# Multiple Views and Tight Coupling in Visualization: A Language, Taxonomy, and System

**Chris North**
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061 USA
north@cs.vt.edu
www.cs.vt.edu/~north/

**Abstract**

*This paper addresses the fundamental issues of multiple views and tight coupling in visualization. It presents a basic theory and formalization for multiple views consisting of a model, language, taxonomy, and system. Armed with this theory, visualization researchers have a better understanding of the design space of multiple views and a firm foundation to build on. It provides visualization designers with guidelines for design possibilities. In addition, the system demonstrates that end users can employ the specification language to gain a new level of flexibility in visualization.*

**Keywords:** visualization, multiple views, tight coupling, taxonomy, specification language.

## 1 Introduction

The use of multiple views in visualization has emerged as a valuable strategy [BWK00] [Shn98] [Rob98] [LRB97] [JBO94]. Multiple views enable the use of different visualizations for different aspects of a data set, thereby giving users additional insight. Tight coupling (sometime called "coordination" or "linking") between the views keeps the views synchronized during interaction, enables users to relate information between the views, and aids in navigation. Empirical studies of multiple views have shown significant improvement in user performance in a variety of situations (browsing software [SSS86], large 2D spaces [BW90], hierarchies [CS94], lists [NS00]).

The design of multiple-view visualizations must be fine tuned according to the target data and user tasks. The choice of views, the choice of data to place in each view, and the choice of interactive tight coupling between the views is highly dependent on these factors. Since needs change with time, a flexible approach is best.

The multiple-view strategy reveals many fundamental issues in visualization that must be explored. Can the concept of tight coupling between multiple views be solidified into a sound theory? This paper offers an initial model, specification language, and taxonomy for tight coupling between multiple views, and an implemented system that uses this specification language to link views in a flexible fashion [Nor00].

## 2 Model and Language

A formal model for multiple views and tight coupling enables the definition of a specification language. First, an individual view is defined simply as a display of a set of data items. This paper is not so concerned with the display itself. The display might be any visualization technique from the literature [CMS99] (e.g. a scatterplot of the data items).

Second, a tight coupling between a pair of views is a constraint between related data items in the views. It links user interface actions on data items in one view to (potentially different) user interface actions on related data items in the other view. Data relationships can be specified in the data itself (e.g. as in a relational database). User interface actions that can be tight coupled are view dependent. Each view has a list of user interface actions that are intrinsic to its display strategy that can be coupled to other views (e.g. selecting and highlighting data-item dots in a scatterplot).

Hence, a specification language that represents a tight coupling between two views consists of a pair of triples:

((viewA, actionA, itemsA),
(viewB, actionB, itemsB))

This specification means that performing user interface action *actionA* on the set of data items *itemsA* in view *viewA* should immediately also perform user interface action *actionB* on the set of data items *itemsB* in view *viewB*, and vice versa. Typically, *itemsA* and *itemsB* will be related by some computable relation (e.g. the identity relation so that *itemsA*=*itemsB*, or by a relational join) so that the tight coupling can be automatically executed by the system.

Tight coupling is commutative (bi-directional). If:

((viewA, actionA, itemsA),
(viewB, actionB, itemsB))

then:

((viewB, actionB, itemsB),
(viewA, actionA, itemsA)).

Tight coupling is also transitive (chain). If:

((viewA, actionA, itemsA),
(viewB, actionB, itemsB))

and

((viewB, actionB, itemsB),
(viewC, actionC, itemsC))

then:

((viewA, actionA, itemsA),
(viewC, actionC, itemsC)).

## 3 Taxonomy

A taxonomy is helpful to understand the design space of possibilities and to classify existing systems. In general, views have two basic classes of user interface actions:

- *Select*: Users can select and highlight data items in the view to express interest in them, or possibly to initiate other forms of manipulation on them. Other possibilities include graying out non-selected items as in the Dynamic Queries [AW95] approach.
- *Navigate*: Users can navigate the view to focus on certain data items or to bring other data items into display. Example navigation actions for different types of views include: scroll, pan, zoom, slice, rotate, ascend/descend tree, follow link, open file, load data, etc.

