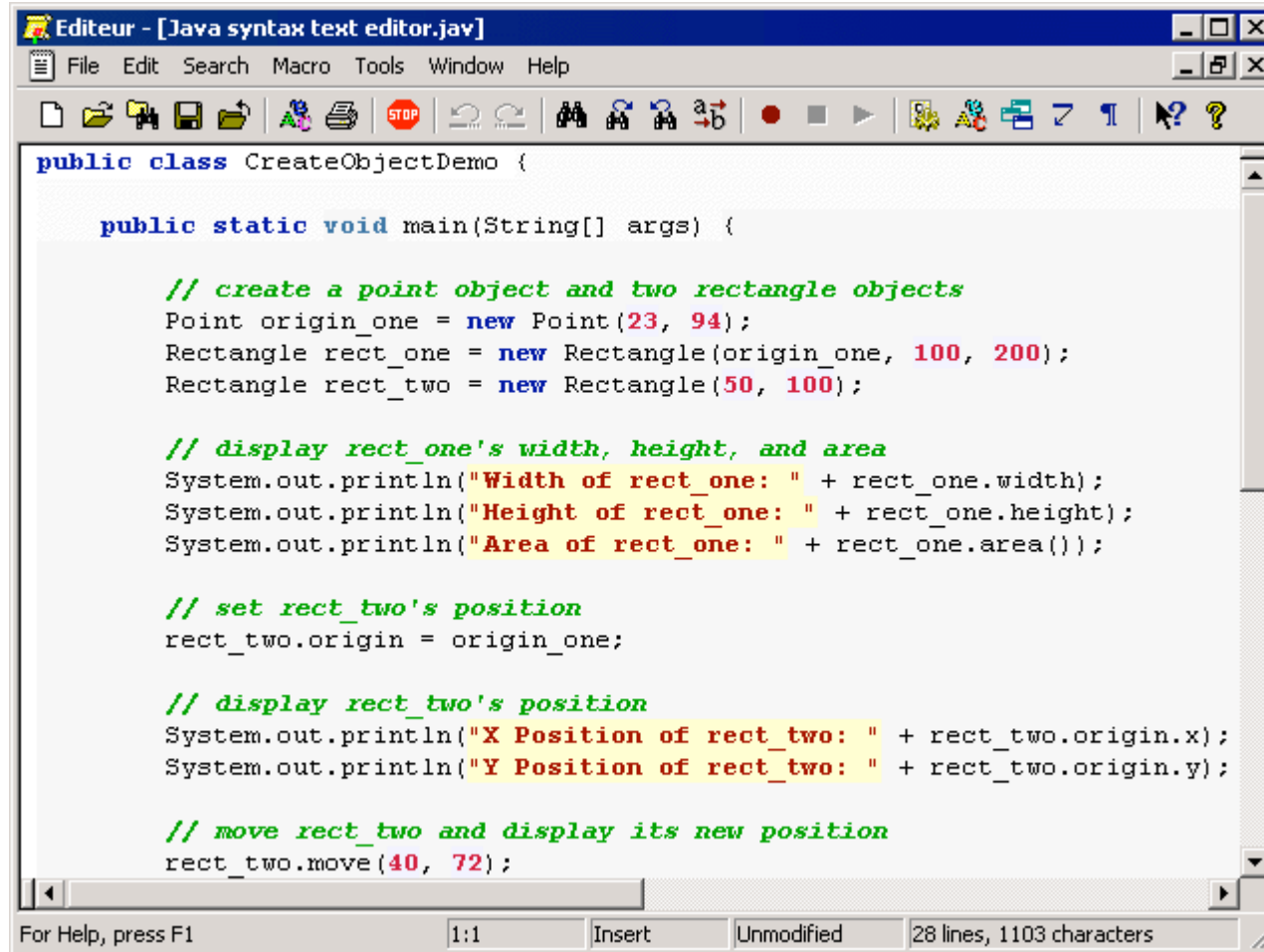


# Using Computer Linguistics to Analyze Software

**Adrian Kuhn**

Software Composition Group  
University of Bern, Switzerland

# Software is built from **source code**, a set of text files with commands



```
public class CreateObjectDemo {

    public static void main(String[] args) {

        // create a point object and two rectangle objects
        Point origin_one = new Point(23, 94);
        Rectangle rect_one = new Rectangle(origin_one, 100, 200);
        Rectangle rect_two = new Rectangle(50, 100);

        // display rect_one's width, height, and area
        System.out.println("Width of rect_one: " + rect_one.width);
        System.out.println("Height of rect_one: " + rect_one.height);
        System.out.println("Area of rect_one: " + rect_one.area());

        // set rect_two's position
        rect_two.origin = origin_one;

        // display rect_two's position
        System.out.println("X Position of rect_two: " + rect_two.origin.x);
        System.out.println("Y Position of rect_two: " + rect_two.origin.y);

        // move rect_two and display its new position
        rect_two.move(40, 72);
    }
}
```

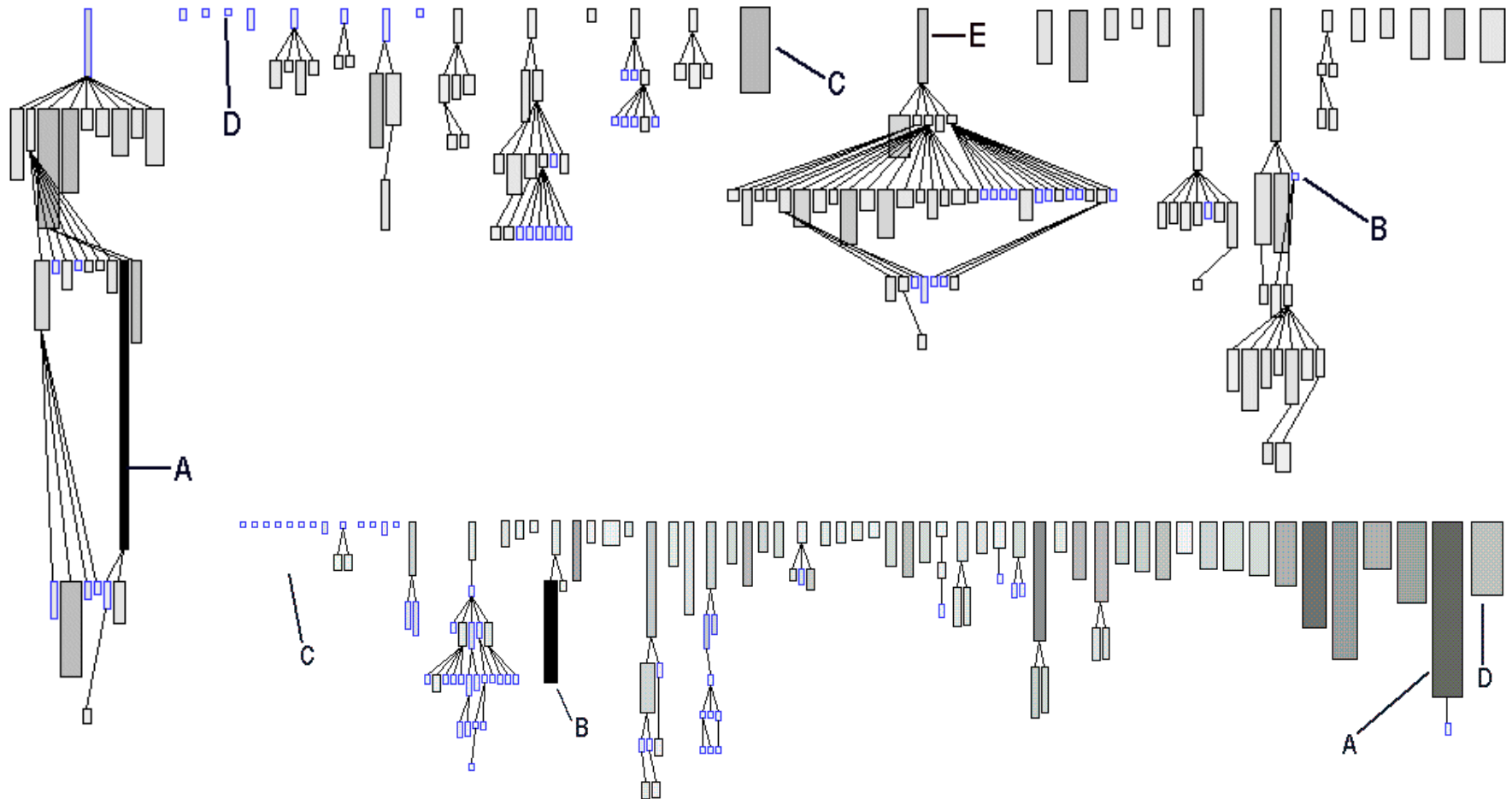
**Source code is by far **more often** read than written.**



