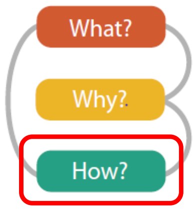
Information Visualization and Visual Analytics (M1522.000500)

How to Design and Validate VIS?

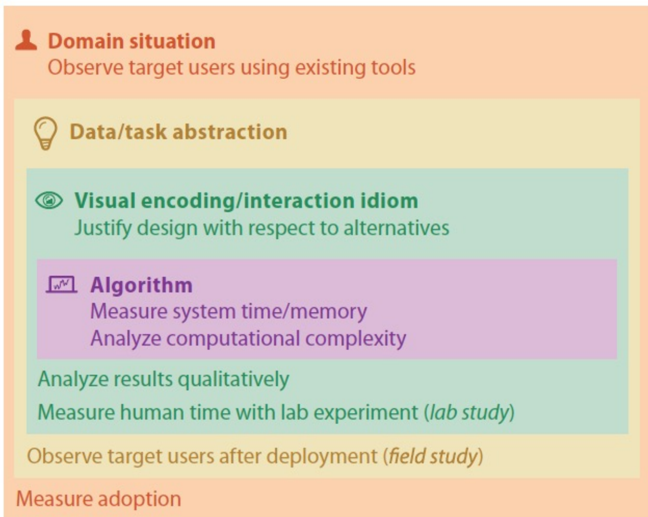
Jinwook Seo, Ph. D.
Professor, Dept. of Computer Science and Engineering
Seoul National University



Model for Design and Validation of Vis Systems

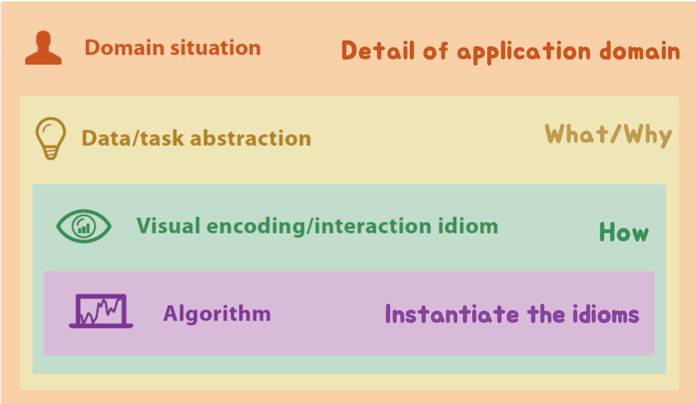
- Four nested levels of vis design

- Why validate?
  - *VIS Design space is huge, and most designs are ineffective*
  - *valuable to think about how you might validate your choices from the very beginning of the design process*



Nested Model unifying Design and Validation

- guidance on when to use what validation method
- different threats to validity at each level of model



Who is in detail of application domain

Four kinds of threats to validity

- What could fail you in your VIS design?

Four kinds of **threats** to validity

- wrong **problem**
  - they don't do that

domain problem characterization

Four kinds of **threats** to validity

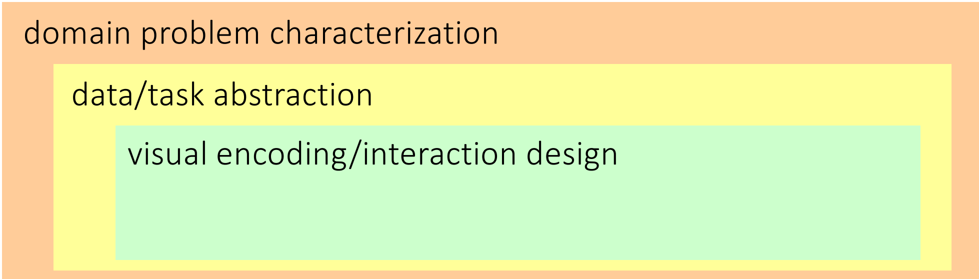
- wrong problem
  - they don't do that
- wrong abstraction
  - you're showing them the wrong thing