Hence, a tight coupling between a pair of views links one of these actions in one view to another action in the other view. The taxonomy classifies tight couplings by the three possible combinations of these actions (Figure 1):

1. ((viewA, *select*, itemsA),
   (viewB, *select*, itemsB))
2. ((viewA, *navigate*, itemsA),
   (viewB, *navigate*, itemsB))
3. ((viewA, *select*, itemsA),
   (viewB, *navigate*, itemsB))

Note that there are only 3 distinct combinations not 4, since combination 3 is equivalent to ((viewB, *navigate*, itemsB), (viewA, *select*, itemsA)) because tight coupling is commutative.

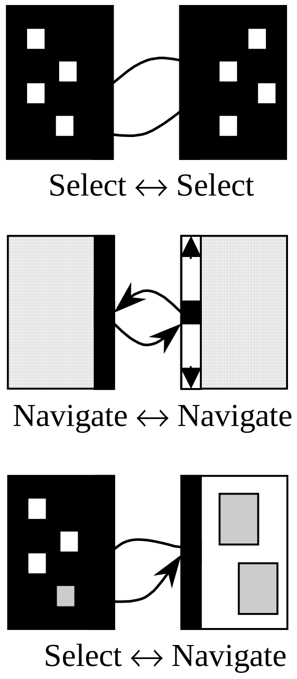


**Figure 1:** Taxonomy

### 3.1 Select ↔ Select

The *Select↔Select* class of tight coupling links selection actions on items in one view to selection actions on items in the other view, and vice versa.

For example, in EDV [EW95] (Figure 2) users can display a data table in many different types of plots. Then, users can select and highlight a subset of the data items in one of the plots using the mouse. Immediately, the same subset of items are also automatically highlighted (in bright yellow) in all the other plots. This tight coupling strategy is commonly called 'brushing and linking', and helps users correlate equivalent or related items across views. Many exploratory data analysis systems use this strategy to visualize multi-dimensional data point sets with multiple tightly coupled views. The specification for the tight coupling between the two scatterplots in Figure 2 is:

((scatterplotA, select, items),
(scatterplotB, select, items)).

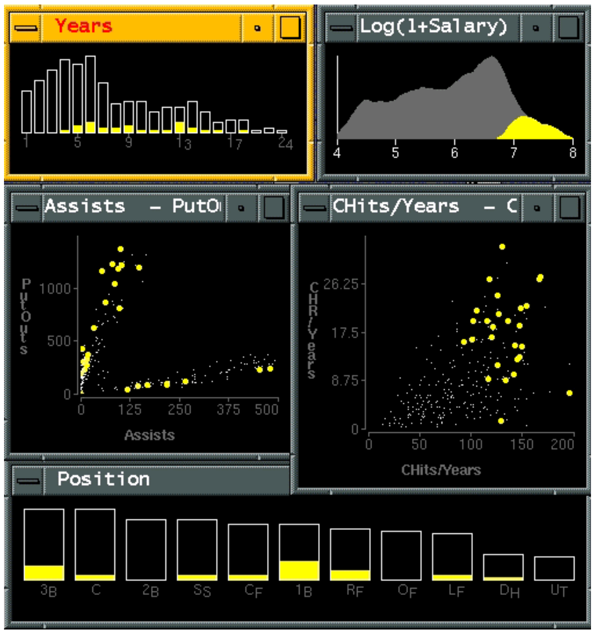


**Figure 2:** Brushing with EDV

For an example with a different type of data and views, the Navigational View Builder [MFH95] (Figure 3) enables users to visualize and navigate large web sites. It displays the entire web site as a hierarchy of web pages in a ConeTree view (left), which emphasizes the tree structure, and also a TreeMap view (center), which emphasizes numerical and categorical node attributes. Selecting a node from either view also highlights that node in the other view, and displays the selected web page in the web browser view (right). The specification for the ConeTree and TreeMap is:

((ConeTree, select, node),
(TreeMap, select, node)).

