IT University of Copenhagen

Software Architecture

Architecture Reconstruction

IV: Visualization

Assoc. Prof. Mircea Lungu

mlun@itu.dk

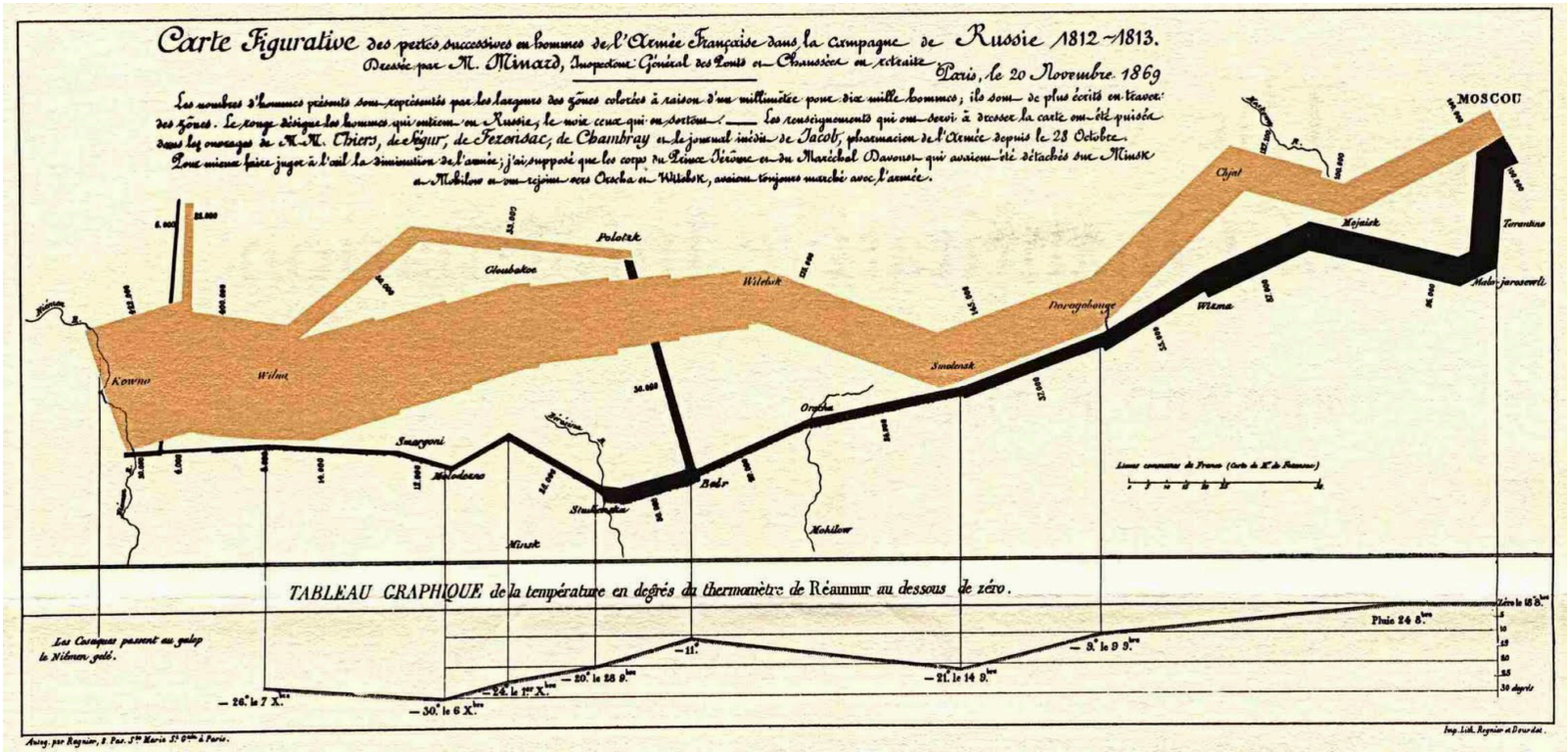
github.com/mircealungu/reconstruction

Outline

- The importance of presenting information
- Polymetric Views - Metrics-Enhanced (Architectural) Views
- Student Submission Examples

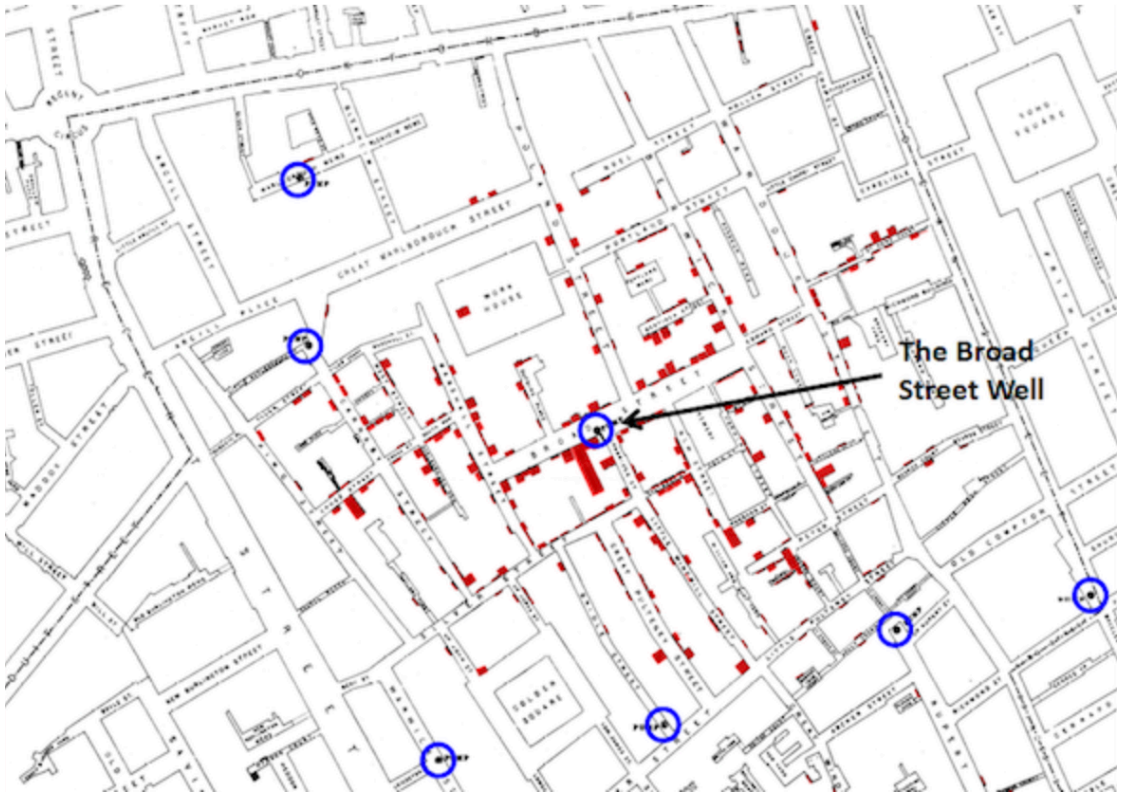
The Importance of Presenting Information

Napoleon's Invasion of Russia in One Infographic



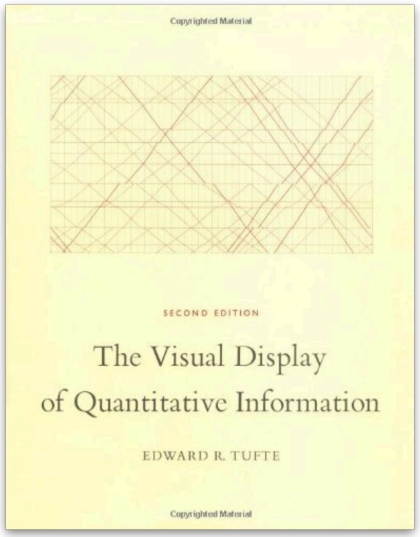
The Visual Display of Quantitative Information, E. Tufte

