Chapter 4

Analysis: Four Levels for Validation

4.1 The Big Picture

Figure 4.1 shows four nested levels of design: domain situation, task and data abstraction, visual encoding and interaction idiom, and algorithm. The task and data abstraction level addresses the *why* and *what* questions, and the idiom level addresses the question of *how*. Each of these levels has different threats to validity, so it's a good idea to choose your validation method with these levels in mind.

4.2 Why Validate?

Validation is important for the reasons discussed in Chapter 1: the vis design space is huge, and most designs are ineffective. In that chapter, I also discuss the many reasons that validation is a tricky problem that is difficult to get right. It's valuable to think about how you might validate your choices from the very beginning of the design process, rather than leaving these considerations for the end as an afterthought.

This chapter introduces two more levels of design to consider, one above the *why–what* abstraction level and one below the *how* idiom level. While this book focuses on the two middle levels, considering all four is helpful when thinking about how to validate whether a given design has succeeded.

4.3 Four Levels of Design

Splitting the complex problem of vis design into four cascading levels provides an analysis framework that lets you address different concerns separately. Figure 4.2 shows these four levels.

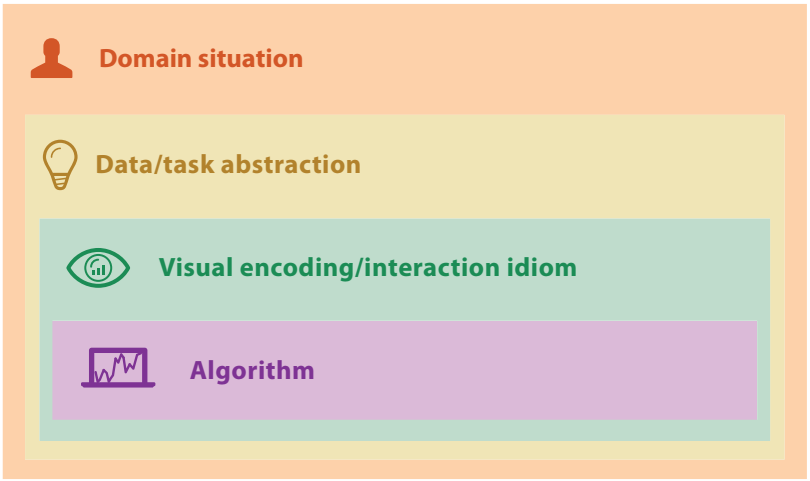


Figure 4.2. The four nested levels of vis design.

At the top is the situation level, where you consider the details of a particular application domain for vis. Next is the *what–why* abstraction level, where you map those domain-specific problems and data into forms that are independent of the domain. The following *how* level is the design of idioms that specify the approach to visual encoding and interaction. Finally, the last level is the design of algorithms to instantiate those idioms computationally.

These levels are nested; the output from an **upstream** level above is input to the **downstream** level below. A **block** is the outcome of the design process at that level. The challenge of this nesting is that choosing the wrong block at an upstream level inevitably cascades to all downstream levels. If you make a poor choice in the abstraction stage, then even perfect choices at the idiom and algorithm levels will not result in a vis system that solves the intended problem.

The value of separating these concerns into four levels is that you can separately analyze the question of whether each level has been addressed correctly, independently of whatever order design decisions were made in the process of building the vis tool. Although I encourage you to consider these four levels separately for analysis, in practice you wouldn't finalize design decisions at one level before moving on to the next. Vis design is usually a highly iterative refinement process, where a better understanding of the

blocks at one level will feed back and forward into refining the blocks at the other levels. Thus, it is one of many examples of the principle of *design as redesign* [Green 89].

4.3.1 Domain Situation

Blocks at this top level describe a specific **domain situation**, which encompasses a group of target users, their domain of interest, their questions, and their data. The term **domain** is frequently used in the vis literature to mean a particular field of interest of the target users of a vis tool, for example microbiology or high-energy physics or e-commerce. Each domain usually has its own vocabulary for describing its data and problems, and there is usually some existing workflow of how the data is used to solve their problems. The group of target users might be as narrowly defined as a handful of people working at a specific company, or as broadly defined as anybody who does scientific research.

One example of a situation block is a computational biologist working in the field of comparative genomics, using genomic sequence data to ask questions about the genetic source of adaptivity in a species [Meyer et al. 09]. While one kind of situation is a specific set of users whose questions about their data arise from their work, situations arise in other contexts. For example, another situation is members of the general public making medical decisions about their healthcare in the presence of uncertainty [Micallef et al. 12].

At this level, situation blocks are *identified*: the outcome of the design process is an understanding that the designer reaches about the needs of the user. The methods typically used by designers to identify domain situation blocks include interviews, observations, or careful research about target users within a specific domain.

Developing a clear understanding of the requirements of a particular target audience is a tricky problem for a designer.⋆ While it might seem obvious to you that it would be a good idea to understand requirements, it's a common pitfall for designers to cut corners by making assumptions rather than actually engaging with any target users.

In most cases users know they need to somehow view their data, but they typically cannot directly articulate their analysis needs in a clear-cut way. Eliciting system requirements is not easy, even when you have unfettered access to target users fluent in the vocabulary of the domain and immersed in its workflow. Asking

★ Working closely with a specific target audience to iteratively refine a design is called **user-centered design** or **human-centered design** in the human–computer interaction literature.

users to simply introspect about their actions and needs is notoriously insufficient: what users say they do when reflecting on their past behavior gives you an incomplete picture compared with what they actually do if you observe them.

The outcome of identifying a situation block is a detailed set of questions asked about or actions carried out by the target users, about a possibly heterogeneous collection of data that's also understood in detail. Two of the questions that may have been asked by the computational biologist working in comparative genomics working above were "What are the differences between individual nucleotides of feature pairs?" and "What is the density of coverage and where are the gaps across a chromosome?" [Meyer et al. 09]. In contrast, a very general question such as "What is the genetic basis of disease?" is not specific enough to be useful as input to the next design level.

4.3.2 Task and Data Abstraction

Design at the next level requires abstracting the specific domain questions and data from the domain-specific form that they have at the top level into a generic representation. Abstracting into the domain-independent vocabulary allows you to realize how domain situation blocks that are described using very different language might have similar reasons why the user needs the vis tool and what data it shows.

Questions from very different domain situations can map to the same abstract vis tasks. Examples of abstract tasks include browsing, comparing, and summarizing. Task blocks are identified by the designer as being suitable for a particular domain situation block, just as the situation blocks themselves are identified at the level above.

► Chapter 3 covers abstract tasks in detail.

Abstract data blocks, however, are *designed*. Selecting a data block is a creative design step rather than simply an act of identification. While in some cases you may decide to use the data in exactly the way that it was identified in the domain situation, you will often choose to transform the original data from its upstream form to something quite different. The data abstraction level requires you to consider whether and how the same dataset provided by a user should be transformed into another form. Many vis idioms are specific to a particular data type, such as a table of numbers where the columns contain quantitative, ordered, or categorical data; a node–link graph or tree; or a field of values at every point in space. Your goal is to determine which data type would support

a visual representation of it that addresses the user's problem. Although sometimes the original form of the dataset is a good match for a visual encoding that solves the problem, often a transformation to another data type provides a better solution.

► Chapter 2 covers abstract data types, and Section 3.4.2.3 discusses transforming and deriving data.

