

## 5. Four Levels for Validation

Prof. Pattabiraman. V

SCSE, VIT, Chennai

# Outline



- Why Validate?
- Four Levels of Design
- Angles of Attack
- Threats to Validity
- Validation Approaches
- Validation Examples

# Why Validate?

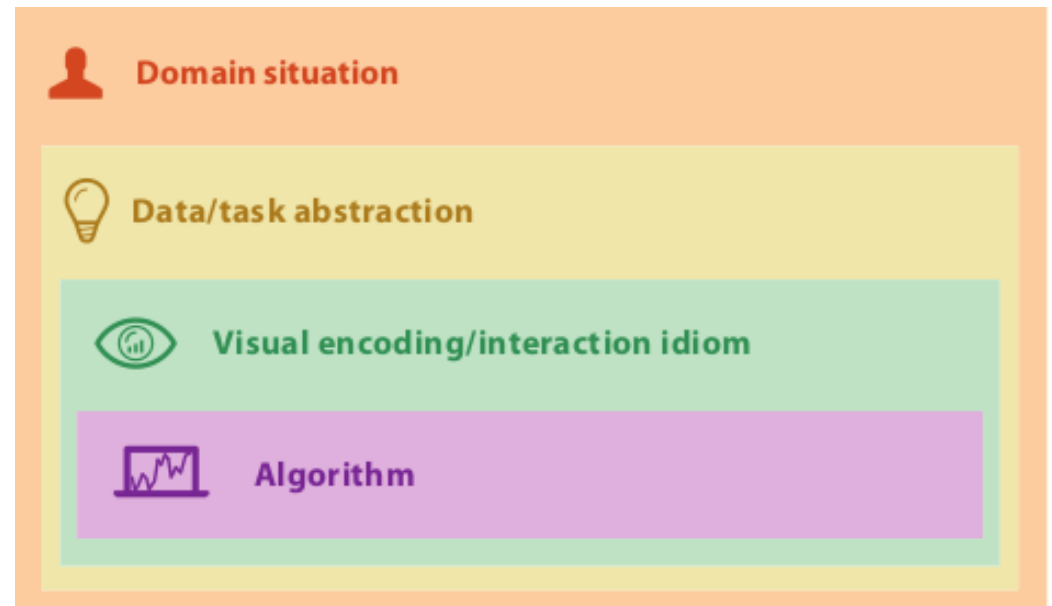


- Most designs are ineffective and validation is a tricky problem that is difficult to get right.
- It's valuable to think about how you might validate your choices from the very beginning of the design process, rather than leaving these considerations for the end as an after thought.

# 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.



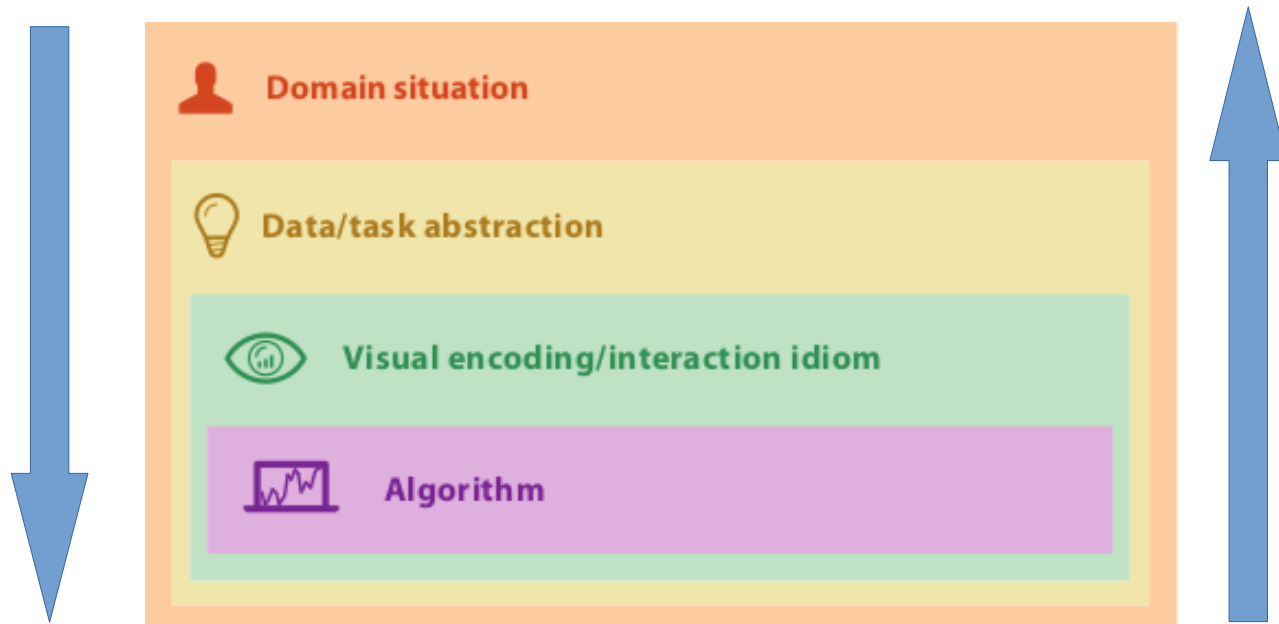
# Four Levels of Design



- 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

# Angles of Attack

- There are two common angles of attack for vis design: **top down** or **bottom up**.



