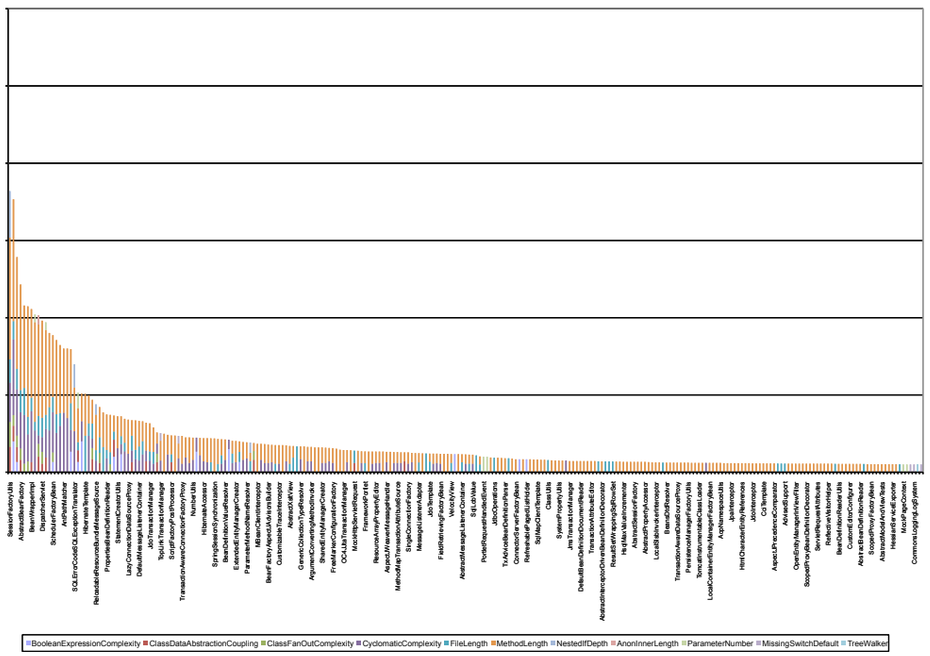
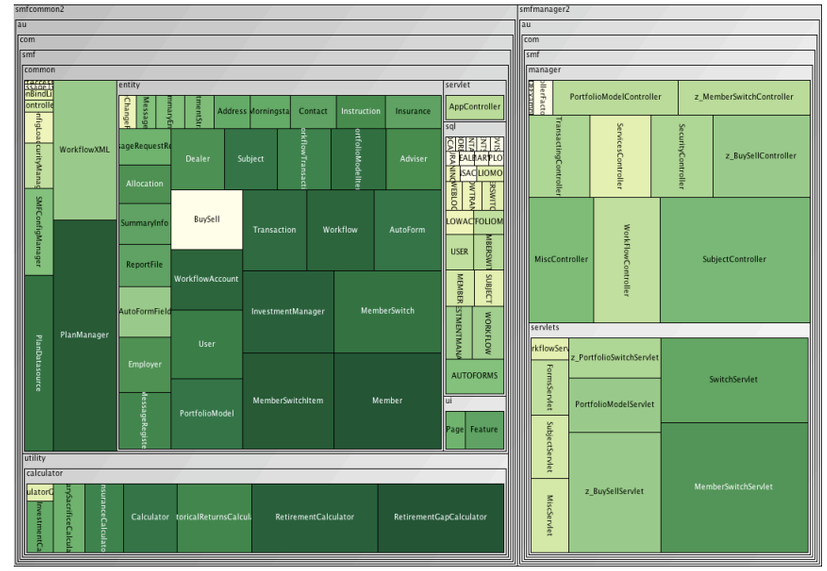
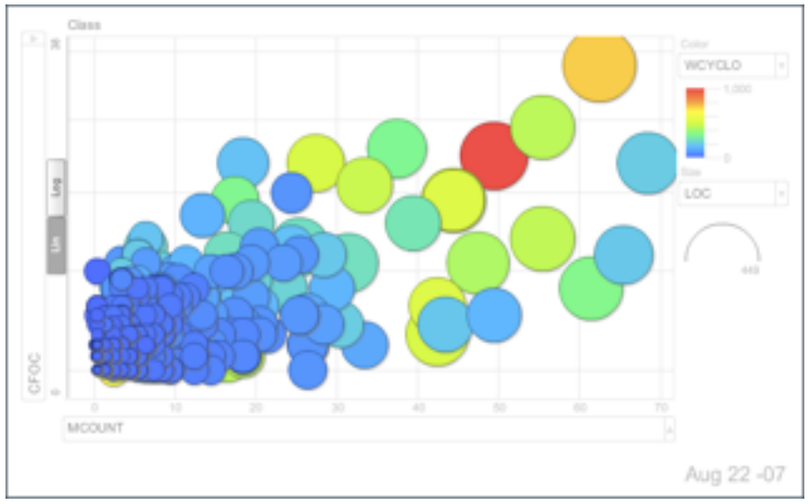