Explicitly considering the choices made in abstracting from domain-specific to generic tasks and data can be very useful in the vis design process. The unfortunate alternative is to do this abstraction implicitly and without justification. For example, many early web vis papers implicitly posited that solving the "lost in hyperspace" problem should be done by showing the searcher a visual representation of the topological structure of the web's hyperlink connectivity graph. In fact, people do not need an internal mental representation of this extremely complex structure to find a page of interest. Thus, no matter how cleverly the information was visually encoded at the idiom design level, the resulting vis tools all incurred additional cognitive load for the user rather than reducing it.

4.3.3 Visual Encoding and Interaction Idiom

At the third level, you decide on the specific way to create and manipulate the visual representation of the abstract data block that you chose at the previous level, guided by the abstract tasks that you also identified at that level. I call each distinct possible approach an **idiom**. There are two major concerns at play with idiom design. One set of design choices covers how to create a single picture of the data: the **visual encoding** idiom controls exactly what users see. Another set of questions involves how to manipulate that representation dynamically: the **interaction** idiom controls how users change what they see. For example, the Word Tree system [Wattenberg and Viegas 08] shown in Figure 4.3 combines the visual encoding idiom of a hierarchical tree representation of keywords laid out horizontally, preserving information about the context of their use within the original text, and the interaction idiom of navigation based on keyword selection. While it's often possible to analyze encoding and interaction idioms as separable decisions, in some cases these decisions are so intertwined that it's best to consider the outcome of these choices to be a single combined idiom.

Idiom blocks are designed: they are the outcome of decisions that you make. The design space of static visual encoding idioms is already enormous, and when you consider how to manipulate them dynamically that space of possibilities is even bigger. The

► Chapters 7 through 14 feature a thorough look at the design space of vis idioms.

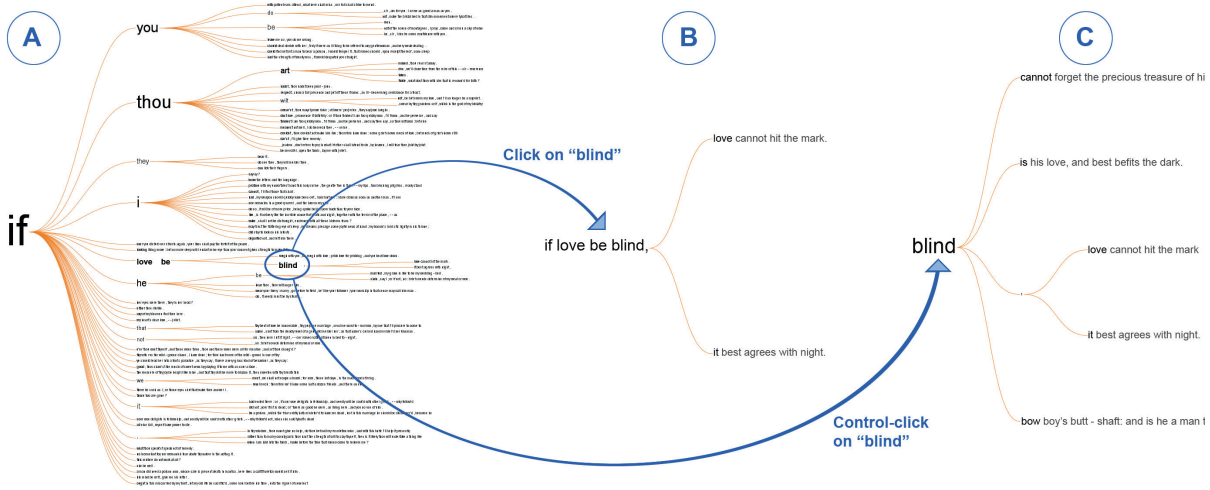


Figure 4.3. Word Tree combines the visual encoding idiom of a hierarchical tree of keywords laid out horizontally and the interaction idiom of navigation based on keyword selection. From [Wattenberg and Viegas 08, Figure 3].

► Chapters 5 and 6 cover principles of human perception and memory that are relevant for making idiom design choices.

nested model emphasizes identifying task abstractions and deciding on data abstractions in the previous level exactly so that you can use them to rule out many of the options as being a bad match for the goals of the users. You should make decisions about good and bad matches based on understanding human abilities, especially in terms of visual perception and memory.

While it's common for vis tools to provide multiple idioms that users might choose between, some vis tools are designed to be very narrow in scope, supporting only a few or even just a single idiom.

4.3.4 Algorithm

The innermost level involves all of the design choices involved in creating an **algorithm**: a detailed procedure that allows a computer to automatically carry out a desired goal. In this case, the goal is to efficiently handle the visual encoding and interaction idioms that you chose in the previous level. Algorithm blocks are also designed, rather than just identified.

You could design many different algorithms to instantiate the same idiom. For example, one visual encoding idiom for creating images from a three-dimensional field of measurements, such as scans created for medical purposes with magnetic resonance imag-

ing, is direct volume rendering. Many different algorithms have been proposed as ways to achieve the requirements of this idiom, including ray casting, splatting, and texture mapping. You might determine that some of these are better than others according to measures such as the speed of the computation, how much computer memory is required, and whether the resulting image is an exact match with the specified visual encoding idiom or just an approximation.

The nested model emphasizes separating algorithm design, where your primary concerns are about computational issues, from idiom design, where your primary concerns are about human perceptual issues.

Of course, there is an interplay between these levels. For example, a design that requires something to change dynamically when the user moves the mouse may not be feasible if computing that would take minutes or hours instead of a fraction of a second. However, clever algorithm design could save the day if you come up with a way to precompute data that supports a fast enough response.

4.4 Angles of Attack

There are two common angles of attack for vis design: top down or bottom up. With **problem-driven** work, you start at the top domain situation level and work your way down through abstraction, idiom, and algorithm decisions. In **technique-driven** work, you work at one of the bottom two levels, idiom or algorithm design, where your goal is to invent new idioms that better support existing abstractions, or new algorithms that better support existing idioms.

In problem-driven vis, you begin by grappling with the problems of some real-world user and attempt to design a solution that helps them work more effectively. In this vis literature, this kind of work is often called a **design study**. Often the problem can be solved using existing visual encoding and interaction idioms rather than designing new ones, and much of the challenge lies at the abstraction level. However, sometimes the problem motivates the design of new idioms, if you decide that no existing ones will adequately solve the abstracted design problem.

Considering the four levels of nested model explicitly can help you avoid the pitfall of skipping important steps in problem-driven work. Some designers skip over the domain situation level completely, short-circuit the abstraction level by assuming that the

first abstraction that comes to mind is the correct one, and jump immediately into the third level of visual encoding and interaction idiom design. I argue against this approach; the abstraction stage is often the hardest to get right. A designer struggling to find the right abstraction may end up realizing that the domain situation has not yet been adequately characterized and jump back up to work at that level before returning to this one. As mentioned above, the design process for problem-driven work is almost never strictly linear; it involves iterative refinement at all of the levels.

In technique-driven work, your starting point is an idea for a new visual encoding or interaction idiom, or a new algorithm. In this style of work, you start directly at one of the two lower levels and immediately focus design at that level. Considering the nested model can help you articulate your assumptions at the level just above your primary focus: either to articulate the abstraction requirements for your new idiom, or to articulate the idiom requirements for your algorithm.