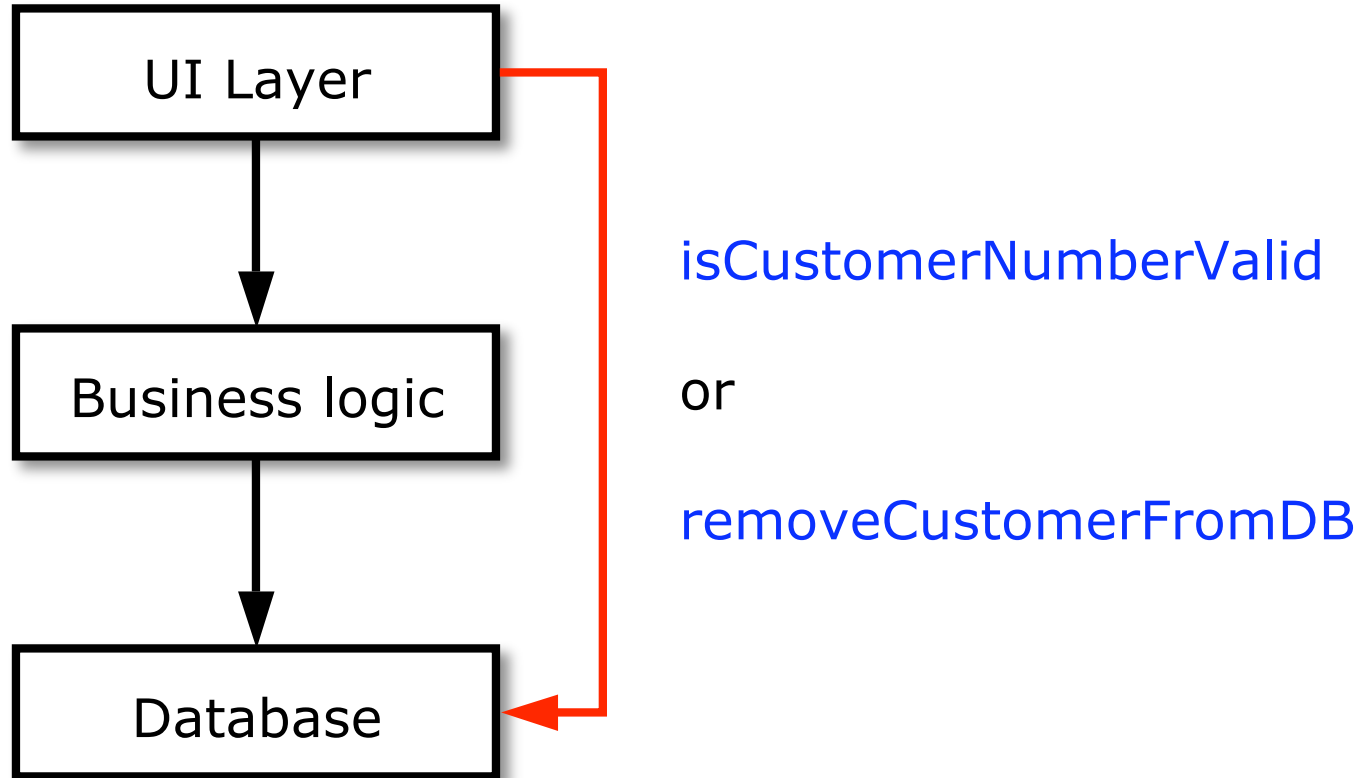
## Context

- You are not the author.
- It's your first contact.
- There are millions of lines of code.
- You have to go fast.

# Most software analysis approaches are based on **boxology**.



# Problem: what is the meaning of that all?



## **Solution: look at the **vocabulary** of the source code**

We use search engine technology.

Based in these observations

- Developers use meaningful names that convey domain concepts.
- Artifacts that use similar identifier names belong to the same concept.

# Thus we apply **Information Retrieval** on the source code

Hapax™

Hapax searches over 10,000 documents containing about 20,000 terms.

The term "hapax legomenon" is Greek and refers to a word that occurs only once in a given body of text. Hapax is a tool built on top of the Moose re-engineering framework.

# In a nutshell: Semantic Clustering

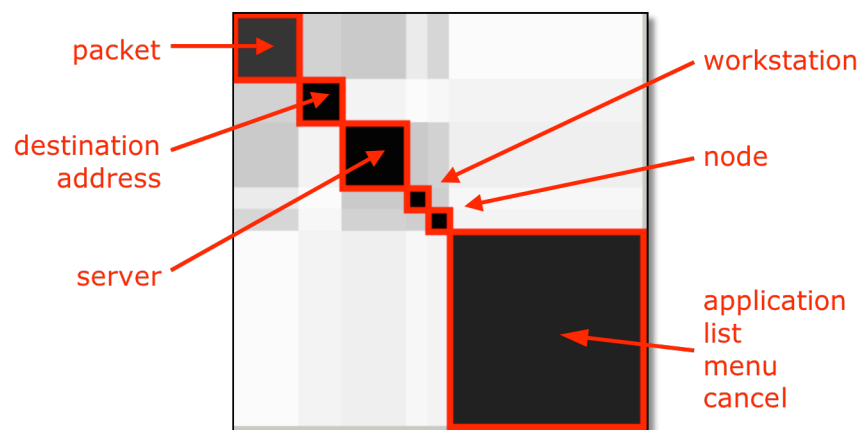
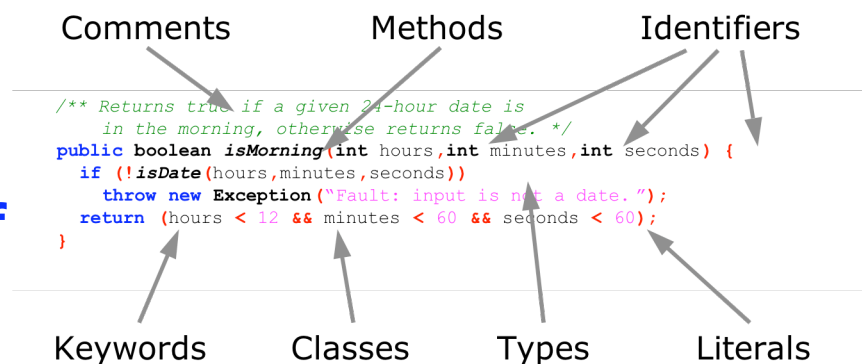
We treat source code as text documents.