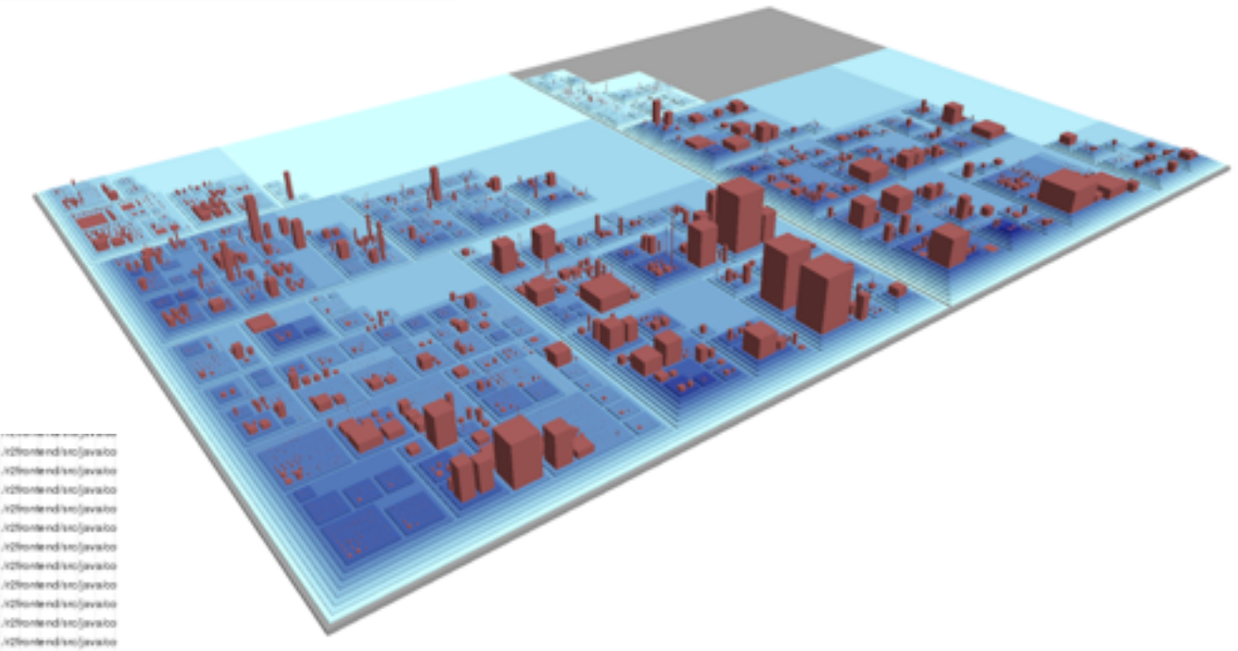
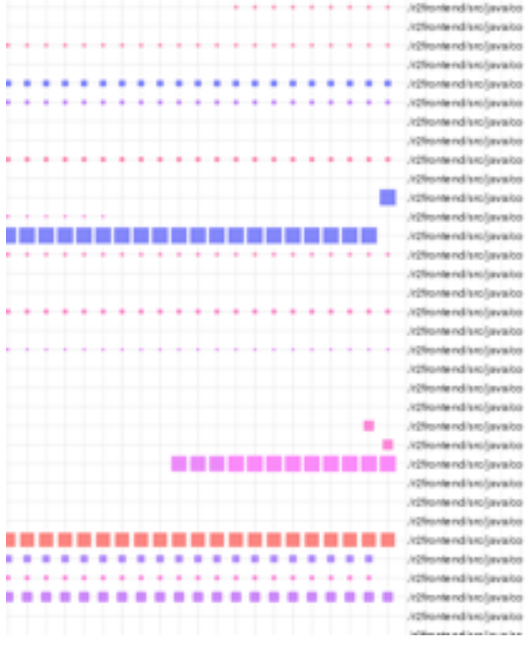
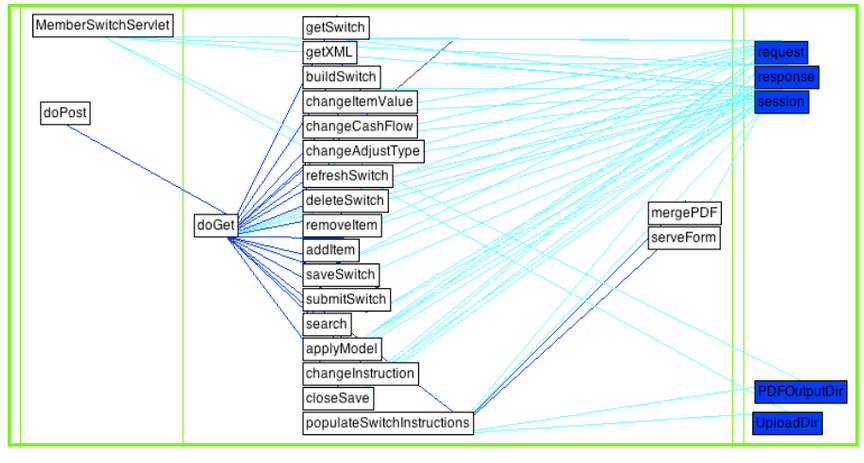