The analysis framework of this book is focused on the *what–why* abstraction and *how* idiom levels and is intended to help you work in either direction. For problem-driven work, it allows you to work downward when searching for existing idioms by analyzing what design choices are appropriate for task abstraction that you have identified and data abstraction that you have chosen. For technique-driven work, it allows you to work upward by classifying your proposed idiom within the framework of design choices, giving you a clear framework in which to discuss its relationship with previously proposed idioms. Similarly, it is helpful to explicitly analyze a new algorithm with respect to the idioms that it supports. Although in some cases this analysis is very straightforward, it can sometimes be tricky to untangle connections between algorithms and idioms. Can your new algorithm simply be switched for a previous one, providing a faster way to compute the same visual encoding? Or does your new algorithm result in a visual encoding different enough to constitutes a new idiom that requires justification to show it's a good match for human capabilities and the intended task?

4.5 Threats to Validity

► Section 1.12 presented many questions to consider when validating a vis design.

Validating the effectiveness of a vis design is difficult because there are so many possible questions on the table. Considering the validity of your decisions at each level of the nested model separately

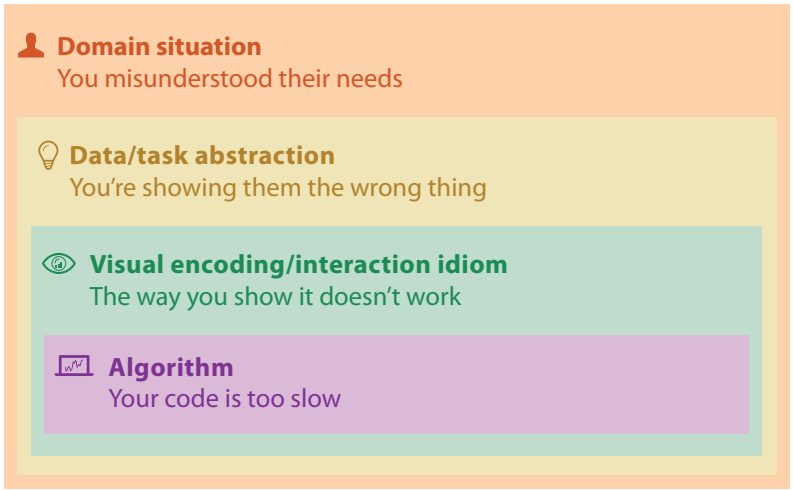


Figure 4.4. The four nested levels of vis design have different threats to validity at each level.

can help you find your way through this thicket of questions about validating your decisions, in the same way that the levels also constrain the decision-making process itself.

Each of the four levels has a different set of **threats to validity**: that is, different fundamental reasons why you might have made the wrong choices.* Figure 4.4 summarizes the four classes of threats, where **they** means the target users and **you** means the vis designer:

- Wrong problem: You misunderstood their needs.
- Wrong abstraction: You're showing them the wrong thing.
- Wrong idiom: The way you show it doesn't work.
- Wrong algorithm: Your code is too slow.

4.6 Validation Approaches

Different threats require very different approaches to validation. Figure 4.5 shows a summary of the threats and validation approaches possible at each level. The rest of this section explains

* I have borrowed the evocative phrase *threats to validity* from the computer security domain, by way of the software engineering literature. I use the word **validation** rather than **evaluation** to underscore the idea that validation is required for every level and extends beyond user studies and ethnographic observation to include complexity analysis and benchmark timings. In software engineering, **validation** is about whether you have built the right product, and **verification** is about whether you have built the product right. Similarly, in the simulation community, **validation** of the scientific model with respect to real-world observations is similarly considered separately from **verification** of the implementation, and connotes a level of rigor beyond the methods discussed here. My use of **validation** includes both of these questions.

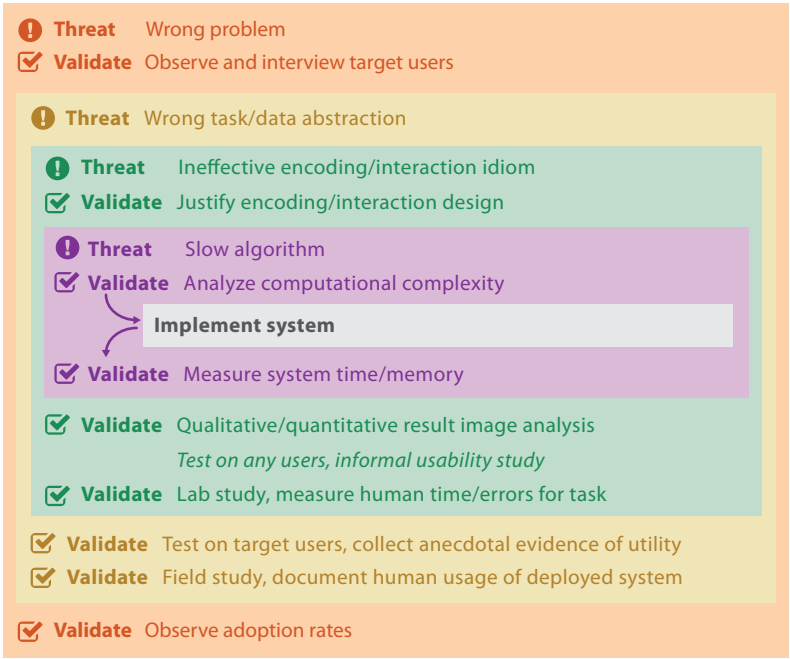


Figure 4.5. Threats and validation at each of the four levels. Many threats at the outer levels require downstream validation, which cannot be carried out until the inner levels within them are addressed, as shown by the red lines. Any single project would only address a subset of these levels, not all of them at once.

these ideas in more detail. I give only a brief outline of each validation method here; the Further Reading section at the end of this chapter has pointers to more thorough discussions of their use.

The analysis below distinguishes between **immediate** and **downstream** validation approaches. An important corollary of the model having nested levels is that most kinds of validation for the outer levels are not immediate because they require results from the downstream levels nested within them. These downstream dependencies add to the difficulty of validation: a poor showing of a test may misdirect attention upstream, when in fact the problem results from a poor choice at the current level. For example, a poor visual encoding choice may cast doubt when testing a legitimate abstraction choice, or poor algorithm design may cast doubt when testing an interaction technique. Despite their difficulties,

the downstream validations are necessary. The immediate validations only offer partial evidence of success; none of them are sufficient to demonstrate that the threat to validity at that level has been addressed.

This model uses the language of *immediate* and *downstream* in order to make the discussion of the issues at each level easier to understand—but it is not always necessary to carry out the full process of design and implementation at each level before doing any downstream validation. There are many rapid prototyping methodologies for accelerating this process by creating low-fidelity stand-ins exactly so that downstream validation can occur sooner. For example, paper prototypes and Wizard of Oz testing [Dow et al. 05] can be used to get feedback from target users about abstraction and encoding designs before diving into designing or implementing any algorithms.

4.6.1 Domain Validation

At the top level, when characterizing the domain situation, a vis designer is asserting that particular problems of the target audience would benefit from vis tool support. The primary threat is that the problem is mischaracterized: the target users do not in fact have these problems. An immediate form of validation is to interview and observe the target audience to verify the characterization, as opposed to relying on assumptions or conjectures. A common approach for this case is a **field study**, where the investigator observes how people act in real-world settings, rather than by bringing them into a laboratory setting. Field studies for domain situation assessment often involve gathering qualitative data through semi-structured interviews. The method of contextual inquiry [Holtzblatt and Jones 93], where the researcher observes users working in their real-world context and interrupts to ask questions when clarification is needed, is typically better suited for vis designers than silent observation because of the complex cognitive tasks that are targeted.