Dr. John Snows Cholera Map of London



The Visual Display of Quantitative Information, E. Tufte

The Visual Display of Quantitative Information, E. Tufte



Introduces: **The Five Laws of Data-Ink (Example)**

1. Above all else, show the data
2. Maximize Data-Ink Ratio
3. Erase non-data ink
4. Erase redundant data ink
5. Revise and edit

Polymetric Views

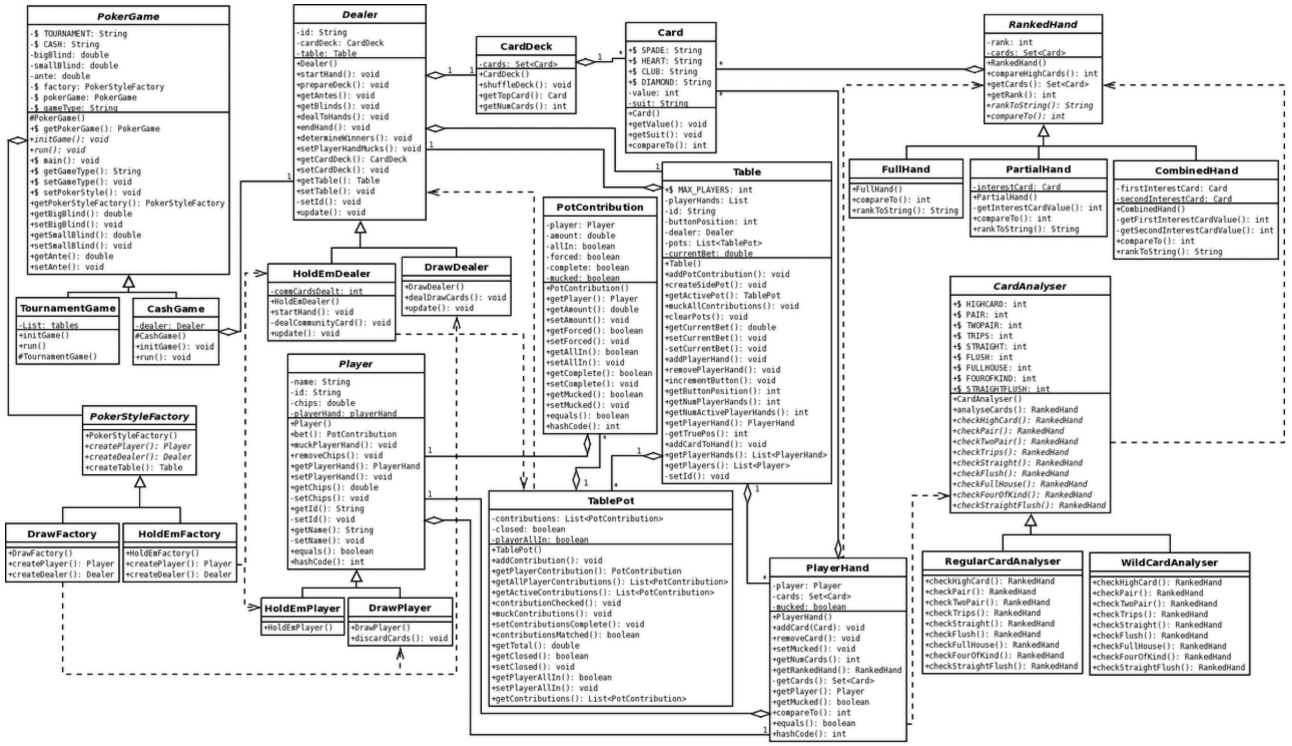
Software Visualization

"Software visualization techniques represent the intangible structures, interrelations, and interactions of software via visual metaphors in 2D and 3D" (Muller et al.)

Aproaches touched today

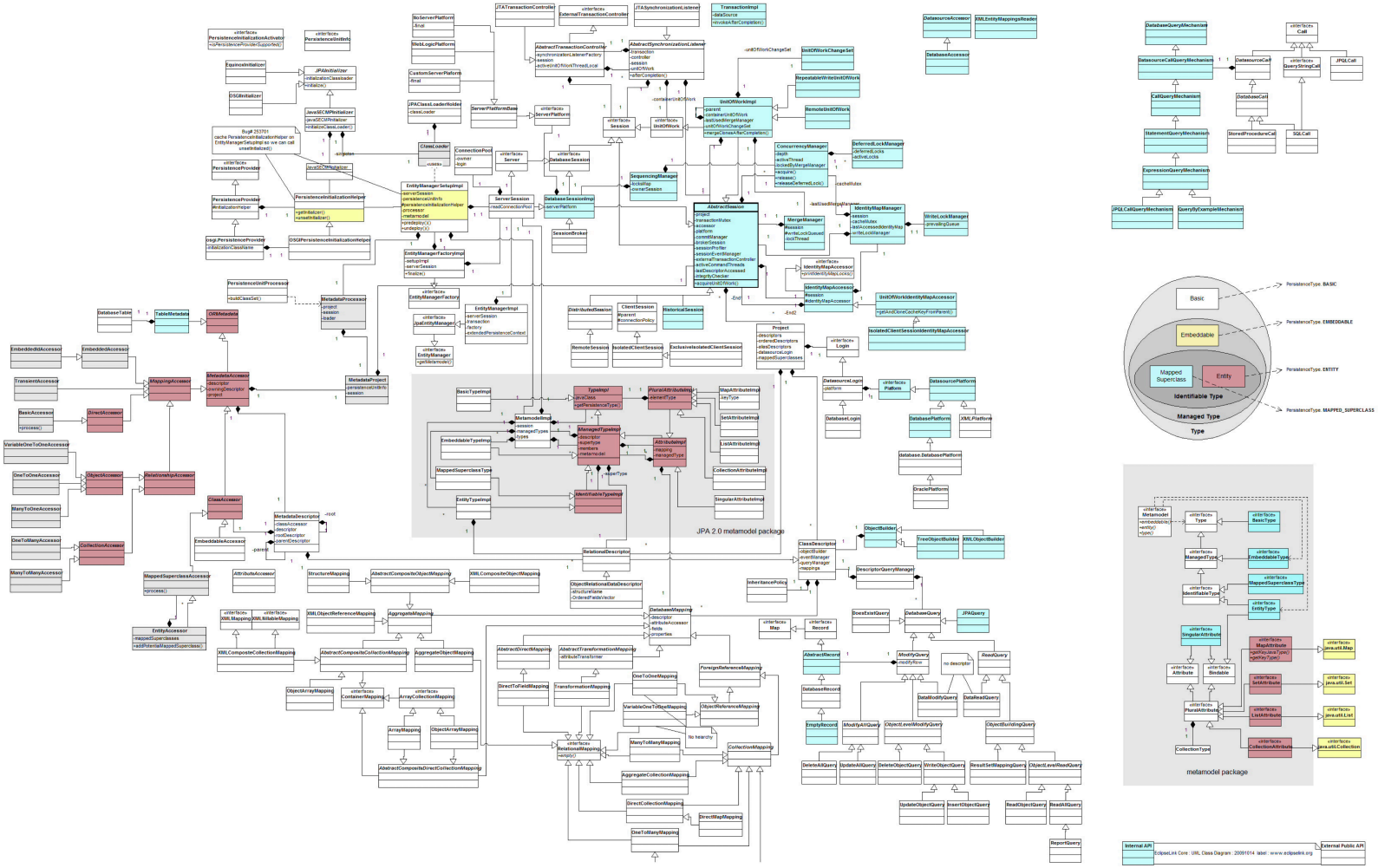
- UML
- Polymetric Views (in 2D & 3D)