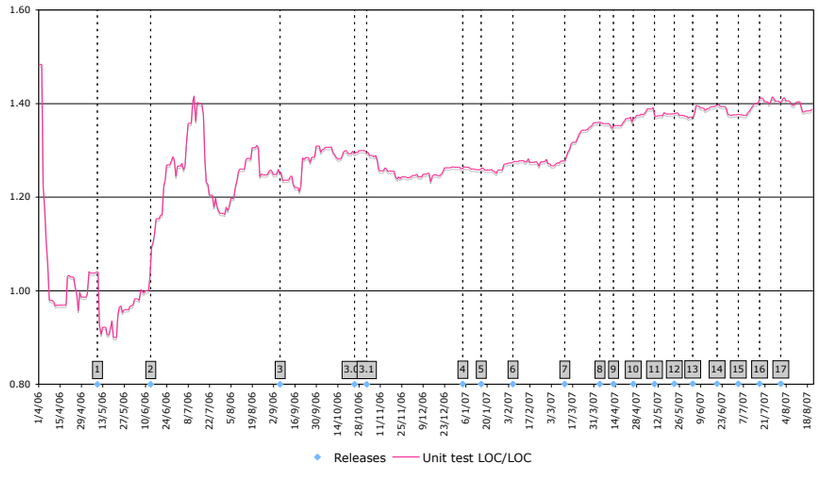








Lines of unit test code per line of production code



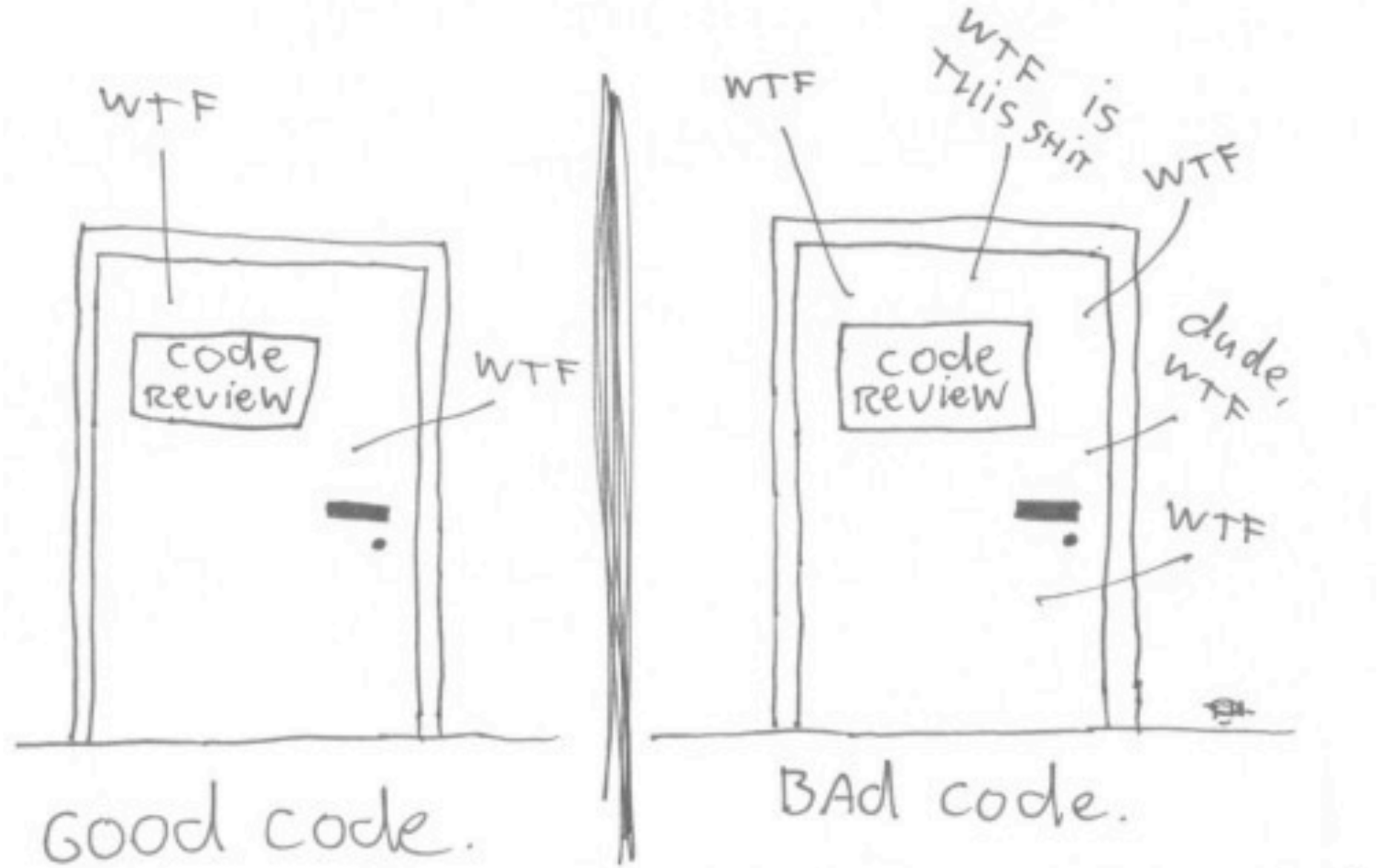
ThoughtWorks®

Software Quality –
you know it when you see it

Erik Doernenburg
ThoughtWorks

@erikdoe // erik.doernenburg.com

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

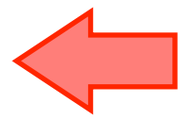
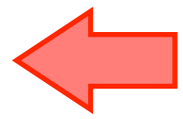