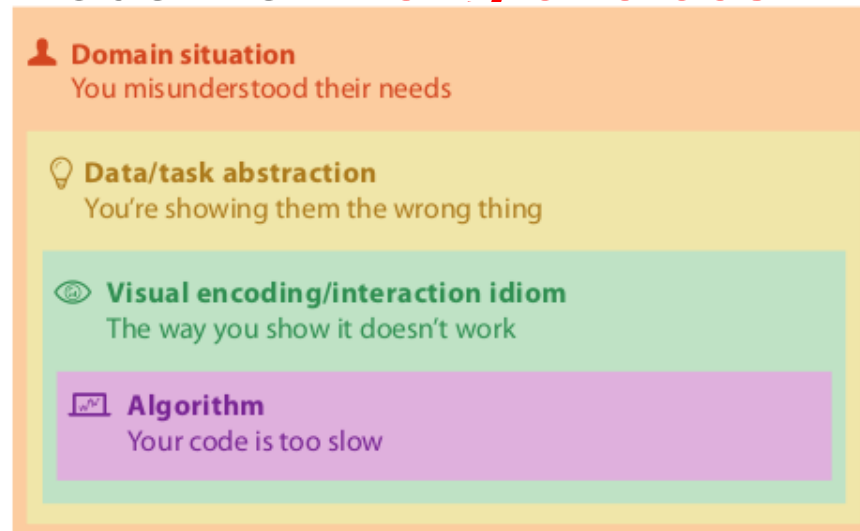
**top down (problem-driven)**

**bottom up (technique-driven)**

- Considering the four levels of nested model explicitly can help you avoid the pitfall of skipping important steps

# Threats to Validity

- 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**.



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

# Validation Approaches

- Different threats require very different approaches

❗ **Threat** Wrong problem

✓ **Validate** Observe and interview target users

❗ **Threat** Wrong task/data abstraction

❗ **Threat** Ineffective encoding/interaction idiom

✓ **Validate** Justify encoding/interaction design

❗ **Threat** Slow algorithm

✓ **Validate** Analyze computational complexity

→ **Implement system**

✓ **Validate** Measure system time/memory

✓ **Validate** Qualitative/quantitative result image analysis

*Test on any users, informal usability study*

✓ **Validate** Lab study, measure human time/errors for task

✓ **Validate** Test on target users, collect anecdotal evidence of utility

✓ **Validate** Field study, document human usage of deployed system

✓ **Validate** Observe adoption rates



# Validation Approaches



- Immediate Versus down-stream validation.
- 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.
- 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.

# Validation Approaches

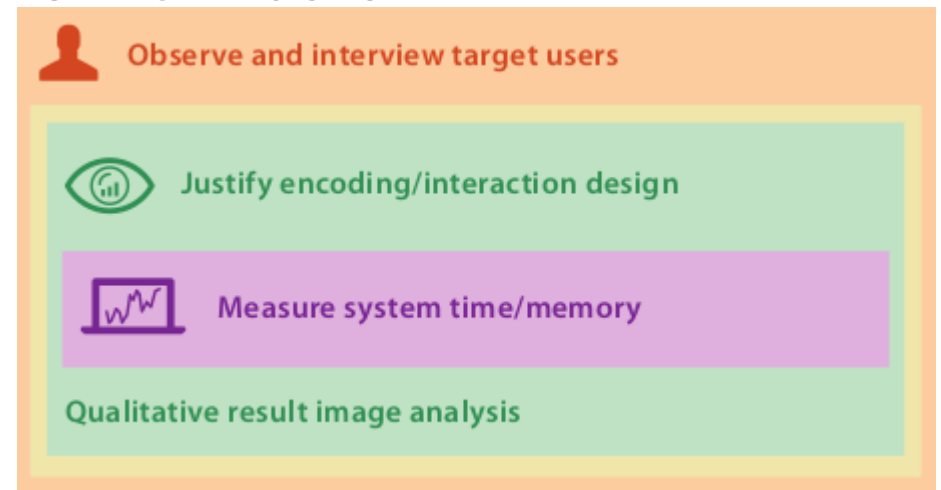


- For example, a poor visual encoding choice may cast doubt when testing a legitimate abstraction choice, or poor algorithm design may cast doubt when testing an interaction technique.
- Despite their difficulties, the downstream validations are necessary. The immediate validations only offer partial evidence of success;

# Validation Examples



- Social Network Analysis- *Matrix Explorer system for social network analysis [Henry and Fekete 06],*
- 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.



# Validation Examples



- At the abstraction level, the 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*.

# Validation Examples



- 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



Thank You