**Two documents are similar if they use the same words.**

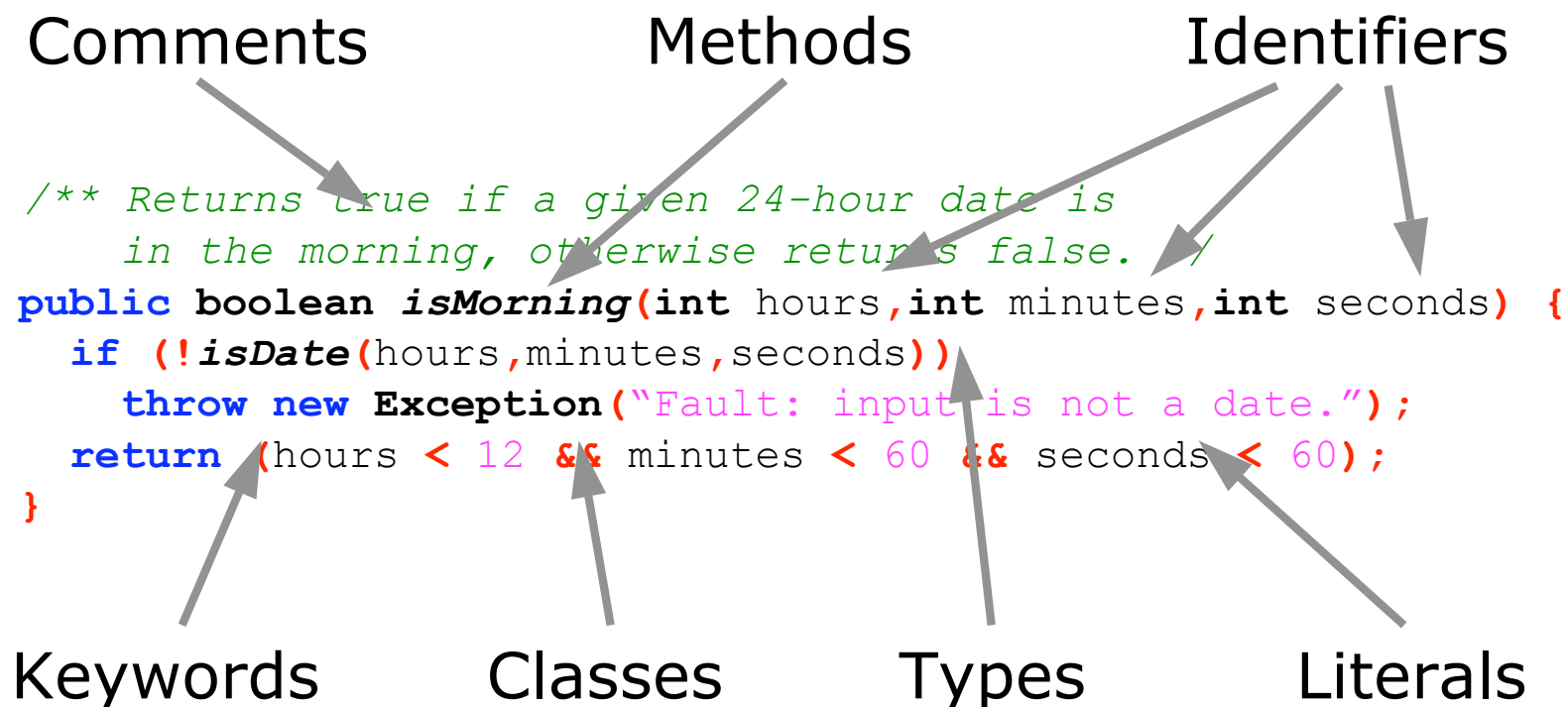
We cluster the documents based on the usage of words.

We use automatically retrieved labels to describe the clusters.

We use a map to illustrate the relation between concepts and structure on the



# Source code is **natural language text**, and each code item has a name



# We use **Latent Semantic Indexing** to analyze linguistic information

Latent Semantic Indexing analyses the distribution of terms over documents.

It is based on a **term-document-matrix** with word frequencies.

[Deerwester90]

# **LSI has been applied to many other problems before.**

## In Software Analysis

To recover links from external documentation. [Maletic01]

And to compare source code artifacts. [Marcus03]

And to categorize whole projects. [Kawaguchi05]

## In other fields

Automatic spell checking and thesaurus.

Automatic essay grading.

Multilingual search engines.

As a model how children acquire language.

And many more...

# LSI uses **Singular Value Decomposition** to compress the term-document-matrix.



# Anything can be used as **document...**

Documents are not necessarily source files.

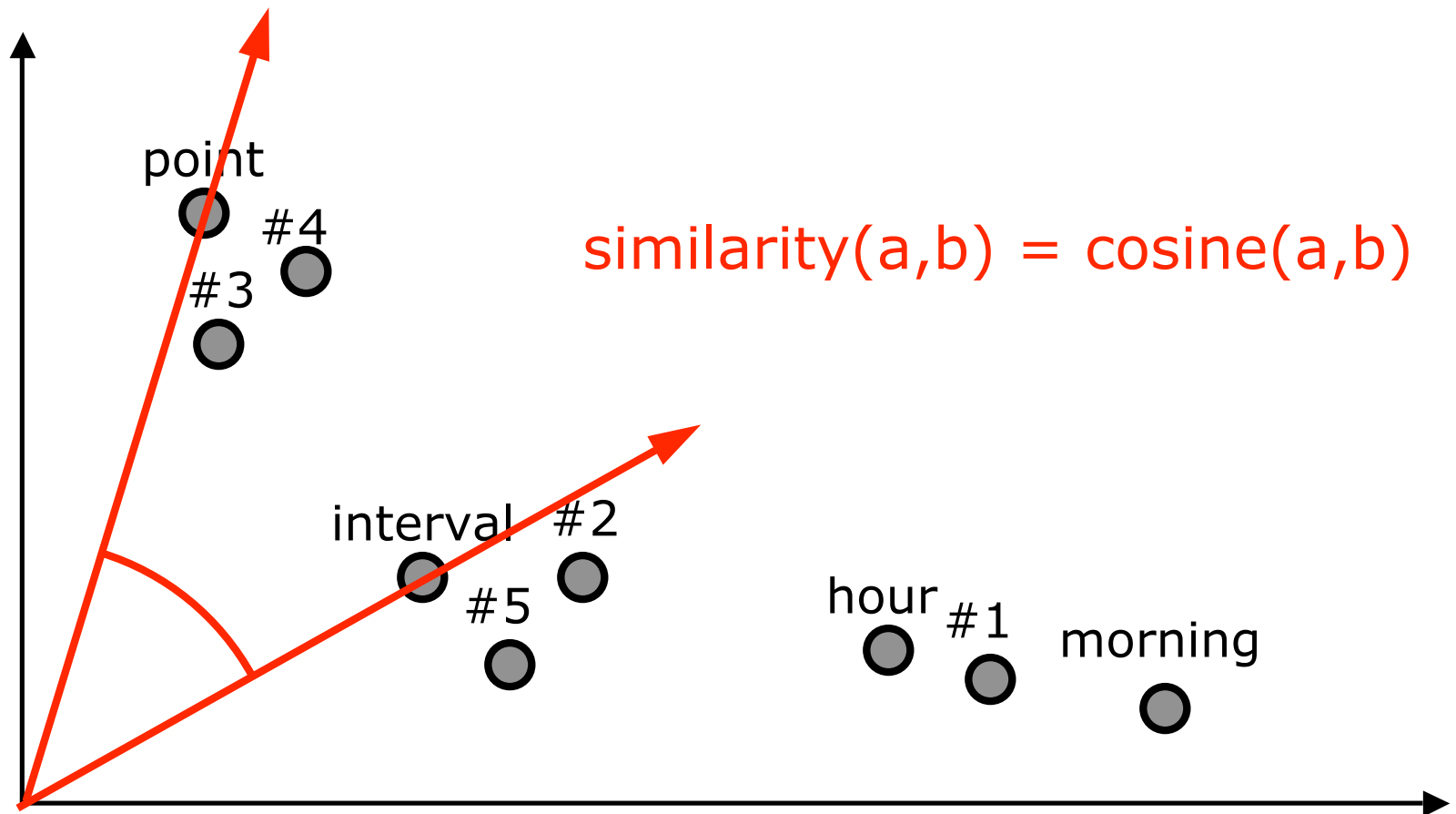
We prefer to split the source at structural levels such as: **package, class, method...**

But any software artifact that has a textual representation is a possible document

