# Software visualization

Tudor Gîrba
www.tudorgirba.com

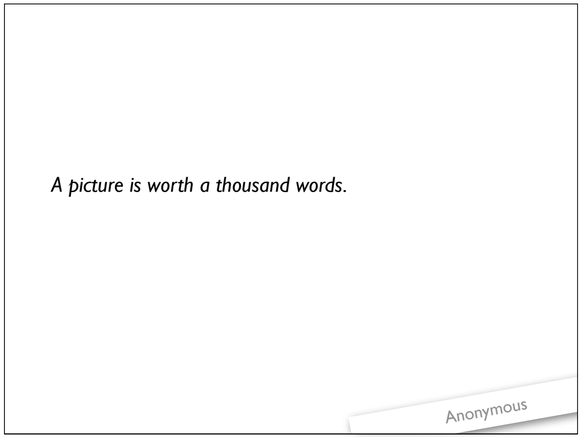
*Software Visualization is the use of typography, graphic design, animation, and cinematography with human-computer interaction and computer graphics to facilitate both the human understanding and effective use of software.*

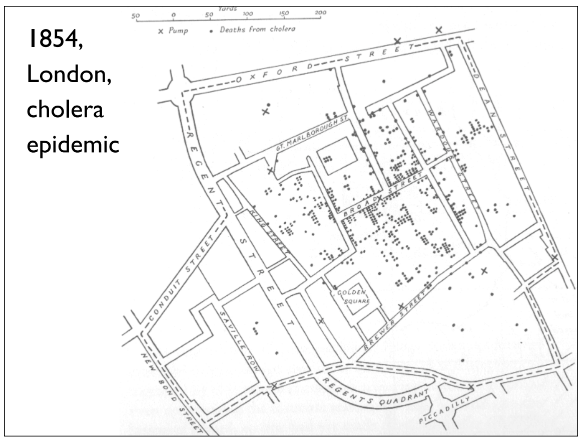
Price etal. 1993

Blaine A. Price, Ronald M. Baecker and Ian S. Small, "A Principled Taxonomy of Software Visualization," Journal of Visual Languages and Computing, vol. 4, no. 3, 1993, pp. 211-266.

Complete definition: Software Visualization is the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of software.

## Why visualization?

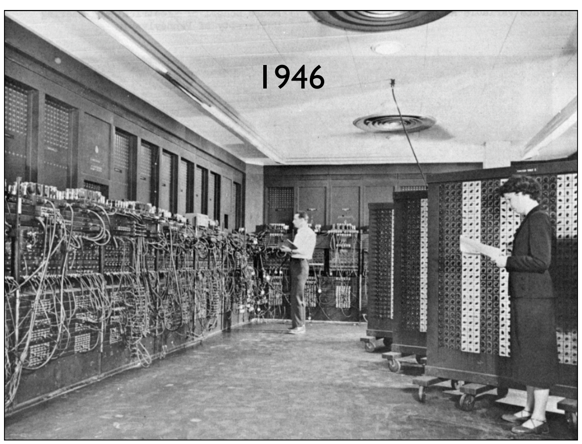



Edward R. Tufte, Visual Explanations, Graphics Press, 1997.

It was not known how cholera was transmitted.
Dr. John Snow had the hypothesis that it gets transmitted via water.
To check this, he plotted on the map of the city:
- the deaths of a new epidemic (dots)
- the water pumps (Xs).

The result was that high number of deaths were detected near infected water pump on Broad Street.

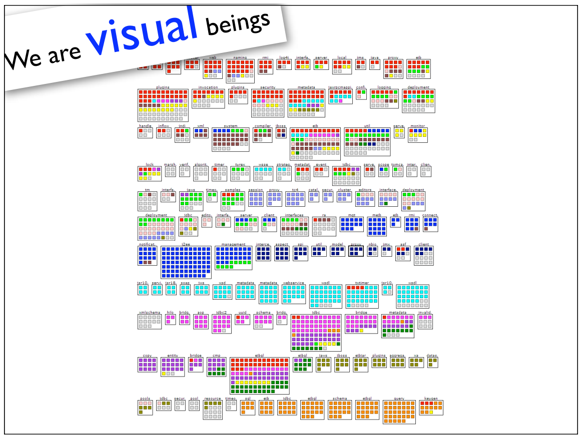



This is a picture of ENIAC I (1946).
(http://en.wikipedia.org/wiki/ENIAC)

In 1946 we used to see the programs. In the picture we can see the complexity of the program in how intricate the cables are. And we see who is working on what. This is no longer the case with modern software systems.

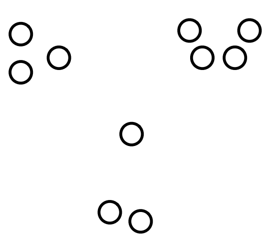
*A picture is worth a thousand words.*

Anonymous

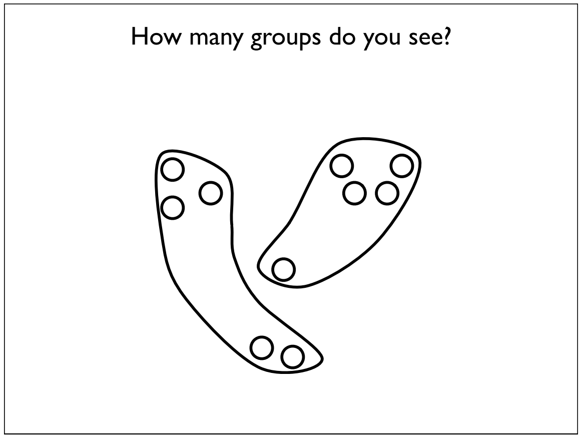



The picture is taken from: Stéphane Ducasse, Tudor Gîrba and Adrian Kuhn, “Distribution Map,” Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM '06), IEEE Computer Society, Los Alamitos CA, 2006, pp. 203-212.

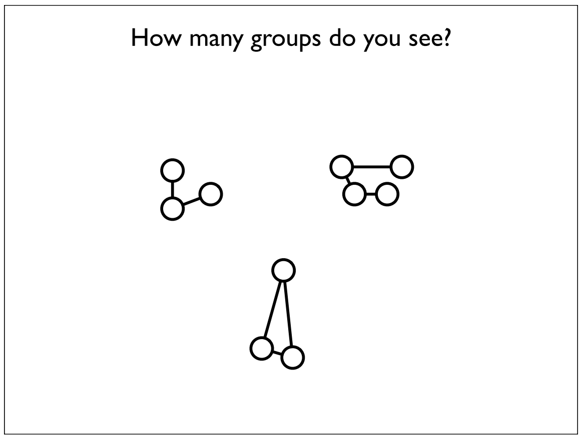
How many groups do you see?