One downstream form of validation is to report the rate at which the tool has been adopted by the target audience. Of course, adoption rates do not tell the whole story: many well-designed tools fail to be adopted, and some poorly designed tools win in the marketplace. Nevertheless, the important aspect of this signal is that it reports what the target users do of their own accord, as opposed to the approaches below where target users are implicitly or explicitly asked to use a tool. In particular, a tool that is actually used by its

intended users has reached a different level of success than one that has only been used by its designers.

4.6.2 Abstraction Validation

At the abstraction level, the threat is that the identified task abstraction blocks and designed data abstraction blocks do not solve the characterized problems of the target audience. The key aspect of validation against this threat is that the system must be tested by target users doing their own work, rather than doing an abstract task specified by the designers of the vis system.

A common downstream form of validation is to have a member of the target user community try the tool, in hopes of collecting anecdotal evidence that the tool is in fact useful. These anecdotes may have the form of insights found or hypotheses confirmed. Of course, this observation cannot be made until after all three of the other levels have been fully addressed, after the algorithm designed at the innermost level is implemented. Although this form of validation is usually qualitative, some influential work toward quantifying insight has been done [Saraiya et al. 05]. As with the level above, it's important to distinguish between a discovery made by a target user and one that you've make yourself; the former is a more compelling argument for the utility of the vis tool.

A more rigorous validation approach for this level is to conduct a field study to observe and document how the target audience uses the deployed system, again as part of their real-world workflow. The key difference between field studies at this level and those just described for assessing domain situations is that you're observing how their behavior changes after intervening with the deployment of a vis tool, as opposed to documenting their existing work practices.

4.6.3 Idiom Validation

At the visual encoding and interaction idiom level, the threat is that the chosen idioms are not effective at communicating the desired abstraction to the person using the system. One immediate validation approach is to carefully justify the design of the idiom with respect to known perceptual and cognitive principles. Evaluation methods such as heuristic evaluation [Zuk et al. 08] and expert review [Tory and Möller 05] are a way to systematically ensure that no known guidelines are being violated by the design.

► Perceptual and cognitive principles will be covered in Chapters 5 and 6.

A downstream approach to validate against this threat is to carry out a **lab study**: a controlled experiment in a laboratory setting.* This method is appropriate for teasing out the impact of specific idiom design choices by measuring human performance on abstract tasks that were chosen by the study designers. Many experimental designs include both quantitative and qualitative measurements. It's extremely common to collect the objective measurements of the time spent and errors made by the study participants; subjective measurements such as their preferences are also popular. Other kinds of quantitative data that are sometimes gathered include logging actions such as mouse moves and clicks by instrumenting the vis tool, or tracking the eye movements of the participants with external gear. Qualitative data gathering often includes asking participants to reflect about their strategies through questionnaires. In this context, the expected variation in human behavior is small enough that it is feasible to design experiments where the number of participants is sufficient to allow testing for statistical significance during the analysis process.

★ The term **user study** is common in the vis literature, but it's used ambiguously: sometimes it's narrowly used to mean only a **lab study**, whereas other times it might also be applied to a **field study**. I use it broadly, to mean both of these.

Another downstream validation approach is the presentation of and qualitative discussion of results in the form of still images or video. This approach is downstream because it requires an implemented system to carry out the visual encoding and interaction specifications designed at this level. This validation approach is strongest when there is an explicit discussion pointing out the desirable properties in the results, rather than assuming that every reader will make the desired inferences by unassisted inspection of the images or video footage. These qualitative discussions of images sometimes occur as usage scenarios, supporting an argument that the tool is useful for a particular task–dataset combination.

A third appropriate form of downstream validation is the quantitative measurement of result images created by the implemented system; these are often called **quality metrics**. For example, many measurable layout metrics such as number of edge crossings and edge bends have been proposed to assess drawings of node–link networks. Some of these metrics have been empirically tested against human judgement, while others remains unproved conjectures.

Informal usability studies do appear in Figure 4.5, but I specifically refrain from calling them a validation method. As Andrews eloquently states: "Formative methods [including usability studies] lead to better and more usable systems, but neither offer validation of an approach nor provide evidence of the superiority of an approach for a particular context" [Andrews 08]. They are listed

at this level because it is a very good idea to do them upstream of attempting a validating laboratory or field study. If the system is unusable, no useful conclusions about its utility can be drawn from a user study. I distinguish usability studies from informal testing with users in the target domain, as described for the level above. Although the informal testing with target users described at the level above may uncover usability problems, the goal is to collect anecdotal evidence that the system meets its design goals. Such anecdotes are much less convincing when they come from a random person rather than a member of the target audience. In contrast, in an informal usability study, the person using the system does not need to be in the target audience; the only constraint is that the user is not the system designer.

4.6.4 Algorithm Validation

At the algorithm level, the primary threat is that the algorithm is suboptimal in terms of time or memory performance, either to a theoretical minimum or in comparison with previously proposed algorithms. Obviously, poor time performance is a problem if the user expects the system to respond in milliseconds but instead the operation takes hours or days.

► The issue of matching system latency to user expectations is discussed in more detail in Section 6.8.

An immediate form of validation is to analyze the computational complexity of the algorithm, using the standard approaches from the computer science literature. While many designers analyze algorithm complexity in terms of the number of items in the dataset, in some cases it will be more appropriate to consider the number of pixels in the display.

The downstream form of validation is to measure the wall-clock time and memory performance of the implemented algorithm. This type of measurement is so common that it's nearly mandatory for papers claiming a contribution at the algorithm level. The primary consideration is typically scalability in terms of how dataset size affects algorithm speed. One of the trickier questions is to determine what data you should use to test the algorithm. Considerations include matching up with standard **benchmarks**, which are used in previous papers, and incorporating a sufficiently broad set of data.

Another threat is incorrectness at the algorithm level, where the implementation does not meet the specification from the idiom level above. The problem could come from poor algorithm design, or the implementation of the algorithm could have bugs like any computer program. Establishing the correctness of a computer program is a notoriously difficult problem, whether through careful testing or formal methods.

The threat of algorithm incorrectness is often addressed implicitly rather than explicitly within the vis literature. Presenting still images or videos created by the implemented algorithm is one form of implicit validation against this threat, where the reader of a paper can directly see that the algorithm correctness objectives have been met. Explicit qualitative discussion of why these images show that the algorithm is in fact correct is not as common.

4.6.5 Mismatches

A common problem in weak vis projects is a mismatch between the level at which the benefit is claimed and the validation methodologies chosen. For example, the benefit of a new visual encoding idiom cannot be validated by wall-clock timings of the algorithm, which addresses a level downstream of the claim. Similarly, the threat of a mischaracterized task cannot be addressed through a formal laboratory study, where the task carried out by the participants is dictated by the study designers, so again the validation method is at a different level than the threat against the claim. The nested model explicitly separates the vis design problem into levels in order to guide validation according to the unique threats at each level.

However, it would be impossible for any single research paper to address all four levels in detail, because of limitations on space and time—such a paper would require hundreds of pages and might take a decade to finish! Instead, any individual research paper would use only a small subset of these validation methods, where careful consideration is made of which methods match the levels of design where research contributions are being claimed.

