashboard assembly frameworks or collections of "slice and dice" widgets for laboring through generic bar chart data exploration. By using multiple, separate views in unrelated layouts, over multiple screen pages, people must rely on visual memory to retain and compare information, and human visual memory is weak [3]. Generic slice and dice approaches also do not embody task centric information visual structures. These slice and dice approaches fail to provide the necessary depth or breadth of insight required for effective situation awareness and decision making from complex and massive data. They also fail to provide a sufficiently rapid, intuitive, and repeatable means of fulfilling the challenges of representing information facets for routine or ad-hoc tasks, and communicating those information facets to the task performer and to others. As shown in Figure 1, this results in a gap for rapid insight in complex decision-making.

There is an opportunity to provide a new, agile (i.e. flexible and rapid) approach to more often achieve optimal visualization solutions, rapidly and with less effort. In designing the Aperture framework, we set out to create an intuitive, easy to understand programming vocabulary implementing a high level grammar of visualization [4], allowing the user to focus more on domain specifics.

Key design objectives for the Aperture framework include: 1) minimize the time and effort required to design and test best practices as well as innovative visualizations, 2) provide visualization and interaction designs appropriate for complex and large data, 3) encode principles that respect and exploit human perception and cognition, 4) use a flexible and agile development process, and 5) produce an open framework with open standards and source code.

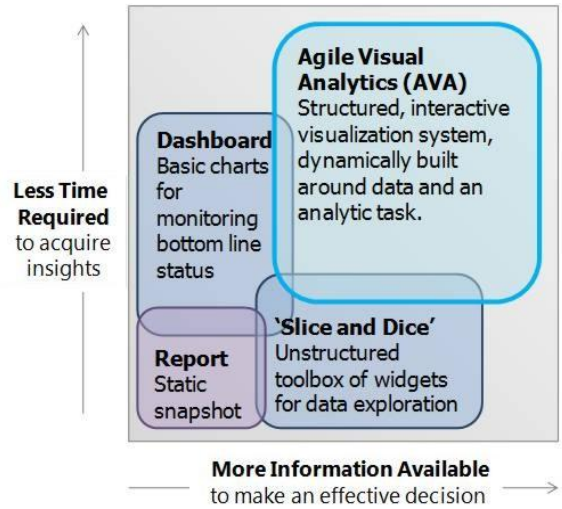

Figure 1: The gap between simplicity for rapid insight and complexity for decision making. Agile visual analytics bridges the gap for rapid insight in complex decision-making.

4. Related work

A wide selection of off-the-shelf solutions for rapid data visualization is readily available. Dashboards or business intelligence (BI) tools provide basic charting capabilities, restricted to a limited number of data dimensions per chart.

Tools such as Spotfire [5] or Tableau [6] allow for "slicing and dicing" the data into linked views of different dimensions. This enables a more detailed look into the data, but requires the user to deal with the complexity of mapping the data to appropriate views, and then exploring for insight among this collection of views.

Tailored visualization solutions (e.g. GeoTime [27]) provide more optimal fit with specific data and analytic tasks, but require experienced, specialized practitioners to design and implement.

© 2012 Oculus Info Inc.

Specialized development tools such as R [7], or Python with Numpy [8]/ Scipy [9]/ Matplotlib [10] functional libraries, provide analytic capabilities with some plotting extensions to display results.

Visually oriented tools such as Adobe Flash [11] and processing [12], or graphics libraries such as Raphaël [13] can aid UI, graphics, interaction and visualization development, but analytic capabilities must be custom coded.

Visualization engineering toolkits such as Protovis [14], its predecessor Prefuse [15], and its heir, D3 [16], as well as Improvise [17], attempt to bridge these two differing approaches by providing libraries specifically for making custom interactive visualizations.

These toolkits provide abstractions of graphical elements and their associated behavior, as building blocks to be combined together by the developer. The open, declarative approach taken by Protovis has proved to be a succinct grammar for such constructions [18]. However, because of the way these grammar elements are combined, they must be lower level in nature to preserve flexibility in visualization design. In practice, we find these lower level abstractions are not always intuitive, even to a visualization expert, and can obscure the higher level analytic task at hand.