Software Quality

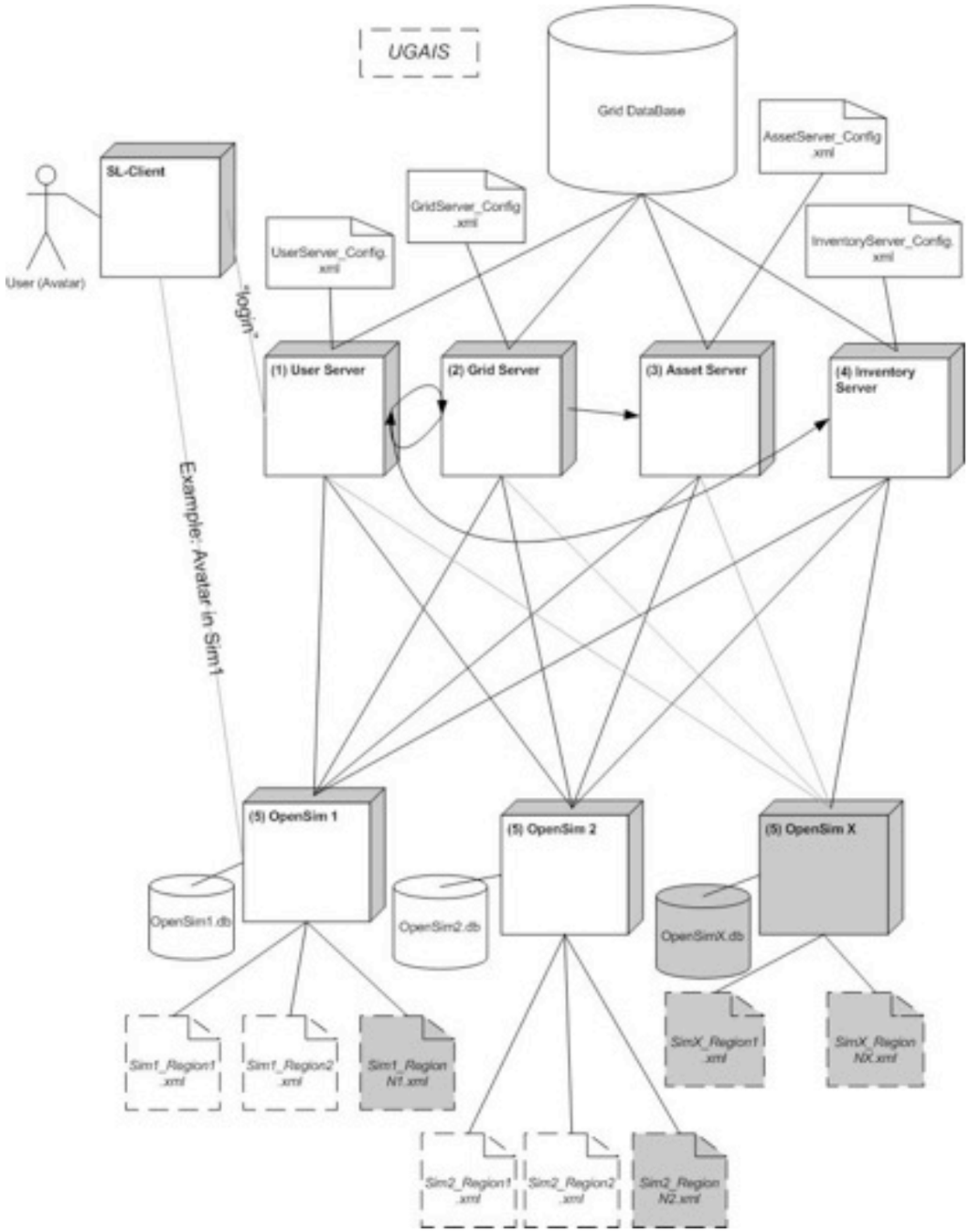
External perspective

- Is the software of value to its users?

Internal perspective

- How appropriate is the design?
- How easy is it to understand and extend? 
- How maintainable is the software? 

30.000ft and ground level




```
public void mergePluginOutput(BuildDetail build, Map paramete
    Iterator iterator = lines().iterator();
    while (iterator.hasNext()) {
        try {
            assemblePlugin(build, parameters, (String) iterat
        } catch (Exception e) {
            logger.error(e);
            continue;
        }
    }
}

void assemblePlugin(BuildDetail build, Map parameters, String
    String className = line.trim();
    if (className.startsWith("#") || StringUtils.isEmpty(clas
        return;
    }
    Class clazz = Class.forName(className);
    Widget digesterService = (Widget) clazz.newInstance();
    mergeParameters(build, parameters);
    build.addPluginOutput(digesterService.getDisplayName(), c
            .getOutput(parameters));
}

private void mergeParameters(BuildDetail build, Map parameter
    parameters.put(Widget.PARAM_CC_ROOT, configuration.getCCH
```

http://opensimulator.org/wiki/Grid_Architecture_Diagram

The 1000ft view

- Is at the “right” level
- Aggregates data and metrics
- Utilises visualisation techniques
- Makes every pixel count

http://97things.oreilly.com/wiki/index.php/Get_the_1000ft_view

Metrics

- lines of code
- method length
- class size
- cyclomatic complexity
- weighted methods per class
- coupling between (object) classes

