Nested Model

Fig 4.2, VAD, Munzner
Domain Situation

- Each domain has its own vocabulary for describing data and problems
- Common pitfall: Cutting corners by making assumptions rather than actually talking to target users
- Outcome: A detailed set of questions asked about or actions carried out by target users

Data/Task Abstraction

- Abstracting specific domain questions and data into a generic representation
- Task blocks are identified by the designer
  - browsing, comparing, summarizing, ...
- Abstract data blocks are designed
  - use original data, transform data, derive data, ...
- Many vis idioms are specific to a particular data type
  - which data type would support a visual representation of the data that addresses the user’s problem
Visual Encoding/Interaction

• Decide on the specific way to create and manipulate the visual representation of the abstract data block

• Visual encoding idiom
  – controls exactly what users see

• Interaction idiom
  – controls how users change what they see

Algorithm

• Algorithm used to instantiate the idiom

• Deals mainly with computer graphics implementation details

• In most cases, we’re using d3 constructs to build the idioms, so we have no control over the underlying rendering algorithm

• Some may use this level in efficiently loading/pre-filtering data to reduce interaction delay
Domain Validation

• Threat: Problem is mis-characterized

• Immediate Validation: Field Study
  – investigator observes how people act in real-world settings

• Downstream Validation: Adoption Rates
  – how many people are actually using the system
Abstraction Validation

• Threat: Wrong task/data abstraction

• Immediate Validation: *None*

• Downstream Validation: Test the system
  – by target users doing their own work

Idiom Validation

• Threat: Ineffective encoding/interaction idiom

• Immediate Validation: justification
  – carefully justify the design of the idiom with respect to known perceptual and cognitive principles

• Downstream Validation: qualitative image analysis, quality metrics, lab study
  – qualitative discussion of results, usage scenarios – show examples
  – quality metrics – quantitative measurement of result images (e.g., number of edge crossings)
  – lab study – controlled experiment in lab setting
Idiom Validation

• Threat: Ineffective encoding/interaction idiom

• Immediate Validation: justification
  – *carefully justify the design of the idiom with respect to known perceptual and cognitive principles*

• Downstream Validation: qualitative image analysis, quality metrics, lab study
  – *qualitative discussion of results, usage scenarios – show examples*
  – quality metrics – quantitative measurement of result images (e.g., number of edge crossings)
  – lab study – controlled experiment in lab setting

Algorithm Validation

• Threat: Slow algorithm

• Immediate Validation: analyze complexity
  – use standard algorithm analysis methods
  – based on number of items in the dataset or pixels in the display

• Downstream Validation: measure performance
  – measure wall-clock time and memory performance for the implemented algorithm
  – typical consideration is how dataset size affects algorithm speed
Example

Justify encoding/interaction design
Measure system time/memory
Qualitative result image analysis

Domain

Observe and interview target users
Justify encoding/interaction design
Measure system time/memory
Qualitative result image analysis

MatrixExplorer: a Dual-Representation System to Explore Social Networks
Nathalie Henry and Jean-Daniel Fekete

Abstract—MatrixExplorer is a network visualization system that uses two representations: node-link diagrams and matrices. Its design comes from a list of requirements formulated after several interviews and a participatory design session conducted with social science researchers. Although matrices are commonly used in social network analysis, very few systems support the matrix-based representations to visualize and analyze networks.

MatrixExplorer provides several novel features to support the exploration of social networks with a matrix-based representation, in addition to the standard interactive filtering and visualizing functions. It provides tools to reorder (by default) methods to position and compact nodes along different layers and hide connections among different clusters. MatrixExplorer also supports node-link diagram views which are familiar to most users and remain a convenient way to publish or communicate results. MatrixExplorer includes a mechanism to synchronize node-link diagrams views and representations keeping them ordered at all stages of the exploration process. MatrixExplorer includes a mechanism to synchronize node-link diagrams views and representations keeping them ordered at all stages of the exploration process.

Index Terms—social network visualization, node-link diagrams, matrix-based representations, exploratory process, matrix ordering, interactive clustering, consensus.

