domain specific languages for
Interactive Data Analysis

Jeffrey Heer  Stanford University
**End-User Programmers**

People who write programs, but not as their primary job function.

Instead, they must write programs in support of achieving their main goal, which is something else, such as accounting, designing a web page, doing office work, scientific research, etc.

Myers, Ko & Burnett 2006

---

**End-User Programming Methods**

- Domain Specific Languages
- Keyword Programming
- Programming-by-Demonstration
- Visual Programming

---

**Today:**

A DSL for making interactive visualizations (*Protovis*)

An interactive visual tool for making statements in a DSL (*Wrangler*)
How do people create visualizations?

**Chart Typology**
- Pick from a stock of templates
- Easy-to-use but limited expressiveness
- Prohibits novel designs, new data types

**Component Architecture**
- Permits more combinatorial possibilities
- Novel views require new operators, which requires software engineering.

---

**Chart Typologies**
- Excel, Many Eyes, Google Charts

**Visual Analysis Languages**
- Tableau VizQL, ggplot2, HiVE

**Component Model Architectures**
- Improvise, Prefuse, Flare

**Graphics APIs**
- OpenGL, Java2D, GDI+, Processing

---

**Protovis**: A Declarative Language for Visualization

A graphic is a composition of data-representative marks.

with Mike Bostock & Vadim Ogievetsky
Protovis
Create customized visualizations using a declarative specification language.

```javascript
var vis = new pv.Panel();
vis.add(pv.Bar)
data([1, 1.2, 1.7, 1.5, .7])
.bottom(10)
.width(50)
.height(function(d) d * 70)
.left(function() this.index * 25 + 20);
vis.render();
```

Protovis (http://protovis.org) – Declarative Visualization Specification

Bach's Prelude #1 in C Major | Jieun Oh
Exploiting Declarative Specification

Protovis has led to faster designs, less code
Job Voyager: 5x less code, 10x less dev time
Over 40,000 downloads and widely in use
Multiple implementations: JavaScript & Java
Behind-the-scenes optimization & parallelization
20x scalability over prior systems (in Java)

Graph Viewer
Protovis vs. Prefuse
Nodes of degree 10
N from 100 to 100k

Interactive Graph Layout Performance

20x
**Design Process**

Determine the domain entities and operators
Iterative development with domain expert (me)
Generate alternative designs
Write hypothetical code; compare & contrast
Minimize surface area
Expressiveness, efficiency, accessibility
Cognitive Dimensions of Notation [Green et al]
Data Wrangling (n):
A process of iterative data exploration and transformation that enables analysis.

The goal of wrangling is to make data useful:
- Map data to a form readable by downstream tools (database, stats, visualization, …)
- Identify, document, and (where possible) address data quality issues.

DataWrangler with Sean Kandel, Philip Guo, Ravi Parikh, Andreas Paepcke & Joe Hellerstein

From UI to running code...

```java
split('data').on(NEWLINE).max_splits(NO_MAX)
split('split').on(COMMA).max_splits(NO_MAX)
columnName().row(0)
delete(isEmpty())
extract('Year').on(/.*\s*/).after(/in /)
fill('extract').method(COPY).direction(DOWN)
delete('Year starts with "Reported crime in"')
columnName('extract').to('State')
```
Wrangler in 2 Parts...

1. Declarative data transformation language
   - **Tuple mapping** - split, merge, extract, delete
   - **Reshaping** - e.g., fold, unfold (cross-tabulation)
   - **Lookups and joins** - e.g., FIPS code to US state
   - **Sorting, aggregation, etc.**

   Informed by prior work in databases:
   - Potter’s Wheel & SchemaSQL

Wrangler in 2 Parts...

1. Declarative data transformation language
   + 2. Mixed-initiative interface for data transforms
     - **Select** data elements of interest
     - **Suggest** applicable transforms
     - Enable rapid **preview and refinement**

Transform Suggestion

- **Interaction**
  - Infer Operands
  - Generate Transforms
  - Rank Transforms
  - Present Top-N

Transform Suggestion

- **Interaction**
  - Infer Operands
  - Generate Transforms
  - Rank Transforms
  - Present Top-N

- **Text Selection**
- **Text Editing**
- **Row Selection**
- **Column Selection**
- **Transform Menu**
- **Click Data Quality Meter**
### Transform Suggestion

**Interaction**

**Infer Operands**

**Generate Transforms**

**Rank Transforms**

**Present Top-N**

Map user input to transform operands. Example: text highlight maps to row, column, and text selections. Inferred text selections include string indices and regular expressions.

### Text Selection Inference

**Series Id**: LNU02000000

→ `^ STR WS STR SYM WS STR NUM $`

**Series Id**: LNU02000000

MATCH  Indices 11-22
MATCH  LNU02000000
MATCH  LNU NUM
MATCH  STR NUM
AFTER  : WS

### Text Selection Inference

**Series Id**: LNU02000000

MATCH  Indices 11-22
MATCH  LNU02000000
MATCH  LNU NUM
MATCH  STR NUM
AFTER  : WS

### Transform Suggestion

**Interaction**

**Infer Operands**

**Generate Transforms**

**Rank Transforms**

**Present Top-N**

Enumerate transforms that accept inferred operands as input. Set unmatched params to default values. Apply filter heuristics: No-ops, delete-all, and overly sparse outputs.
**Transform Suggestion**

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Sort transforms by:
Selection status
Specification difficulty
Frequency in corpus
Description length
↓
Rank Transforms
↓
Present Top-N

**Comparative Evaluation**

Compared Wrangler performance to Excel with 3 data cleaning tasks on small data sets.
Median completion time for Wrangler at least twice as fact in all tasks ($p < 0.001$).
Suggestions and visual previews used heavily.

**Conclusions**

Performance, Portability, Productivity
Expressiveness, Efficiency, Accessibility
DSLs should support domain reasoning
... by end user programmers
... by optimizing compilers
... by development tools
All three might influence DSL design.
Future work: models for DSL program inference.