ckjm

http://www.campwoodsw.com/sourcemonitor.html

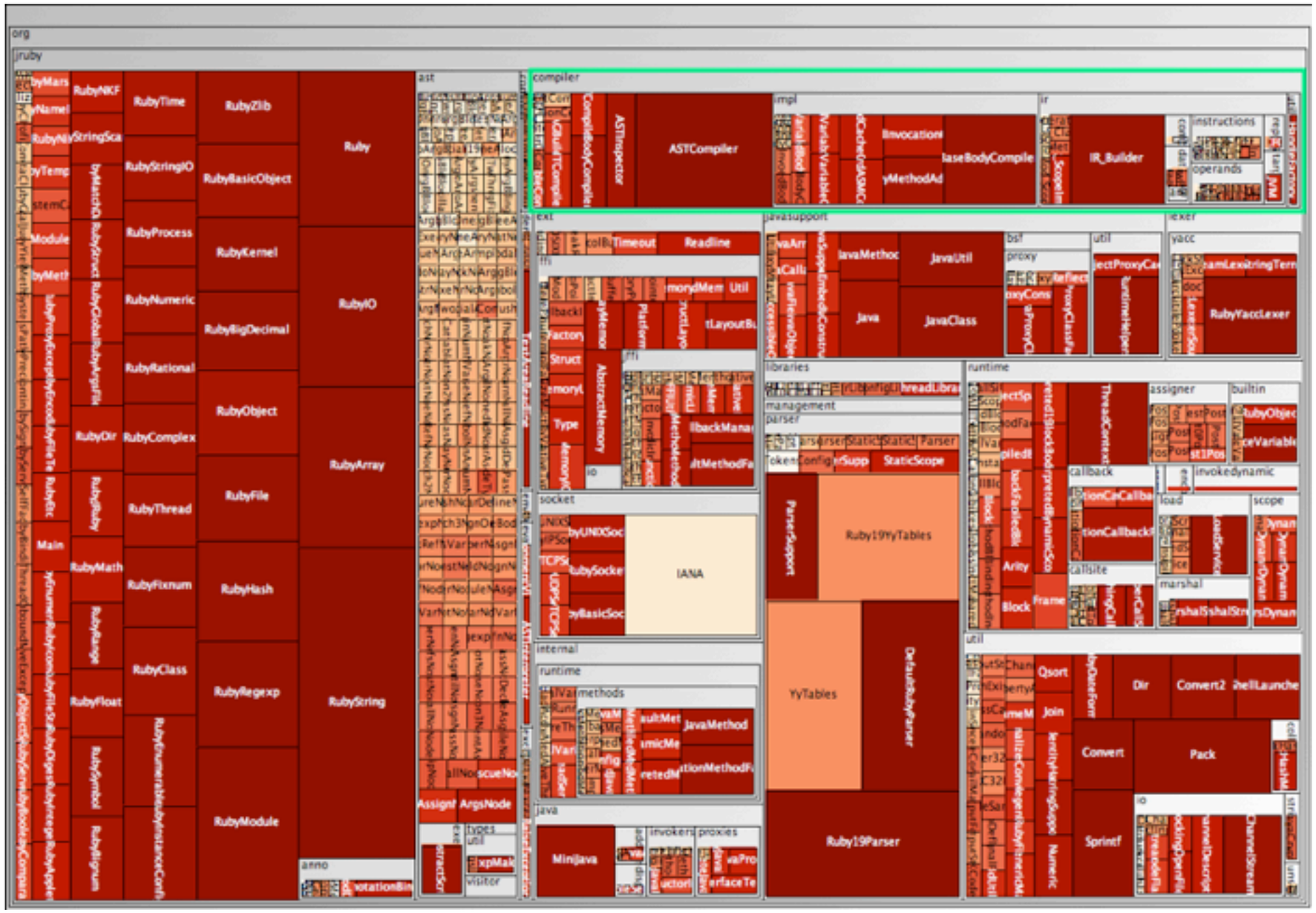
More metrics

- duplication
- check-in counts
- coverage
- testability
- test/code ratio

simian

http://www.redhillconsulting.com.au/products/simian/

Metrics tree maps

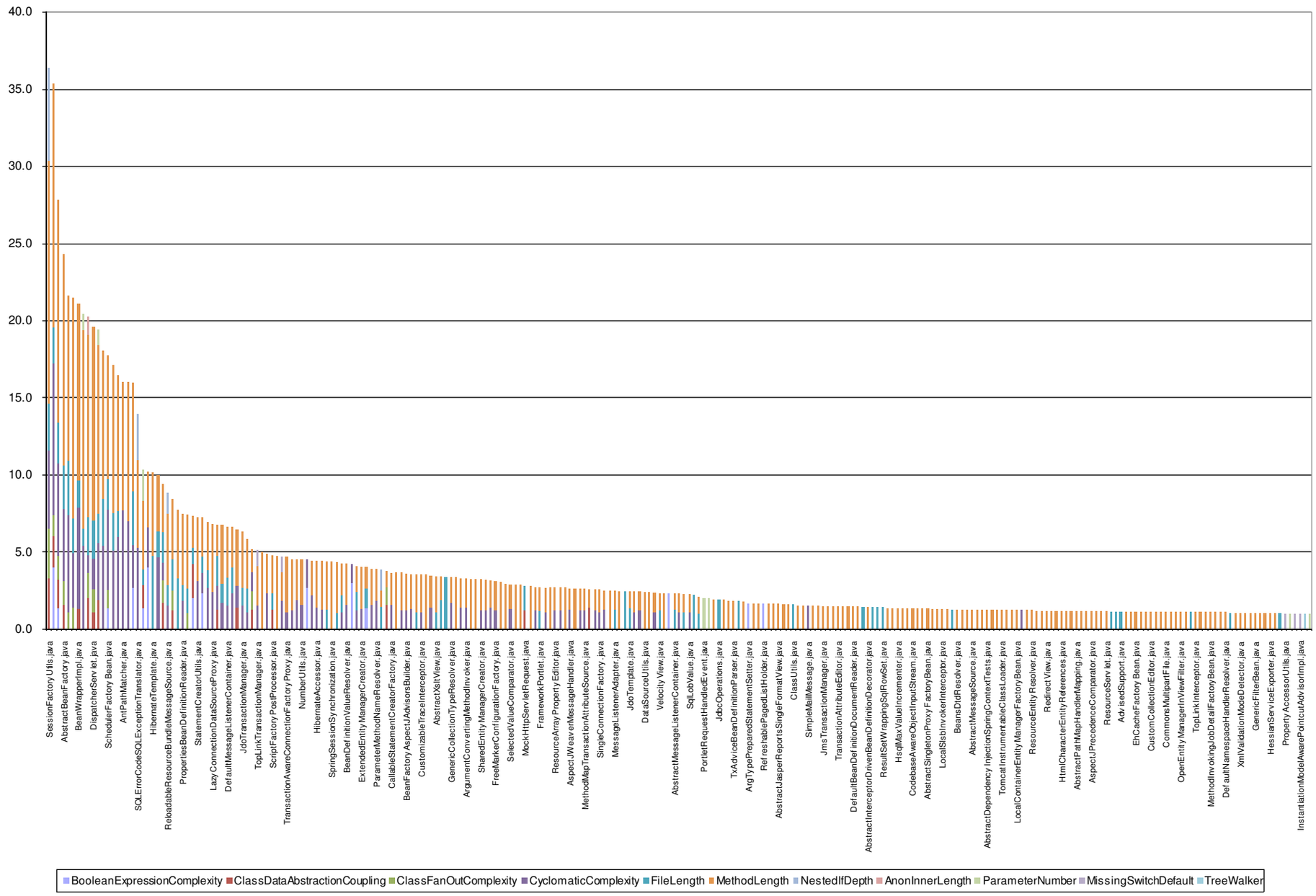


Shows distribution of metrics

Created with checkstyle and InfoVis

http://erik.doernenburg.com/2010/05/metrics-tree-maps/

Toxicity chart

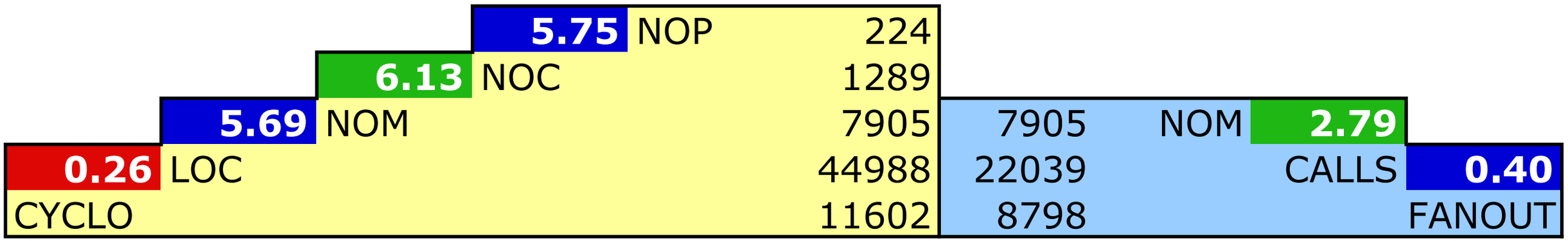


Provides easy to compare overview of quality Developed by ThoughtWorks using Excel

