Informations Visualisierung
SoSe 19

Benedikt Lüken-Winkels

July 9, 2019

Contents

1 1. Lecture

1.1 Orga

- Website: st.uni-trier.de/lectures/S19/IV/
- Tutorial: TBD (beginning: 22.-26.04.)
- Final exam: Do, 11.07. (elfths of July) 12-14 (H12)

1.2 Visualisation-Basics

- Combine different kinds of information in one graphic (geographical, temporal, historical, numeric, etc.)
- Sharing and visualising abstract data, without physical representation
- Visualisation is not:
  - scientific visualisation (non-abstract data)
  - computer graphics
  - graphic design
- **Example** Treemap
  - representation of a hierarchy of a filesystem
  - no border used for a square (compression)
  - light effect shows curvature, indicating where the squares/areas end
  - $\Rightarrow$ only 4 pixels needed instead of 9
  - Several drawbacks (alternative: tree view)

Abstract Data

- Text, table
- Hierarchy
- Composed data (Multivariate data): Example Napoleon (Slide 1)
- Time series: multivariate data with time as a dimension

Definition: Visualisation comprehend and extract data, visualisation produced automatically (not manually by humans)

Visualisation process

- graphical user interface
- interaction to create and manipulate the visualisation (**Visual steering**)

2 2. Lecture

2.1 Diagrams

Pie charts

- applicable to part-whole relation
- Several issues
  - hard to compare values
  - hard to compare different pie charts

Other Diagrams

- Timelines
- Sparklines: Reduction to show trend and the change of values over time

2.2 Metaphors and Symbols

Make constructs/concepts more accessible/imaginable

2.3 Symbols

highly simplified representation of objects and acitvities

Isotype Present quantity/value by number of pictograms

2.4 Infographics

- Eyecatcher to get people interested in the presented data
- Contain few text
- Self-explanatory
- Should tell a **story** $\Rightarrow$ express an opinion

3 3. Lecture

3.1 Visual Memory

- The brain fills empty gaps
- Distraction by environment (contrast/structure)
- $\Rightarrow$ visual perception is selective

3.2 Visula Information Processing

3 Phases of processing

1. Simple patterns and colors are recognized
2. Action system: reflexes
3. Visual working meomry/visual query

Human Eye

Usage of the properties of visual perception (Anticipation, pattern recognition)

- Eye Tracking (works by measuring the reflection form the eye's curvature)

3.3 Color Perception

3-Color-Theory

- Each color consists of rgb

Opponent-Color-Theory

- After image effect: color-receptors are getting exhausted, so white cannot be 'produced'
- three chemical processes with two opponent colors each
- Color is perceived by the difference between the opponent colors

⇒ Color and brightness are relative

Design Recommendations

- Emphasize with color
- Differences with brightness
- Coding of categories: max 6 to 12 different colors
- Color scales should vary in color and brighntess
- Color perception depends on culture
- Motion to grap attention/indicate a relation
- Strong colors/contrast can cause interta (ghost images)

3.4 Preattentive vision

- Detect patterns before an eye movement
- Motion is preattentive
- ⇒ Use preattentive patterns to encode information (spot an outlier)

3.5 Pattern Recognition

- Edge detection
- Simple patterns (detect small distortions)
- Complex patterns
- Object recognition (compare observation with learned patterns to recognise an object)

3.6 Motion recognition

Different elements perform similar motions

- Recognize patterns to identify object
- Recognize change after each frame
- Movements seem related, when they are in synch
- ⇒ Indicate a relation with a synchronous animation
- Motion can induce causality

4 Lecture

Visualization of Graphs: **Graph drawing**

Application

- Map-drawing: indicate multiple data sets in one map (London Underground)
- Ego(-centric) network: graph with personal connections

Visual Encoding

- Thickness, color of edges
- Color of nodes

Asthetic Criteria Readability does not induce asthetic

- min edge crossings
- min drawing
- min edge length
- min number of bends
- max symmetry
- uncover clusters
- max continuity amongst paths

4.1 Layouting algorithms

Radial Layout

- fair node weight, every node's representation is equal
- lots of edge crossings
- applicable, if there is no further info about the data