UML As an Information Presentation Mechanism



The UML Diagram of a Pocker Game: 25 Classes

A More Realistic UML Diagram



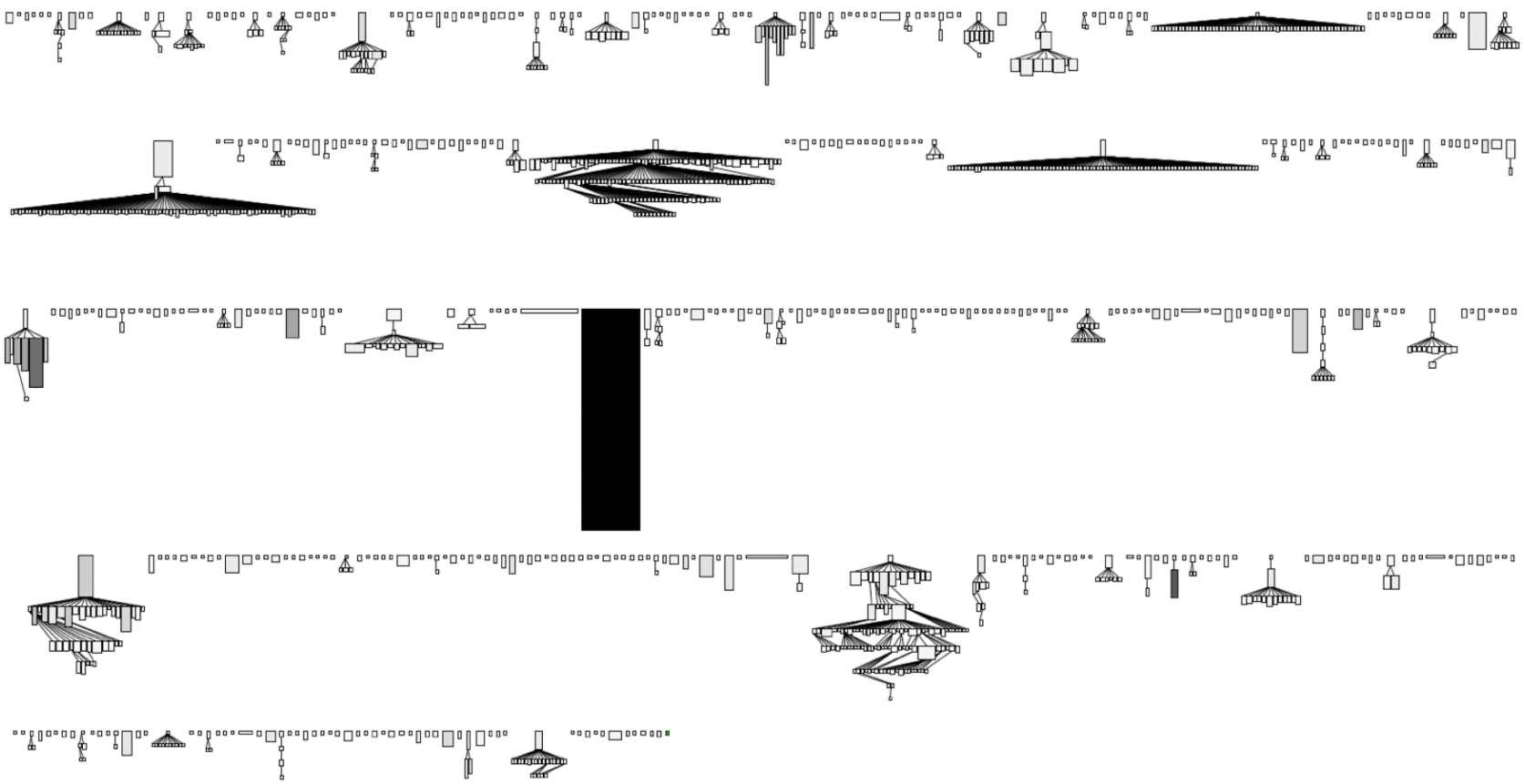
Java Link API - A persistence API: >100 Classes

UML Class Diagrams - Limitations

- Do not scale well
- Designed as a **modeling** language (thus for specification)
- Not used much for documentation(ArgoUML had no class diagrams about itself)

Alternative? Learn from Infovis!

System Complexity of ArgoUML > 1000 classes

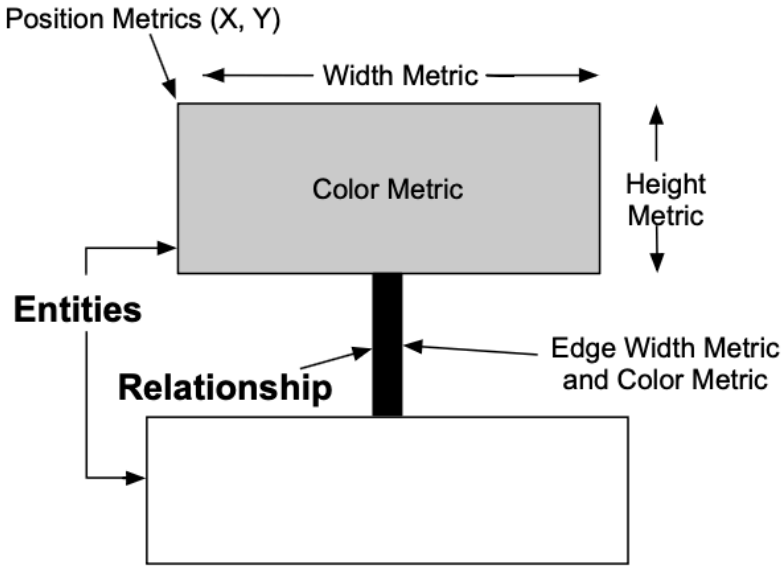


Entities: Classes: Relationships: inheritance. Height: # of Attributes; Width: # of Methods; Color: LOC

Polymetric View

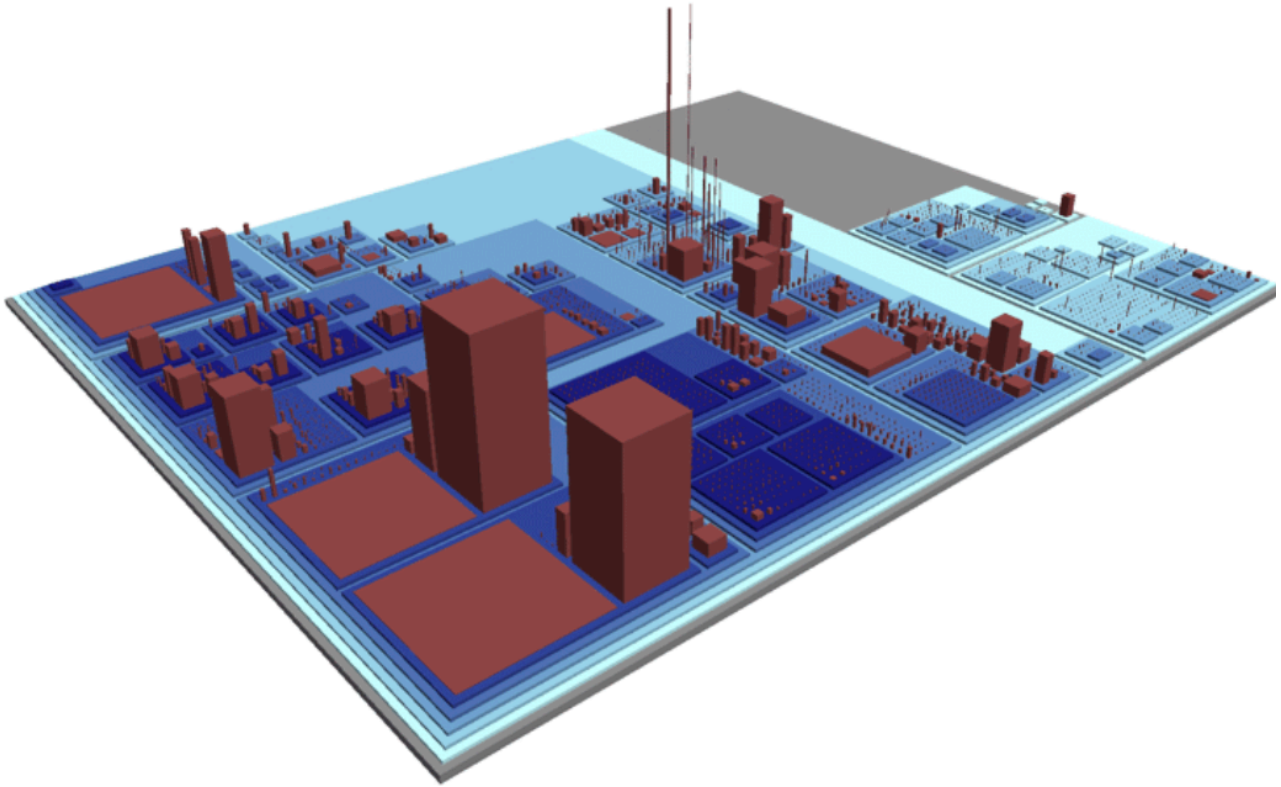
A view that visualizes multiple metrics together with system structure

Visual properties on which to map metrics: width, height, color, edges, etc.