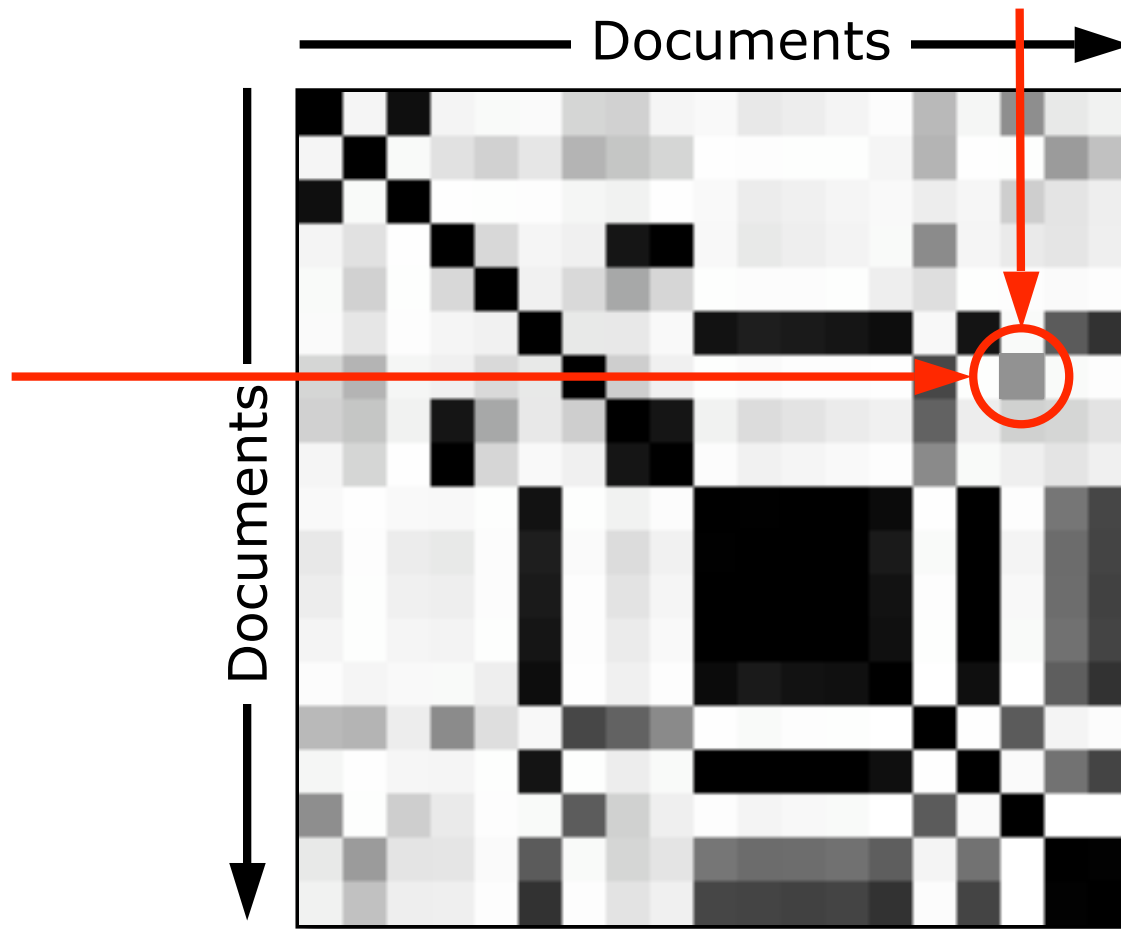
# We count the **word frequencies** in each document

	doc 1	doc 2	doc 3	doc 4	doc 5	...
hour	2	3		3	2	
interval		2	1		1	
morning	1				1	
point			4	4		
...						

# Latent semantic indexing puts all the documents and terms in one space



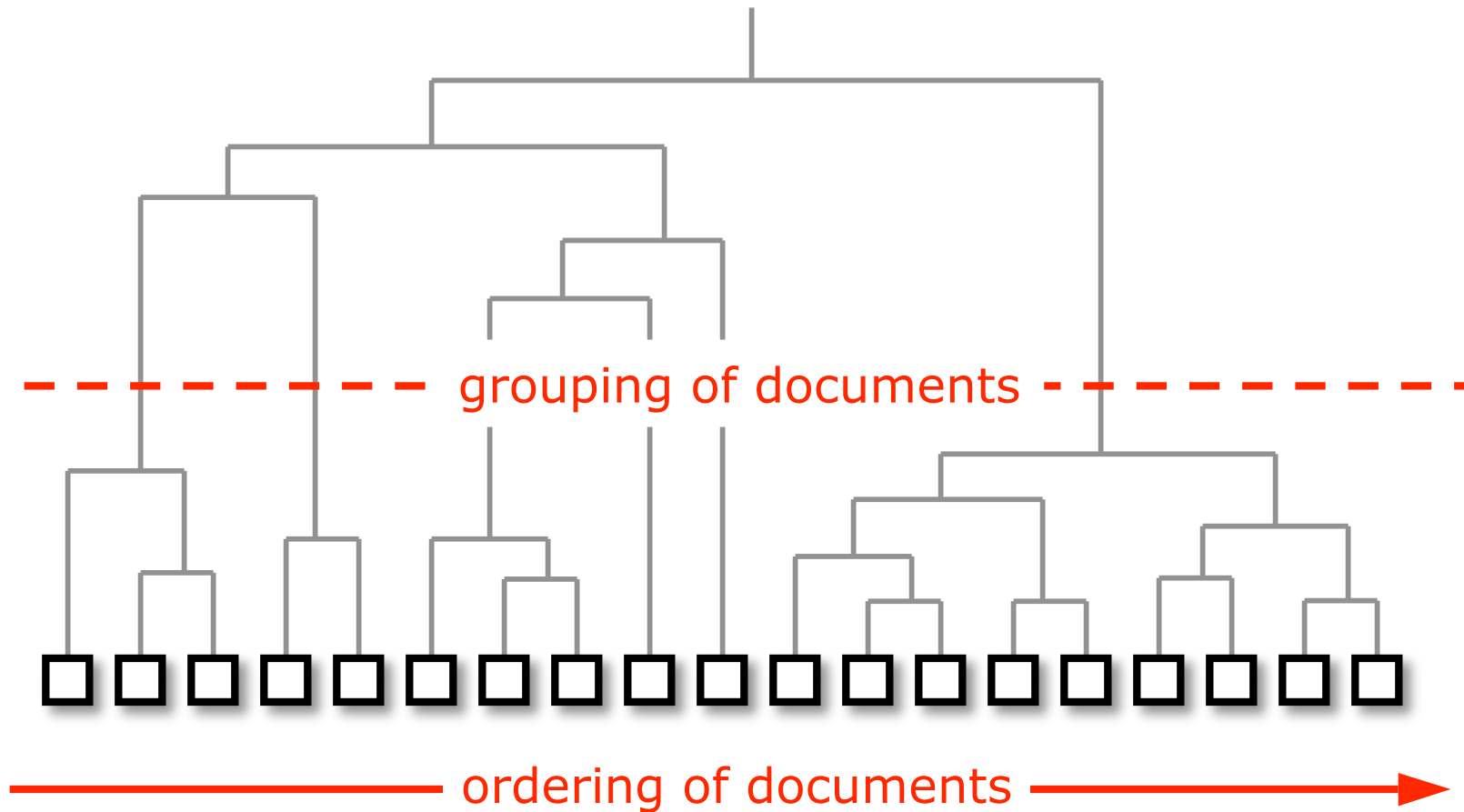
The **correlation matrix** shows all similarities between all documents



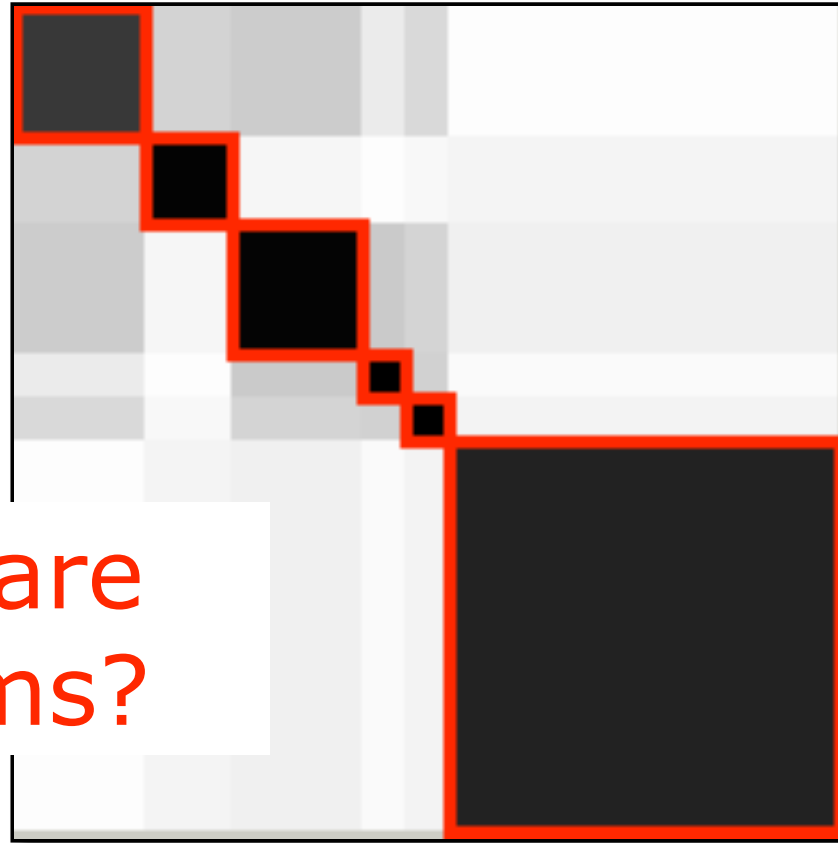
But an **unordered** matrix looks like television tuned to a dead channel...