http://erik.doernenburg.com/2008/11/how-toxic-is-your-code/

Size & complexity pyramid

	Low	Medium	High
CYCLO / Line	0.16	0.20	0.24
LOC / method	7	10	13
NOM / class	4	7	10
NOC / package	6	17	26
CALLS / method	2.01	2.62	3.20
FANOUT / call	0.56	0.62	0.68



Developed at Universities of Berne and Lugano

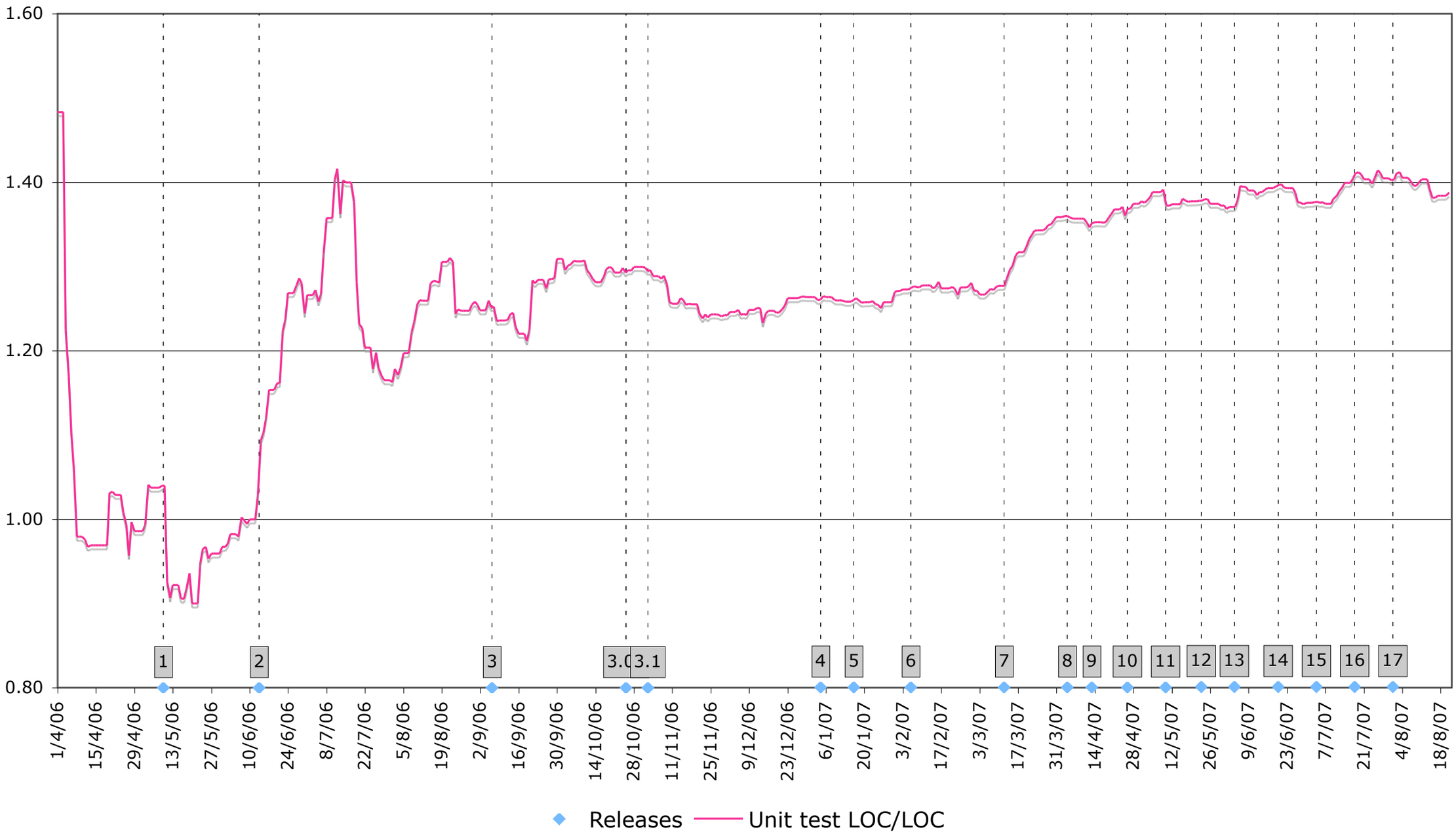
Shows key metrics and their relationships

Allows comparison to “industry standards”

