Distribution Map

Stephane Ducasse, Tudor Gırba Adrian Kuhn (ICSM 2006)

Introduction

- Understanding how a given phenomenon is distributed across a large software system is a key information for the overall comprehension of the system.

- For example, suppose we know which developer owns which file in the system, we would like to answer several questions:
  - What are the zones of interest of the developers?
  - Is the work of a developer focused on only one part of the system, or is it scattered throughout the system?
  - Is a package implemented by one or several developers?

Distribution Map

- Given the software system S as a set of software artifacts and two partitions P and Q of that set, we introduce the Distribution Map as a means to visualize Q compared to P.

- The visualization is composed of large rectangles containing small squares in different colors.
- There is a small square for each element of S,
- The partition P is used to group the squares into large rectangles
- The partition Q is used to color the squares.

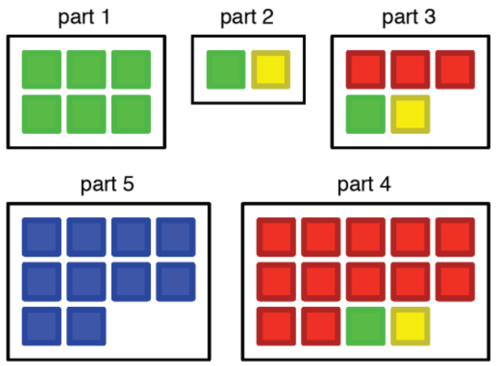


Figure 1. A *Distribution Map* **showing five packages and four properties: Red, Blue Green and Yellow.**

Distribution Map

- Reference partition:
  - The partition P corresponds to a well understood partition.
  - Typically the reference partition represents the intrinsic structure of the software system (e.g., the package structure).

- Comparison partition:
  - the partition Q is the result of an analysis.
  - Mostly, this partition is either a set of clusters, or some mutually exclusive properties associated with the elements of S.

- In our example from Figure 1, about the properties we say that Blue is well-encapsulated, that Yellow is cross-cutting and that Green is like an octopus because it has a body and tentacles spread over the system.