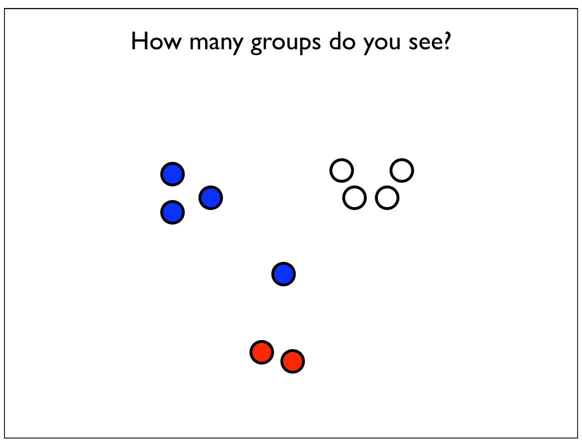
Some see 3 groups and some see 4 groups. Those that see 3, see the circle in the center as belonging to the group formed by the two circles at the bottom.



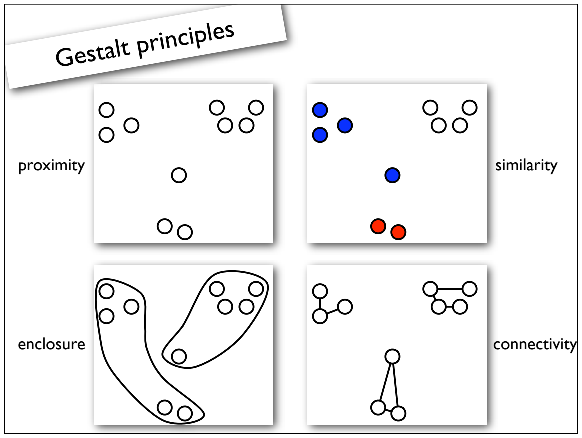

Enclosing clarifies the situation.




Again, no problem in identifying 3 groups when circles are connected with edges.



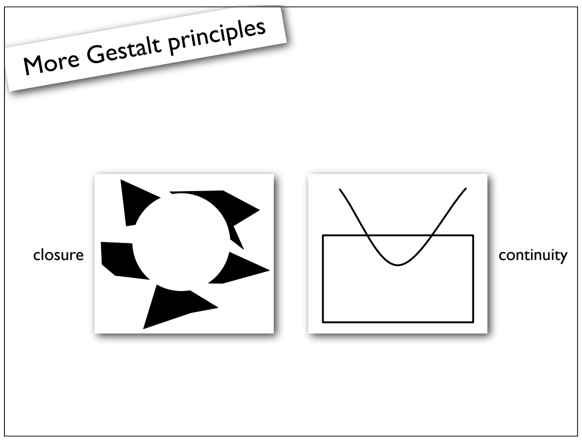

The same happens when the circles share the same visual shape.




Stephen Few, Show me the numbers: Designing Tables and Graphs to Enlighten, Analytics Press, 2004.

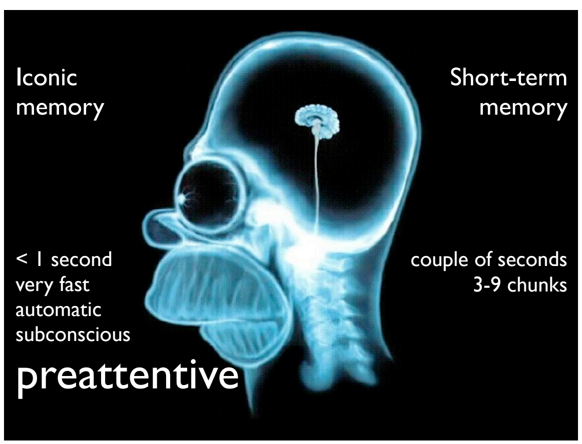
http://en.wikipedia.org/wiki/Gestalt_psychology

These are some examples of so called Gestalt principles. According to these, we perceive the world as a whole rather than as a sum of parts.




Stephen Few, Show me the numbers: Designing Tables and Graphs to Enlighten, Analytics Press, 2004.

Two more examples of Gestalt principles.




Our brain is a computer with 3 types of memory:

Iconic
Short-term
Long-term

| | | | |
|---|---|---|---|
| Orientation | Line Length | Line Width | Size |
| Shape | Curvature | Added Marks | Enclosure |

If eyes are computers, visualizations are programs.

These attributes of form are the primitive instructions we can use for building these programs.

How many times does 5 appear?

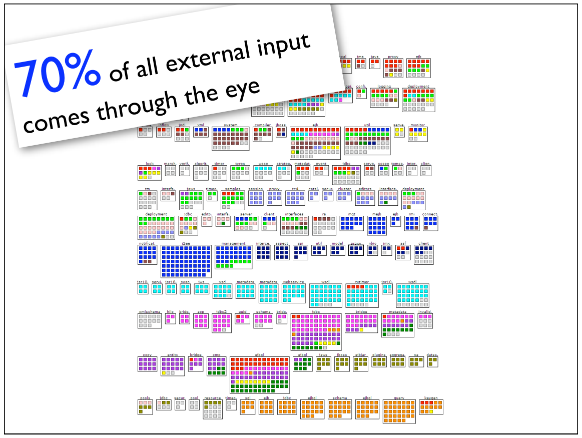
8789364082376403128764532984732984732094873290845
389274-0329874-32874-23198475098340983409832409832
049823-0984903281453209481-0839393947896587436598

Exemplifying Preattentive Processing
Colin Ware, Information Visualisation, Elsevier, Sansome Street, San Fransico, 2004.
p150

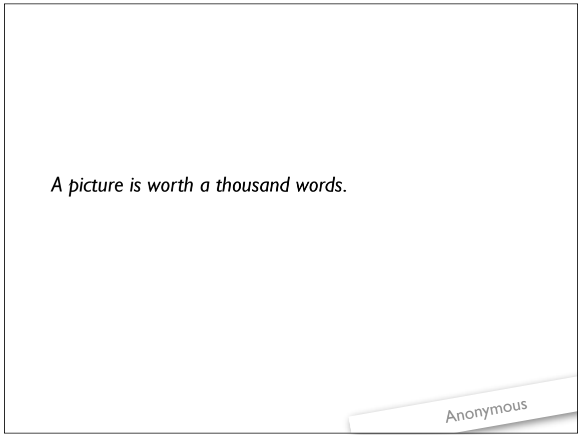
How many times does 5 appear?