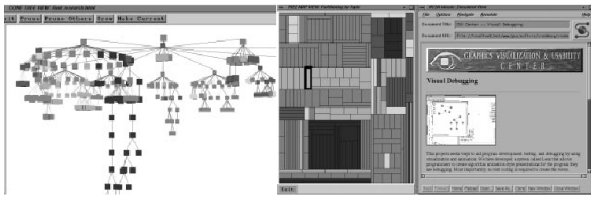


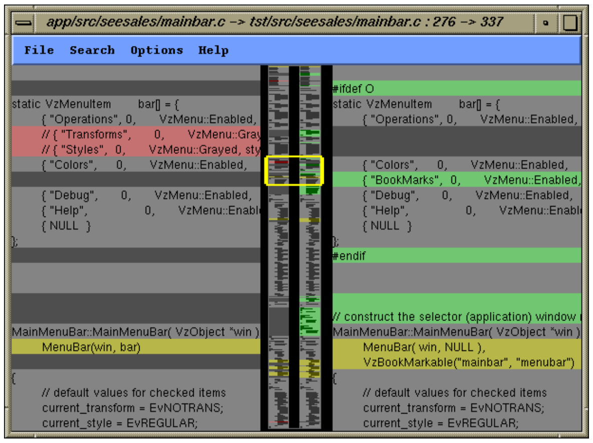
**Figure 3:** Navigational View Builder

## 3.2 Navigate ↔ Navigate

The *Navigate↔Navigate* class of tight coupling links navigation actions in one view to simultaneous navigation actions in the other view, and vice versa. This maintains synchronization of views while navigating (e.g. scrolling, panning, zooming, slicing, traversing, etc.) through correlated information spaces.

For example, the synchronized-scrolling strategy tightly couples the scroll bars of two visualizations. SeeDiff [BE96] synchronizes scrolling through two version of a source code file for analyzing changes (Figure 4). This approach avoids losing the relationship between the files and saves users from tedious repetition of scrolling actions in each view. The specification for SeeDiff assumes there is a source code line matching function called *diff*:

((oldFile, scroll, line),
(newFile, scroll, diff(line))).


app/src/seesales/mainbar.c –> tst/src/seesales/mainbar.c : 276 –> 337
File Search Options Help
```
static VzMenuItem       bar[] = {
    { "Operations", 0,      VzMenu::Enabled,
    // { "Transforms",      0,      VzMenu::Gray
    // { "Styles",  0,      VzMenu::Grayed, sty
    { "Colors",     0,      VzMenu::Enabled,

    { "Debug",      0,      VzMenu::Enabled,
    { "Help",               0,      VzMenu::Enabl
    { NULL  }
};


MainMenuBar::MainMenuBar( VzObject *win )
    MenuBar(win, bar)

{
    // default values for checked items
    current_transform = EvNOTRANS;
    current_style = EvREGULAR;
```

```
#ifdef O
static VzMenuItem       bar[] = {
    { "Operations", 0,      VzMenu::Enabled,

    { "Colors",     0,      VzMenu::Enabled,
    { "BookMarks",  0,      VzMenu::Enabled,
    { "Debug",      0,      VzMenu::Enabled,
    { "Help",               0,      VzMenu::Enabl
    { NULL  }
};
#endif

// construct the selector (application) window
MainMenuBar::MainMenuBar( VzObject *win )
    MenuBar( win, NULL ),
    VzBookMarkable("mainbar", "menubar")
{
    // default values for checked items
    current_transform = EvNOTRANS;
    current_style = EvREGULAR;
```
**Figure 4:** Synchronized scrolling with SeeDiff

As an example of how this taxonomy might suggest new possibilities, the Navigational View Builder above (Figure 3) could use this class of tight coupling to better support navigating very large websites. Zooming in on a node's subtree in the TreeMap view could exploit the

ConeTree's ability to bring a node into focus by rotating it to the front of the tree. This would help users stay oriented while navigating to lower levels of the tree structure. The specification for this new functionality is:

((ConeTree, focus, node),
 (TreeMap, zoom, node)).