4.7 Validation Examples

This section presents examples of several vis research papers, analyzed according to the levels of vis design that they target and the methods used to validate their benefits. These projects also provide a preview of many approaches to vis that are discussed in more detail later in the book.

4.7.1 Genealogical Graphs

McGuffin and Balakrishnan present a system for the visualization of genealogical graphs [McGuffin and Balakrishnan 05]. They pro-

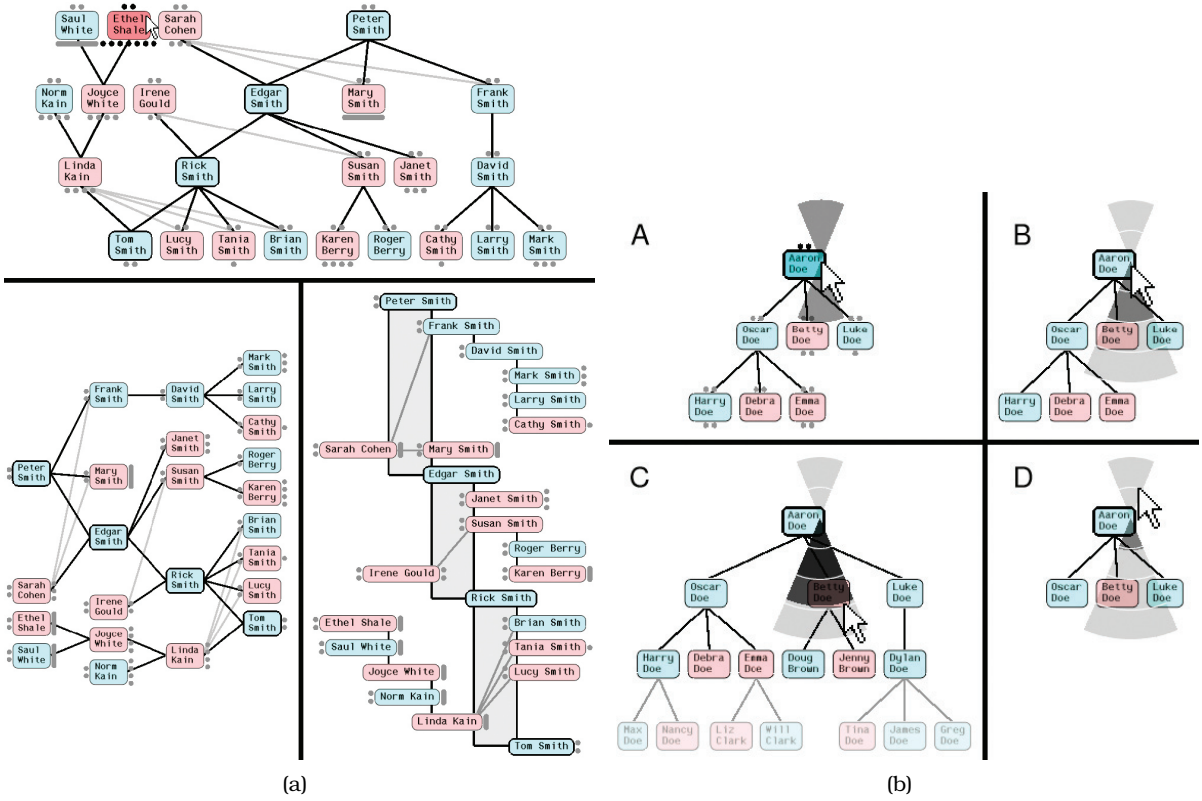


Figure 4.6. Genealogical graphs. (a) Three layouts for the dual-tree: classical node–link top-to-bottom at the top, classical left-to-right on the left, and the new indented outline algorithm on the right. (b) Widget for subtree collapsing and expanding with ballistic drags. From [McGuffin and Balakrishnan 05, Figures 13 and 14].

pose multiple new visual encoding idioms, including one based on the *dual-tree*, a subgraph formed by the union of two trees, as shown in Figure 4.6(a). Their prototype features sophisticated interaction idioms, including automatic camera framing, animated transitions, and a new widget for ballistically dragging out subtrees to arbitrary depths as shown in Figure 4.6(b).

This exemplary paper explicitly covers all four levels. The first domain situation level is handled concisely but clearly: their domain is genealogy, and they briefly discuss the needs of and current tools available for genealogical hobbyists. The paper particularly shines in the analysis at the second abstraction level. They point out that the very term *family tree* is highly misleading, be-

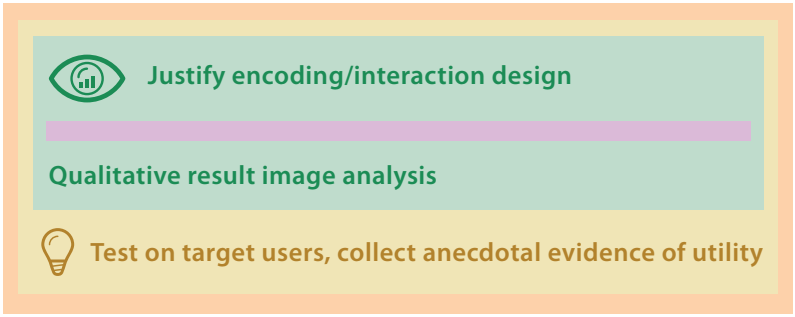


Figure 4.7. Genealogical graphs [McGuffin and Balakrishnan 05] validation levels.

cause the data type in fact is a more general graph with specialized constraints on its structure. They discuss conditions for which the data type is a true tree, a multitree, or a directed acyclic graph. They map the domain problem of recognizing nuclear family structure into an abstract task of determining subgraph structure. At the third level of the model, they discuss the strengths and weaknesses of several visual encoding idiom design choices, including using connection, containment, adjacency and alignment, and indentation. They present in passing two more specialized encoding idioms, fractal node–link and containment for free trees, before presenting in detail their main proposal for visual encoding. They also carefully address interaction idiom design, which also falls into the third level of the model. At the fourth level of algorithm design, they concisely cover the algorithmic details of dual-tree layout.

► Design choices for visual encoding idioms for network data are discussed in Chapter 9.

Three validation methods are used in this paper, shown in Figure 4.7. There is the immediate justification of encoding and interaction idiom design decisions in terms of established principles, and the downstream method of a qualitative discussion of result images and videos. At the abstraction level, there is the downstream informal testing of a system prototype with a target user to collect anecdotal evidence.

