VIM: A Framework for Intelligence Analysis

T. Alan Keahey
Visintuit LLC
keahey@visintuit.com

Kenneth C. Cox
Visintuit LLC
kcc@visintuit.com

ABSTRACT. Intelligence analysts receive thousands of facts from a variety of sources. In addition to the bare details of the fact – a particular person, for example – each fact may have provenance, reliability, weight, and other attributes. Each fact may also be associated with other facts, e.g. that one person met another at a particular location. The analyst's task is to examine a huge collection of such loosely-structured facts, and try to "connect the dots" to perceive the underlying and unknown causes – and their possible future courses. We have designed and implemented a Java platform called VIM to support intelligence analysts in their work.

ACM Descriptors: information visualization, intelligence analysis, XML, knowledge representation

1. INTRODUCTION.

A critical need for intelligence analysts is to be able to extract meaningful and actionable intelligence information from large and loosely structured collections of heterogeneous information. That information is then shared with others either as linear reports to superiors who make decisions based upon the recommendations, or with other analysts via intelligence reports. Traditionally the analyst works with an internally constructed mental model of the facts and hypotheses that go into an analysis (supported by text and image notes), and then produces a linear or tabular description of that model for sharing with others. This introduces a significant gap between the analyst's internal thought processes and how the information is portrayed to others, which in turn presents a significant opportunity for errors in communication and understanding to occur. When provided with an intelligence report, there is usually no way for the reader to "drill down" into the analyst's original thought processes to better understand the reasoning and support for the recommendations. Our goal with this project is to help close the conceptual gap between the highly individualized and creative analysis process, and the ways that the results of that process are presented so that the analytical insights can best be shared. Towards that end we have created a lightweight, modular, adaptable and extensible framework that allows analysts to visually manipulate, query, analyze and share intelligence information and hypotheses in a collaborative environment.

Our framework has similarities with a number of established techniques including visualization of relational databases [1, 3], node-link displays of intelligence information [2], highly interactive visual exploration of heterogeneous information [4] and logical query lenses [5]. Our framework also has a number of novel features that are tuned to the analysis process, such as new visual metaphors, extensive use of new WWW technologies to facilitate system construction and collaboration, sophisticated mappings between conventional data sources and our internal data structures, and also mappings between information and visual representation. We believe that taken together these represent a fundamentally novel approach to the way that intelligence analysts can perform their analyses and share their results with others.

2. SYSTEM DESCRIPTION.

The VIM platform provides a shared collection of intelligence data called a Knowledge Space. The analyst can create any number of views of this data, choosing the view from a large (and growing) collection of both standard and novel visual representations of data. VIM has a number of interesting features which help support the analysis task.

Knowledge representation. The VIM Knowledge Space contains node, edge, and set items. All items have a type as well as other attributes. Nodes are used to represent individual intelligence facts. Node types include concepts such as Person, Place, Event, and Weapon, while the attributes are as appropriate to each type – name and birthdate for a person, or name, latitude, and longitude for a place. Edges are used to represent connections between nodes, and their types are correspondingly linking concepts such as Communication and Transaction. Sets are collections of any number of related nodes.

The platform also provides tools to convert other data, such as relational tables and web sites, into knowledge spaces. In addition, many of the views provide mechanisms to modify the knowledge space through interactions with the graphics, e.g., by generating new items using mouse and menu commands.

Visual representation. Each view has a Visual Space containing the visual items displayed by the view. The Visual Space structure parallels that of the Knowledge Space; each visual item in the space is conceptually a node, edge, or set. New types of visual items (new Java subclasses of the node, edge, and set base classes) can be created by the programmer of a view. Visual items have attributes, such as color, edge style, and icon, and each visual item may represent a knowledge item.

The view defines a mapping which generates the visual space from the knowledge space. There is no requirement that this mapping be one-to-one, or that knowledge nodes map to visual nodes; for example, we might represent a knowledge set as multiple visual edges, connecting the visual nodes that represent the knowledge nodes contained in the set, as in our parallel-axes view (Figure 1). When the knowledge space changes, the visual space is updated using the mapping; for example, in the parallel axis view of Figure 1, if a node is added to the knowledge space, the view will add the new node to any axis that should contain the node, along with any needed edges.

We have implemented such standard views as node-link networks, geographic maps, and parallel axes, as well as novel views such as order-of-interest grids and a stack view (supporting the common analyst methodology of working with stacks of paper, each stack containing associated facts), using this single underlying data representation.

XML. VIM uses XML extensively for data manipulation. The XML description of the knowledge schema (item types and attributes) is used during system build to *automatically* generate code for storing, retrieving, and modifying knowledge spaces. Knowledge spaces themselves – that is, the collections of facts

used by the analyst – are also stored in XML. Finally, the entire state of all the VIM views and queries can be saved in an XML file, and later re-loaded into the system to re-create the state.

Queries. A query tests a visual item or knowledge space item. VIM makes extensive use of queries, both to define subsets of data to be displayed and in other contexts. For example, every axis in every view has a query which is used to select the subset of knowledge space items which are displayed by that axis. These queries can be edited by the user, to change the set of items shown in the axis, as shown in Figure 1.