domain problem characterization

data/task abstraction

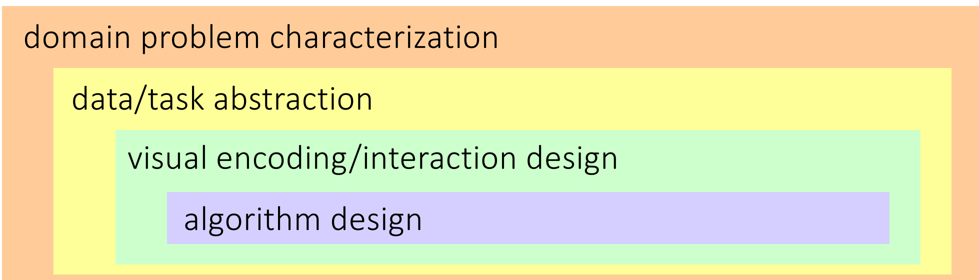
Four kinds of **threats** to validity

- wrong problem
  - they don't do that
- wrong abstraction
  - you're showing them the wrong thing
- wrong **encoding/interaction technique**
  - the way you show it doesn't work



Four kinds of **threats** to validity

- wrong problem
  - they don't do that
- wrong abstraction
  - you're showing them the wrong thing
- wrong encoding/interaction technique
  - the way you show it doesn't work
- wrong **algorithm**
  - your code is too slow



Match validation method to contributions

- each validation works for only one kind of threat to validity

threat: wrong problem

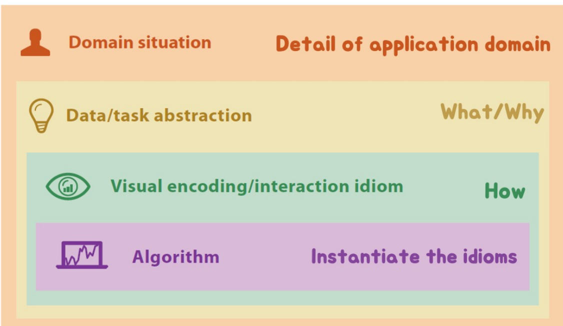
threat: bad data/operation abstraction

threat: ineffective encoding/interaction technique

threat: slow algorithm

이런 허들을 넘어서 비쥬얼라이제이션을 체크함

The Four Levels



- Value of separating these concerns into four levels
  - 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.
  - in practice you wouldn't finalize design decisions at one level before moving on to the next -> iterative design
- Nested levels
  - Output of **upstream** level → Input to the **downstream** level
  - **challenge**: upstream errors inevitably cascade down
    - if poor abstraction choice made, even perfect technique and algorithm design will not solve intended problem

Domain Situation

- Situation about particular field of interest of the target users
  - Group of target users / Domain of interest / Question / Data Collections
  - User-centered design

- Identify situation blocks
  - Users typically cannot directly (verbally) articulate their needs clearly
  - Reach the needs of target users
    via *interviews*, *observation*, *research about target users*
  - Result : Detailed set of questions or actions by target users

Task and Data Abstraction

- Abstraction of specific domain questions and data
  - Domain specific → Domain independent representation
  - Browsing, comparing, summarizing, …

- Design abstract data blocks (data transformation/**derivation**)
  - In which form the data should be used?
  - Vis idioms are specific to the data type!
    - determine which data type would support a visual encoding that solves the user’s problem

Task and Data Abstraction

- Explicitly consider the **decisions made in abstracting** from domain-specific to **generic**
- Justify your decision by comparing it to alternatives

- Assumptions for many early web vis papers : 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**.
- People do not need an internal mental representation of this extremely complex structure to find a page of interest

Visual Encoding and Interaction Idiom

- Decide on the specific way of **creating** and **manipulating** the visual representation of the abstract data block
  - Each distinct possible approach => Idiom
  - **Visual encoding idioms** for controlling *what* users see
  - **Interaction idioms** for controlling *how* users change what they see