5. The Aperture framework

Our goal was to combine visual and analytic approaches, and create an expressive medium for programming graphics and visualizations in a more efficient manner through building data structures and semantics. In addition, we wanted to enable ease of usage for developers without specialized expertise in graphics programming or visualization.

The Aperture API provides a high level vocabulary of visualization constructs, such as plots, bar and line series, indicators, nodes and links, rather than low level graphic primitives that must be chained together. This is a vocabulary more familiar to analysts, intended to help preserve context when assembling visualizations.

While combining primitive visual elements in novel ways provides one means of expressiveness, the approach in Aperture is to layer more intuitive and highly functional constructs in novel combinations. An extensible set of layer forms are provided, including extension of existing broad capabilities of the OpenLayers [19] open source toolkit to overlay dynamically interactive, geo-located visual entity representations. Chart, timeline and network forms can also be constructed and combined, with reference implementations as described in section 5.7.

Aperture is designed for fully interactive visual analytic applications, including multi-touch support. Filters can be used to highlight information of interest for interactive exploration. Additionally, layers monitor the state of what is added, changed or removed. State changes can be animated (fade in, transition, fade out) to provide visual affordances for continuity in transitions.

5.1. Layer based visualization assembly

Aperture was developed to enable richly layered, coordinated expressive combinations of map, chart, network and timeline "vizlets" or small visualization mini-applications in a service-oriented Web 2.0 environment. Examples are shown in Figures 2 and 3. Using layers allows a merging or fusing of multiple information facets into a more unified, task centric display of information.

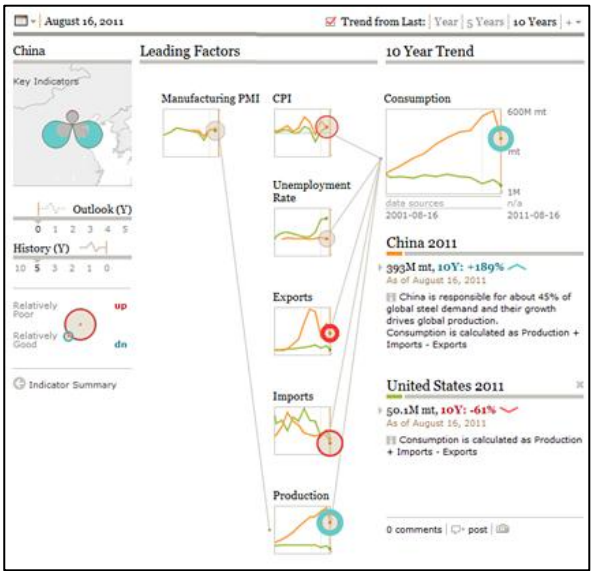

Figure 2: Layered visualization approach for creative, interactive, expressive visualization forms in a browser.

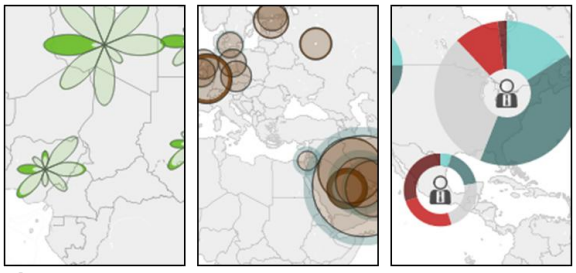

Figure 3: Any number of representational layers can be added to parent layers, including maps and charts.

This unified layer based approach to visualization assembly is intended to enable more powerful combinations of visual forms [20], with greater ease,

© 2012 Oculus Info Inc.

than a standard widget based approach with limited configuration options.

In Aperture, a layer represents a set of like graphical elements, which are mapped in a spatial context. Types of layers currently implemented include, but are not limited to, map, chart, network and timeline.

Unlike conventional geospatial layers that remain independent from one another, Aperture provides hierarchical nesting through layer composability. Children inherit mappings and data of their parent. These can also optionally be overridden in the child layer. This provides the organizational benefits of recursive groups (e.g. charts on a map) with the shared property management benefits of flat sets of visuals.