## 3.3 Select ↔ Navigate

The *Select↔Navigate* class of tight coupling links selection actions in the first view to navigation actions in the second view, and vice versa (i.e. navigate in the second view to select in the first view).

For example, to finish the Navigational View Builder (Figure 3) specification, users can select any node in the ConeTree or TreeMap views to load and display the selected web page into the web browser view:

((ConeTree, select, node),
 (WebBrowser, load, node.URL))
((TreeMap, select, node),
 (WebBrowser, load, node.URL)).

This specification also suggests that, to maintain consistency, following a link in the web browser to another page on the site should highlight that new page in the ConeTree and TreeMap. The specification language helps to avoid such potential inconsistency 'bugs'.

The *Select↔Navigate* tight coupling strategy is often called "Overview+Detail". Users select items from the overview (in this case, ConeTree or TreeMap) to navigate to corresponding detailed information in the detail view (web browser). Likewise, navigating in the detail view highlights the corresponding selection in the contextual overview. Overviews provide a global map of the information, and detail views provide detailed information about a small portion. Tightly coupling the views indicates the location of and provides a mechanism for navigating the detail from within the context of the overview. This is advantageous over detail-only visualizations since overviews indicate what information is available, provide context for details, guide browsing, promote exploration, and help avoid getting lost. This strategy has become commonplace in user interface design. It is used in many standard tools such as Microsoft Word and Windows Explorer and is also used with frames on many web pages.

The *Select↔Navigate* tight coupling strategy can also be used to drill down through multiple layers of a database, with separate visualizations for each layer. CASCADE [SMH96] (Figure 5) provides four layers of tightly coupled views for zooming through 4 successive levels of scale within a large document database: the Docuverse level (collection of up to 5000 documents organized by related areas), Webview (up to 500 documents within a selected area), Landmarks (a single selected document), and Preview (details of a selected individual object in a document, such as properties of a hyperlink). The specification is:

((Docuverse, select, area),
 (Webview, load, area.documents))
((Webview, select, document),
 (Landmarks, load, document.objects))
((Landmarks, select, object),
 (Preview, load, object.properties)).

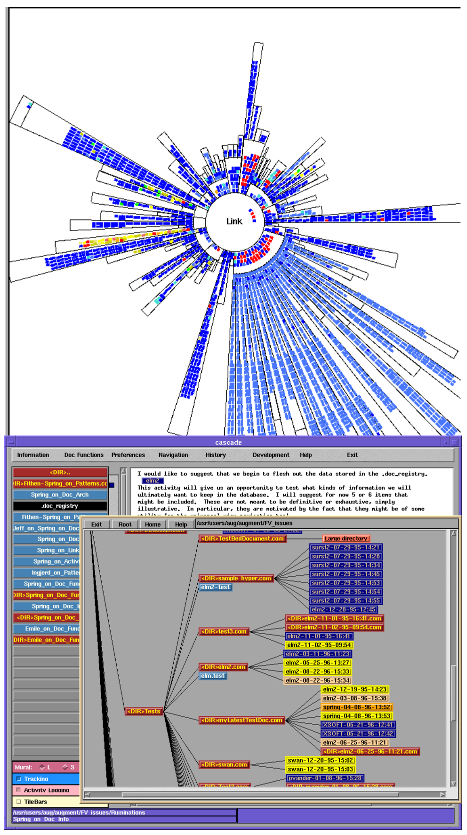


**Figure 5:** Drilling down with CASCADE

## 4 System

The Snap-Together Visualization (Snap) system [NS00] implements this specification language for tight coupling in a simple graphical user interface. It enables visualization flexibility by allowing users to easily link together multiple views as needed for their data and tasks. Users first load portions of their data into desired set of views, and then tightly couple the views together. As the specification language indicates, to tightly couple views the user simply chooses the pair of views, the desired user interface actions for each view, and the relationship between the data items.