# Clustering yields **both** an ordering and a grouping of documents

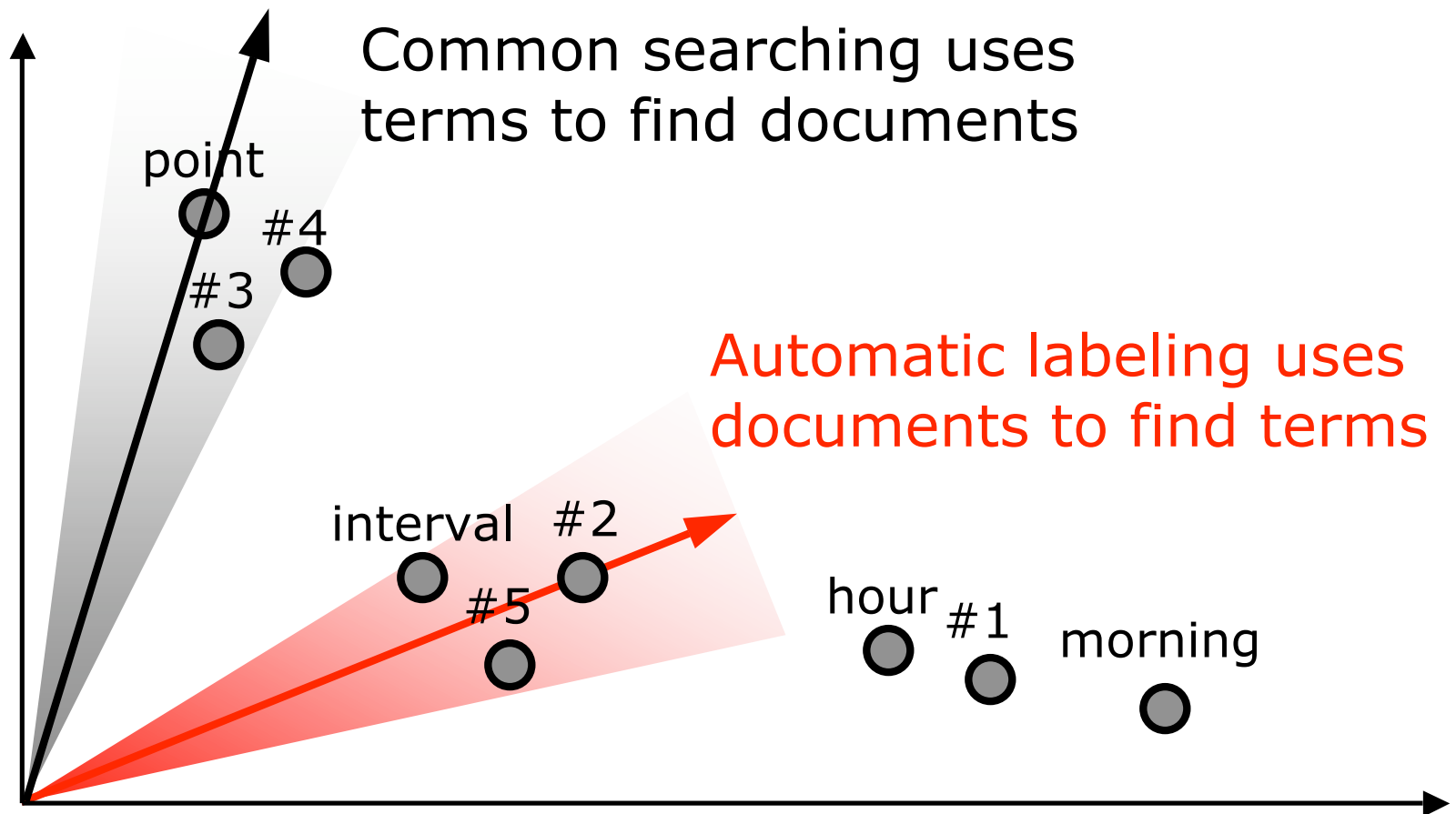


A **cluster** is set of documents which use similar terms

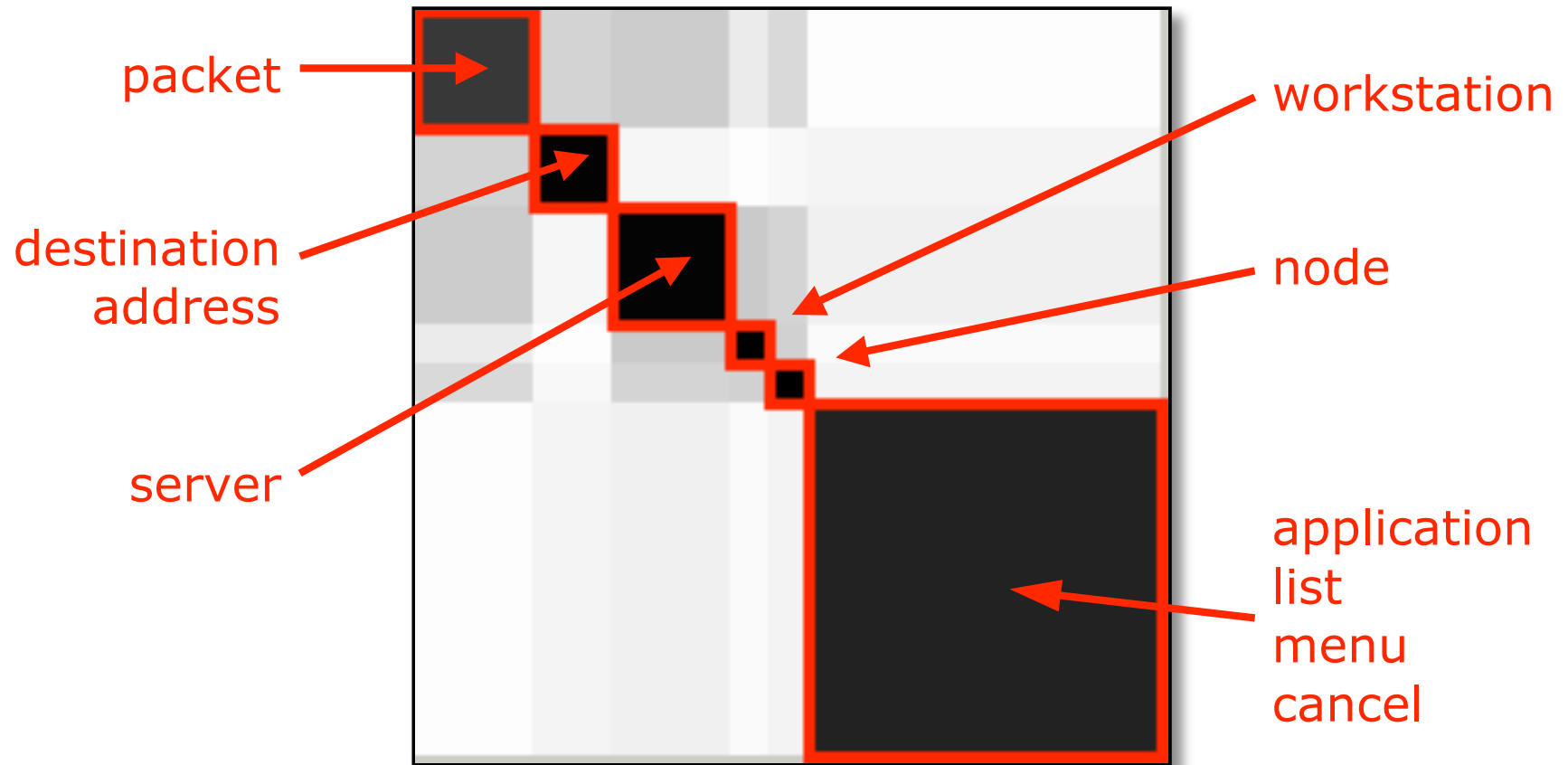


But what are these terms?