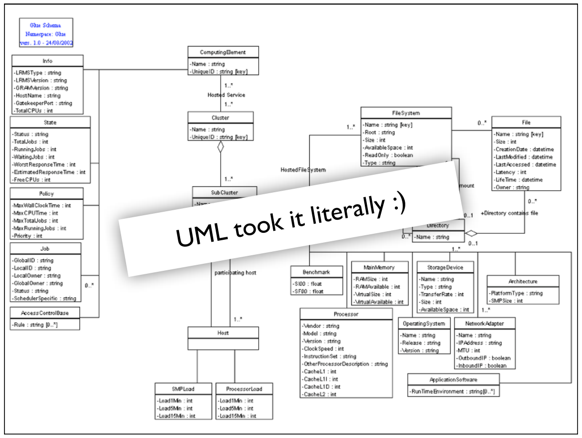
8789364082376403128764**5**3298473298473209487329084**5**
389274-0329874-32874-231984**75**098340983409832409832
049823-09849032814**5**3209481-08393939478965**8**7436598

Colin Ware, Information Visualisation, Elsevier, Sansome Street, San Fransico, 2004.
p150

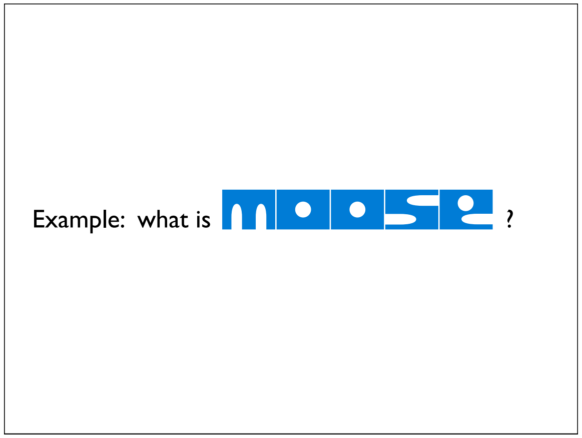



To see is often used a synonym for to understand. Do you see my point?



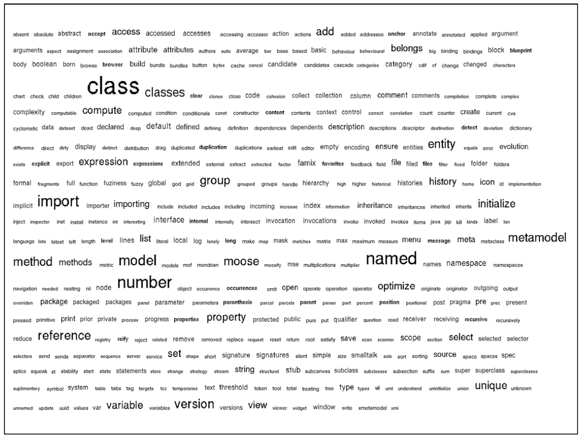




This picture shows approximately 350 words for a tiny system.

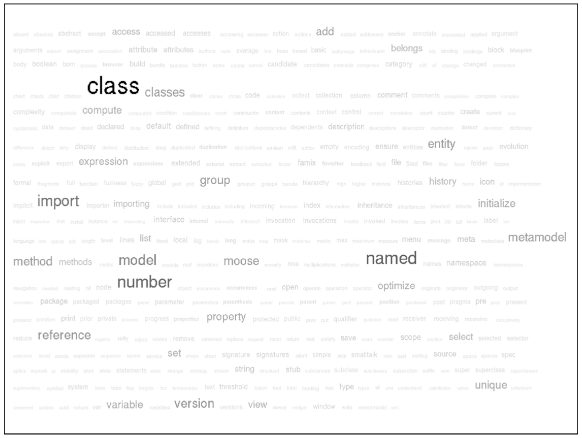



Let's see what else can we do with 1000 words.

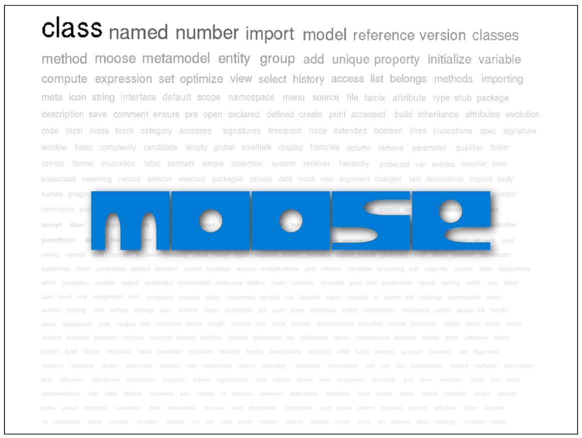
If we display all of them equally, we cannot identify much.




Increasing the font size leads to a tag cloud visualization. The small text is hardly readable, but it still competes for attention.




If we decrease the visual importance of the small text, we still know it's there but the important words stand out more. Still, alphabetical order is not necessarily the most relevant.




Ordering the words eases our task.

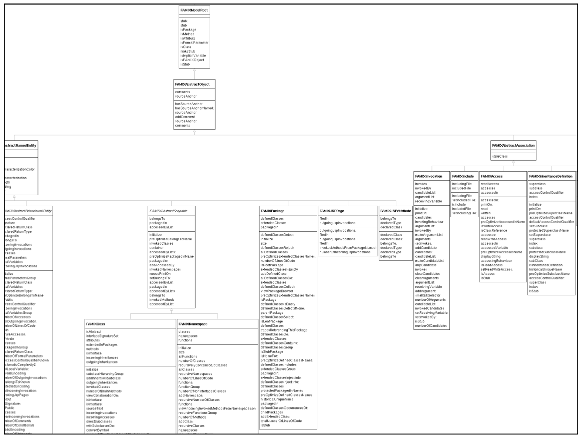
An example of what can be done with 1000 words.

What to visualize?
How to visualize?

What to visualize?

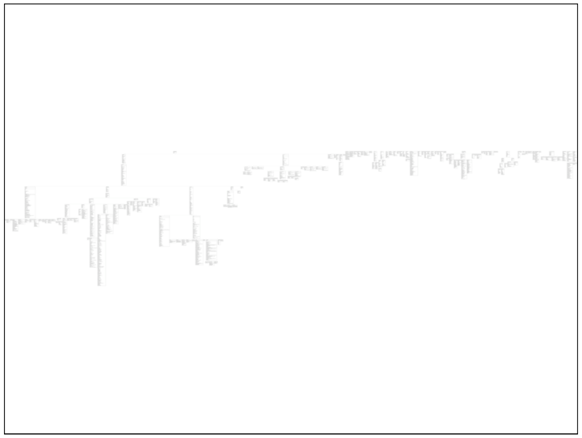
What to visualize?

Software structure

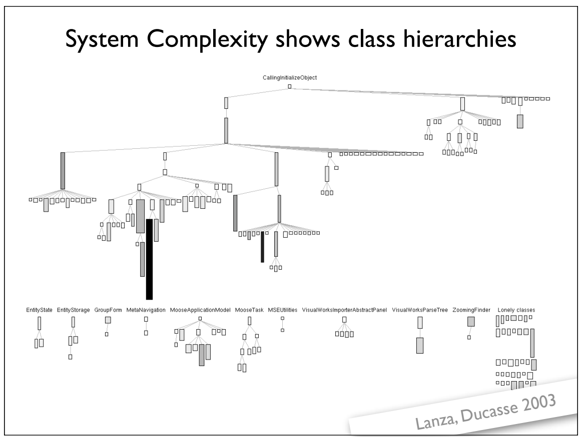


UML is a nice visual language for expressing ideas, but it is hardly useful as a visualization.