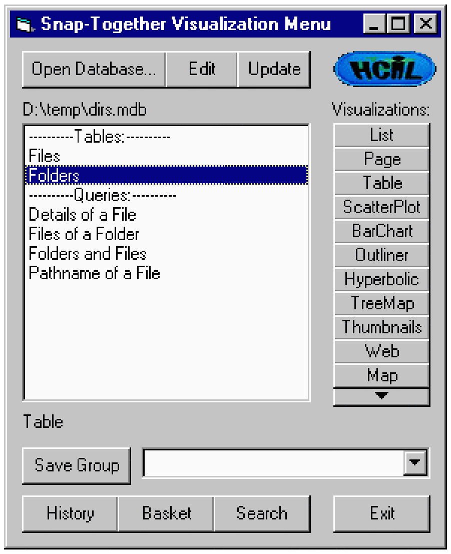
Currently, Snap can be used for visualizing data stored in relational databases (using ODBC). Snap's main menu window (Figure 6) has a pallet of different types of views that can be displayed. These views are actually third party visualization components, which can be easily added to Snap's open architecture using a simple API. Users drag tables and/or queries from the database onto the views to display them.

Snap displays a small snap icon on each view. Users can indicate they wish to tightly couple two views by dragging the snap icon from one view to the other view. This opens the Snap Specification dialog box (Figure 7), in which users select the user interfaces actions for each view that they want to tightly couple together. The relationship between the data items is determined automatically from the data schema of the database. Then, the views are tightly coupled and the user can interact with them as a unified multiple-view visualization.

### 4.1 Scenario

For example, the visualization in Figure 8 was quickly snapped together using the Snap system to visualize Census population data. It is composed of 6 views, 3 of U.S. states data (left) and 3 of counties data (right). Users can explore from nominal, geographic and numeric perspectives. The states views show an overview of the whole country. The scatterplot reveals a clear relationship between college education and income per capita. The map, scatterplot, and list views are tightly coupled for brushing (*select↔select*). Selecting Maryland on the map or list indicates in the scatterplot that it ranks very high in terms of income per capita and percent college graduates.

The states views are tightly coupled to the counties views using the overview+detail strategy (*select↔navigate*). Selecting Maryland from the states views reveals details about Maryland's counties. Maryland has two counties that have much higher percentage of college graduates than the other counties. One of these, Montgomery County, has the highest per capita income and is clearly located just north of Washington DC.

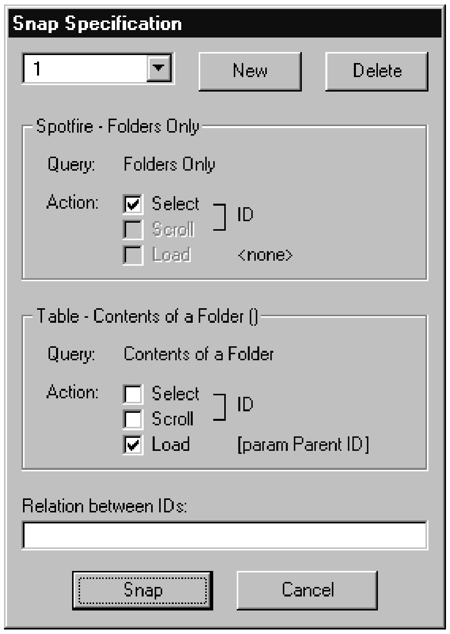


D:\temp\dirs.mdb

---------Tables:---------
Files
Folders
---------Queries:---------
Details of a File
Files of a Folder
Folders and Files
Pathname of a File