Mappings. A mapping is a type of query which produces an attribute for a visual item. For example, a mapping may produce the image icon used when rendering a visual item. The most common form of mapping uses the knowledge space item represented by the visual item to determine the generated attribute, but mappings can use any other information associated with the visual item to generate the property.

Every view has a list of mappings. When an item is being drawn and an attribute is needed, each mapping in the list from is checked to see if it generates the needed attribute for that item. The first to produce the property is used. (This is a conceptual model of the behavior; the implementation, of course, makes use of cached results for speed.) The last element of the list is a *knowledge-visual binder*, shared among all views, which provides all properties for all visual items, unless overridden by a mapping earlier in the list.

Magic lenses. We use the magic lens (Figure 2) as an example bringing together several VIM concepts. A lens is simultaneously a visual item, a query, and a mapping. As a visual item, the lens has a graphical representation in the view. As a query, the lens tests items; the simplest form of lens test is geometric, accepting a visual item if the visual item intersects the area of the lens. Finally, as a mapping the lens produces some graphical property used when rendering the visual item.

When a lens is created, it is added to the visual space (as a visual item) and to the mapping list (as a mapping). Because it is in the visual space, it is drawn as part of the view, and is treated like all other visual items. This means that (among other things) the user can drag the lens to different locations in the view. As the dragging occurs, the lens (as a query) reports changes in the query, caused by changes in the intersection of the lens with other visual items. This causes the view to redraw, during which the visual items access the lens (as a mapping) to determine their highlight color property. Those visual items which intersect with the lens thus get the color property from the lens mapping, while those visual items which do not intersect get the property from some mapping further down the list.

Like all VIM queries, the lens query can be edited by the user and made more useful. For example, the query can be modified to only accept those items whose knowledge space item is a Group node that has a name attribute containing either the string "Qaeda" or "Qaida". This lens will then highlight only that particular subset of the nodes, ignoring any other visual items that intersect the lens area (Figure 2).

3. Conclusions and Future Work.

Vim was delivered to working intelligence analysts in July, 2004. We are receiving feedback on the usability and capabilities of the system from these analysts. We are particularly interested in the effectiveness of the collaborative capabilities which we have built into VIM, such as the ability to save and share the complete system state. We also anticipate receiving additional suggestions for views and capabilities, particularly in the area of report generation.

4. Acknowledgements.

This work was supported and monitored by the Advanced Research and Development Activity (ARDA) and the National Geospatial Intelligence Agency (NGA) under Contract Number NMA401-02-C-0016. The views, opinions, and findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision, unless so designated by other official documentation.

References.

[1] Aiken, A., J. Chen, et al. (1996). ""Tioga-2: A Direct Manipulation Database Visualization Environment." ICDE: 208-217.

[2] i2, L. (2003). Analysts Notebook.

[3] Cruz, I.F., M. Averbuch, et al. (1997). "Delaunay: a Database Visualization System." ACM-SIGMOD Intl. Conf. on Management of Data.

[4] Roth, S. et al. (1996). "Visage: A User Interface Environment for Exploring Information." Proceedings IEEE Information Visualization.

[5] Stone, M. C., K. Fishkin, et al. (1994). "The Movable Filter as a User Interface Tool." ACM CHI Conference.

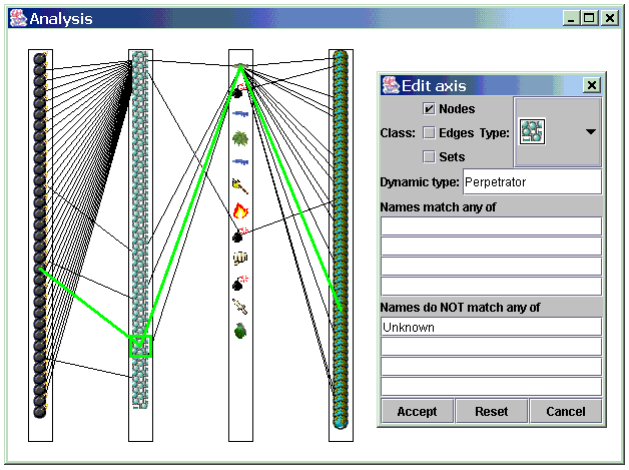

Figure 1. Vim parallel-axis view. Each column contains nodes representing a subset of knowledge-space items, as defined by a query. The green marks a particular selected set containing nodes from each column. The query for the second column is being edited in the "Edit axis" window.

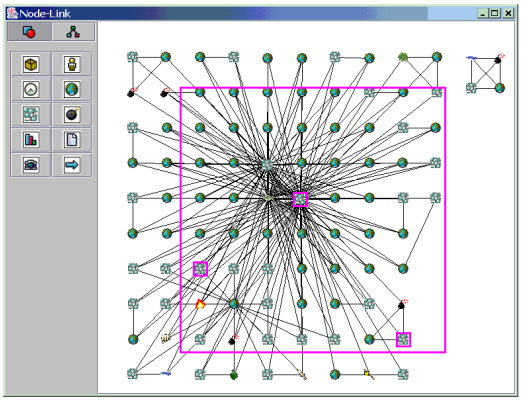

Figure 2. A node-link view with a lens query. The lens (large magenta rectangle) generates a highlight color attribute which is used to highlight all items matching the associated logical query, here finding Group nodes with a name matching Qaeda or Qaida.