In this example, we show a small fragment of the model hierarchy in Moose. Still even at this level of zoom, we cannot see the details.

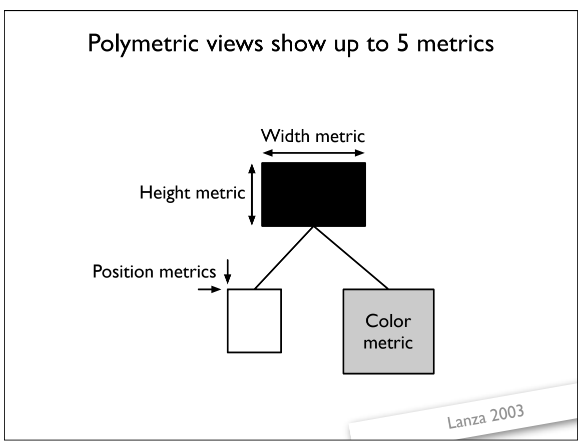


If we zoom out, all we see is the shape of hierarchies, and the shape of classes.



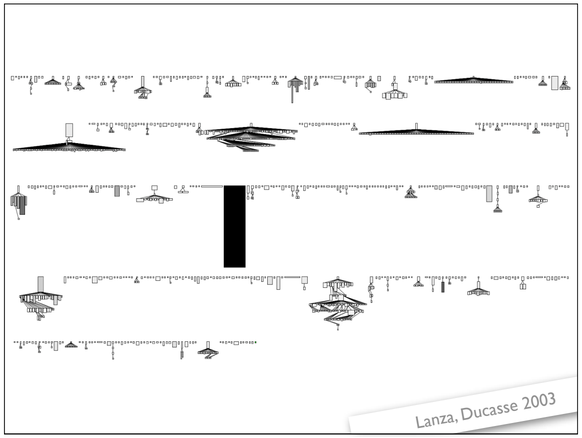

Michele Lanza and Stéphane Ducasse, "Polymetric Views—A Lightweight Visual Approach to Reverse Engineering," IEEE Transactions on Software Engineering, vol. 29, no. 9, September 2003, pp. 782-795. http://www.iam.unibe.ch/~scg/cgi-bin/scgbib.cgi/abstract=yes?Lanz03d

System complexity is a polymetric view that does a better job at showing the shape of hierarchies and of individual classes.



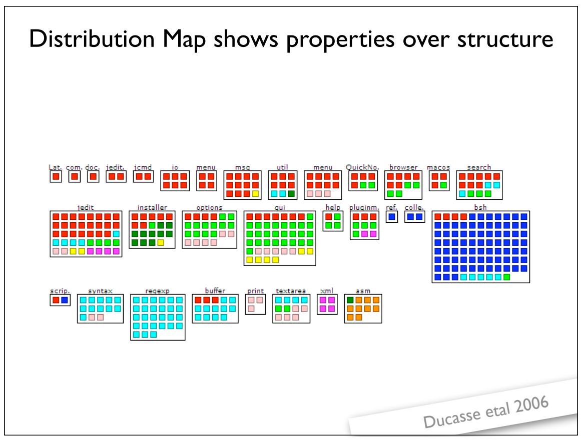

Michele Lanza, "Object-Oriented Reverse Engineering — Coarse-grained, Fine-grained, and Evolutionary Software Visualization," Ph.D. thesis, University of Berne, May 2003. http://www.iam.unibe.ch/~scg/cgi-bin/scgbib.cgi/abstract=yes?Lanz03b

Polymetric views are graphs enriched with metric information.



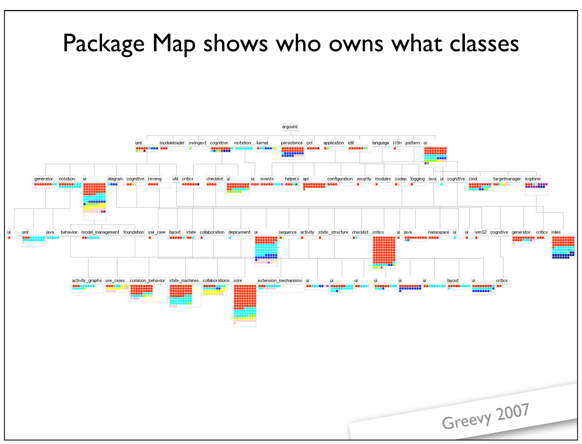

System complexity of ArgoUML.

Node = class
Edge = inheritance
Node width = number of attributes (NOA)
Node height = number of methods (NOM)
Node color = number of lines of code (LOC)

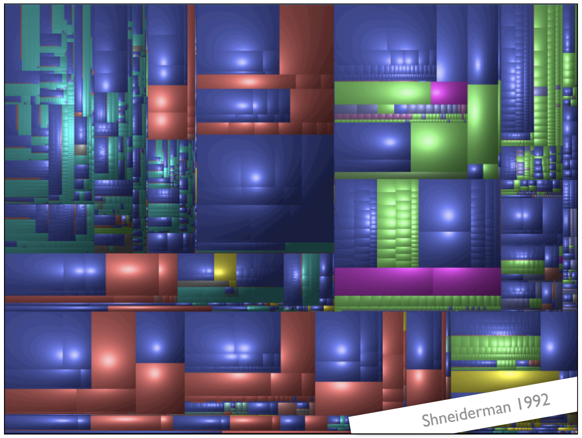



Stéphane Ducasse, Tudor Gîrba and Adrian Kuhn, “Distribution Map,” Proceedings International Conference on Software Maintainance (ICSM 2006), IEEE Computer Society, Los Alamitos CA, 2006, pp. 203-212.
This picture shows classes (small squares) grouped in packages (large rectangles). The color of the classes is given by the predominant concept. The labels show the package name, but they are cropped when longer that the package width. The classes in the package are arranged to get a ration between width and height as close as possible to the golden ratio (1.61803399).

31 packages, 394 classes and 9 concepts of JEdit.




Orla Greevy, Tudor Gîrba and Stéphane Ducasse, “How Developers Develop Features,” Proceedings of 11th European Conference on Software Maintenance and Reengineering (CSMR 2007), IEEE Computer Society, Los Alamitos CA, 2007, pp. 256—274.