- **Design** idiom blocks
  - Should match task/data abstractions (the data type)
  - Consider human abilities: **visual perception** and **memory**
  - Vis may contain one or more visual idioms that can be chosen

Four Levels of Design

Algorithm

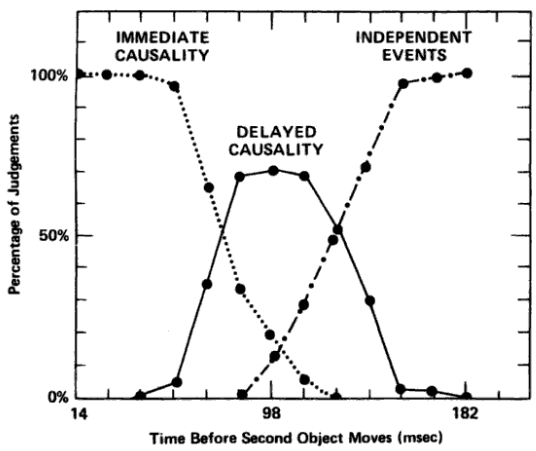
- Detailed procedure of computer to carry out desired goal
  - Efficiently handle visual encoding and interaction idioms

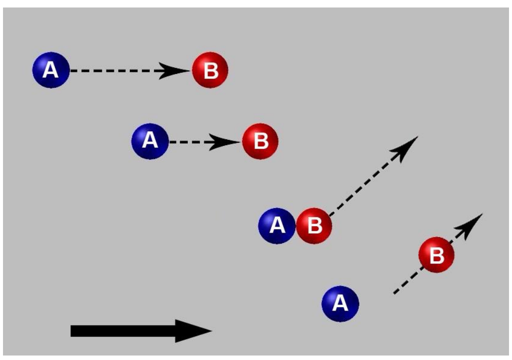
- **Design** algorithm blocks
  - Computation speed / memory / level of approximation
  - Computational issues
  - perceptual issues to consider
    - feedback within 100ms for immediate response

Perceptual Causality

Perceptual Fusion

- Perceptual Fusion: Two stimuli within a perceptual processor cycle appear *fused*
  → the first event appears to *cause* the other





Angles of Attack for Designing Vis

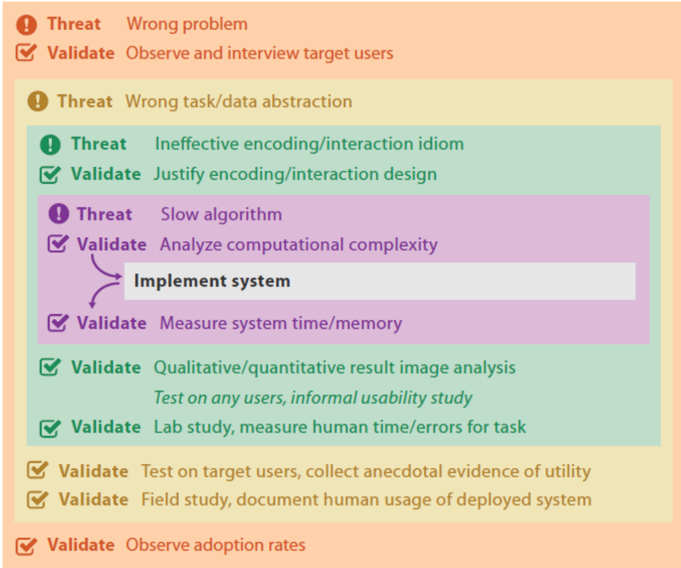
- Top down
  - **Problem driven**: search for existing idioms to solve real world user's problem → Design study
- Bottom up
  - **Technique driven**: new encoding, new interaction
    - articulate your assumptions at a level above
- Levels of design help both approaches to designing vis
  - Top down: What idiom to choose/make?
  - Bottom up: your idiom's relationship between existing idioms?