Layers are designed to be easily combined in many ways—e.g. pie charts on maps, line charts as nodes of a node-link graph—including use as dominant form or small multiple.

5.2. Data mapping API

Aperture's visual mapping and filter API provides adaptability to data schemas and transformation of data into succinct interactive visual forms. An easily understood sentence-style chain of function calls is used to define the mapping of each visual property from a domain-specific data object. Keys that define these are shareable across vizlets for consistency of scales.

Data values are mapped to visual properties as constants or from a data source. Data variables can optionally be represented as a scalar range, using interpolation when mapping values to visual properties, or ordinal range, for properties such as instance series colors or up/down indicators.

Aperture filters are functions that are applied to visual properties after a visual value is calculated using the associated data mapping. Filters can be used to alter the visual value, for example making a color brighter or increasing size of a visual element. Used in conjunction with filters, sets can provide functionality such as linked selection. Filters can be applied to elements of sets, which can be added, removed, toggled, and checked for containment.

5.3. Example use cases

To illustrate use of the Aperture visualization assembly and data mapping API, we present a use case in which we wish to visualize opinion poll data for various cities. Figure 4 shows the desired visualization, with a radial indicator layer overlaid on a map displaying notional opinion poll data for four cities.

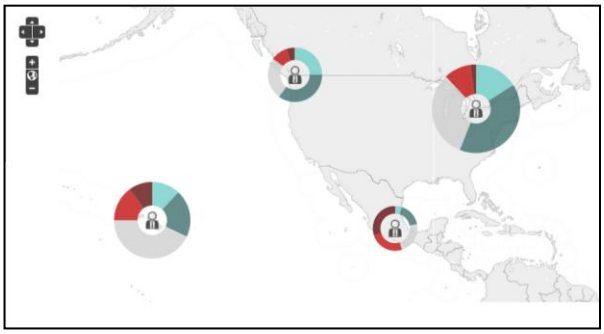

Figure 4: Radial pie charts visualization of opinion poll data.

When the mouse is over a segment of one of the indicators, the corresponding response is displayed in text form and all wedges of that type highlight (Figure 5). This example makes use of mouse hover event listeners and visual mapping filters to apply alternate styles to hovered data.

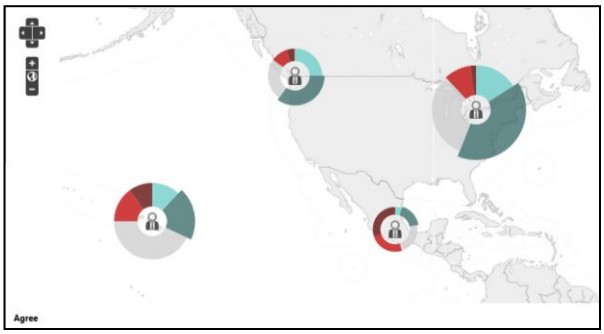

Figure 5: Effect of mouseover on a chart segment, causing wedges for the same type of poll response in all charts to grow in radius.

In Figure 6 we see a sample of the data to be visualized. Figures 7 to 11 show the steps required to create a map, add a chart layer, map data variables to visual properties, and set interactive behavior.

© 2012 Oculus Info Inc.

```
var data = [
    {name:'Honolulu', lon:-157.1, lat:21.3, respondents: 500000, responses: [
        {response:'Strongly Agree', percent:12},
        {response:'Agree', percent:20},
        {response:'Undecided', percent:43},
        {response:'Disagree', percent:15},
        {response:'Strongly Disagree', percent:10}
        ]},
];
```

Figure 6: Sample data to be visualized. In addition to Honolulu data, example visualization dataset also includes geographic location, number of poll respondents, and response percentages for Toronto, Vancouver and Mexico City.

```
// Create the map in the DOM                                    (a)
var map = new aperture.geo.Map('#map');

// Create a content position layer                              (b)
var locations = map.addLayer( aperture.geo.MapNodeLayer );
locations.map('latitude').from('lat');
locations.map('longitude').from('lon');
locations.data(data);
```