3 EXPLORATORY ANALYSIS REQUIREMENTS

We used participatory design techniques described in [29] to understand the needs of social science researchers. After several interviews, we organized a participatory design session with professional social science researchers, selected for their frequent use of social network analysis tools. The participants included: a sociologist, a psychologist, a social network analysis specialist, two historians and five computer science researchers in the fields of HCI and Information Visualization. We focused on three specific questions:

1. How would you like to create a social network?
2. How would you like to edit a created social network?
3. How would you like to explore an unknown social network?

The session was organized in four stages. First, we presented participants the state-of-the-art tools in the domain of social network analysis and a broad range of novel HCI and InfoVis techniques for interacting with graphs and data. We explicitly avoided guiding them towards specific design techniques or tools. In the second stage, they split into small groups and generated ideas in a brainstorming session, which were then ranked. In the third stage, participants captured their ideas by creating paper prototypes (Figure 2) and then filming what it would be like to interact with them. In the last stage, we reviewed the ideas altogether and gathered the common and important ones. Summarizing the working sessions, we ended up with a list of requirements for social networks exploratory analysis.

Fig. 1. MatrixExplorer showing two synchronized representations of the same network: matrix on the left and node-link on the right.

Fig. 2. Video Brainstorming showing a historian describing her ideas about using matrix-based representations to compare two networks.
Encoding/Interaction - Immediate

2.2 Matrix-Based Representations

Bertin in “Semiology of graphics” [5] introduced visual matrices to represent networks. Ghoniem et al. [18] showed that matrices outperform node-link diagrams for large graphs or dense graphs in several low-level reading tasks, except path finding. Bertin showed that matrices can be used to exhibit high-level structures by finding good permutations of their rows and columns. Thus, he qualified matrices as "mendable." Reordering rows and columns of an adjacency matrix is similar to computing the layout for a node-link diagram: finding a layout that reveals some structure in the data. Related works can be divided into two categories: automatic and interactive systems.

Algorithm

To do this, we use the matrix of shortest paths (SP matrix) instead of the adjacency matrix. Our algorithm is:

1. Compute connected components
2. For each component
3. Compute the SP matrix
4. Compute a matrix of distances between rows
5. Apply the algorithm to find a linear order
6. Compute a matrix of distances between columns
7. Apply the algorithm to find a linear order
8. End for.

Connected components are independent blocks in the adjacency matrix so an order for their rows and columns is computed for each of them. Computing the SP matrix improves notably the order quality: it reduces the impact of noise (which is important in real datasets) and gives more information for low density vertices (for which the rows and columns are very sparse). Computing the SP matrix is quadratic, as is the computation of the distance matrix for rows and the distance matrix for the columns. This has an important impact, since we want to use automatic ordering interactively. Therefore, we chose two fast ordering algorithms from the bioinformatics field. The first one is based on hierarchical clustering, followed by a seriation (HCS) and is described in [14]: the second one is based on the traveling salesman problem (TSP) as presented in [9]. To solve TSP, we use a fast heuristic described in [21]. Matrices up to 1000 rows* 1000 columns can be ordered in seconds. Ordering larger matrices introduces a noticeable delay. So far, our user did not provide us with networks having a connected component larger than a thousand actors. However, we are investigating faster algorithms such as AMADO [7].
Encoding/Interaction - Downstream

Fig 4.9, VAD, Munzner

Evaluation Paper

Not all papers have (or even should have) validation for every level
What-Why-How Table

• There are some nice examples of how to frame this in Chapter 15.

• For each of the visualization tools, there’s a table that describes things like "what: data", "what: derived", "why: tasks", "how: encode".
  – what – see Chapter 2
  – why – see Chapter 3
  – how: encode – see Chapters 7-10
  – how: manipulate – see Chapter 11
  – how: facet – see Chapter 12
  – how: reduce – see Chapters 13-14

• The "what" and "why" tasks should be driving how you developed the "how".

• Your system may not include all of the “how” items, but you need to address the ones that it does have.

Wilkinson, Anand, Grossman. “Graph-Theoretic Scagnostics”, InfoVis 2005
Seo, Shneiderman. “Interactively Exploring Hierarchical Clustering Results”, IEEE Computer, 2002