Threats to Validity

- Validating (Evaluating) the effectiveness of a vis design
  - Is hard
  - You may have made the wrong decision

Validation Approaches

- **Immediate**
- **Downstream**
  - Require result from downstream level
- (rapid) Prototyping
  - Downstream validation occur earlier
  - Wizard of OZ



Domain Validation

- **Problem** being mischaracterized
- Interview and observe target audience
  - Not just relying on assumptions or conjectures
  - **Field study** to observe target users in real-world setting
  - **Contextual inquiry** (observation in real context with questions for clarifications during the inquiry)
- Report adoption rate
  - Not the whole story

Data/Task Abstraction Validation

- Task and data abstractions do not solve the *specific topic of the target audience*
  - Must be tested after implementation
- So **no immediate** validation approach

- Let target users try the tool → anecdotal evidence
- Field study
  - Different from field study of domain validation
    - Observe how users use your design
    - Observe **change** of behavior

Visual Encoding Idiom Validation

- Is the idiom effective?
- Justify the design of idiom
  - According to perceptual and cognitive **theories** and **principles**

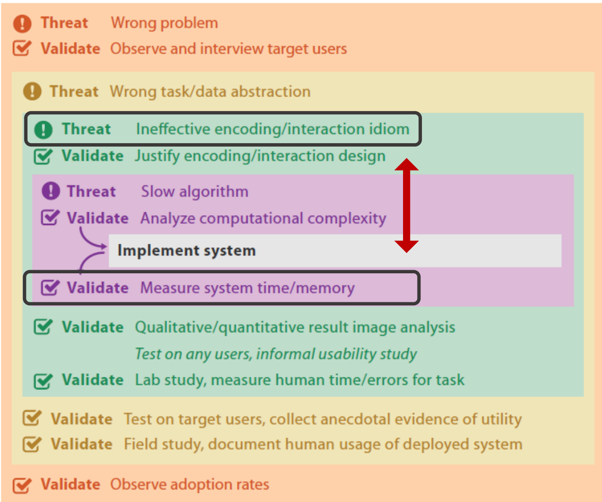
- Lab study
  - Controlled experiment with quantitative/qualitative measure
- Presentation and qualitative discussion of result (image analysis)
  → Usage scenario
- Quality Metric: Measure quality of result (e.g., # of edge crossings)

Algorithm Validation

- Time/memory performance
- Calculate computational complexity
- Measure wall-clock time / memory performance of the implemented
  - **Scalability**, Benchmarks
  - Implementation not same as expected speed

Avoid mismatches

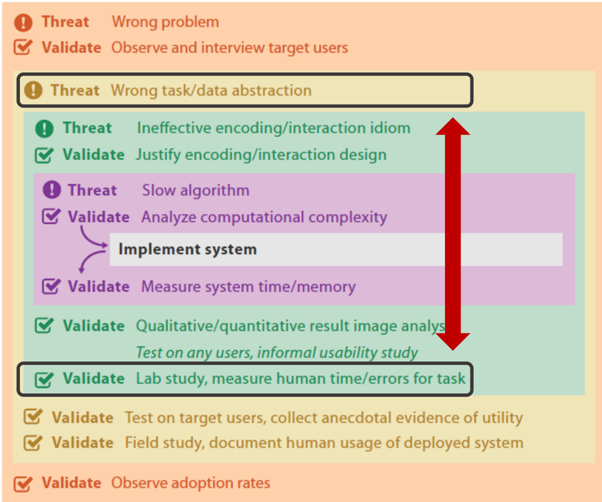
- can't validate encoding with wall clock timings