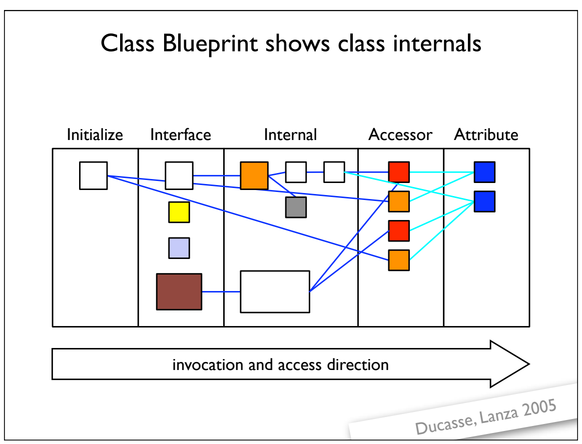

http://www.cs.umd.edu/hcil/treemap-history/index.shtml

Treemaps show the hierarchical structure by filling completely the given space.

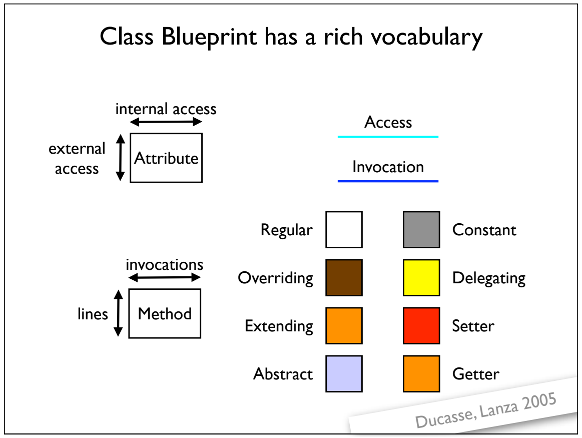
The picture shows the files colored by type of ArgoUML 0.26 and it was generated with Disk Inventory (http://www.derlien.com/).
Blue are Documents, Red are Jars, Cyan are Pictures, Cyan are Java

What to visualize?

Software structure
Software relationships

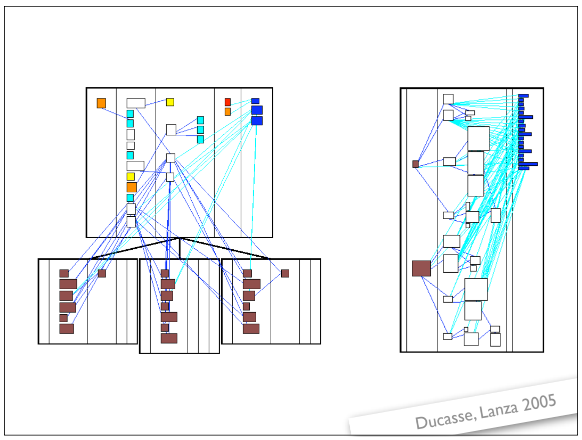



Stéphane Ducasse and Michele Lanza, “The Class Blueprint: Visually Supporting the Understanding of Classes,” IEEE Transactions on Software Engineering, vol. 31, no. 1, January 2005, pp. 75-90. http://www.iam.unibe.ch/~scg/cgi-bin/scgbib.cgi/abstract=yes?Duca05b
The class is split into 5 layers:
- The initialize layer contains constructor methods,
- The Interface layer contains methods called from outside the class,
- The Internal implementation layer contains methods called from within the class,
- The Accessor layer contains setters and getters,
- The Attribute layer contains the attributes :).
Blue edges represent method invocations. Cyan edges represent attribute access.



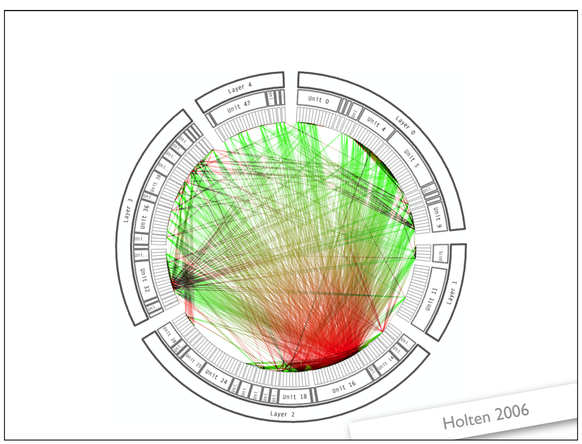

The blueprint has a rich vocabulary.

On both dimensions of methods and attributes are mapped metrics. On the color of the methods are mapped different attributes of the method.



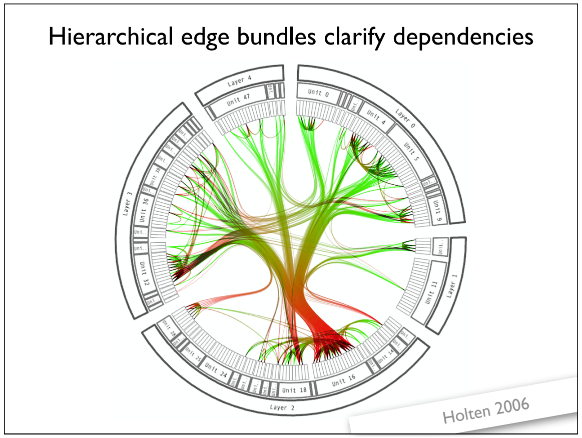

The picture on the left shows the blueprint of one superclass and 3 of its subclasses. The subclasses have the same shape and color (given by overriding methods). Hence the name Siamese twins.

The picture on the right shows a class that has two distinct interests, because the methods on top do not have a direct relationship with the methods on the bottom (we see that because there is no blue edge in between).



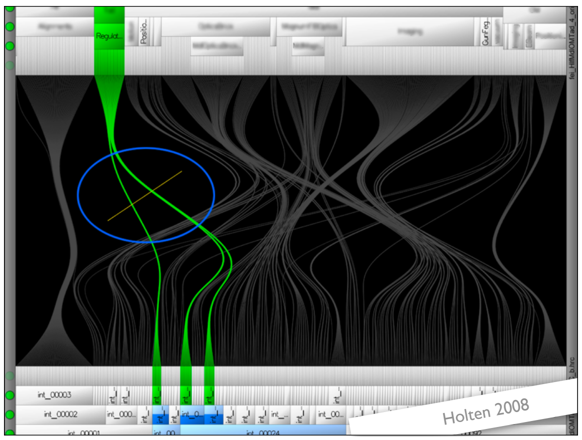

Danny Holten, “Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data”, IEEE Transactions on Visualization and Computer Graphics (TVCG; Proceedings of Vis/InfoVis 2006), Vol. 12, No. 5, Pages 741 - 748, 2006.

One straightforward way of representing relationships is to display entities in a circle and draw edges between them. The picture shows classes organized in a module structure and the arrows are dependencies (red=called, green=caller). We know that Units 16 and 18 are called many times, but we do not know exactly where from. The picture is too noisy.