4.7.2 MatrixExplorer

Henry and Fekete present the MatrixExplorer system for social network analysis [Henry and Fekete 06], shown in Figure 4.8. Its design comes from requirements formalized by interviews and partic-

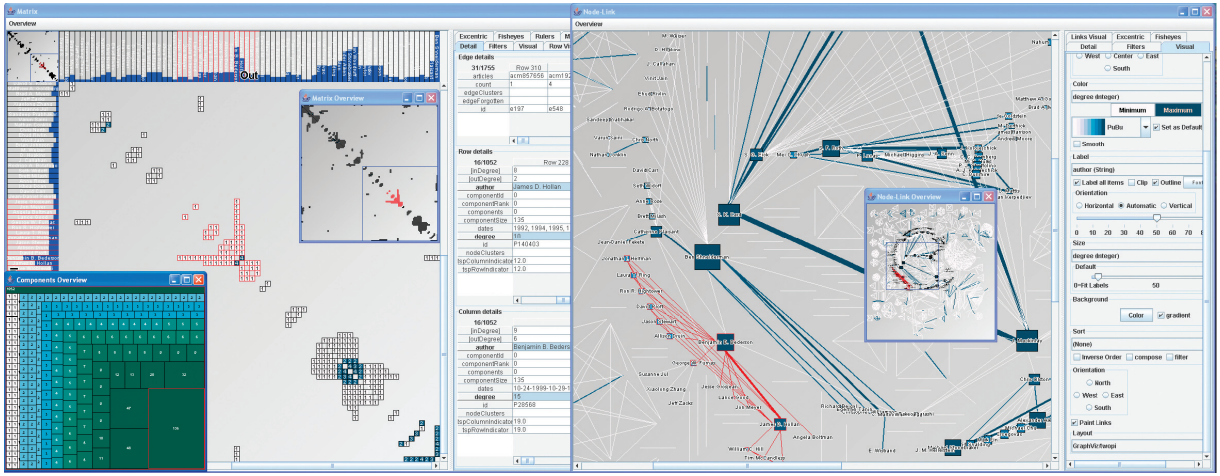


Figure 4.8. MatrixExplorer features both node–link and matrix representations in an interface designed for sociologists and historians to explore social networks. From [Henry and Fekete 06, Figure 1].

▶ The strengths and weaknesses of matrix and node–link representations of networks are discussed in Section 9.4.

ipatory design sessions with social science researchers. They use both matrix representations to minimize clutter for large and dense graphs and the more intuitive node–link representations of graphs for smaller networks.

All four levels of the model are addressed, with validation at three of the levels, shown in Figure 4.9. At the domain situation level, there is explicit characterization of the social network analysis domain, which is validated with the qualitative techniques of interviews and an exploratory study using participatory design methods with social scientists and other researchers who use social network data. At the abstraction level, the paper includes a detailed list of requirements of the target user needs discussed in terms of abstract tasks and data. There is a thorough discussion of the primary encoding idiom design decision to use both node–link and matrix views to show the data, and also of many secondary encoding issues. There is also a discussion of both basic interaction idioms and more complex interaction via interactive reordering and clustering. In both cases the authors use the immediate validation method of justifying these design decisions. There is also an extensive downstream validation of this level using qualitative discussion of result images. At the algorithm level, the focus is on the reordering algorithm. Downstream benchmark timings are mentioned very briefly.

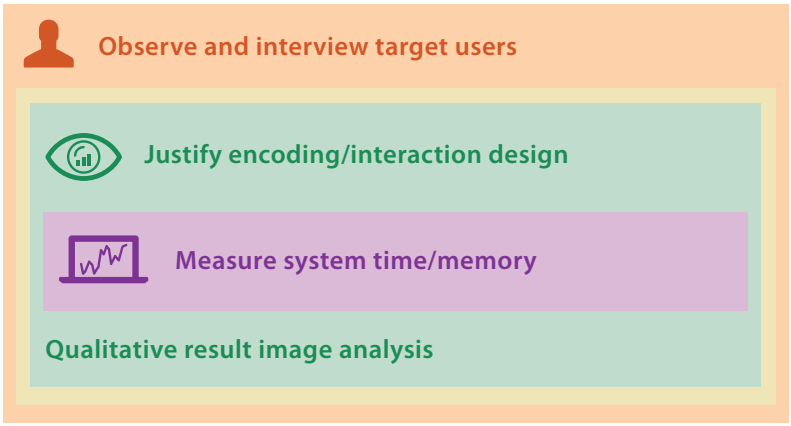


Figure 4.9. MatrixExplorer [Henry and Fekete 06] validation methods.

4.7.3 Flow Maps

Phan et al. propose a system for creating **flow maps** that show the movement of objects from one location to another, and demonstrate it on network traffic, census data, and trade data [Phan et al. 05]. Flow maps reduce visual clutter by merging edges, but most previous instances were hand drawn. They present automatic techniques inspired by graph layout algorithms to minimize edge crossings and distort node positions while maintaining relative positions, as shown in Figure 4.10. Figure 4.10(a) shows migration to California, while Figure 4.10(b) shows the top ten states sending migrants to California and New York.

► The visual encoding of geographic data is discussed in Section 8.3.1.

In their paper, Phan et al. focus on the innermost algorithm design level, but the idiom and abstraction levels are also covered. Their analysis of the useful characteristics of hand-drawn flow maps falls into the abstraction level. At the idiom level, they have a brief but explicit description of the goals of their layout algorithm, namely, intelligent distortion of positions to ensure that the separation distance between nodes is greater than the maximum thickness of the flow lines while maintaining left–right and up–down ordering relationships. The domain situation level is addressed more implicitly than explicitly: there is no actual discussion of who uses flow maps and why. However, the analysis of hand-drawn flow maps could be construed as an implicit claim of longstanding usage needs.

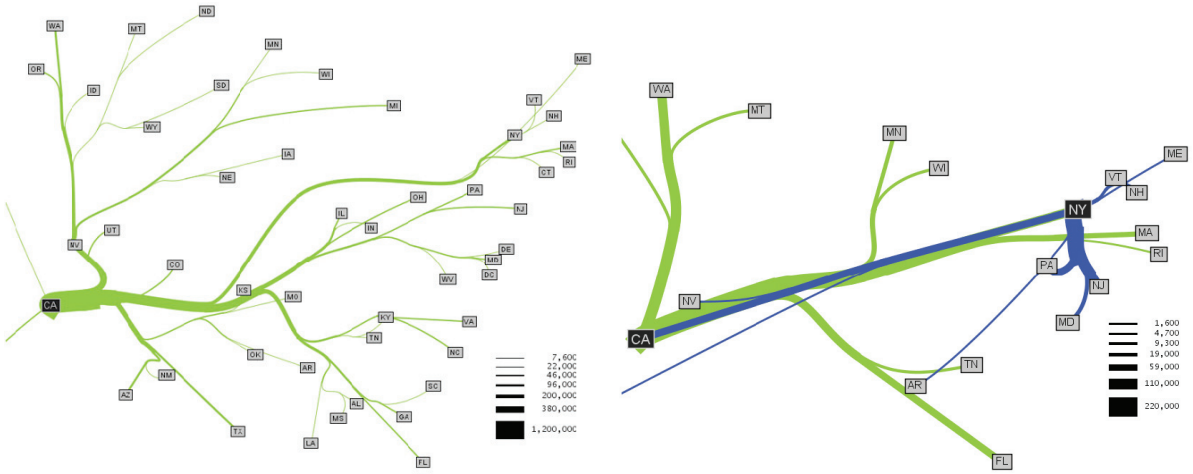


Figure 4.10. Flow maps showing migration patterns from 1995 to 2000 US Census data. (a) Migration from California. (b) The top ten states that sent migrants to California shown in green, and to New York in blue. From [Phan et al. 05, Figures 1c and 10].