Match Validation Approach

Avoid mismatches

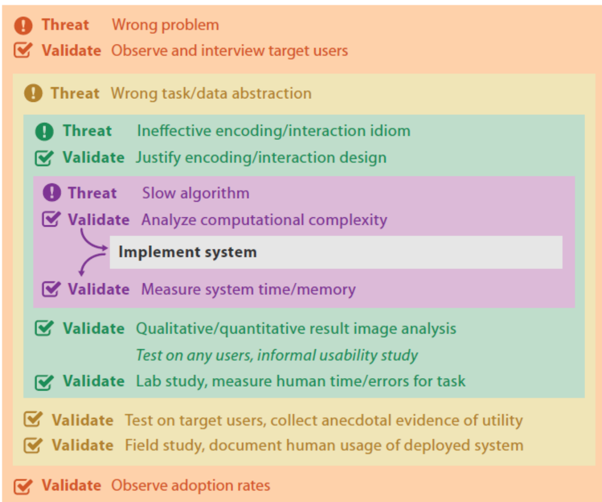
- can’t validate abstraction with lab study



Research Perspective

Single paper would include only subset

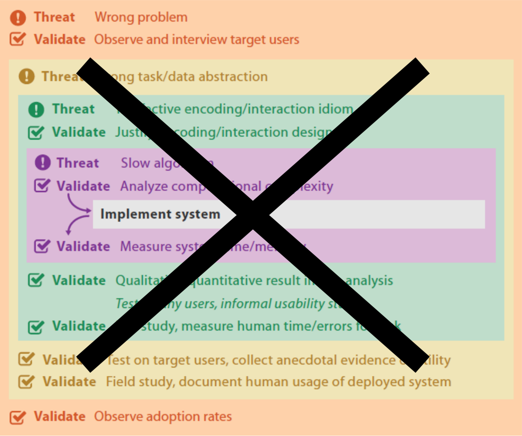
- can’t do all for same project → not enough space in paper or time to do work



Research Perspective

Single paper would include only a subset

- Pick validation method according to contribution claims
- level at which the benefit is claimed : validation method



Real Design Process

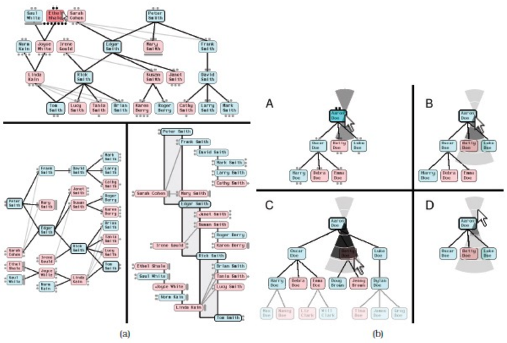
Iterative Design Process

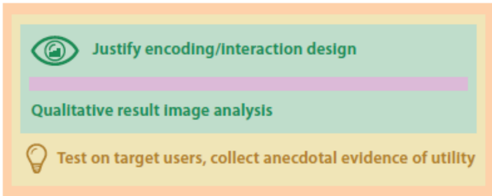
- Vis design is usually a highly ***iterative refinement*** process
  - the act of design is inherently about revisiting and reinterpreting existing designs: design as redesign
  - a better understanding of the blocks at one level will **feed back and forward** into refining the blocks at the other levels
  - levels don't need to be done in strict order
  - intellectual value of level separation
    - framework for exposition and analysis

- shortcut across inner levels + implementation
  - rapid prototyping, etc.
    - low-fidelity stand-ins so downstream validation can happen sooner

Examples (1)

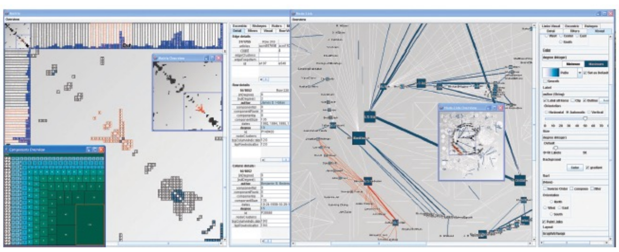
- Genealogical Graphs
  - New tree-based visual idioms