Polymetric Views – A Lightweight Visual Approach to Reverse Engineering , Lanza & Ducasse

A 3D Polymetric View



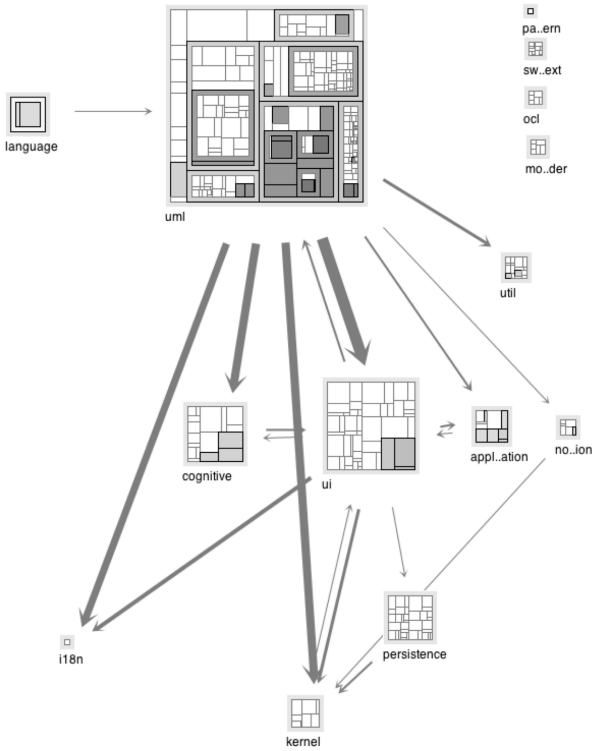
1: CodeCity visualizing ArgoUML, release 0.24. The buildings represent classes, the height is mapped to the number of methods and the width to the number of attributes in that class. Districts depict packages and their color is a measure of their position in the package hierarchy.

Developer-centric analysis of SVN Ecosystems, J. Malnati

Polymetric *Module* Views

Show dependencies between modules together with multiple metrics.

e.g. from Softwarenaut



- Edge width: number of low-level method calls
- Node size: LOC
- Node figure: treemap of contents

Example Visualisation from Past Student Submissions

Not bad… but how could we improve?

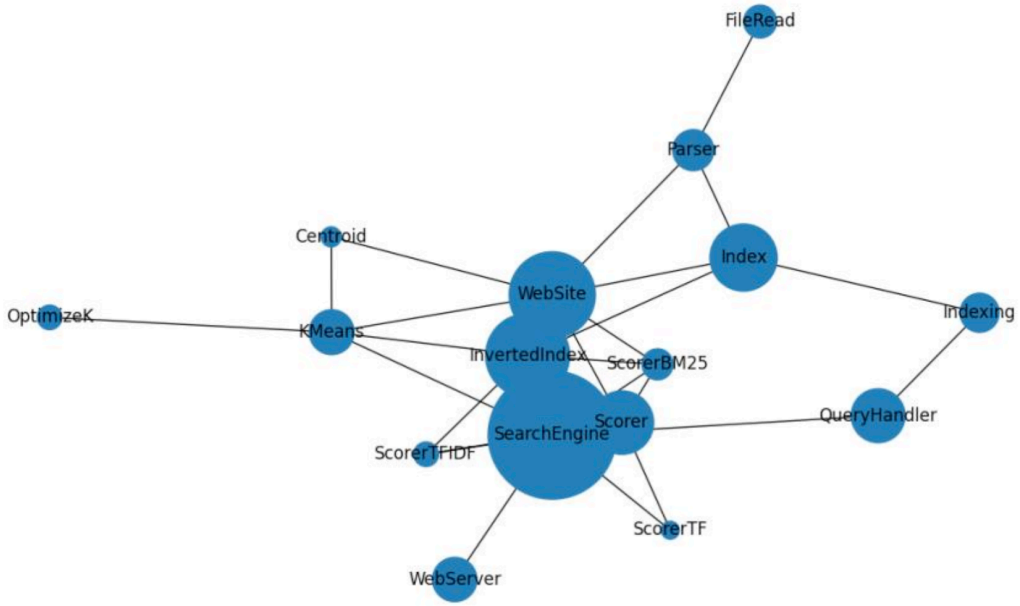


Figure 6: Evolutionary Hotspot-Dependencies

Not bad... but how could we improve?

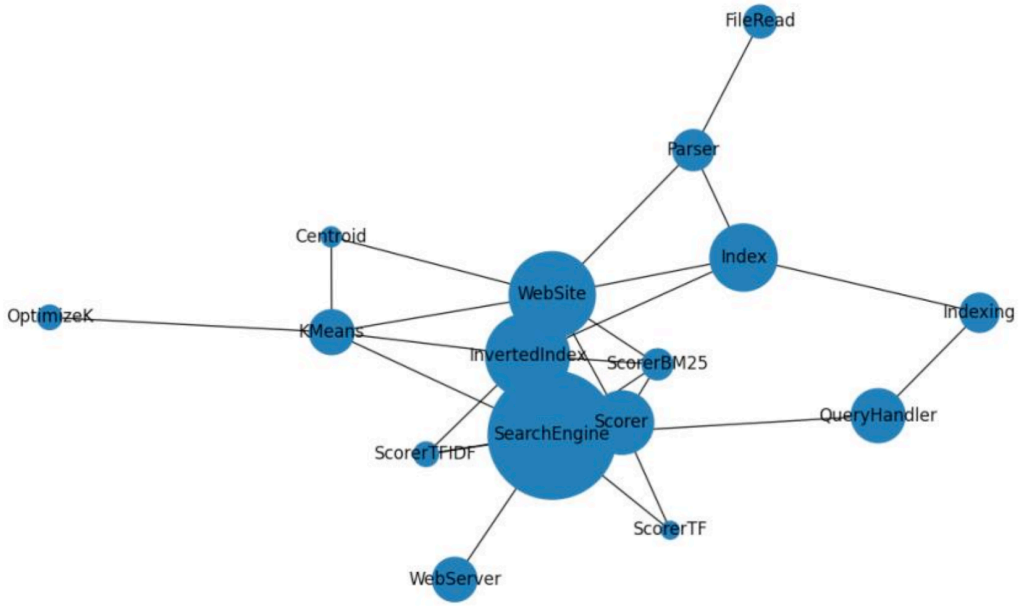
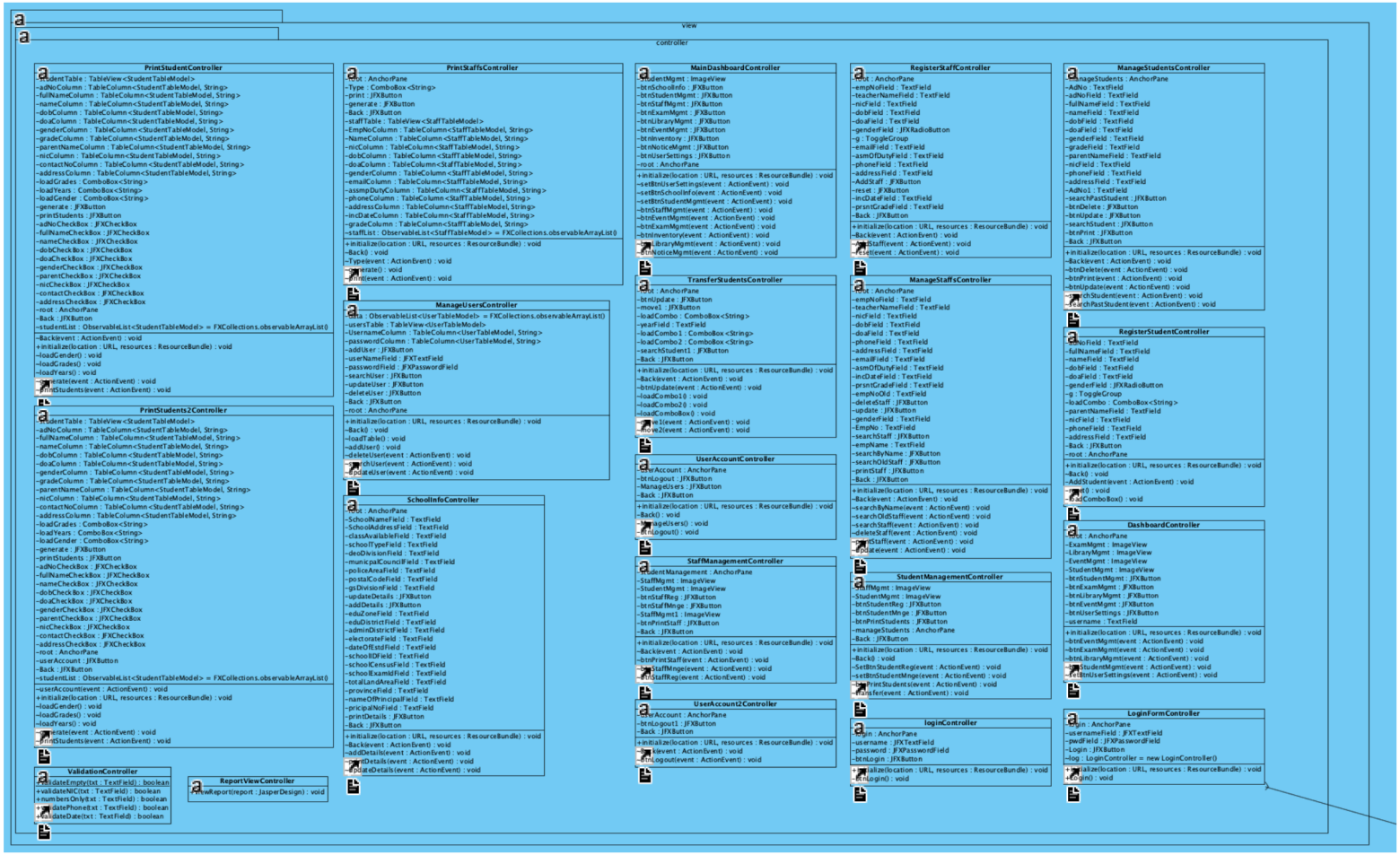