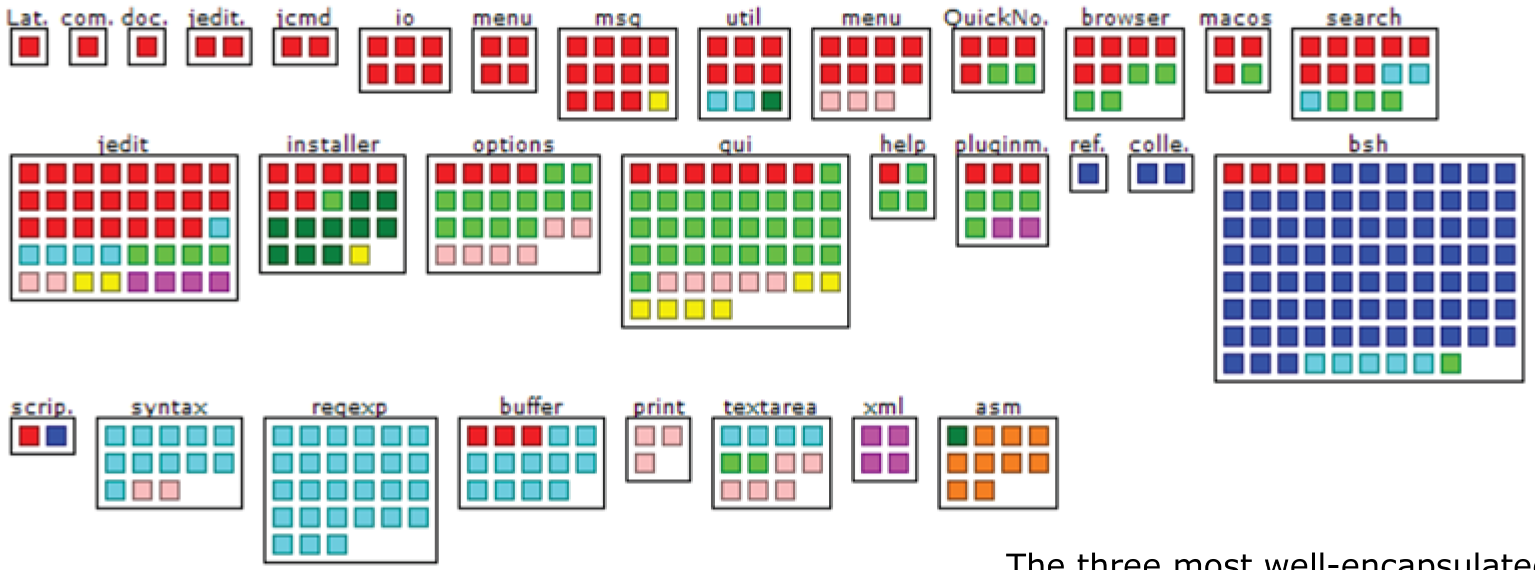
Application: Semantic Clustering JEdit

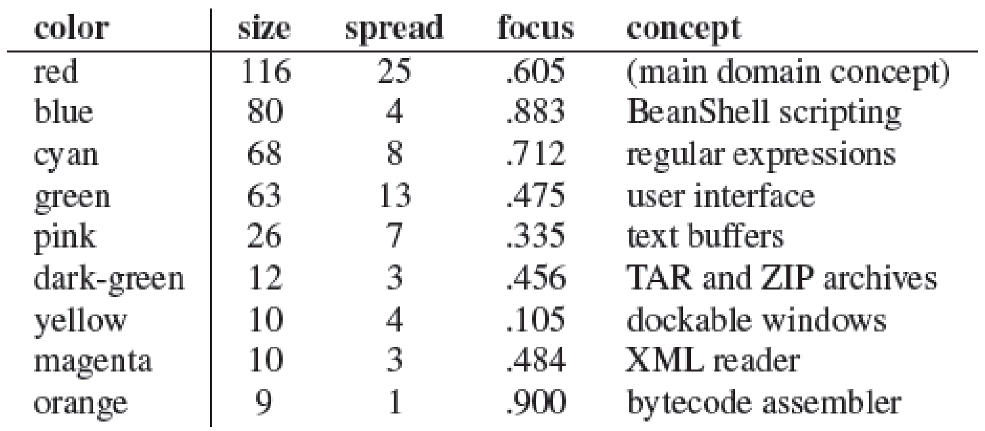
- JEdit is a text editor written in Java

- The source has a total of 394 classes in 31 packages
- The source uses a vocabulary of 1603 distinct terms, applying semantic clustering results in nine domain concepts

- Figure 2 illustrates how the retrieved domain concepts are distributed over the package structure: the parts are the packages, the elements are the classes and the colors refer to their concepts.

Semantic Clustering JEdit





color	size	spread	focus	concept
red	116	25	.605	(main domain concept)
blue	80	4	.883	BeanShell scripting
cyan	68	8	.712	regular expressions
green	63	13	.475	user interface
pink	26	7	.335	text buffers
dark-green	12	3	.456	TAR and ZIP archives
yellow	10	4	.105	dockable windows
magenta	10	3	.484	XML reader
orange	9	1	.900	bytecode assembler

The three most well-encapsulated concepts (Orange, Blue and Cyan), implement clearly separated concepts such as scripting and regular expres-sions.