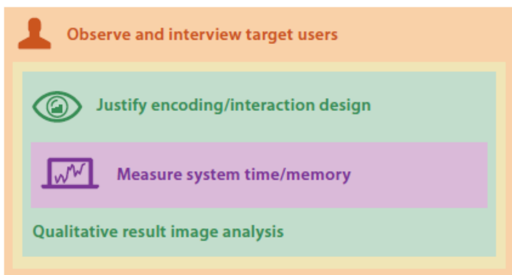




Examples (2)

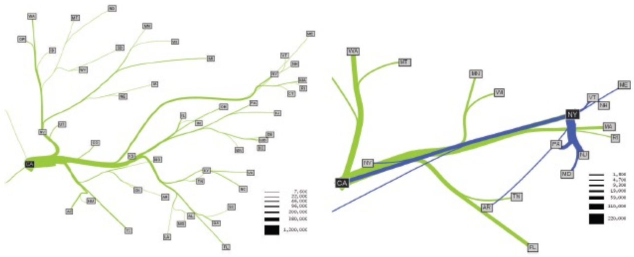
- Matrix Explorer
  - Tool for social science researchers used at social network analysis



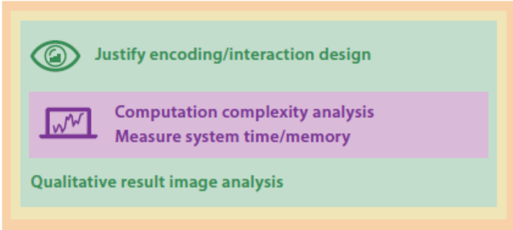


Examples (3)

- Flow Maps
  - Map of movement – reduced clutter by merging edges

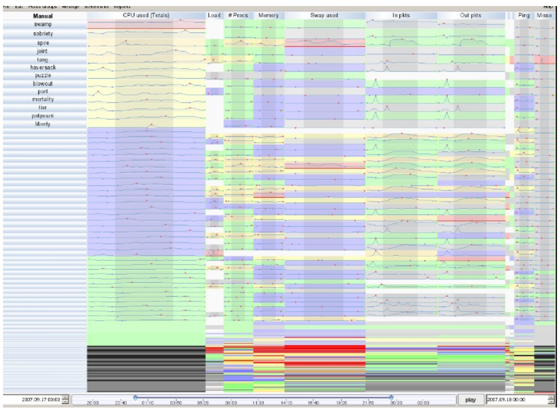


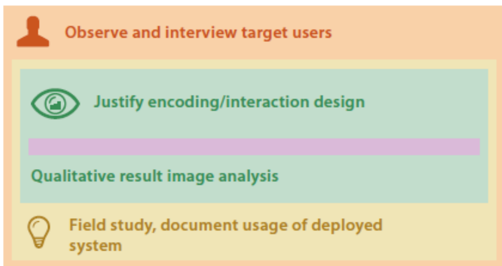
(a) (b)



Examples (4)

- LiveRAC
  - Time series data observation for system management





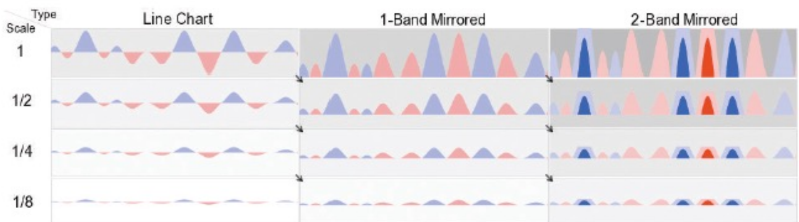
Examples (5)

- LinLog
  - Energe model for graph designed to reveal clusters

Qualitative/quantitative result image analysis

Examples (6)

- Horizon Graphs
  - Is Horizon graphs effective when chart height is reduced?



Lab study, measure human time/errors for task

Note

- Questions?