Table

**Figure 6:** Snap main menu



1

Spotfire - Folders Only
Query: Folders Only
Action: [✓ Select] ID
Scroll
Load <none>

Table - Contents of a Folder ()
Query: Contents of a Folder
Action: Select ID
Scroll
[✓ Load] [param Parent ID]

Relation between IDs:

**Figure 7:** Snap Specification dialog

The specification used to tightly couple the views in this example visualization is (note that it is not necessary to show the specifications for all pairs of the 6 views since some tight couplings can be derived by transitivity):

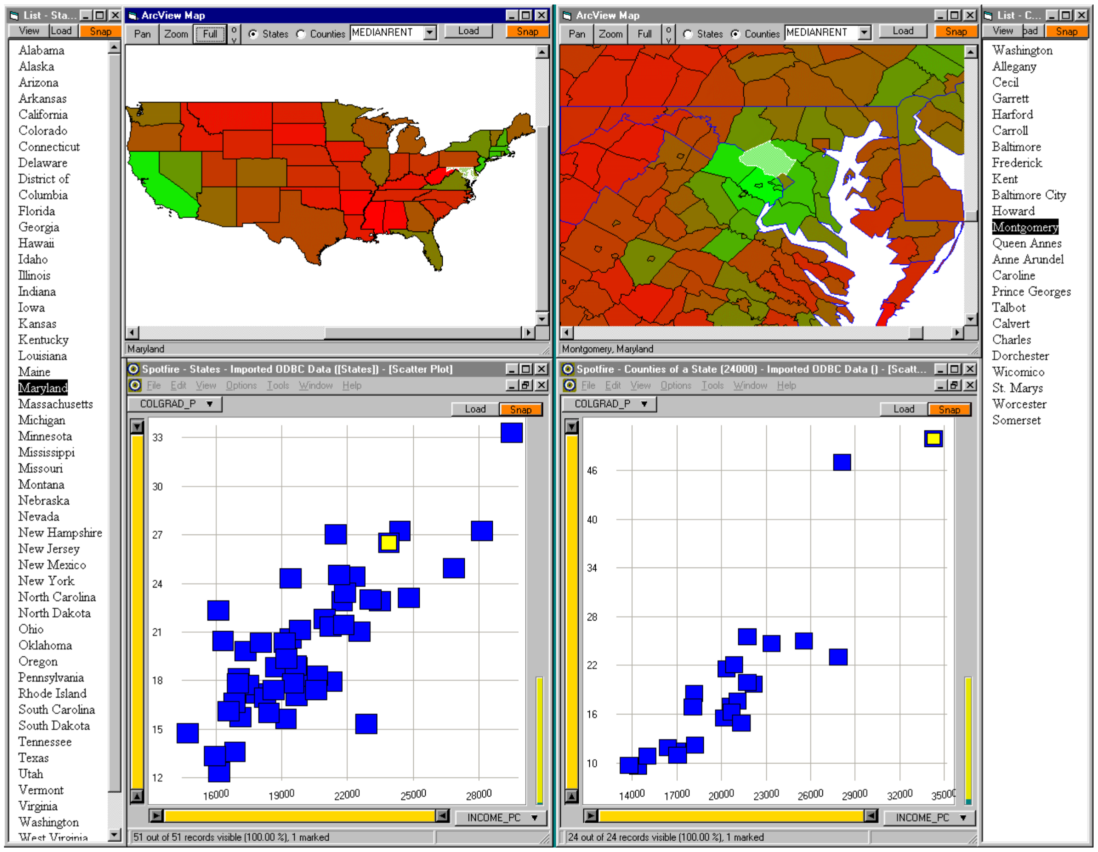
((map_states, select, state),
 (scatterplot_states, select, state))
((map_states, select, state),
 (list_states, select, state))
((map_counties, select, county),
 (scatterplot_counties, select, county))
((map_counties, select, county),
 (list_counties, select, county))
((map_states, select, state),
 (map_counties, zoom, state))
((map_states, select, state),
 (plot_counties, load, state.counties))
((map_states, select, state),
 (list_counties, load, state.counties))

This example demonstrates the use of ESRI MapObjects, a component of the popular commercial ArcView GIS software package, for the geographic views. It also uses Spotfire [AW95], a commercial data visualization tool, for the scatterplots.