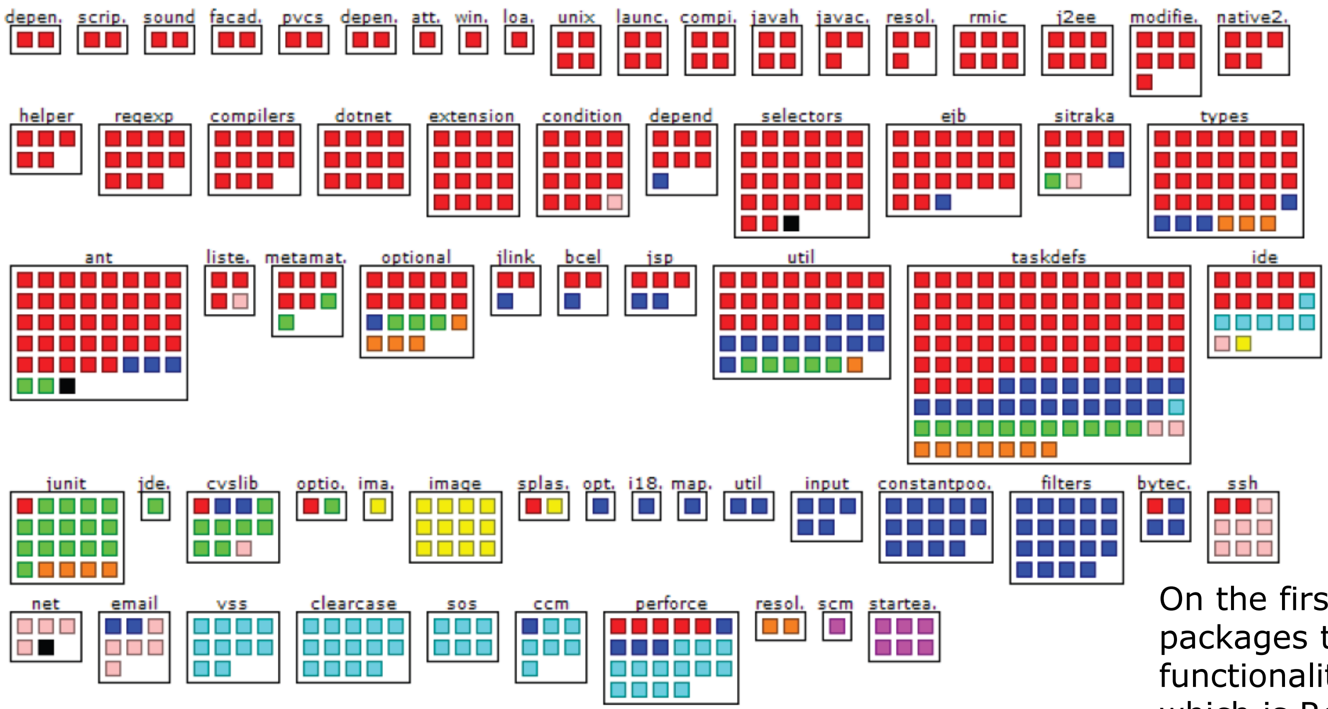
The concepts with the lowest focus crosscut the system: Yellow implements dockable windows, a custom GUI-feature, and Pink is about handling text buffers.
These two concepts are good candidates for a closer inspection, since we might want to refactor them into packages of their own.

Semantic Clustering of Ant

- Ant is a popular development tool for Java.

- Its source has a total of 665 classes in 66 packages
- The source uses a vocabulary of 1787 distinct terms.
- Applying semantic clustering resulted in nine domain concepts

Semantic Clustering of Ant



color	size	spread	focus	concept
red	390	47	.808	(main domain concept)
blue	103	22	.555	string processing
cyan	56	7	.803	version control clients
green	48	10	.421	unit-test support
pink	23	9	.557	network protocols
orange	21	6	.210	XML handling
yellow	15	4	.904	image and graphics
magenta	7	2	1.0	another versioning client
black	3	3	.087	FTP and filesystem

On the first three rows there are packages that implement the core functionality of Ant, the largest concept which is Red, dominates this part of the figure.

On row four we find the octopus concepts Blue and Green that are used by the core packages in the third row.