Four validation methods were used in this paper, shown in Figure 4.11. At the algorithm level, there is an immediate complexity analysis. There is also a brief downstream report of system timing, saying that all images were computed in a few seconds. There is also a more involved downstream validation through the qualita-

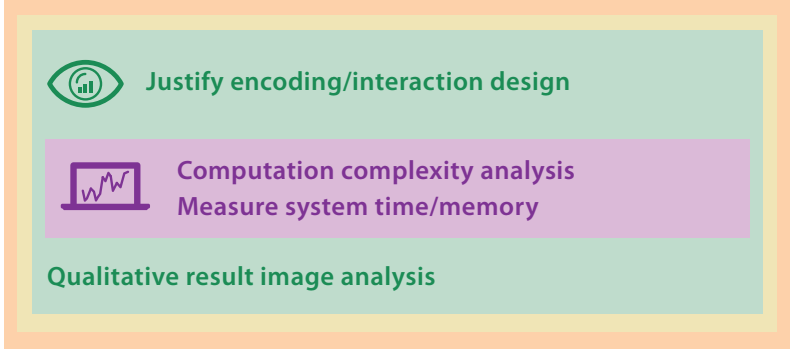
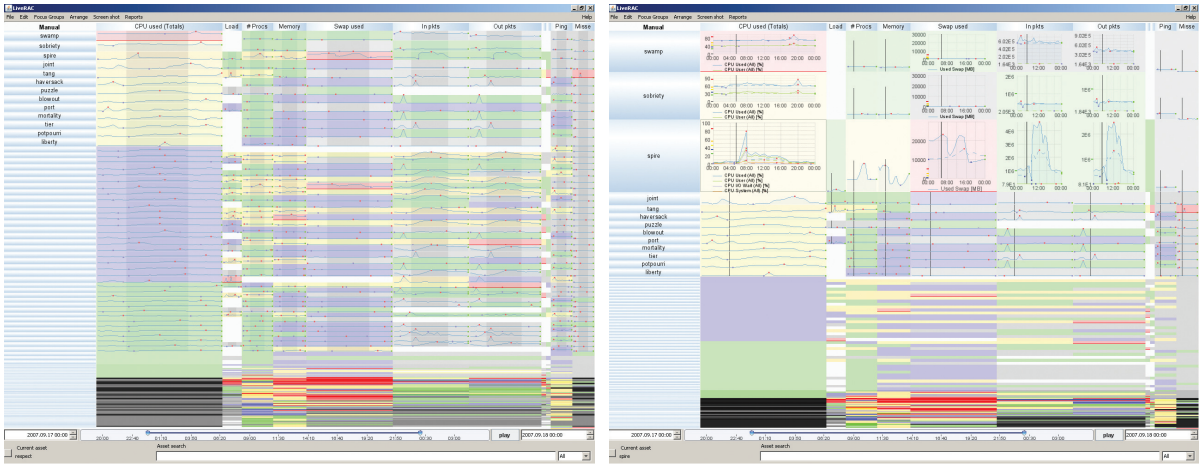


Figure 4.11. Flow map [Phan et al. 05] validation methods.



(a) (b)

Figure 4.12. LiveRAC supports exploration of system management time-series data with a reorderable matrix and semantic zooming. (a) The first several dozen rows have been stretched out to show sparklines for the devices. (b) The top three rows have been enlarged more, so the charts appear in full detail. From [McLachlan et al. 08, Figure 3].

tive discussion of result images generated by their system. In this case, the intent was mainly to discuss algorithm correctness issues at the innermost algorithm level, as opposed to addressing the visual encoding idiom level. At the idiom level, the authors justify their three fundamental requirements as the outcome of analyzing hand-drawn diagrams: intelligent distortion of positions, merging of edges that share destinations, and intelligent edge routing.

4.7.4 LiveRAC

McLachlan et al. present the LiveRAC system for exploring system management time-series data [McLachlan et al. 08]. LiveRAC uses a reorderable matrix of charts with stretch and squish navigation combined with semantic zooming, so that the chart's visual representation adapts to the available space. Figure 4.12(a) shows a mix of small boxes showing only a single attribute encoded with color and somewhat larger boxes showing concise line charts. The top three rows have been enlarged in Figure 4.12(b), providing enough room that the representation switches to detailed charts with axes and labels. The paper reports on an informal longitudinal field study of its deployment to operators of a large corporate

► Reorderable matrix alignments are covered in Section 7.5.2, semantic zooming is covered in Section 11.5.2, and stretch and squish navigation is covered in Section 14.5.

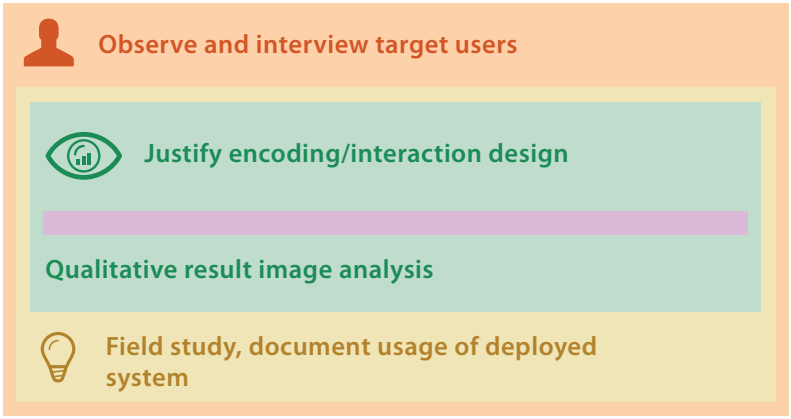


Figure 4.13. LiveRAC [McLachlan et al. 08] validation methods.

web hosting service. Four validation methods were used in this paper, shown in Figure 4.13.

At the domain situation level, the paper explains the roles and activities of system management professionals and their existing workflow and tools. The validation approach was interviews with the target audience. The phased design methodology, where management approval was necessary for access to the true target users, led to a mix of immediate and downstream timing for this validation: many of these interviews occurred after a working prototype was developed. This project is a good example of the iterative process alluded to in Section 4.3.

At the abstraction level, the choice of a collection of time-series data for data type is discussed early in the paper. The rationale is presented in the opposite manner from the discussion above: rather than justifying that time-series data is the correct choice for the system management domain, the authors justify that this domain is an appropriate one for studying this data type. The paper also contains a set of explicit design requirements, which includes abstract tasks like search, sort, and filter. The downstream validation for the abstraction level is a longitudinal field study of the system deployed to the target users, life cycle engineers for managed hosting services inside a large corporation.

At the visual encoding and interaction level, there is an extensive discussion of design choices, with immediate validation by jus-

tification in terms of design principles and downstream validation through a qualitative discussion of the results. Algorithms are not discussed.

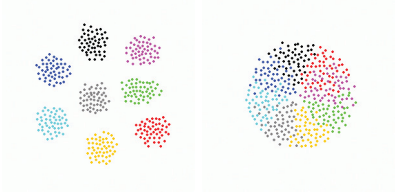
4.7.5 LinLog

Noack's LinLog paper introduces an energy model for graph drawing designed to reveal clusters in the data, where clusters are defined as a set of nodes with many internal edges and few edges to nodes outside the set [Noack 03]. Energy-based and force-directed methods are related approaches to network layout and have been heavily used in information visualization. Previous models strove to enforce a layout metric of uniform edge lengths, but Noack points out that creating visually distinguishable clusters requires long edges between them. Figure 4.14(a) shows the success of this approach, in contrast to the indifferentiated blob created by a previously proposed method shown in Figure 4.14(b).

▶ Force-directed placement is discussed in Section 9.2.