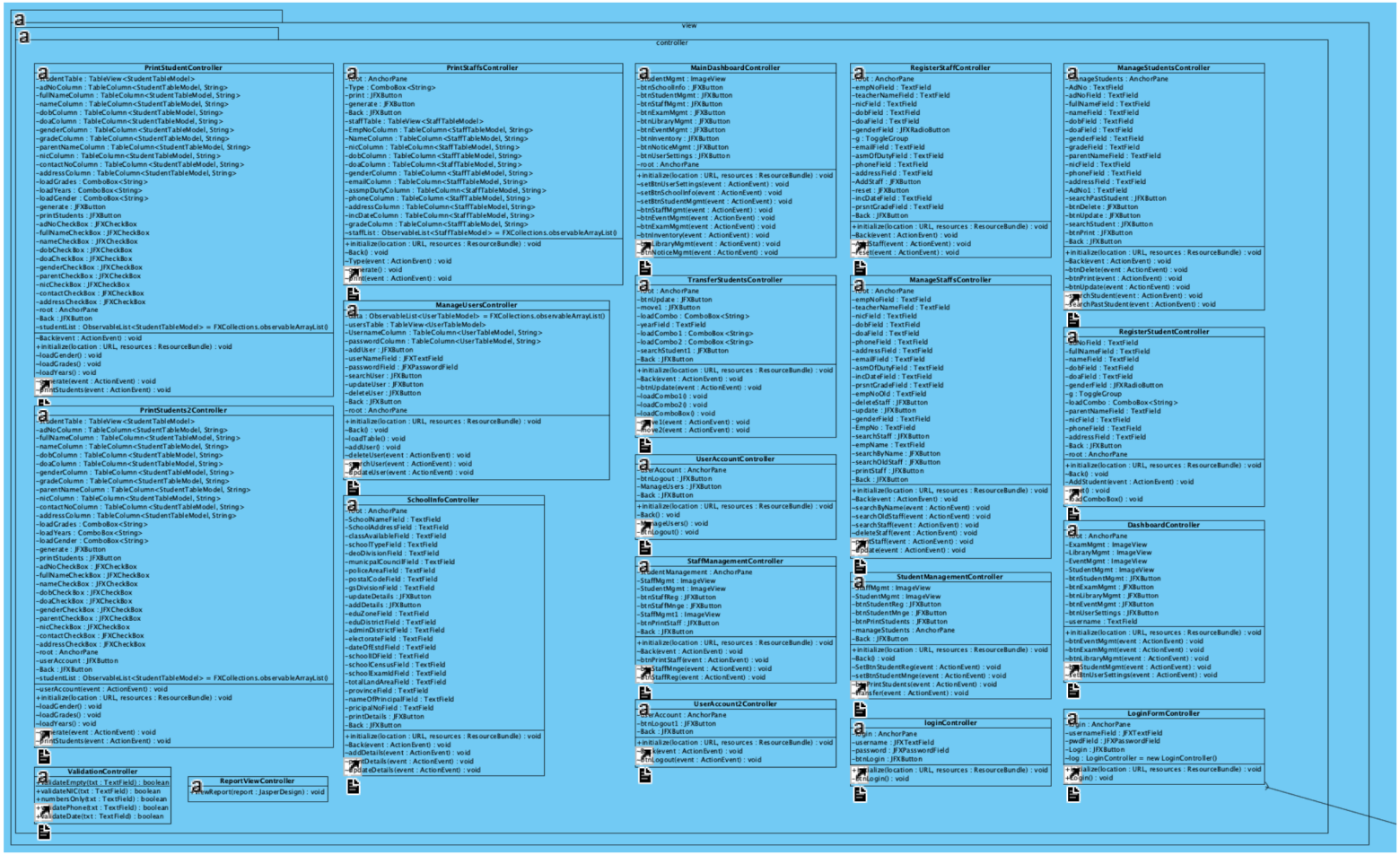
Figure 6: Evolutionary Hotspot-Dependencies

- Text readability - increase contrast or increase font
- Layout - increase distance between the central nodes
- Add Legend & Scale - otherwise how do I know whether this is 1KLOC or 1MLOC?

Not good... but why?



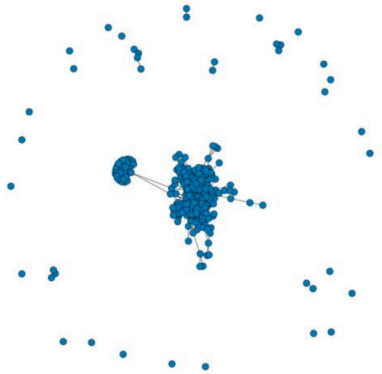
Not good... but why?



- Too much information; missing the abstraction step
- Unreadable: too low contrast between freground and background

Not good - but why?

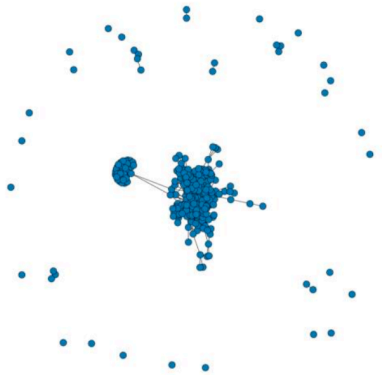
We could even open the files and look for “import” statements to see what other files are imported in each of them. Using the semi-automatic approach we could use a program like “pyter” to search for import statements containing another file name in the Zeeguu-core-package. This would tell us the dependencies between the files. Combining these two approaches with a graph tool in e.g. jupyter could help us illustrate dependencies:



Remember: *The Rational Design Process and Why to Fake it*

Not good - but why?