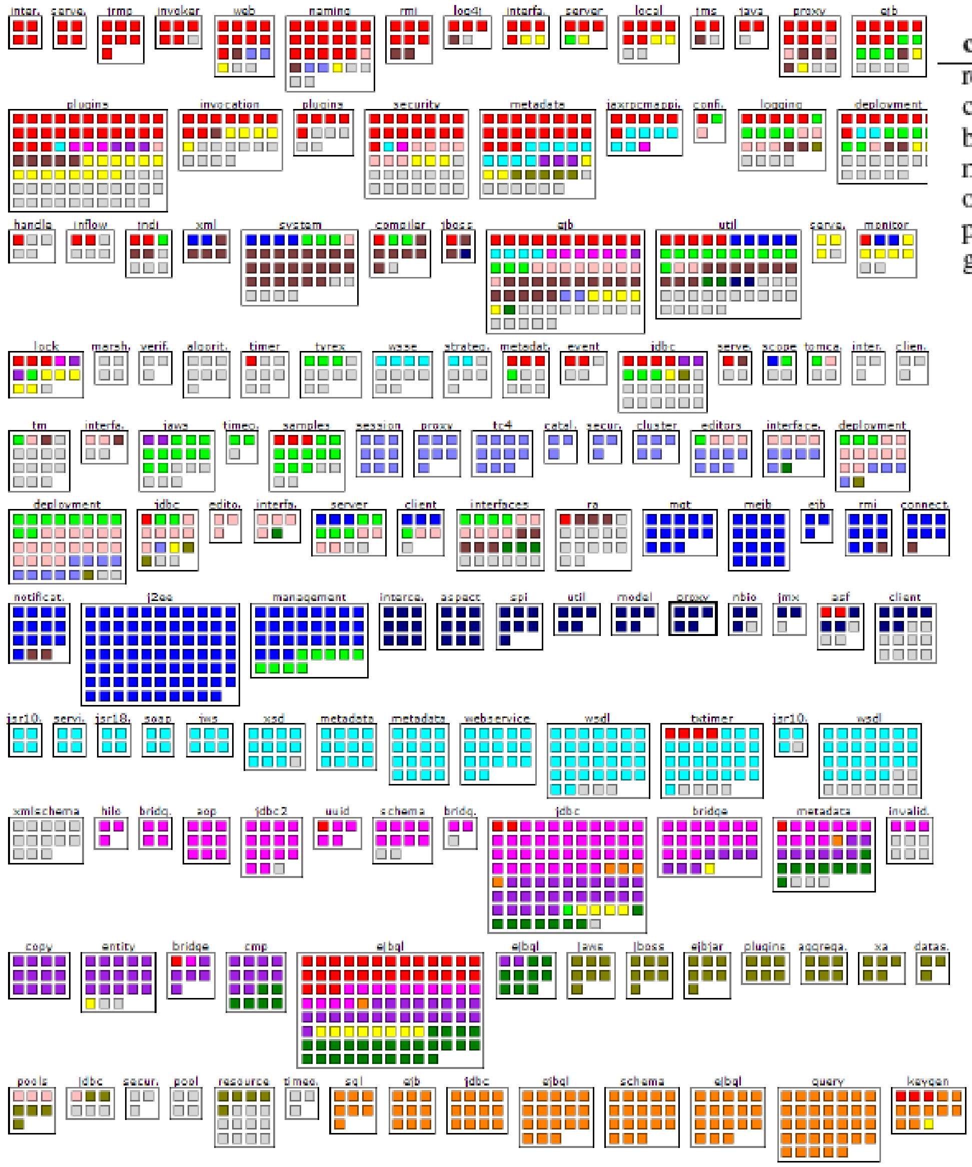
On the last row there are well-encapsulated plug-ins such as the Pink parts, which implement network protocols, or the Cyan and Magenta parts, which implement plug-ins for different version control systems.

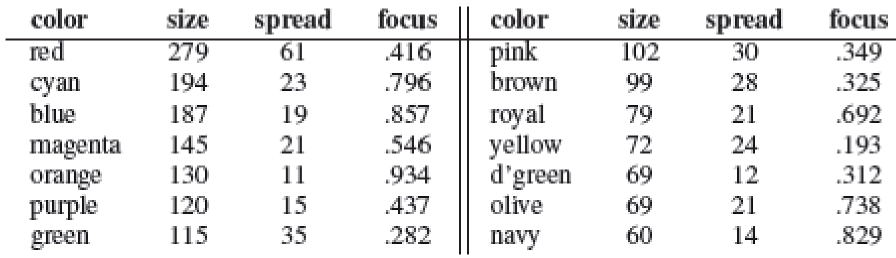
Application: The Distribution of File Ownership

- In this case, the artifacts are the files, and the properties are the owner of the file.
- To obtain the owner of a file, we use the heuristic that a file is owned by the developer that wrote the most lines of code in that file.

- JBoss is an open source Java application server
- The system is written by 133 authors, and 14 of them have implemented over 80% of the lines.
- Thus we select those 14 to be shown on the visualization as colored properties
- Figure 4 shows how the files owned by these authors are distributed over the file-folder structure: the parts are the folders that ever existed in the system, the elements are the file that ever existed in the system and the colors refer to the owners.

File Ownership in JBoss





color	size	spread	focus	color	size	spread	focus
red	279	61	.416	pink	102	30	.349
cyan	194	23	.796	brown	99	28	.325
blue	187	19	.857	royal	79	21	.692
magenta	145	21	.546	yellow	72	24	.193
orange	130	11	.934	d'green	69	12	.312
purple	120	15	.437	olive	69	21	.738
green	115	35	.282	navy	60	14	.829

The focus metric for example covers the whole ranges from 0.1 to 0.9, revealing that there are all-rounders as well as specialists among the main developers of the system.