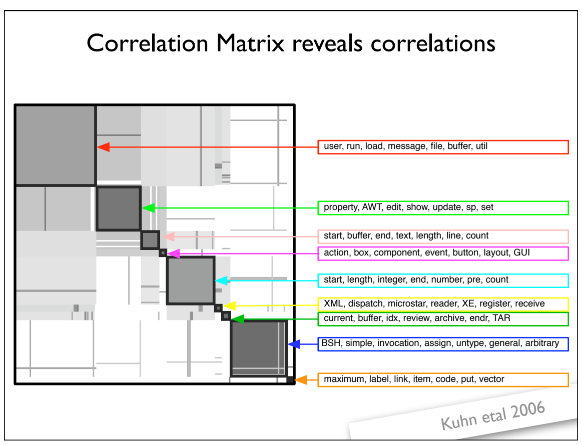

Danny Holten, “Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data”, IEEE Transactions on Visualization and Computer Graphics (TVCG; Proceedings of Vis/InfoVis 2006), Vol. 12, No. 5, Pages 741 - 748, 2006.

Hierarchical edge bundles make use of the hierarchical structure of entities to make relationships between larger parts clearer.




Visual Comparison of Hierarchically Organized Data" (PDF available through EUROGRAPHICS / Blackwell Publishing), Danny Holten and Jarke J. van Wijk, 10th Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum; Proceedings of EuroVis'08), 2008.

This visualization uses hierarchical edge bundles to show the relationships between two hierarchies of data.

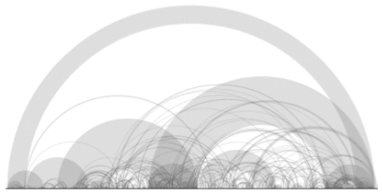



A correlation matrix displays the same entities both on the rows and on the columns. Each cell in the matrix is colored based on the strength of the correlation. This notation is useful for identifying similarities (e.g., code duplication).

In this picture, the matrix displays the similarity of vocabulary used in the classes of JEdit. Furthermore, the classes are grouped to reveal clusters of classes that use similar vocabulary.

Adrian Kuhn, Stéphane Ducasse and Tudor Gîrba, “Semantic Clustering: Identifying Topics in Source Code,” Information and Software Technology, vol. 49, no. 3, March 2007, pp. 230—243.

## Arc diagrams show duplications



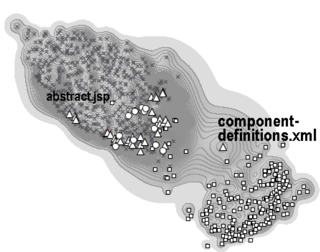
Wattenberg 2002

Martin Wattenberg, Arc diagrams: visualizing structure in strings, In Proceedings of IEEE Symposium on Information Visualization, 2002 (INFOVIS 2002), 110-116

## What to visualize?

Software structure
Software relationships
Metaphors

## Software Map reveals software geography

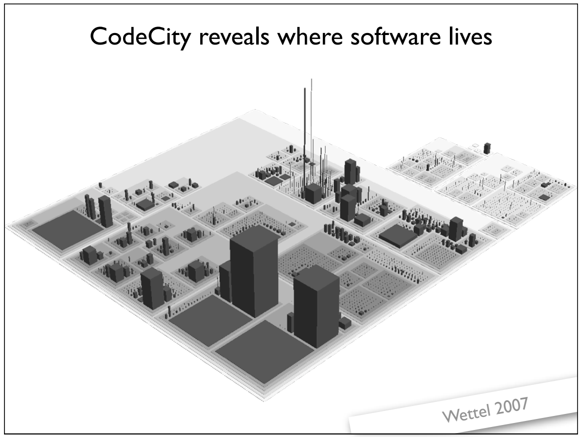



Kuhn etal 2008

It took thousands of years to build the first abstract representation of the real world using x and y axis. Software on the other hand, has no physical shape and one challenge is to lay it out so that the distance between entities has a meaning.

This visualization proposes a cartography metaphor to represent software. The entities are distributed based on the vocabulary used.

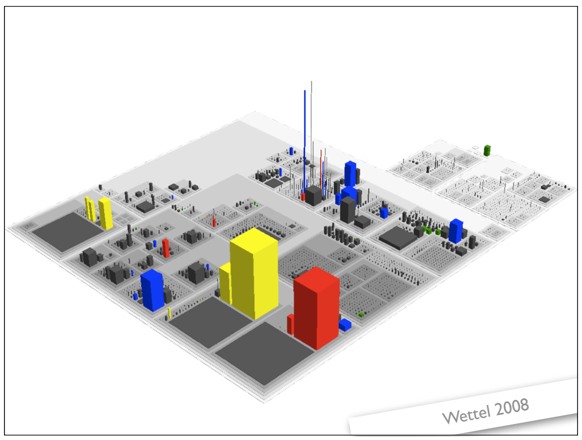
Adrian Kuhn, Peter Loretan and Oscar Nierstrasz, “Consistent Layout for Thematic Software Maps,” Proceedings of 15th Working Conference on Reverse Engineering (WCRE'08), IEEE Computer Society Press, Los Alamitos CA, October 2008, pp. 209—218.




Richard Wettel and Michele Lanza, “Visualizing Software Systems as Cities,” Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis), 2007, pp. 92—99.

CodeCity represents the system structure as a city. The packages generate quarters, while the classes are buildings.

http://www.inf.unisi.ch/phd/wettel/codecity.html




Richard Wettel, Michele Lanza “Visually Localizing Design Problems with Disharmony Maps” In Proceedings of Softvis 2008 (4th International ACM Symposium on Software Visualization), pp. 155 - 164, ACM Press, 2008.

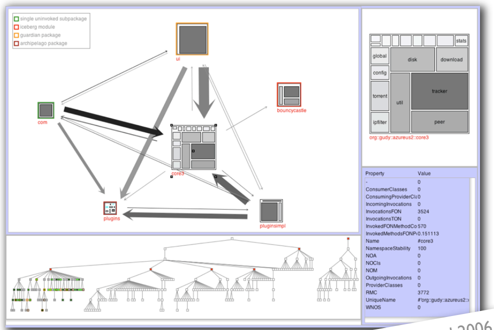
In this work, design flaw suspects are highlighted with different colors.

http://www.inf.unisi.ch/phd/wettel/codecity.html

What to visualize?

Software structure
Software relationships
Metaphors
Interaction

Softwarenaut discovers architecture



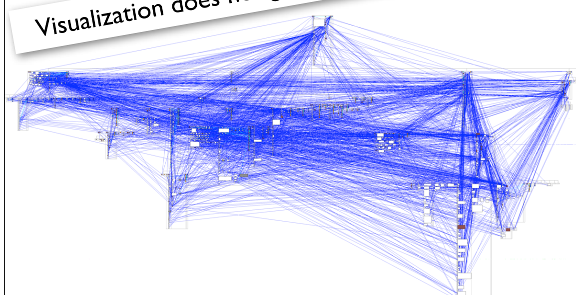
Lungu etal 2006

Mircea Lungu, Michele Lanza and Tudor Gîrba, “Package Patterns for Visual Architecture Recovery,” Proceedings of CSMR 2006 (10th European Conference on Software Maintenance and Reengineering), IEEE Computer Society Press, Los Alamitos CA, 2006, pp. 185—196.