We could even open the files and look for “import” statements to see what other files are imported in each of them. Using the semi-automatic approach we could use a program like “pyter” to search for import statements containing another file name in the Zeeguu-core-package. This would tell us the dependencies between the files. Combining these two approaches with a graph tool in e.g. jupyter could help us illustrate dependencies:

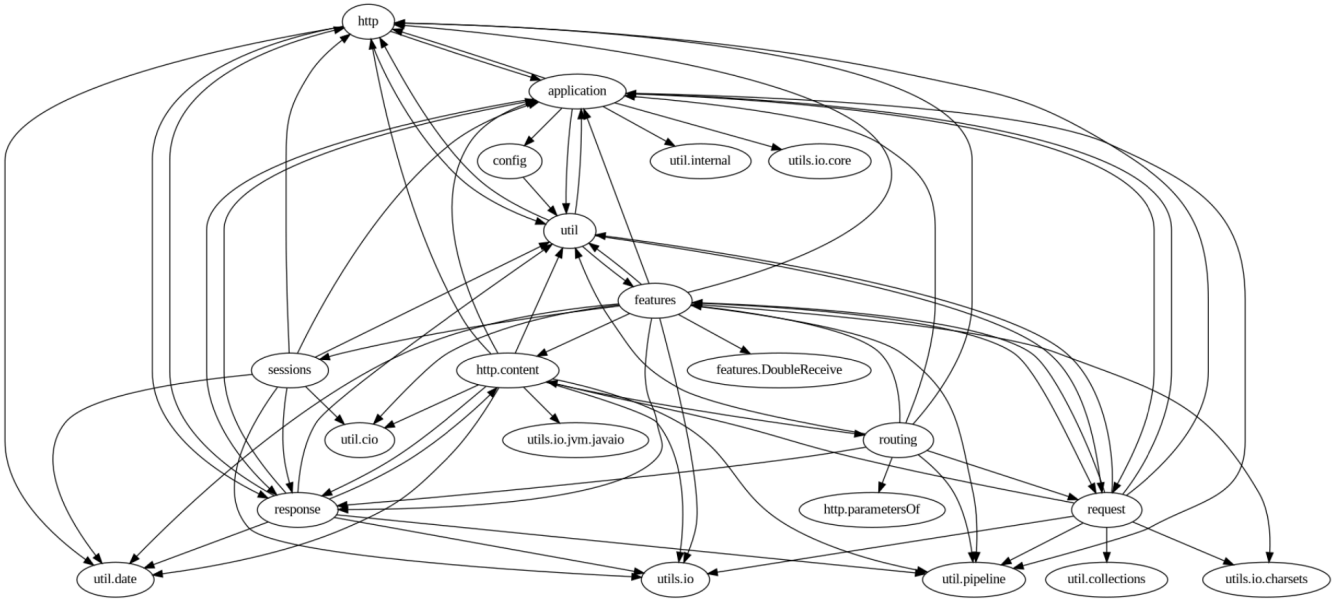


Remember: *The Rational Design Process and Why to Fake it*

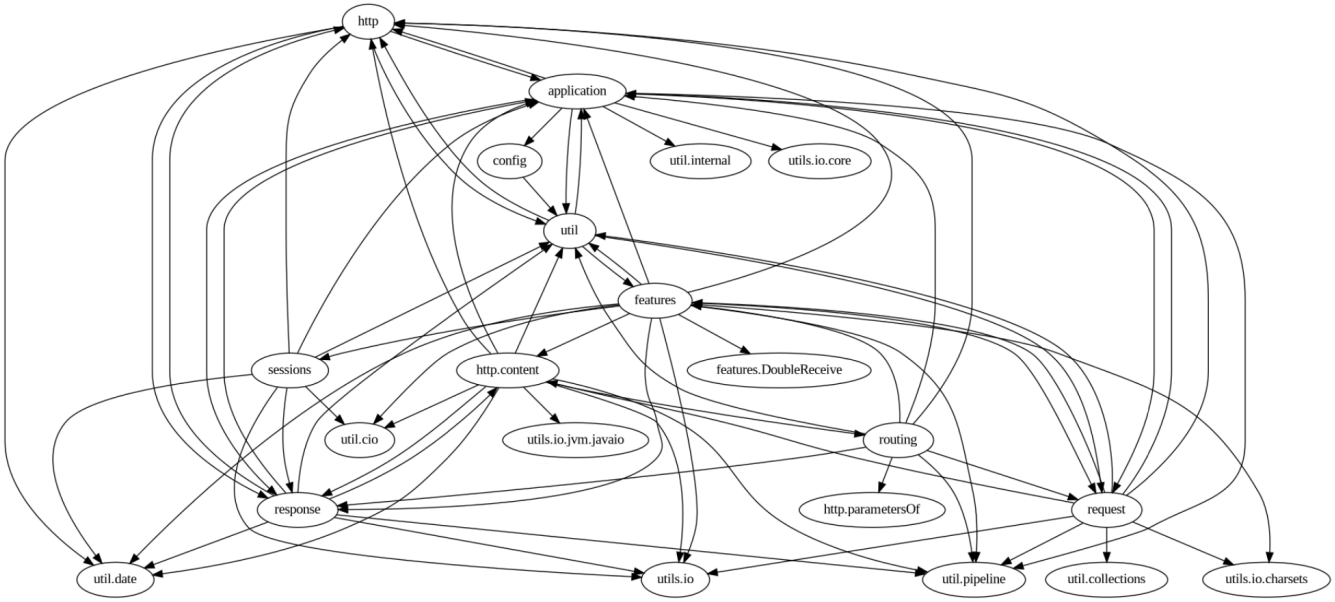
Intermediate step

- You do not have space for that
- Also… Nobody wants to read that

Not bad - But how to improve?



Not bad - But how to improve?



Filter out `util`, almost everybody depends on it!

Make view *polymetric*:

- Color intensity on nodes
- Depedency width

Not good - why?

To register a student, the admin should provide the required information as in Figure 6.

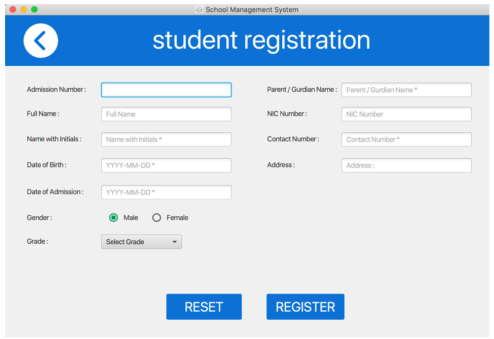


Figure 6. Student registration

As for managing the students, the admin can search a student by providing his ID and then he can update, delete or print that information.

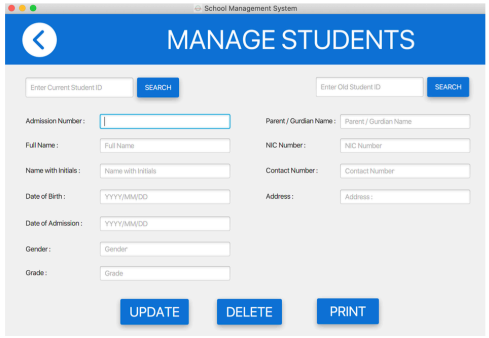


Figure 7. Manage students

On the other hand, 'Print Students' option has the same view as the view that appears when logged in as a teacher and supports the same functionalities.

Not good - why?

To register a student, the admin should provide the required information as in Figure 6.