The two authors with the lowest focus, Green and Yellow, cross-cut the system touching about 30 parts, while the author with the next lowest focus, which
is Dark green, touches only 12 folders.

On the other end of the spectrum there are Orange, Blue and Navy as most
focused authors, each of them owns a dozen parts just by themselves, and author Red, that owns the most files and touches a third of the system but with a moderate focus

Number of Commits in JBoss

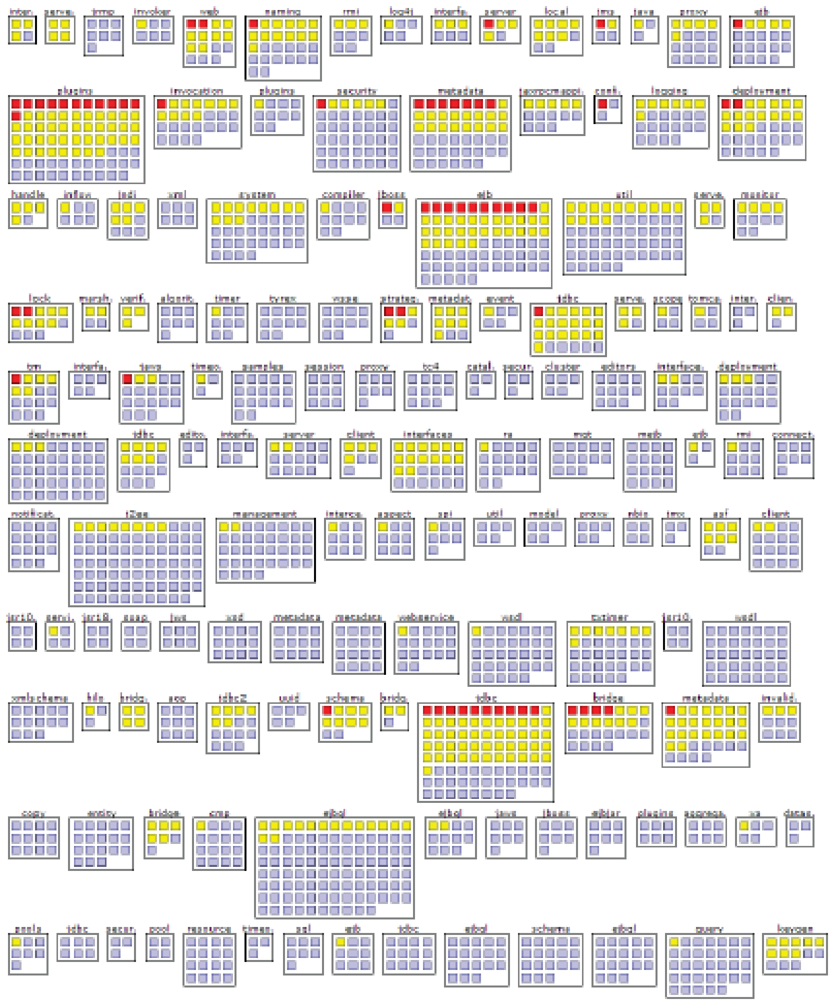


Figure 5. Distribution Map of the number of commits in JBoss (204 parts, 2107 elements and 3 properties).

In Figure 5 we show a variation of Figure 4 where we use the same layout, but we show the number of commits on the files.

We employed the heat scale:
Red denotes more than 50 commits, Yellow more than 20 commits, and Light blue less than 20 commits.

The figure
is zoomed out to the size of a thumbnail and contains 2000 elements: yet it is still readable

On the Limitations of Colors

- The study of human perception shows that there are only eight colors which are recalled by humans with more than 75% probability.
- This means that we can not discriminate more than a dozen different colors at once, which puts an upper limit on the number of properties being displayed on the Distribution Map.
- Sometimes it is possible to work around this limitation using the Pareto principle, which is the well known rule-of-thumb that for many phenomena, 80% of the consequences stem from 20% of the causes.
- See for example the case study in Section 3.2, where 80% of the systems code is written by only 14 out of 133 authors.