Figure 7: Create a map (a) and add a content position layer to situate visual entities in child layers (b). Aperture integrates the open source OpenLayers JavaScript library to provide full-featured client side map rendering capabilities.

```
// value ranges                                                 (a)
var percentRange = new aperture.Scalar('Percent', [0, 100]);
var totalRange = new aperture.Scalar('Number of Respondents', [0, 650000]);

// simple color definitions.                                    (b)
var agree = new aperture.Color( 'good' ),
    unsure = new aperture.Color( '#ccc' ),
    disagree = new aperture.Color( 'red' );

// create the ordinal mapping between responses and colors      (c)
var responseMapping = new aperture.Ordinal('Response', [
    'Strongly Agree',
    'Agree',
    'Undecided',
    'Disagree',
    'Strongly Disagree'
]).mapping([
    agree,
    agree.blend('black', 0.5),
    unsure,
    disagree.blend('black', 0.25),
    disagree.blend('black', 0.65)
]);
```

Figure 8: Set value ranges and map them to response percentages and number of respondents (a). Specify color definitions (b) and map to poll responses, applying blending for additional color variations (c).

© 2012 Oculus Info Inc.

```
//Create a location layer
var pies = locations.addLayer( aperture.RadialLayer );                                    a
pies.map('opacity').asValue('0.75');
pies.map('radius').from('respondents').using(totalRange.mapping([5, 40]));                b
pies.map('base-radius').asValue(20);
pies.map('sector-count').from('responses.length');
pies.map('sector-angle').from('responses[].percent').using(percentRange.mapping([0, 360]));
pies.map('fill').from('responses[].response').using(responseMapping);

var iconLayer = locations.addLayer( aperture.IconLayer );                                 c
iconLayer.map('type').asValue('person');                                                  d
iconLayer.map('attributes').asValue({role: 'business'});
```

Figure 9: Add a radial chart layer to the content position layer, inheriting geographic location properties (a). Configure chart properties, including mapping chart radius to number of respondents and sectors to poll response percentages (b). Add an icon layer to the content position layer, also inheriting geographic location properties (c), and specify ontological icon type and attributes (d).

```
/*
 * Create a "Set" containing the hovered response                                         a
 * Cause all wedges of this type to grow on hover
 */
var hover = new aperture.Set( 'responses[].response' );
pies.map('radius').filter( hover.scale(1.2) );                                            b

pies.on('mouseover', function(event){                                                     c
    // Given the data object and the index into the data, get the response label
    var response = event.data.responses[event.index[0]].response;
    $('#hover').html( response );
    // Add to highlight group
    hover.add(response);
    map.update(new aperture.Transition(150));                                             d
});
pies.on('mouseout', function(event){
    $('#hover').html('');
    // Clear highlight group
    hover.clear();
    map.update(new aperture.Transition(150));                                             d
});
```

Figure 10: Create a set containing the hovered poll response type (a). On hover, using a filter on the mapped visual property, cause it to grow (b), and add the poll response label to the data set to also appear (c). Apply animation both on mouseover and when clearing the highlighted group (d).

```
//Zoom to the area of the world with the data
map.zoomTo( 35, -120, 3 );                                                                a

map.update();                                                                             b
```

Figure 11: Zoom the map to the desired geographic location (a) and draw the visualization (b).

In the above use case we see how the process of adding and configuring a chart and icon layer is simplified by inheritance of properties from the parent content position layer. In the same fashion, any number and ordering of Aperture layers can be composed together.

Figure 12 shows a more complex visualization. In this example, we see maritime vessel tracks and locations, which can be animated over time through use of the timeline control. Popup charts indicate close encounters with other vessels, displaying tracks for those vessels. This visualization combines multiple

© 2012 Oculus Info Inc.

layers of map, icon, line series, chart and timeline types. The process of composition, however, remains simple and straightforward.