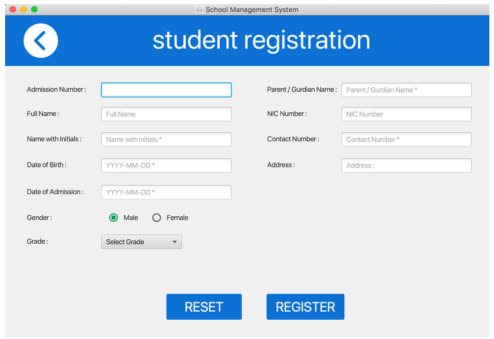


Figure 6. Student registration

As for managing the students, the admin can search a student by providing his ID and then he can update, delete or print that information.

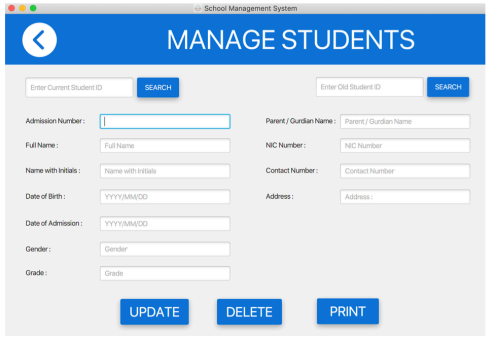


Figure 7. Manage students

On the other hand, 'Print Students' option has the same view as the view that appears when logged in as a teacher and supports the same functionalities.

Do not waste time on showing me screenshots of your system! Summarize the system in a paragraph or two at the beginning

Not very good - why?

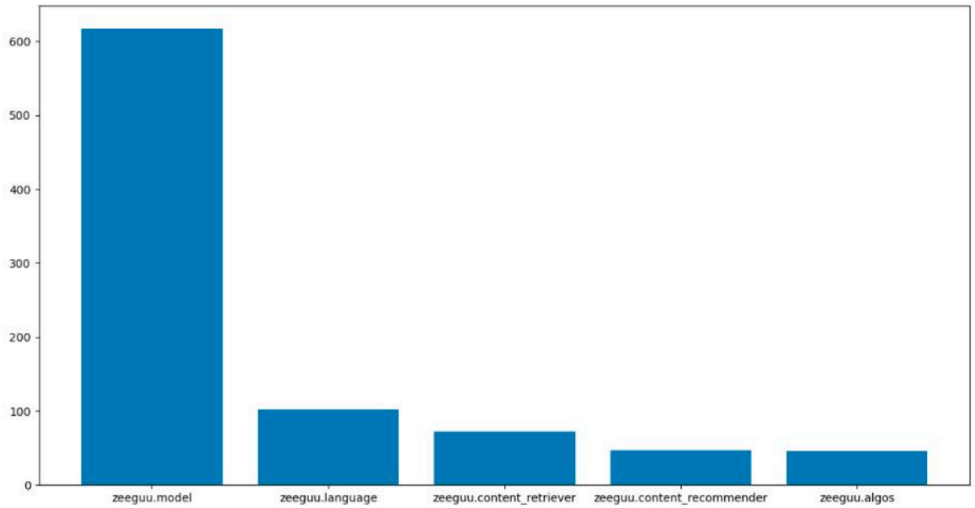


Figure 10: *A viewpoint that highlights the top 5 most active modules in the Zeeguu Core domain*

Not very good - why?

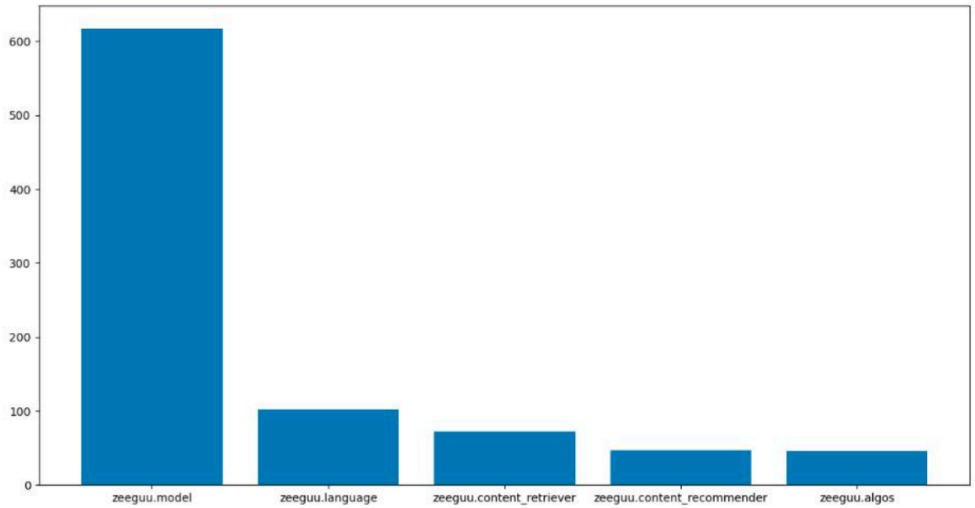
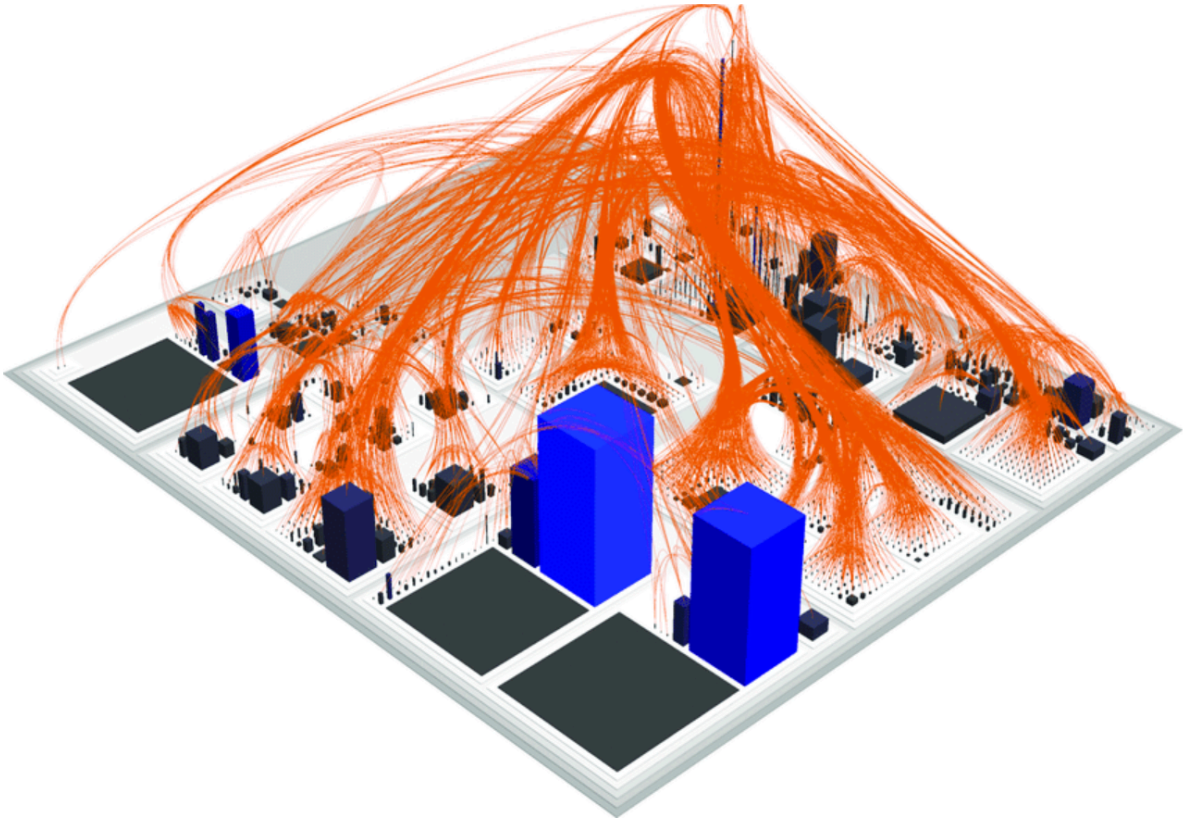