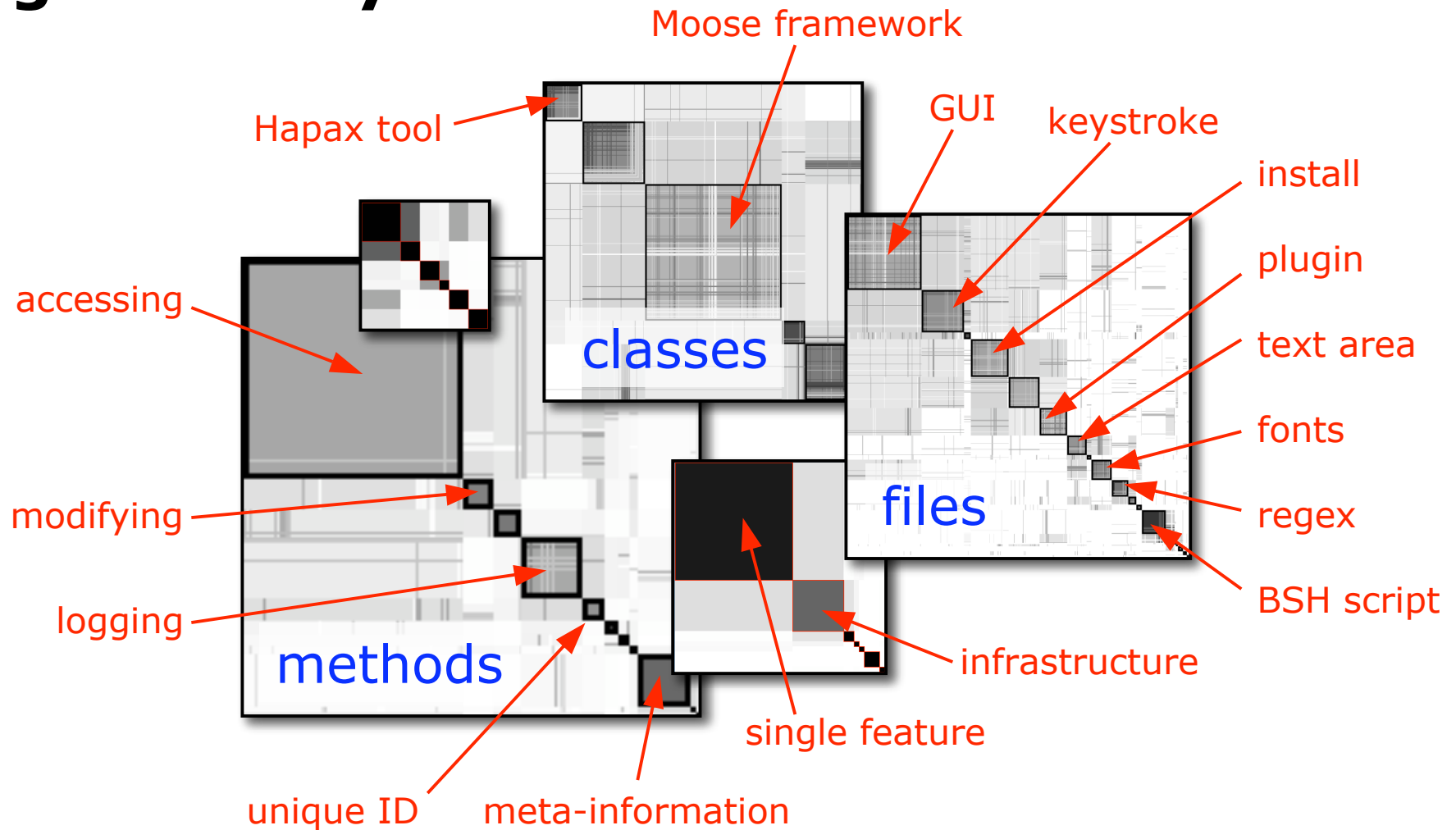
# We use the **documents as query** to search for cluster labels



# Automatically retrieved **labels** describe the clusters



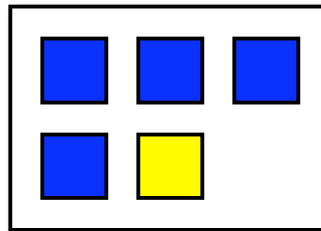
# Hapax was applied at different levels of granularity...



# Distribution Map illustrates the relation between concepts and structure

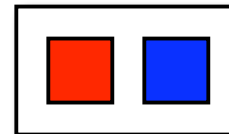
## Module A

5 artifacts  
4 x Blue  
1 x Yellow



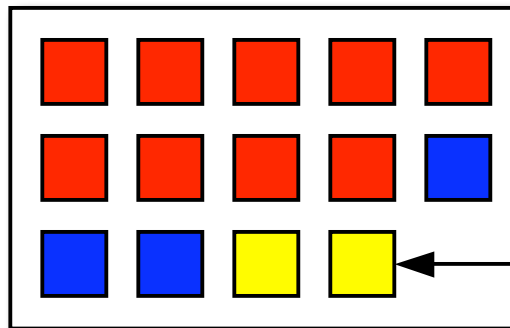
## Module B

2 artifacts  
1 x Red  
1 x Blue



## Module C

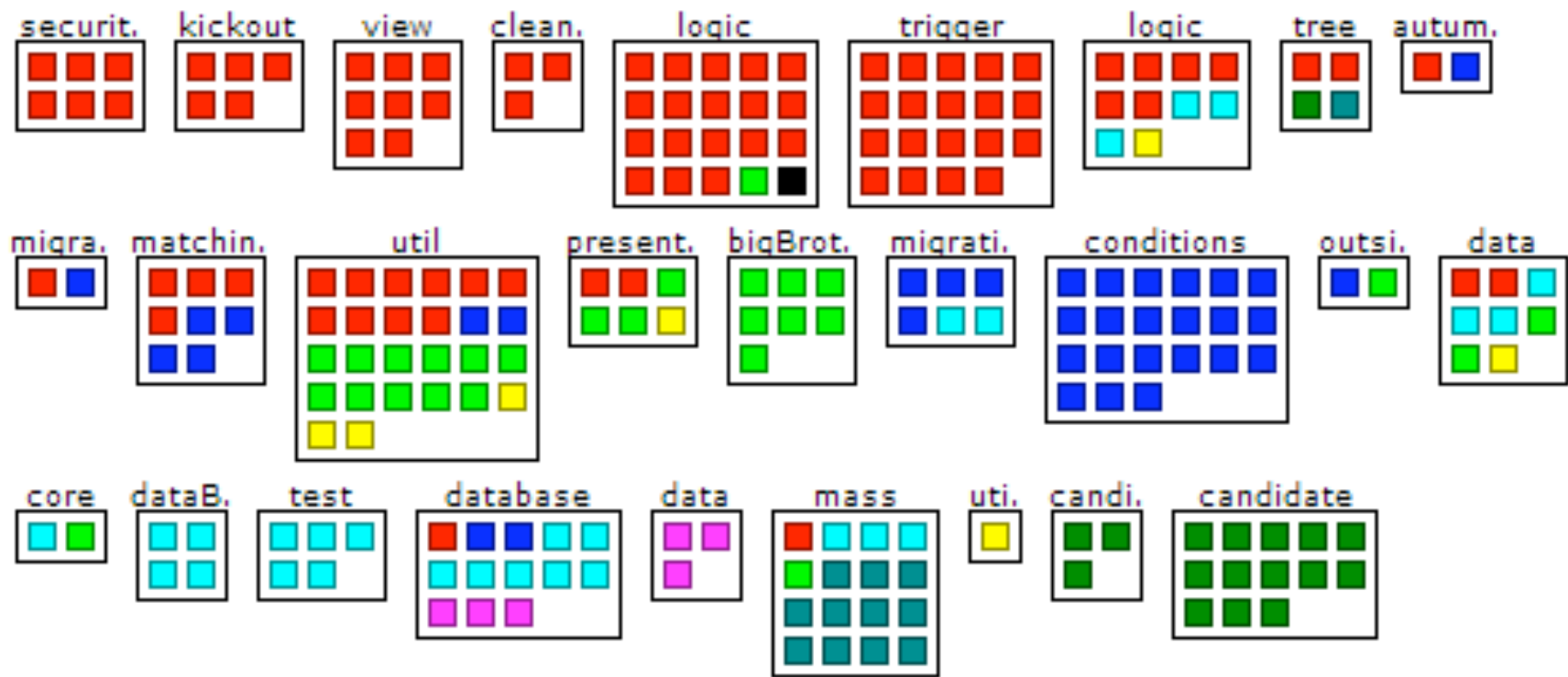
14 artifacts  
9 x Red  
3 x Blue  
2 x Yellow