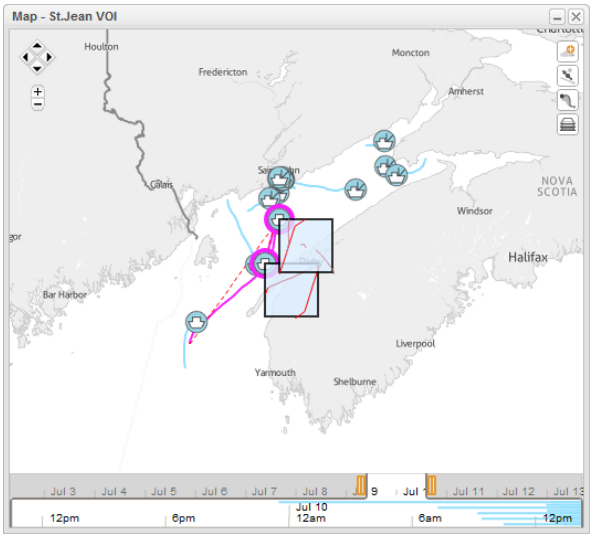

Figure 12: Maritime vessel tracks and locations, with popup charts indicating close encounters with other vessels. Courtesy DRDC Valcartier [28].

The line series layer, for example, that is added to the close encounter popup charts to show tracks for those vessels, inherits properties from the chart layer. This automatically enables the lines to be properly situated and drawn within their corresponding chart.

5.4. Browser based client implementation

Aperture is implemented as a standards based JavaScript visualization library and associated supporting services for creating mashups of maps, charts, networks and timelines and richly layered combinations thereof using one consistent and efficient API.

AJAX (asynchronous JavaScript and XML) offers the highest degree of deployability and cross-device support for client-side visual analytic capabilities. However, without a coordinated framework, such as Aperture, it can be a technically challenging environment for visualization. A legacy of inconsistent support for graphics technologies across browsers is the source of many low level issues, which must be addressed for even the most basic elements of visual expression, such as diagonal lines and curves. Moreover the open and exceptionally lightweight nature of AJAX technology can be both a blessing and a headache.

Often a complex mashup of numerous independent JavaScript libraries and server side libraries is required to build up the technology stack for an application, with dependencies and overlaps that can be difficult to resolve and maintain. Without Aperture, an AJAX chart widget library and a map library addresses many of the same challenges but each provides its own end to end solution with its own proprietary interfaces, nuances, requirements and limitations. Aperture addresses these issues by providing a unified API.

5.5. Interoperability using Web 2.0, SOA and GIS standards

Aperture uses a non-proprietary Web 2.0, Service Oriented Architecture (SOA) approach for greatest interoperability with standards such as OpenSearch, OpenSocial, RSS/Atom/GeoRSS, OAuth, RDF, WS-I Basic Profile, REST, SOAP, XML Schema, WSDL, UDDI, WMS, WFS/ WFS-T, WCS, KML, GML, WPS, CSW, CTS, WMC, and SOS.

Aperture does not need to "own" the visualization container portal, enabling constructed Aperture vizlets to be easily embedded with full interoperability in frameworks that do, such as the Ozone Widget Framework (OWF) [21], or in any simple web page, where open JavaScript APIs provide full interoperability with other DOM elements (Figure 13).

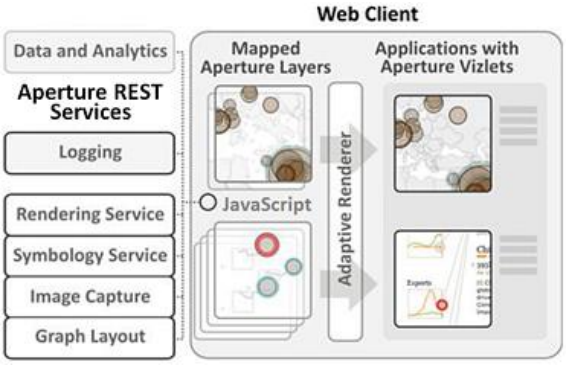

Figure 13: Modular and open architecture enables flexible options for rapid integration and cross-leveraging of capabilities.