## 5 Conclusions

This model, specification language, taxonomy, and system establish an initial fundamental theory for tight coupling of multiple views in visualization. This helps researchers and designers to understand the space of possibilities. In practice, this theory enables a new form of flexibility in visualization. End users can specify custom visualizations composed of tightly coupled views as needed on the fly for their own complex data and changing tasks.

Continued work is needed to expand this initial theory to more complex types of views, user interface actions, tight couplings, and data types. It would also be interesting to explore the combination of this theory for multiple views with other theories concerning the design of individual views.



51 out of 51 records visible (100.00 %), 1 marked

24 out of 24 records visible (100.00 %), 1 marked

**Figure 8:** Visualizing census data with Snap

## 6 References

[AW95] Ahlberg, C., Wistrand, E., "IVEE: An Information Visualization and Exploration Environment", *Proc. IEEE Information Visualization '95*, pp. 66-73, (1995).

[BWK00] Baldonado, M., Woodruff, A., Kuchinsky, A., "Guidelines for Using Multiple Views in Information Visualization", *Proc. Advanced Visual Interfaces 2000*, (2000).

[BE96] Ball, T., Eick, S., "Software visualization in the large", *IEEE Computer*, 29(4):33-43, (April 1996).

[BW90] Beard, D., Walker, J., "Navigational techniques to improve the display of large two-dimensional spaces", *Behaviour & Information Technology*, 9(6), pp. 451-466, (1990).

[CMS99] Card, S., Mackinlay, J., Shneiderman, B. (editors), *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann, (1999).

[CS94] Chimera, R., Shneiderman B., "An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents", *ACM Transactions on Information Systems*, 12(4), pp. 383-406, (Oct. 1994).

[EW95] Eick, S., Wills, G., "High Interaction Graphics", *European Journal of Operations Research*, #81, pp. 445-459, (1995).

[JBO94] Jacobson, A., Berkin, A., Orton, M., "LinkWinds: interactive scientific data analysis and visualization", *Communications of the ACM*, 37(4), pp. 43-52, (April 1994).

[LRB97] Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., Wenger, K., "DEVise: integrated querying and visual exploration of large datasets", *Proc. ACM SIGMOD'97*, pp. 301-312, (1997).

[MFH95] Mukherjea, S., Foley, J., Hudson, S., "Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views", *Proc. ACM CHI'95*, pp. 331-337, (1995).

[Nor00] North, C., "A User Interface for Coordinating Visualizations Based on Relational Schemata: Snap-Together Visualization", *University of Maryland, Computer Science Dept*, Doctoral Dissertation, (May 2000).

[NS00] North, C., Shneiderman, B., "Snap-Together Visualization: Can Users Construct and Operate Coordinated Views?", *Intl. Journal of Human Computer Studies*, Academic Press, 53(5), pg. 715-739, (November 2000).

[Rob98] Roberts, J., "On Encouraging Multiple Views for Visualization", *Proc. IEEE Information Visualization IV'98*, (July 1998).

[Shn98] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Third Edition, Addison-Wesley, (1998).

[SSS86] Shneiderman, B., Shafer, P., Simon, R., Weldon, L., "Display strategies for program browsing: concepts and an experiment", *IEEE Software*, 3(3), pp. 7-15, (March 1986).

[SMH96] Spring, M., Morse, E., Heo, M., "Multi-level Navigation of a Document Space", *Proc. Leveraging Cyberspace Conference*, (October 1996).