What to visualize?

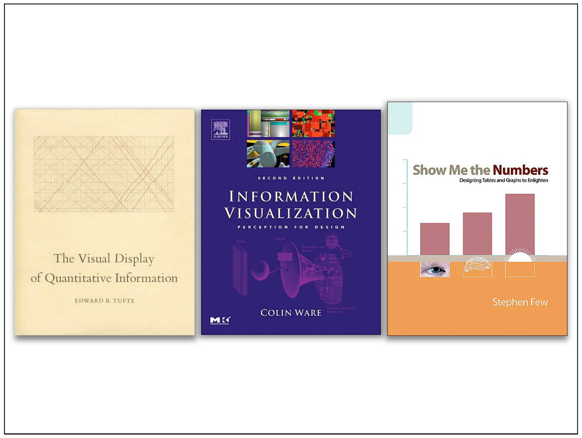
How to visualize?

Visualization does not guarantee understanding



Visualization does not guarantee understanding

Not any picture tells a thousand words.

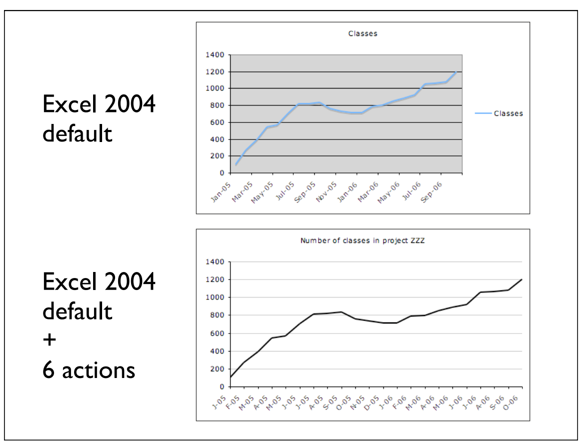



Edward R. Tufte, The Visual Display of Quantitative Information (2nd edition), Graphics Press, 2001.

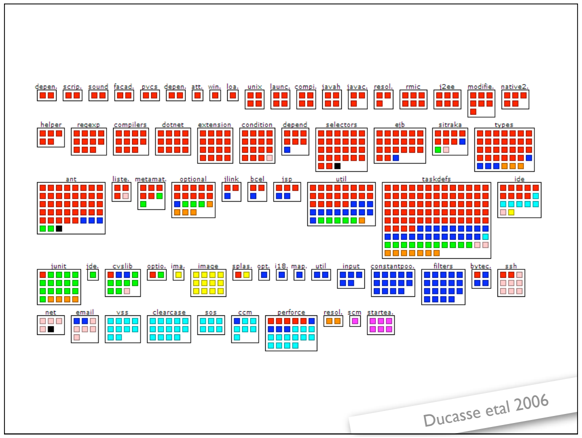
Colin Ware, Information Visualisation, Elsevier, Sansome Street, San Fransico, 2004.

Stephen Few, Show me the numbers: Designing Tables and Graphs to Enlighten, Analytics Press, 2004.


Edward R. Tufte, The Visual Display of Quantitative Information (2nd edition), Graphics Press, 2001.



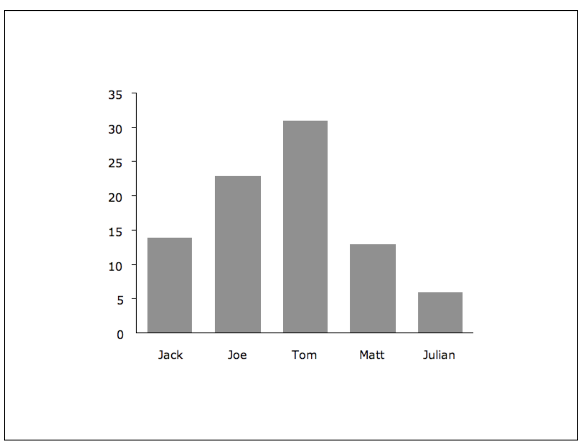

The 6 actions are:
1. Remove background.
2. Remove legend.
3. Add better graph description.
4. Make the series line black for better contrast.
5. Make the grid lines light gray to be less intrusive.
6. Make the dates to start from the origin to avoid confusions.

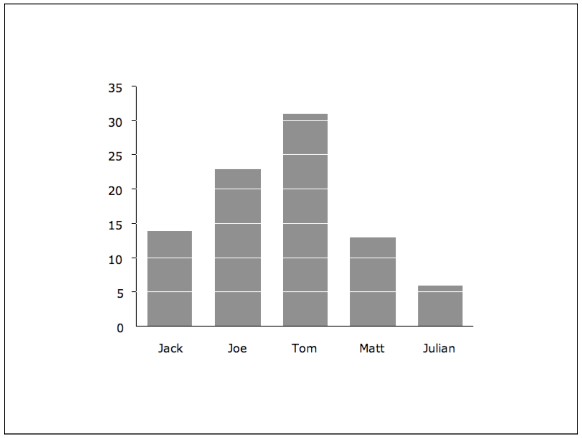


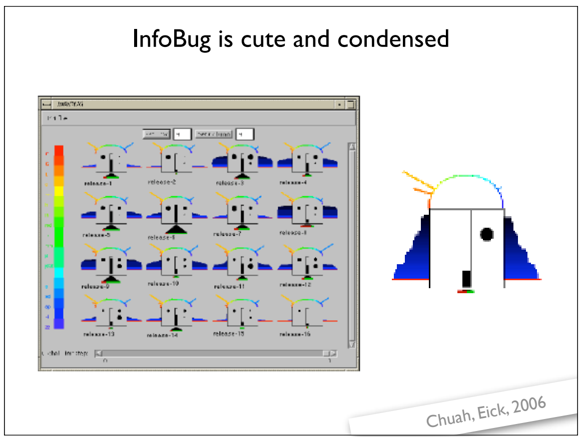

Stéphane Ducasse, Tudor Gîrba and Adrian Kuhn, “Distribution Map,” Proceedings International Conference on Software Maintainance (ICSM 2006), IEEE Computer Society, Los Alamitos CA, 2006, pp. 203-212.

The only element debatable to be chart junk is the black border which could be perhaps made gray.


Edward R. Tufte, The Visual Display of Quantitative Information (2nd edition), Graphics Press, 2001.