On the client side, Aperture makes use of the OpenLayers [19] Geographic Information System (GIS) to provide full featured client side map rendering capabilities, and abstracted renderer options that include Raphaël [13] for interactive cross-browser graphics, with support for modern browsers as well as IE7 and IE8. The libraries or selective components of the library are easily integrated into any Web 2.0 architecture.

The Aperture standards, framework and API are provided under the MIT License (MIT) open source licensing, allowing a broad community to leverage and extend capabilities and even to invent new paradigms.

© 2012 Oculus Info Inc.

5.6. Aperture services

In addition to the visualization assembly and data mapping APIs, rapid visualization development and integration is further supported by a suite of Aperture services:

- Property Palettes, and a graph Layout Service.
- Icon Service API designed to match entity requests to best available symbols, with support for dynamic scaling of vector source icons and common image types. A reference set of socio-cultural oriented icons and supporting service has been implemented.
- Image Capture Service API for preserving a snapshot of client visualization suitable for insertion into a report document. Implementations for both Windows and Linux servers have been developed.
- Application Support Service APIs and industry reference implementations for authentication, CMS storage, io, logging, and a client message bus.
- Modular, Extensible Architecture providing options to include select subsets of Aperture only, and extend and adapt capabilities as necessary.

5.7. Reference vizlet implementations

Aperture includes reference implementations in AJAX of several frequently used "vizlets" or small visualization mini-applications. These serve as useful visualization methods that work right out-of-the-box as well as informative examples of how to implement visualizations in the Aperture framework and standards. The emphasis is on providing functionality with high usability. The vizlets are extendable by third parties to enable specialization, ongoing innovation and community development. Foundation vizlets include:

- Geospatial View: A geo-spatial viewer component is a fundamental part of the Aperture framework. Maps are used to show information visually in relationship to its location as well as spatial, directional and temporal relationships. The reference implementation makes use of the OpenLayers open source toolkit.
- Network Graph: These relationship diagrams may be used to illustrate the social interaction of groups and individuals. Leaders, informal groupings, group dynamics and roles as well as relationships and their natures can be represented. Similarly, these "node and link" graphs may also be used to show relationships among concepts, events, actions, etc.
- Charts and Line Graphs: A line graph visually displays quantitative relationships between two or more groups of information. Graphs have one or two axes utilizing quantitative, ordinal or categorical scales. Reference implementations include timeseries, bar charts and pie charts, supporting their use in combination and in embedded form, with common scales and legends.
- Timeline View: Timeline charts are typically a specialization of bar charts and line graphs that relate events, activities, actions, characteristics, to time.

6. Example application case study

As an example of practicing "agile" (i.e. flexible and rapid) visual analytics using "big data" (8 GB, 158M rows), Aperture was used to rapidly develop a tailored visualization for the IEEE Visual Analytics Science and Technology (VAST) Challenge 2012 [22]. The challenge focused on visual analytics applications for large scale cyber situation awareness and analysis. The contest committee provided two mini-challenges to test analytical skills and challenge the capabilities of visual analytics applications.

The 2012 VAST Challenges were set in BankWorld, a fictional planet with its own unique geography. For "Mini-Challenge 1: Bank of Money Enterprise: Cyber Situation Awareness," supplied datasets provided metadata about the Bank of Money (BOM) network and periodic status reports from all computing equipment in the BOM enterprise. With offices of various sizes all across BankWorld and computers in use throughout the day in each office, the challenge was to achieve cyber situation awareness across an enterprise network of approximately 1,000,000 machines.

We designed a tailored situation awareness and analysis application (Figure 14) showing thumbnail time series charts of policy events, performance events, and derived questionable activity issues. These thumbnails are arranged according to the hierarchical organization of the Bank of Money. In a single view, we display the corporate headquarters (CHQ), the large data centers (DC1-5) and all 50 of the large and small regions. The legend allows event filtering by type, and also by subtype of "questionable activity." The application also contains an interactive map in the upper left to allow analysis of geo-based trends. Interactions include drilling down to view detailed event counts and machine type distributions for each region or center (Figure 15).

© 2012 Oculus Info Inc.

