Ch 4: Validation
Paper: D3

Tamara Munzner
Department of Computer Science
University of British Columbia

CPSC 547, Information Visualization
Day 6: 29 September 2015

http://www.cs.ubc.ca/~tmm/courses/547-15
News

• LAVA Hackathon Oct 24-25
  – http://blogs.ubc.ca/lava/
  – Learning Analytics, Visual Analytics
  – there are no lectures in this class that week
    • if you want to avoid withdrawal :-)


Four Levels of Design and Validation

• four levels of design problems
  – different threats to validity at each level

- **Domain situation**
  You misunderstood their needs

- **Data/task abstraction**
  You’re showing them the wrong thing

- **Visual encoding/interaction idiom**
  The way you show it doesn’t work

- **Algorithm**
  Your code is too slow
Nested Levels of Design and Validation

- **Domain situation**
  Observe target users using existing tools

- **Data/task abstraction**

  - **Visual encoding/interaction idiom**
    Justify design with respect to alternatives

  - **Algorithm**
    Measure system time/memory
    Analyze computational complexity

    - Analyze results qualitatively
    - Measure human time with lab experiment (*lab study*)

- **Behavioral studies**
  Observe target users after deployment (*field study*)

Measure adoption

- mismatch: cannot show idiom good with system timings
- mismatch: cannot show abstraction good with lab study
Directionality

- **Domain situation**
  - **Problem-driven work**

- **Data/task abstraction**
  - **Technique-driven work**

- **Visual encoding/interaction idiom**

- **Algorithm**
Paper: D3

- paper types
  - design studies
  - technique/algorithm
  - evaluation
  - model/taxonomy
  - **system**
    - today’s emphasis

Toolkits

• imperative: how
  – low-level rendering: Processing, OpenGL
  – parametrized visual objects: prefuse
    • also flare: prefuse for Flash

• declarative: what
  – Protoviz, D3, ggplot2
  – separation of specification from execution

• considerations
  – expressiveness
    • can I build it?
  – efficiency
    • how long will it take?
  – accessibility
    • do I know how?
OpenGL

• graphics library
  – pros
    • power and flexibility, complete control for graphics
    • hardware acceleration
    • many language bindings: C, C++, Java (w/ JOGL)
  – cons
    • big learning curve if you don’t know already
    • no vis support, must roll your own everything
  – example app: TreeJuxtaposer

Processing

• layer on top of Java/OpenGL
• visualization esp. for artists/designers
• pros
  – great sandbox for rapid prototyping
  – huge user community, great documentation
• cons
  – poor widget library support
• example app: MizBee

prefuse

• infovis toolkit, in Java
• fine-grained building blocks for tailored visualizations
• pros
  – heavily used (previously)
  – very powerful abstractions
  – quickly implement most techniques covered so far
• cons
  – hasn’t been under active development for
  – nontrivial learning curve
• example app: DOI Trees Revisited

prefuse

- separation: abstract data, visual form, view
  - data: tables, networks
  - visual form: layout, color, size, ...
  - view: multiple renderers

InfoVis Reference Model

- conceptual model underneath design of prefuse and many other toolkits
- heavily influenced much of infovis (including nested model)
  - aka infovis pipeline, data state model

[Redrawn Fig 1.23. Card, Mackinlay, and Shneiderman. Readings in Information Visualization: Using Vision To Think, Chapter 1. Morgan Kaufmann, 1999.]
Declarative toolkits

• imperative tools/libraries
  – say exactly **how** to do it
  – familiar programming model
    • OpenGL, prefuse, ...

• declarative: other possibility
  – just say **what** to do
  – Protovis, D3
Protovis

• declarative infovis toolkit, in Javascript
  – also later Java version
• marks with inherited properties
• pros
  – runs in browser
  – matches mark/channel mental model
  – also much more: interaction, geospatial, trees,…
• cons
  – not all kinds of operations supported
• example app: NapkinVis (2009 course project)

[Fig 1, 3. Chao. NapkinVis. http://www.cs.ubc.ca/~tmm/courses/533-09/projects.html#will]
Protovis Validation

• wide set of old/new app examples
  – expressiveness, effectiveness, scalability
  – accessibility

• analysis with cognitive dimensions of notation
  – closeness of mapping, hidden dependencies
  – role-expressiveness visibility, consistency
  – viscosity, diffuseness, abstraction
  – hard mental operations

D3

• declarative infovis toolkit, in Javascript
• Protovis meets Document Object Model
• pros
  – seamless interoperability with Web
  – explicit transforms of scene with dependency info
  – massive user community, many thirdparty apps/libraries on top of it, lots of docs
• cons
  – even more different from traditional programming model
• example apps: many
D3

• objectives
  – compatibility
  – debugging
  – performance

• related work typology
  – document transformers
  – graphics libraries
  – infovis systems
    • general note: all related work sections are a mini-taxonomy!

D3 capabilities

• query-driven selection
  – selection: filtered set of elements queries from the current doc
    • also partitioning/grouping!
  – operators act on selections to modify content
    • instantaneous or via animated transitions with attribute/style interpolators
    • event handlers for interaction

• data binding to scenegraph elements
  – data joins bind input data to elements
  – enter, update, exit subselections
  – sticky: available for subsequent re-selection
  – sort, filter

D3 Features

• document transformation as atomic operation
  – scene changes vs representation of scenes themselves

• immediate property evaluation semantics
  – avoid confusing consequences of delayed evaluation

• validation
  – performance benchmarks
    • page loads, frame rate
  – accessibility
    • everybody has voted with their feet by now!
Next Time

• to read
  – VAD Ch. 7: Tables
    • paper type: survey
Now

• guest lectures on tools & resources
  – Matt Brehmer
  – http://www.cs.ubc.ca/group/infovis/resources.shtml