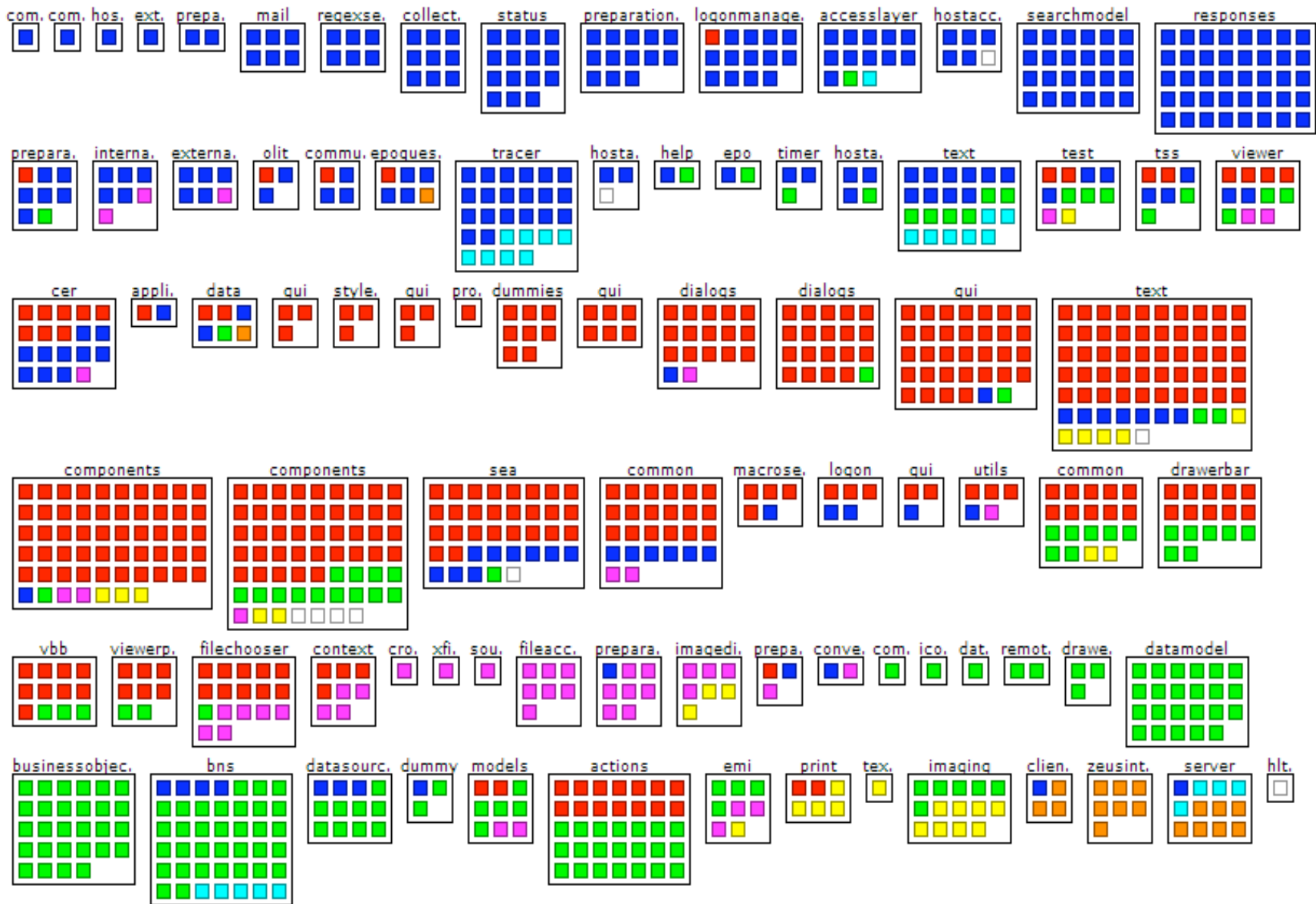


← *a module*

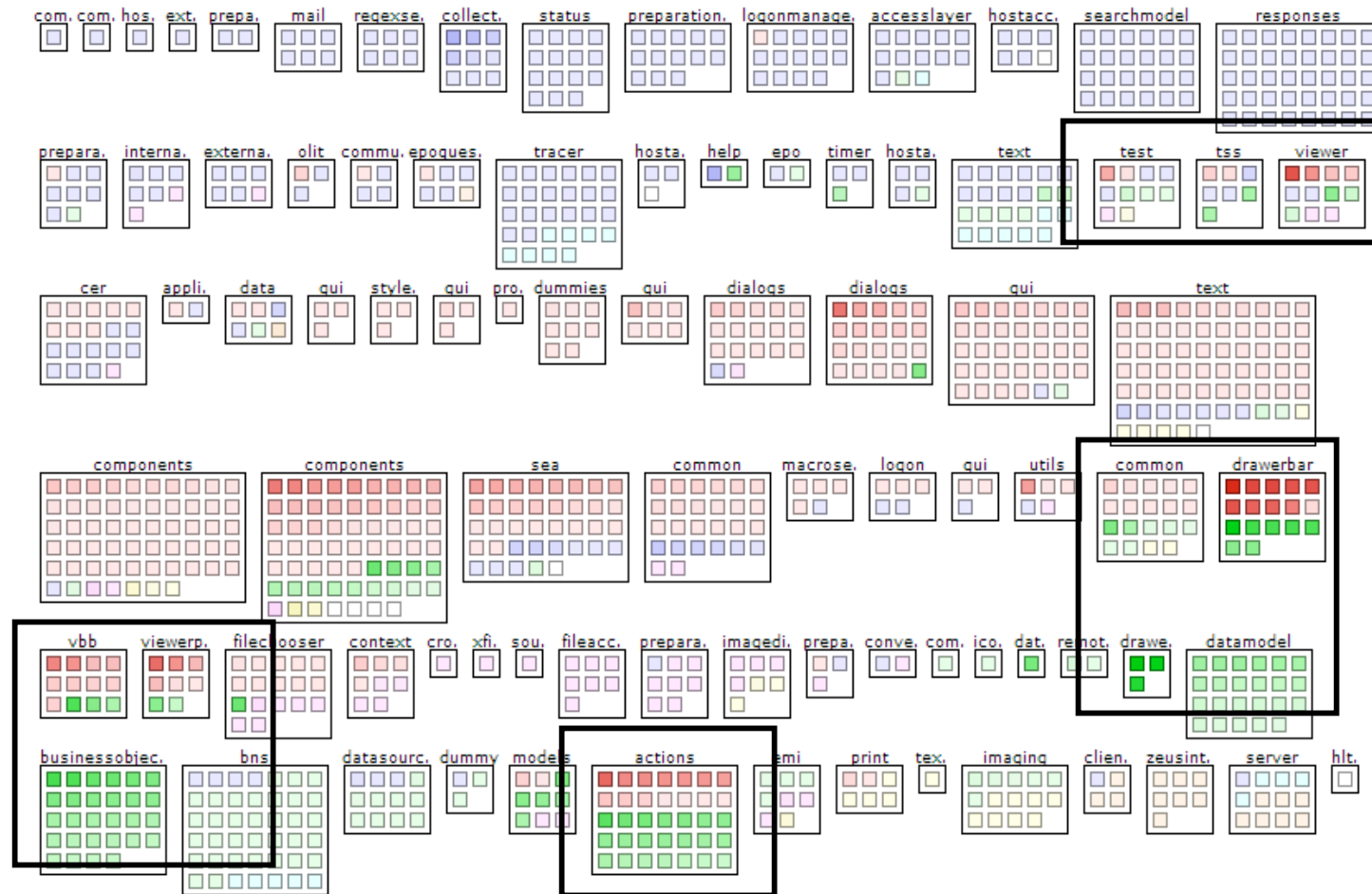
← *an artifact with  
concept color*

# Example: the distribution of concepts in the Oversight case study

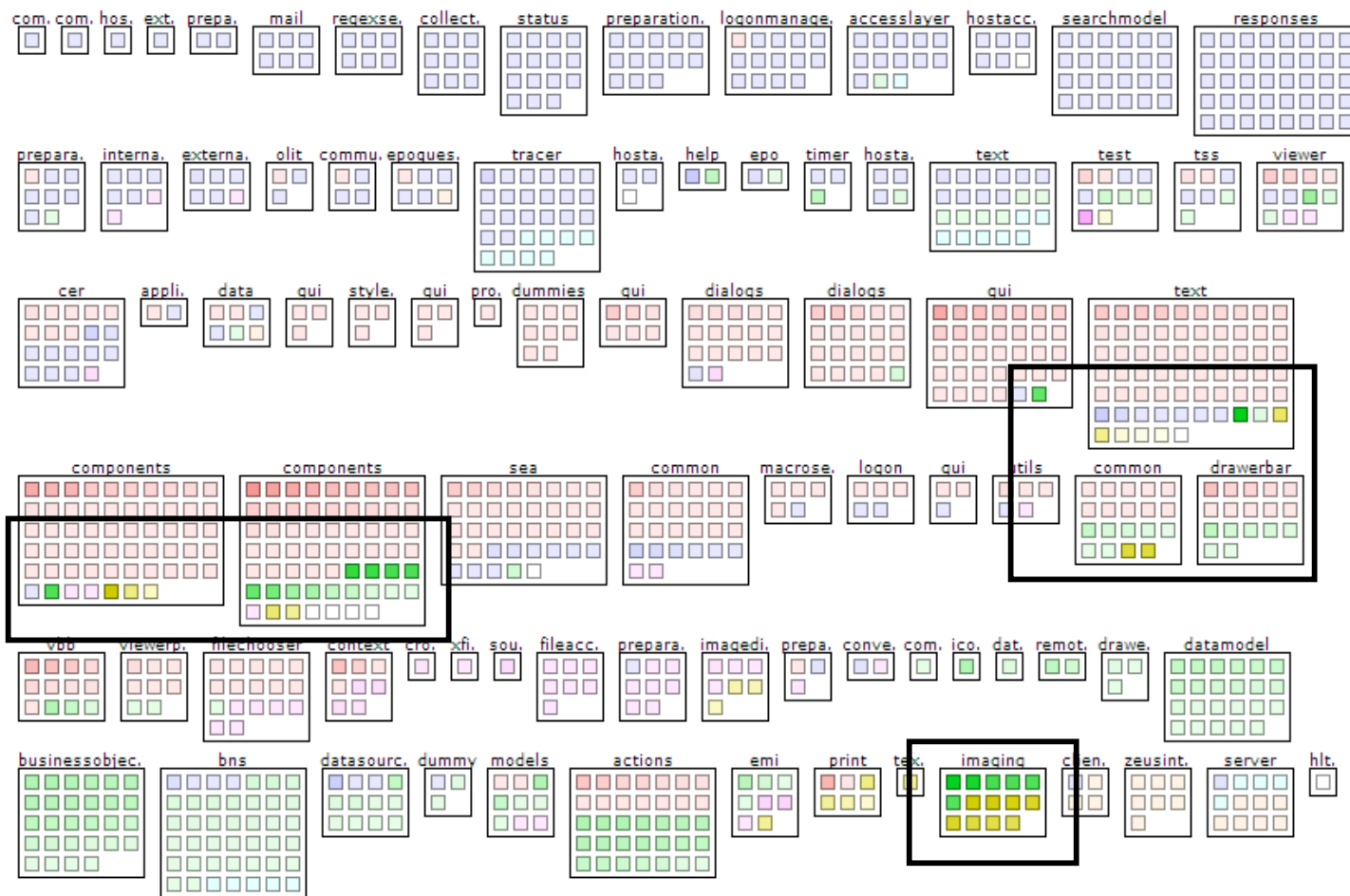




# Search the distribution map for classes related to "drawer".



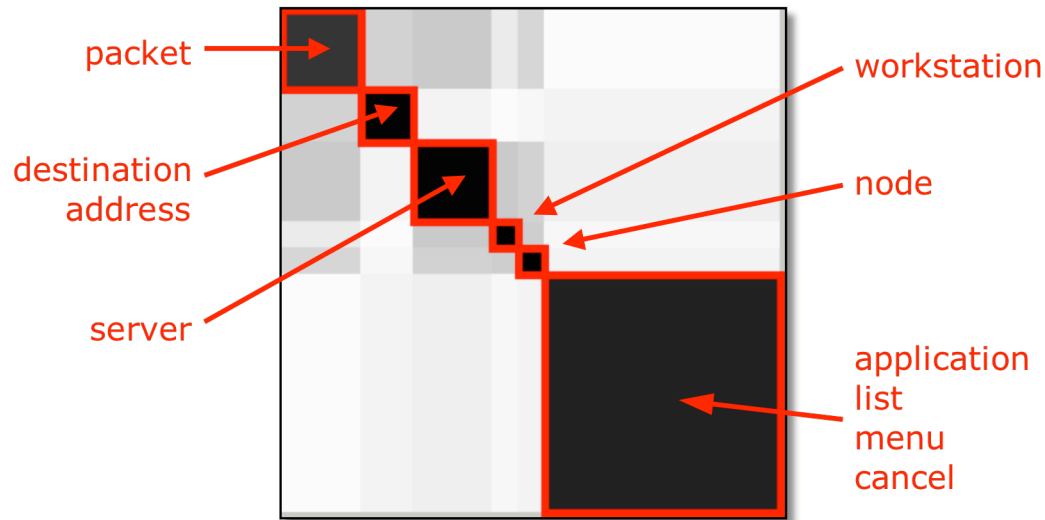
# Search the distribution map for classes related to "save cropped image".



# Moose and Hapax: explore structure and semantics with the same tool

The screenshot displays the Moose IDE interface. On the left, a tree view shows a project structure under 'Root', including folders like 'FAMIXAccess', 'FAMIXAttrib', and 'HapaxSem'. A 'Group Editor' window is open, showing a 'FAMIXClassGroup (314 items)' with a list of classes such as 'Root::Smalltalk::SCG::Moose::EntityType'. A 'Hapax.Similarities 164@164' window is active, displaying a similarity matrix with a 'Matrix Settings' tab. The matrix shows a diagonal of dark squares, indicating self-similarity. To the right of the matrix, there are lists for 'Row Entries' and 'Column Entries' containing method names like 'numberOfMethods()' and 'inheritedAttributesOf:'. A 'Labels' list includes 'inheritance', 'subclass', and 'superclass'. A 'Similarity = 0.774309' value is shown at the bottom right of the matrix window. In the foreground, a small dialog box titled 'Enter search query:' is open, with the text 'remove entity' entered in the input field and 'OK' and 'Cancel' buttons below it.

**Conclusion: we should not just look for boxes, we should look for names too!**



**Questions?**