Figure 14: Cyber-situation awareness application for "BankWorld". Using the legend to filter on policy event type we observe an upward trend in policy violations across all regions in the enterprise. The charts are generated based on normalized counts of events as a percentage of the total number of machines in the region, with the base of each time scale proportional to the number of machines. Regions are sorted by time zone, and shaded areas in each thumbnail indicate periods outside of business hours for that area.

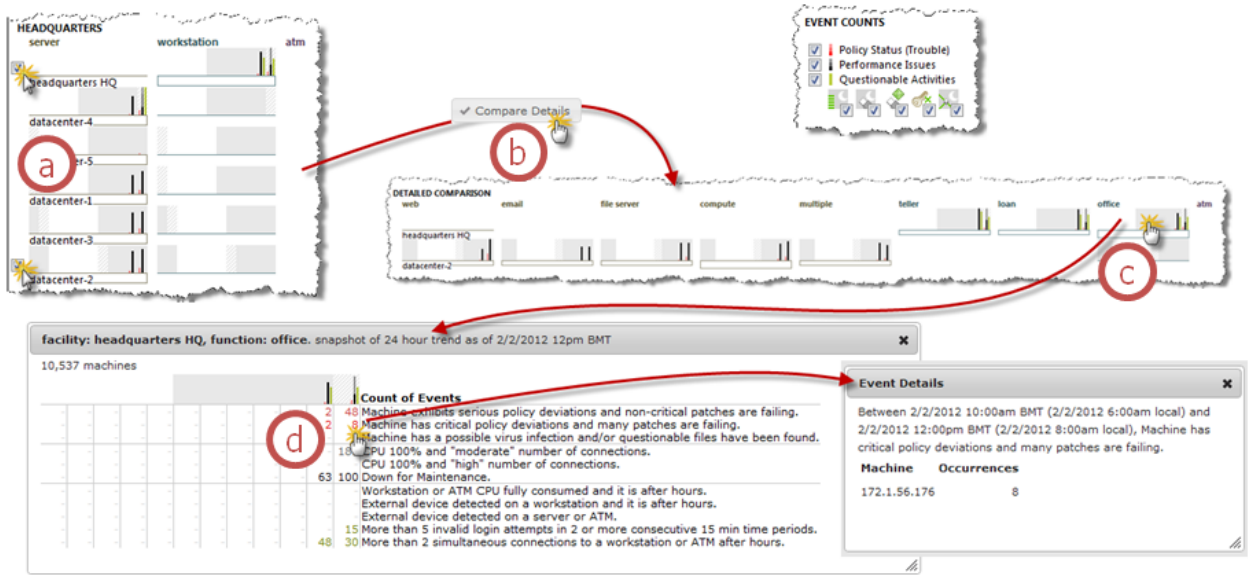

Figure 15: Examining the detailed data for a policy violation anomaly in a CHQ workstation. A "severe" policy violation happened at CHQ and at DC2 (a). We open a detailed comparison of CHQ and DC2 (b), and discover that the events at CHQ occurred on office workstations (c), and the events at DC2 happened on computational servers. From our table view (d), we drill down into the list of IP addresses affected by the notifications. We are able to observe "critical" policy deviations triggered by a workstation in CHQ (172.1.56.176).

© 2012 Oculus Info Inc.

Using the Aperture visualization framework, our time from project inception to problem solution was 20 days and included: analytic visualization design; development (data, analytics, and visualization); testing and tuning the tool; deployment; analysis using the tool; and reporting the solution. The lessons learned from this test will be used to develop a foundation framework for doing agile analysis on extremely large datasets using visual analytics.

7. Conclusion and future work

Aperture has moved beyond proof-of-concept and is now in use on two projects currently underway. One project is for maritime situation awareness and the other is for human social cultural systems analysis. In these projects, the jumpstart that Aperture provides has been critical in meeting the requirements for rich, expressive visualizations and effective visual analytics, within time and budget constraints.