Created by iPlasma tool from source code

http://loose.upt.ro/iplasma/
http://www.intooitus.com/inFusion.html

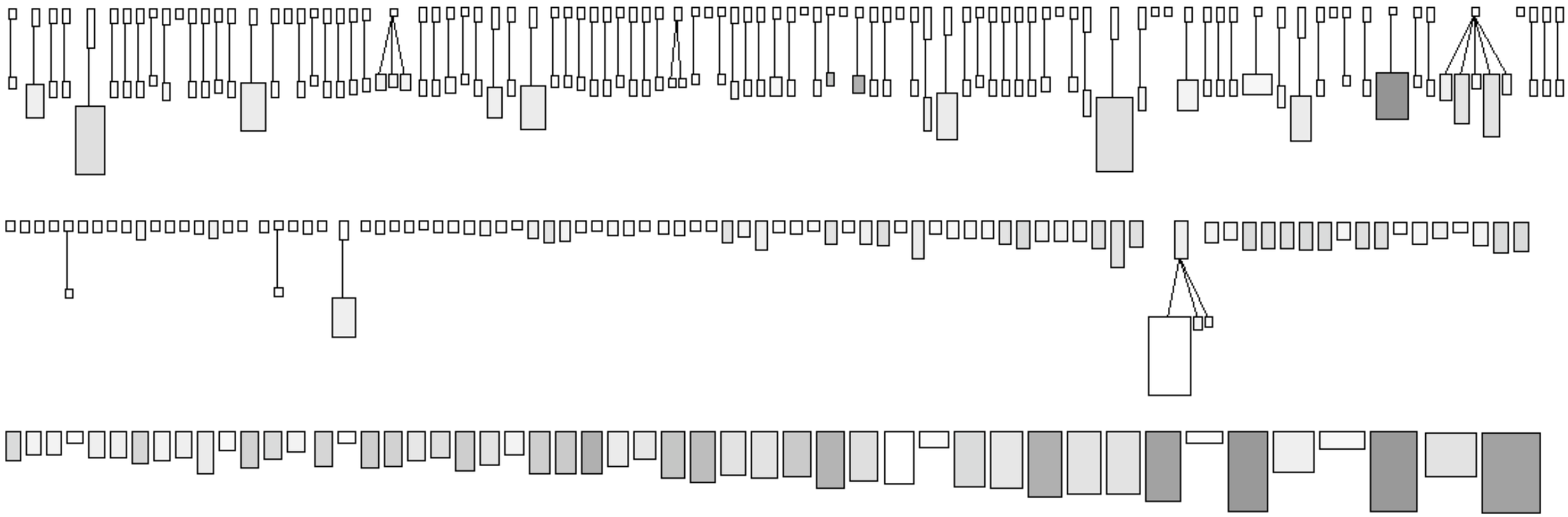
Test to code ratio

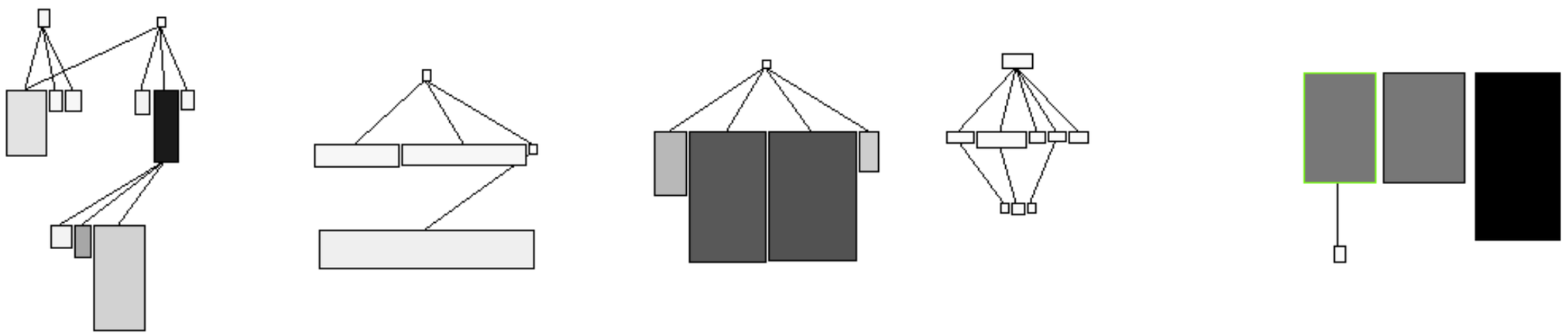


Shows the test to code ratio over time

Created with Unix tools and Excel