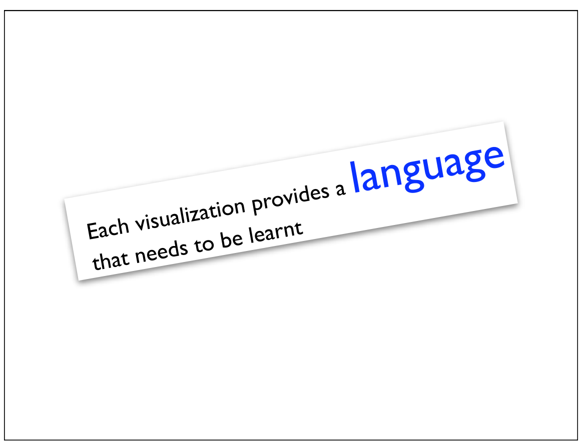




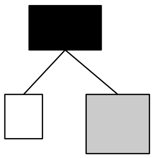

Mei C. Chuah and Stephen G. Eick, “Information Rich Glyphs for Software Management Data,” IEEE Computer Graphics and Applications, vol. 18, no. 4, July 1998, pp. 24—29.

Interesting about this visualization is that each part of the bug bears information, and the result is a pleasant glyph:
- the wings show two time series of lines of code (left) and errors detected (right)
- the antenas show different types of code. For example the orange line shows the amount of C code.
- The eye shows the amount of inheritance relationships.
- With red and green are shown lines added to correct errors (red) or for new functionality (green).



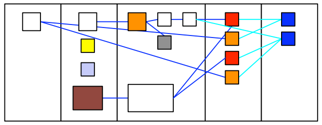
## System Complexity

1 node type
1 edge type
3 metrics



## Class Blueprint

3 node types
2 edge types
3 metrics
8 properties



A small experiment

1, 13, 27, 4, 96

What were the numbers?

Easy!

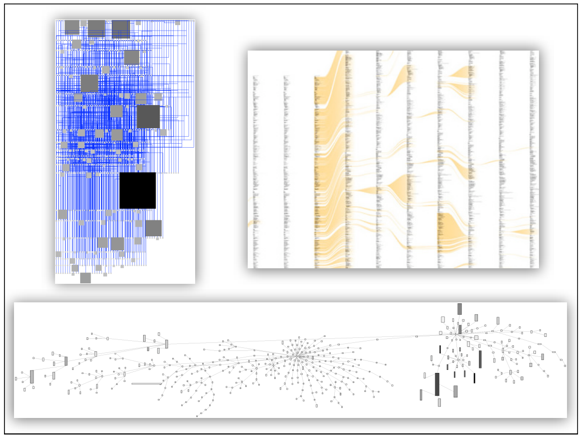
What’s the last advertisement you saw?

Not so easy!

Each visualization provides a language that needs to be learnt

Visualization is art, too

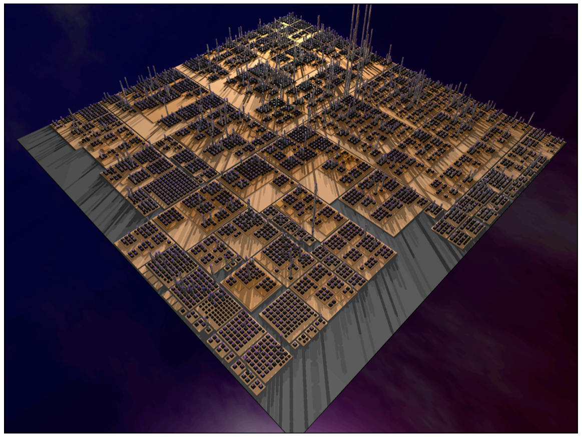
Just at the beginning of 20th century artists sought means of expression that would match the industrial age, now, as we step into the information age we seek new artistic means of expression.



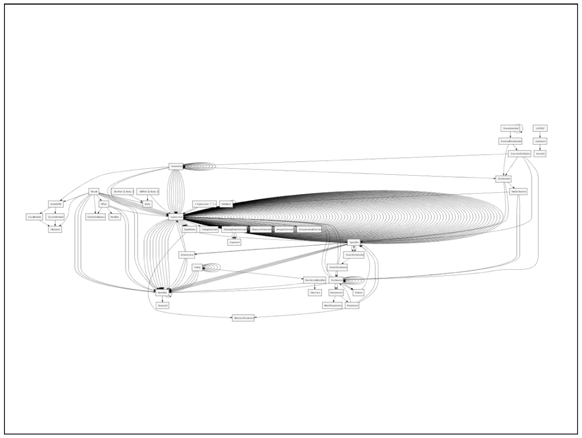
Visualization is art, too.

The picture on the top right: http://acg.media.mit.edu/people/fry/revisionist/
The other two pictures were created with Mondrian.

Two nice collections of visualizations are:
http://infosthetics.com/
http://www.visualcomplexity.com/vc/

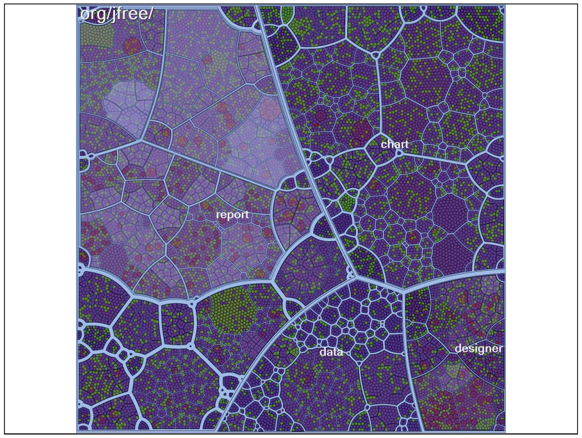


This picture was created by Michele Lanza



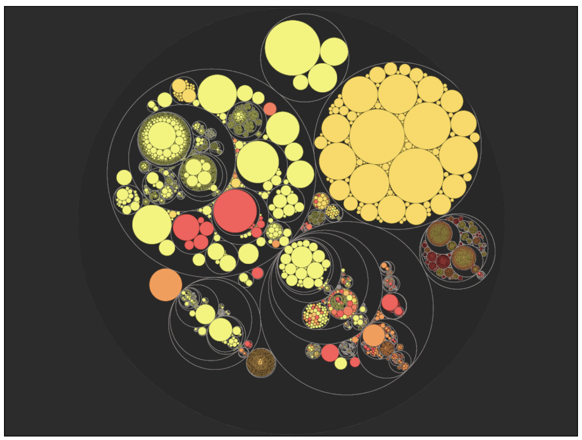
The zeppelin!

The picture shows C grammar dependencies and it was created by Magiel Bruntink, Jurgen Vinju and CWI




Michael Balzer, Oliver Deussen and Claus Lewerentz, “Voronoi treemaps for the visualization of software metrics,” SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization, ACM, New York, NY, USA, 2005, pp. 165—172.

The picture shows a novel way of drawing treemaps.



http://lip.sourceforge.net/ctreemap.html

Tudor Gîrba
www.tudorgirba.com