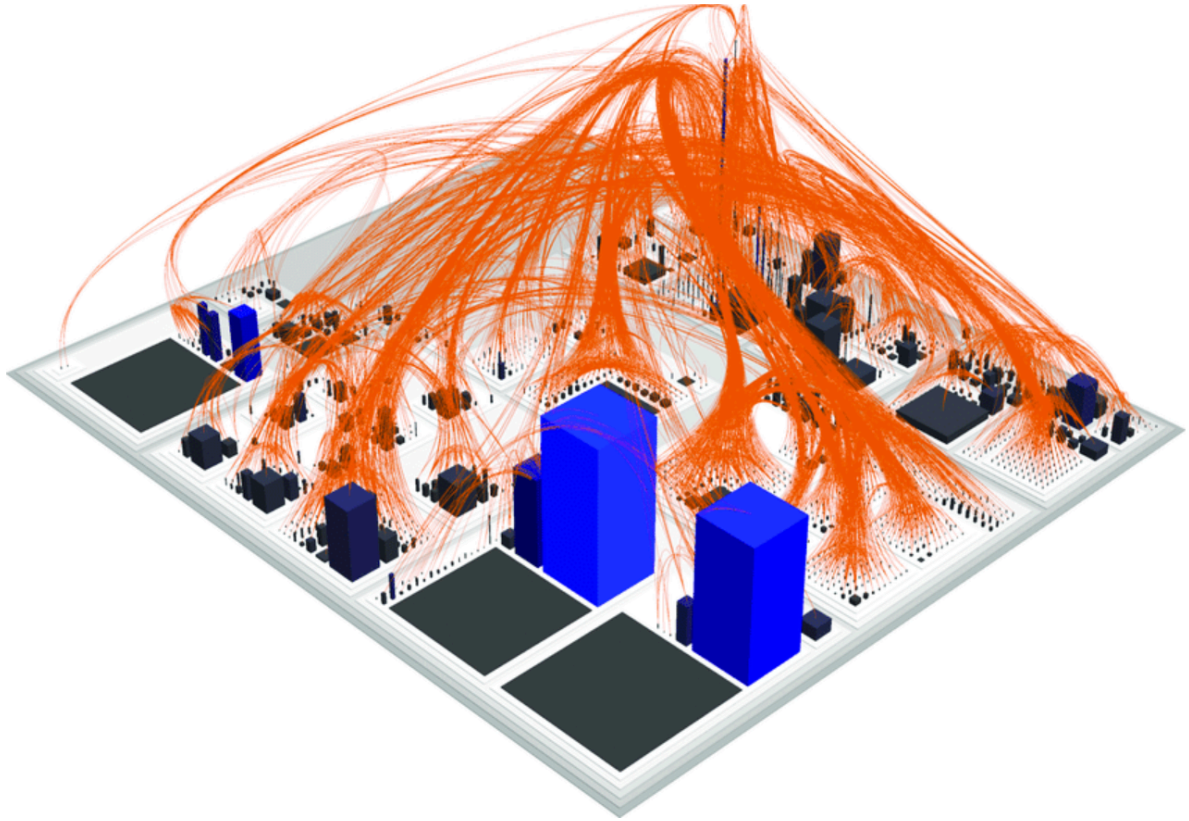
Figure 10: *A viewpoint that highlights the top 5 most active modules in the Zeeguu Core domain*

- Too little information (small data/ink ratio)
- Not very readable (compare font in the image with the font in the caption)

Not very good - why?



Not very good - why?



Lack of abstraction

Good...but can we improve?

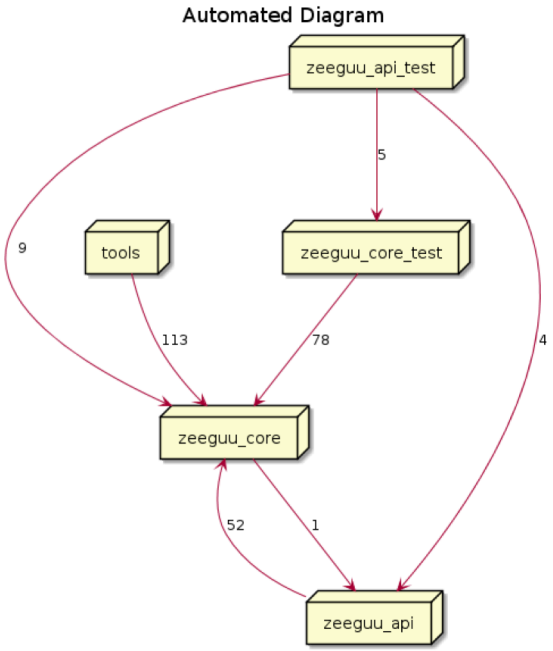


Figure 3: Resulting graph

Good...but can we improve?

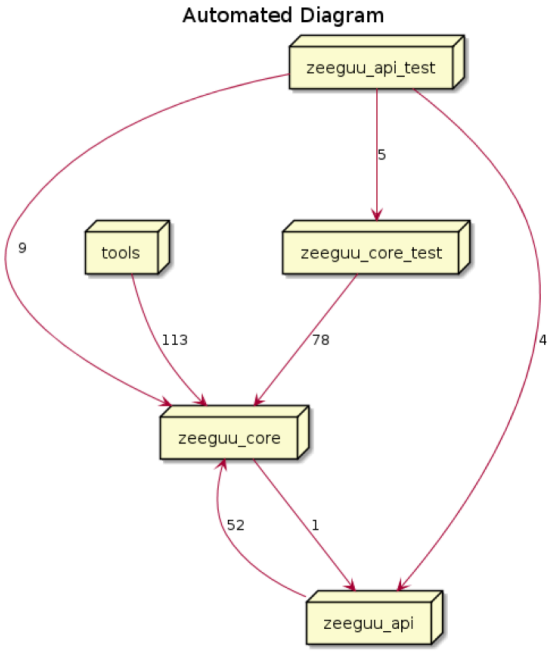


Figure 3: Resulting graph

Caption is bad (explain the figure in a concise manner)

Good...but can we improve?

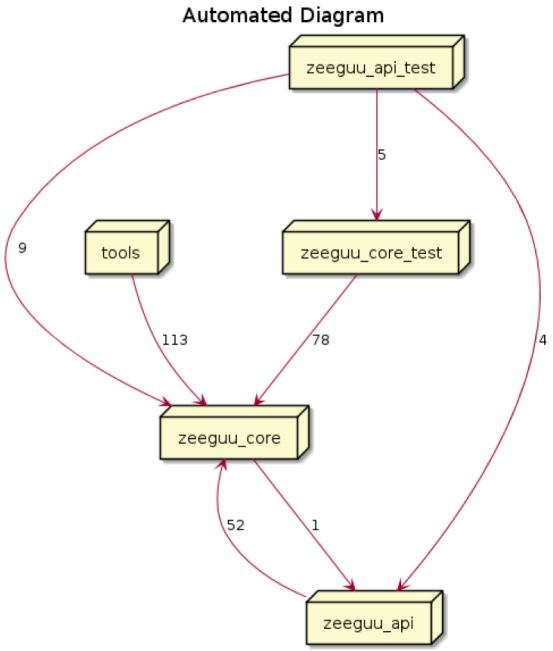


Figure 3: Resulting graph

Caption is bad (explain the figure in a concise manner)

More info about the

- Size of the modules
- Details of most important modules in separate views

Final Words on the Architecture Reconstruction

Advanced topics are still left for ASE

1. Dynamic analysis (extract info from the runtime of the system!)
2. Advanced fact extraction (parsing ASTs)
3. Ecosystem dependency analysis

Final Words on the Architecture Reconstruction

Advanced topics are still left for ASE

1. Dynamic analysis (extract info from the runtime of the system!)
2. Advanced fact extraction (parsing ASTs)
3. Ecosystem dependency analysis

Thesis Topics in this area (github.com/mircealungu/student-projects/)

1. Specifying diagrams in code
2. Integrating diagrams in CI/CD & the IDE
3. Applying ML and NLP techniques
4. Understanding the ecosytem dependencies

Final Words on the Individual Reports

See report description

Remember

- Not more than 1.5K words
- Quality over quantity
- Do not teach back; assume that the reader knows the course materials
- Consider creating GitHub repo & moving away from Collab