Although a quick glance might lead to an assumption that this graph drawing paper has a focus on algorithms, the primary contribution is in fact at the visual encoding idiom level. The two validation methods used in the paper are qualitative and quantitative result image analysis, shown in Figure 4.15.

Noack clearly distinguishes between the two aspects of energy-based methods for force-directed graph layout: the energy model itself versus the algorithm that searches for a state with minimum total energy. In the vocabulary of my model, his LinLog energy model is a visual encoding idiom. Requiring that the edges between clusters are longer than those within clusters is a visual encoding



(a) (b)

Figure 4.14. The LinLog energy model reveals clusters in node–link graphs. (a) LinLog clearly shows clusters with spatial separation. (b) The popular Fructerman-Reingold model for force-directed placement does not separate the clusters. From [Noack 03, Figure 1].

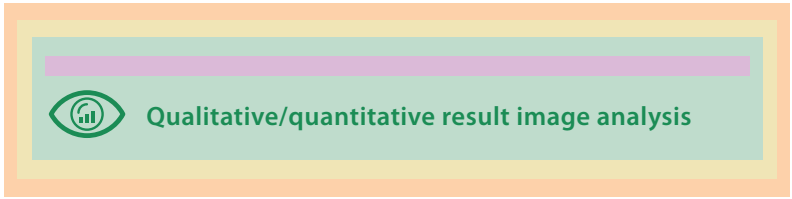


Figure 4.15. LinLog [Noack 03] validation methods.

using the visual channel of spatial position. One downstream validation approach in this paper is a qualitative discussion of result images, which is appropriate for a contribution at the encoding level. This paper also contains a validation method not listed in the model, because it is relatively rare in vis: mathematical proof. These proofs are about the optimality of the model results when measured by quantitative metrics involving edge lengths and node distances. Thus, this model classifies it in the quantitative image analysis category, another appropriate method to validate at the idiom level.

This paper does not in fact address the innermost algorithm level. Noack explicitly leaves the problem of designing better energy-minimization algorithms as future work, using previously proposed algorithms to showcase the results of his model. The domain situation level is handled concisely but adequately by referencing previous work about application domains with graph data where there is a need to see clusters. For the abstraction level, although the paper does not directly use the vocabulary of *task* and *data abstraction*, it clearly states that the abstract task is finding clusters and that the data abstraction is a network.

4.7.6 Sizing the Horizon

► Line charts are discussed in Section 9.2.

Heer et al. compare line charts to the more space-efficient **horizon graphs** [Heer et al. 09], as Figure 4.16 shows. They identify transition points at which reducing the chart height results in significantly differing drops in estimation accuracy across the compared chart types, and they find optimal positions in the speed–accuracy trade-off curve at which viewers performed quickly without attendant drops in accuracy. This paper features lab studies that are designed to validate (or invalidate) specific design choices at the

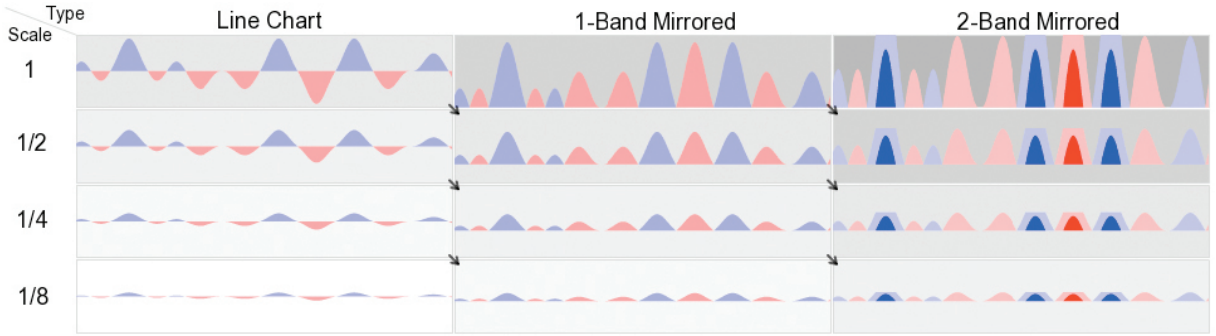


Figure 4.16. Experiment 2 of Sizing the Horizon compared filled line charts, one-band horizon graphs, and two-band horizon graphs of different sizes to find transition points where reducing chart height results in major drops in estimation accuracy across chart types. From [Heer et al. 09, Figure 7].

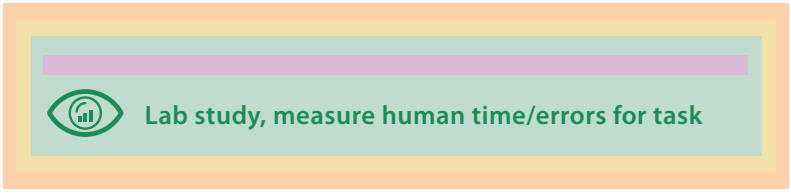


Figure 4.17. Lab studies as a validation method.

visual encoding and interaction idiom level by measuring time and error rates of people carrying out abstracted tasks, as shown in Figure 4.17.

4.8 Further Reading

The Big Picture I first presented the four-level nested model of vis design as a paper [Munzner 09a], with a discussion of blocks and guidelines between them in a follow-up paper [Meyer et al. 13]; both of these contain many more references to previous and related work. McGrath's analysis of the strengths and limitations of different experimental methods is well worth reading [McGrath 94], and it influenced my partition of validation techniques according to levels.

Problem-Driven Work A good entry point for problem-driven vis work is a detailed discussion of design study methodology, with a nine-stage framework for conducting them and suggestions for how to avoid 32 pitfalls [Sedlmair et al. 12]. Another framework for problem-driven work is the Multidimensional In-depth Long-term Case studies (MILC) approach, which also advocates working closely with domain users [Shneiderman and Plaisant 06].

Abstraction Level A recent paper argues that both data and task abstractions are important points of departure for vis designers [Pretorius and van Wijk 09]. The problems at the abstraction level fall into the realm of requirements elicitation and analysis in software engineering; a good starting point for that literature is a recent book chapter [Maalej and Thurimella 13].

Algorithm Level There are several existing books with a heavy focus on the algorithm level, including two textbooks [Telea 07, Ward et al. 10] and a large handbook [Hansen and Johnson 05]. Other options are recent survey papers on a particular topic, or specific research papers for very detailed discussion about a given algorithm. The larger issues of algorithm design are certainly not unique to vis; an excellent general reference for algorithms is a popular textbook that also covers complexity analysis [Cormen et al. 90].

Human–Computer Interaction A comprehensive textbook is a good starting point for the academic human–computer interaction literature [Sharp et al. 07]. A very accessible book is a good starting point for the large literature aimed at practitioners [Kuniavsky 03].

Evaluation Methods A book chapter provides an excellent survey and overview of evaluation and validation methods for vis, including an extensive discussion of qualitative methods [Carpendale 08]. Another discussion of evaluation challenges includes a call for more repositories of data and tasks [Plaisant 04]. A viewpoint article contains the thoughts of several researchers on why, how, and when to do user studies [Kosara et al. 03].

Field Studies For field studies, contextual inquiry is a particularly important method and is covered well in a book by one of its pioneers [Holtzblatt and Jones 93].

Experiment Design For lab studies, my current favorite references for experiment design and analysis are a cogent and accessible recent monograph [Hornbaek 13], a remarkably witty book [Field and Hole 03], and a new textbook with many examples featuring visualization [Purchase 12].