In our next steps, the intent is to measure technical and operational performance. Technical performance metrics for software development frameworks or toolkits depend on design goals, and for Aperture, will include ease of use (e.g. time to develop an application), performance, extensibility, web standards compliance, functionality, interoperability and ease of deployment. Operational performance metrics for visualization support of analysis and decision making, include detailed metrics in each of situation awareness (SA), collaboration, interaction, creativity, utility [23]. Broadly speaking, information and knowledge products expressed in visual form let people see and understand complex information, more quickly and more thoroughly. Improved understanding improves decision-making. The Aperture approach to operational metrics will be to merge Endsley's perception / comprehension / projection levels of SA [24] with the visualization community's current thinking on performance metrics to create a short set of relevant metrics to describe Aperture goals and to gauge Aperture progress.

Planning is also underway to extend the Aperture framework with mixed-initiative partial automation recommendations to support novice visualization developers. To increase success in achieving optimal visualization solutions, the design must be grounded in core scientific principles of perception and cognition [3]. Important principles, guidelines and cognitive ergonomics have been identified that can support recommendations to aid in the rapid and effective composition and implementation of an easy-to-use and perceptually/cognitively correct solution. This is critical because human performance can vary significantly from plus/minus 100 times according to the representation used in problem solving [25]. Mixed initiative interaction supports an efficient interleaving of contributions by users and automated services to converge on solutions to problems [26].

8. References

[1] Card, Stuart, J. Mackinlay, B. Shneiderman, Readings in Information Visualization, Morgan Kaufman, 1999.

[2] Thomas, Jim, and K. Cook (Ed.), Illuminating the Path: The R&D Agenda for Visual Analytics, IEEE Press, 2005.

[3] Ware, Colin, Information Visualization: Perception for Design, 2nd Edition, Morgan Kaufman, 2004.[4] Wilkinson, L., The Grammar of Graphics, Springer, 2005.

[5] http://spotfire.tibco.com/

[6] http://www.tableausoftware.com/

[7] http://www.r-project.org/

[8] http://numpy.scipy.org/

[9] http://www.scipy.org/

[10] http://matplotlib.sourceforge.net/

[11] http://www.adobe.com/flashplatform/

[12] http://processing.org/

[13] http://raphaeljs.com/

[14] Bostock, M. and Heer, J., "Protovis: A graphical toolkit for visualization", IEEE Transactions on Visualization and Computer Graphics, IEEE, 2009.

[15] Heer, J. and Card, S.K. and Landay, J.A., "Prefuse: a toolkit for interactive information visualization", Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2005.

[16] Bostock, M., Ogievetsky, V. and Heer, J., "D3 Data-Driven Documents", IEEE Transactions onVisualization and Computer Graphics, IEEE, 2011.

[17] Weaver, C., "Building highly-coordinated visualizations in improvise", IEEE Symposium on InfoVis, 2004.[18] Heer, J. and Bostock, M., "Declarative language design for interactive visualization", IEEE Transactions on Visualization and Computer Graphics, 2010.

[19] http://openlayers.org/

[20] Javed, W. and Elmqvist, N., "Exploring the design space of composite visualization", IEEE Pacific Visualization Symposium, 2012.

[21] Hellar, D.B. and Vega, L.C., "The Ozone Widget Framework: towards modularity of C2 human interfaces", Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 2012.

[22] http://www.vacommunity.org/VAST+Challenge+2012

[23] Scholtz, J., Beyond Usability: Evaluation Aspects of Visual Analytic Environments, IEEE VAST, 2006.

[24] Endsley, M., Situation Awareness: A Key Cognitive Factor in Effectiveness of Battle Command, Battle for Cognition, Praeger, CT, 2008.

[25] Hanrahan, P, Systems of Thought, EuroVis Keynote, 2009.

[26] Horvitz, E., Uncertainty, Action, and Interaction: In Pursuit of Mixed-Initiative Computing, IEEE Intelligent Systems 14(5), 1999.

[27] Kapler, T. and W. Wright, GeoTime Information Visualization, IEEE InfoVis Conference, 2004.

[28] Courtesy of DRDC Valcartier Maritime Visual Analytics Prototype project, 2012.

 © 2012 Oculus Info Inc.