Force-Directed Layout

- force edges to a certain length
- reorder nodes
- try to find equilibrium, where the forces cancel out each other

Hierarchical Layout

- for cyclic structures: flip the edges that close the cycle while drawing the graph
- depth first search provides a topological ordering of the nodes
- sort nodes on the lower layer until the bottom is reached, then go back to start
- to have a clean layout, put in dummy nodes as a spacer

Orthogonal Layout

- edges follow grid (orthogonal paths)
- shape metrics
  - describe the path the edges take by turns
  - evaluate the paths

Edge Bundling

- structured radial layout
- bundle edges with the same direction

4.2 Matrix visualization of Graphs

Adjacency Matrix

- indicate an edge in a matrix
- uncovering clusters is hard

Layouting

Compound graphs

5 Lecture

5.1 Visualization of dynamic graphs

Dynamic graph: sequence of graph states

Animation see difference between layout and data changes to preserve the mental map of the graph. Examples:

- TimeLine, horizontal development of the graph, vertical orientation of the graph
- TimeSpiderTrees, cirular layout, each ring is one graph
- TimeRadarTrees, cicular layout, outer circles are a representation of the inner. The inner circle shows incoming edges, the outer shows outgoing

5.2 Multivariate data and time series

Boxplots box showing 50 percent of data, outer borders not standardized

Fan Chart wide part shows the mean (similar to the box plot)

Histogram bar represents a range of values (value ragne split into intervals)

6 Lecture

6.1 Software Visualization: Architecture

Pipes and Filters Input stream providing data, putting it into a pipe of filters

Layered Systems Layers provide functionality of upper layers (radial or stacked). Radial: small core, Pyramid: neutral representation

Blackboard-driven Different processes share info on one blackboard

6.1.1 Reverse Engineering

Create higher level of abstraction for a given system and automatically create architecture visualization. The detection of design patterns is non-trivial. To detect, the program is run and traced.

6.1.2 Enriched Node-Link Diagrams

Visuialize/Encode software metrics. Aggregation of information to simplify.

Class Blueprint Categorize methods by name and access attributes (public/protected/private...)

Depenecies Viewer Visualize package graph and dependencies between packages and methods

Dependency Structure Matrix DSM Detect cycles and indirect cycles with highlighting

Software Citites and Maps 2D plane represents system. Hierarchy shown with trees/dimesions. 3rd dimension can be used to show other metrics, like evolution/age/dependencies

Summary Ad-hoc diagrams hard to understand without explanation. With reverse engineering automatic creation for specific techniques are possible

7 Lecture

7.1 Dynamic Program Visualization

Dynamic Data Acquisition invasive mehtod, monitoring the behaviour of a program before/after each instruction. Might alter the program execution. NACHLESEN

8 Lecture

8.1 Visula Debugging

Slices are parts/slices of the huge dependency graph in a program

Static Slice How is a variable changed by other code points. Slice is a samll part

Dynamic Slice

Execution Slices Sequence of program points.

Dice Difference of two Slices.

X-Slice (Heuristic) Compare a run with failing and compiling input. Only the failing program points are highlighted. Color coding coverage data by failure propability and evidence for failure.

Test Blueprint Highlight non-executed program points in the Class Blueprint.

8.2 Software Evolution

aka Software Development Process ⇒ Software changes in its lifetime.

Software Archive version control/collection of the history of a program of any kind.

Color-coding

- Line Representation: indentation/different metrices
- Code Age: when was a file/line changed
- Pixel Representation
- Version-specific Code: highlight eg platform specific code
- Depth of nested blocks
- CVSScan: different versions for a file with LOC as bar height.

Evolution Matrix Classes are represented as boxes. Box height and width encode a certain metric. ⇒ No insight on program structure

Call Graph Which function calls wich function (low level info). Encode program structure. Edge splatting (the more often an edge is drawn the more intense it color gets) shows call clusters.

8.2.1 Visual Data Mining in Software Architecture

Data Mining Process Startgin with a version control program (git)

1. Analysis
2. Extraction
3. Data Mining
4. Visual Data Mining

Coupling

- Evolutionary Coupling artifact are related, when they are changed together.
- Logical Coupling artifacts are related, when they are programmatically calling each other.