System complexity view



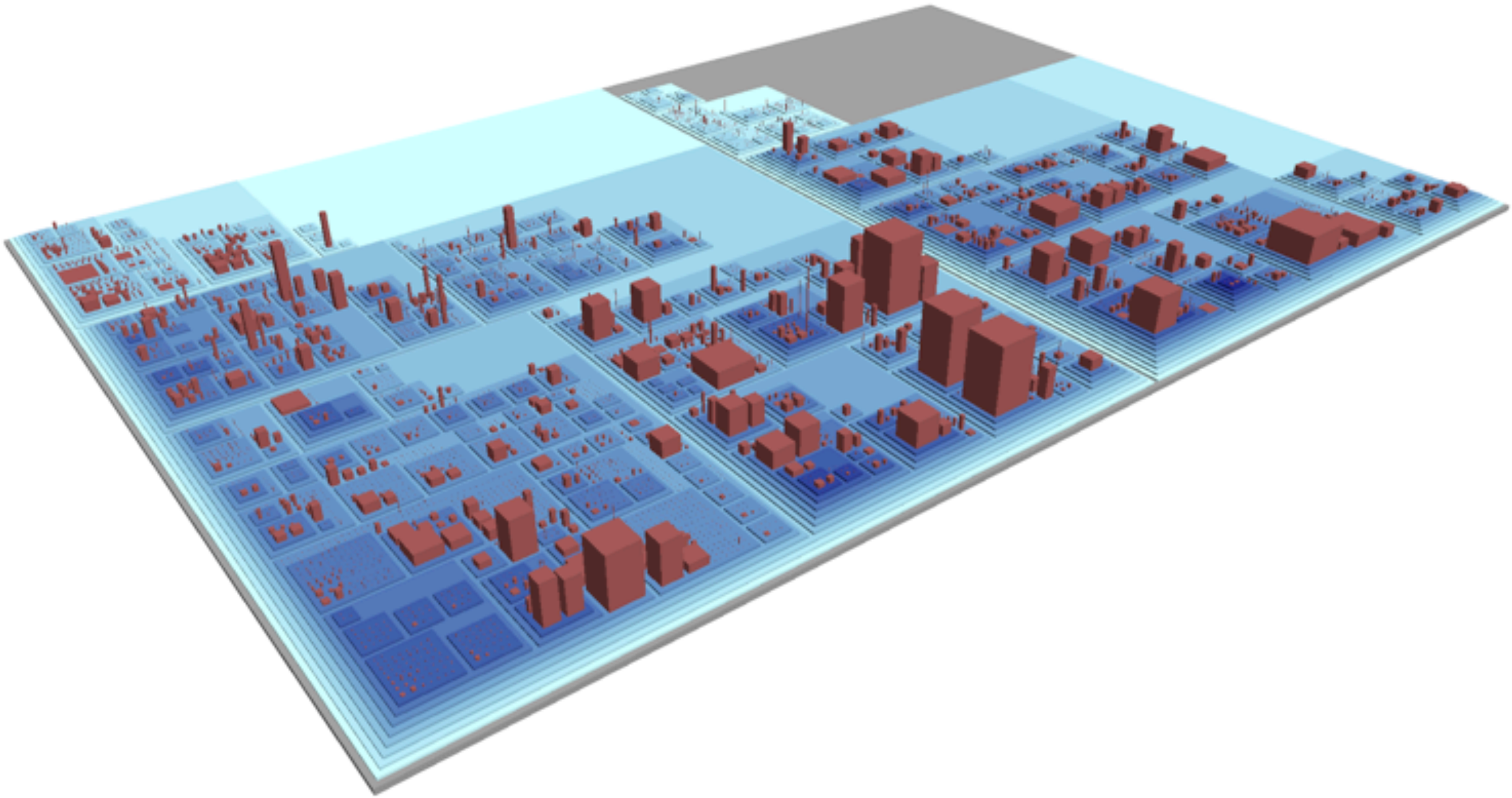


Part of Moose Technology

Polymetric view of class hierarchy

http://www.moosetechnology.org/
http://www.inf.usi.ch/faculty/lanza/codecrawler.html
http://sourceforge.net/projects/java2cdif/

CodeCity

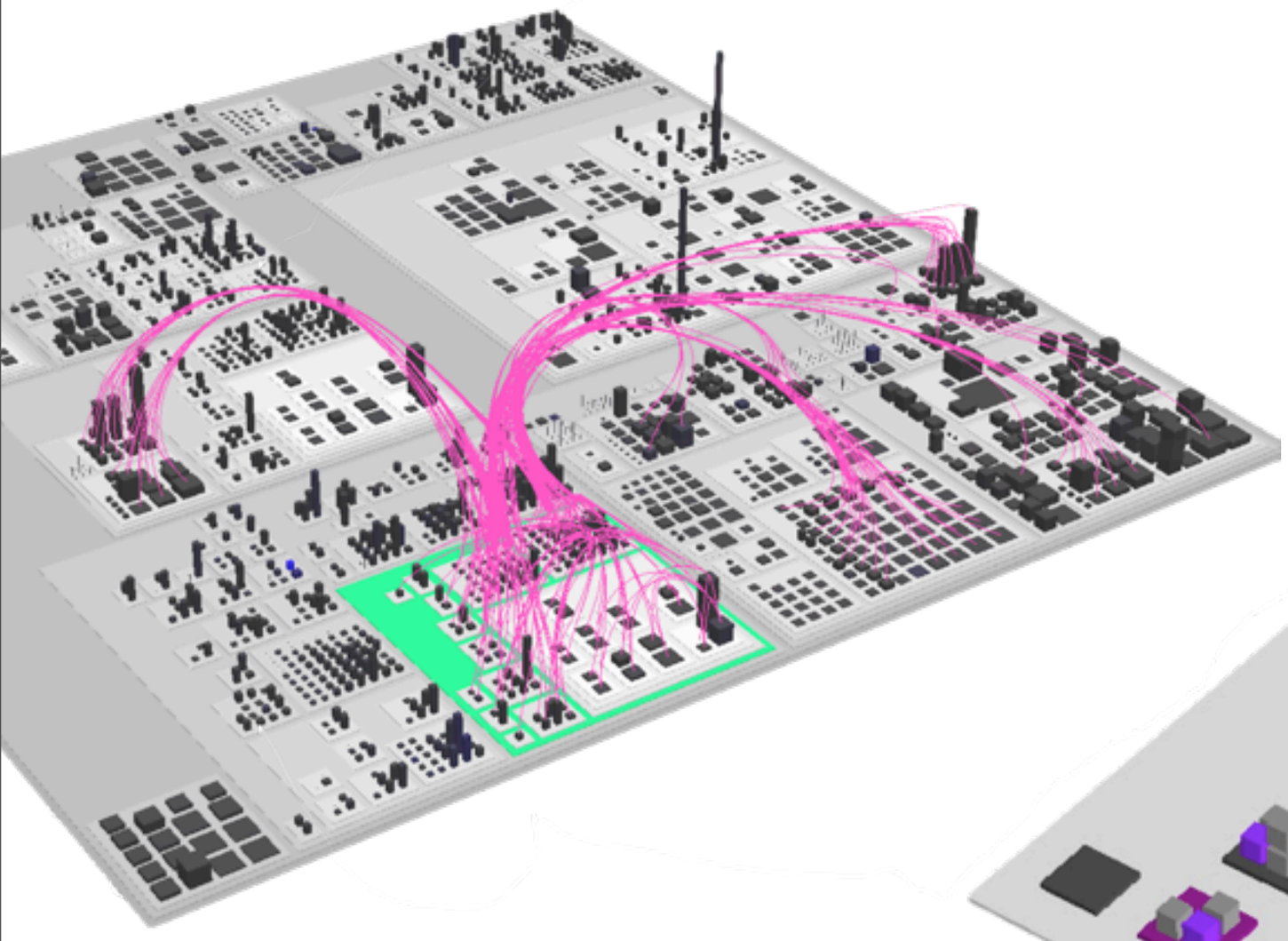


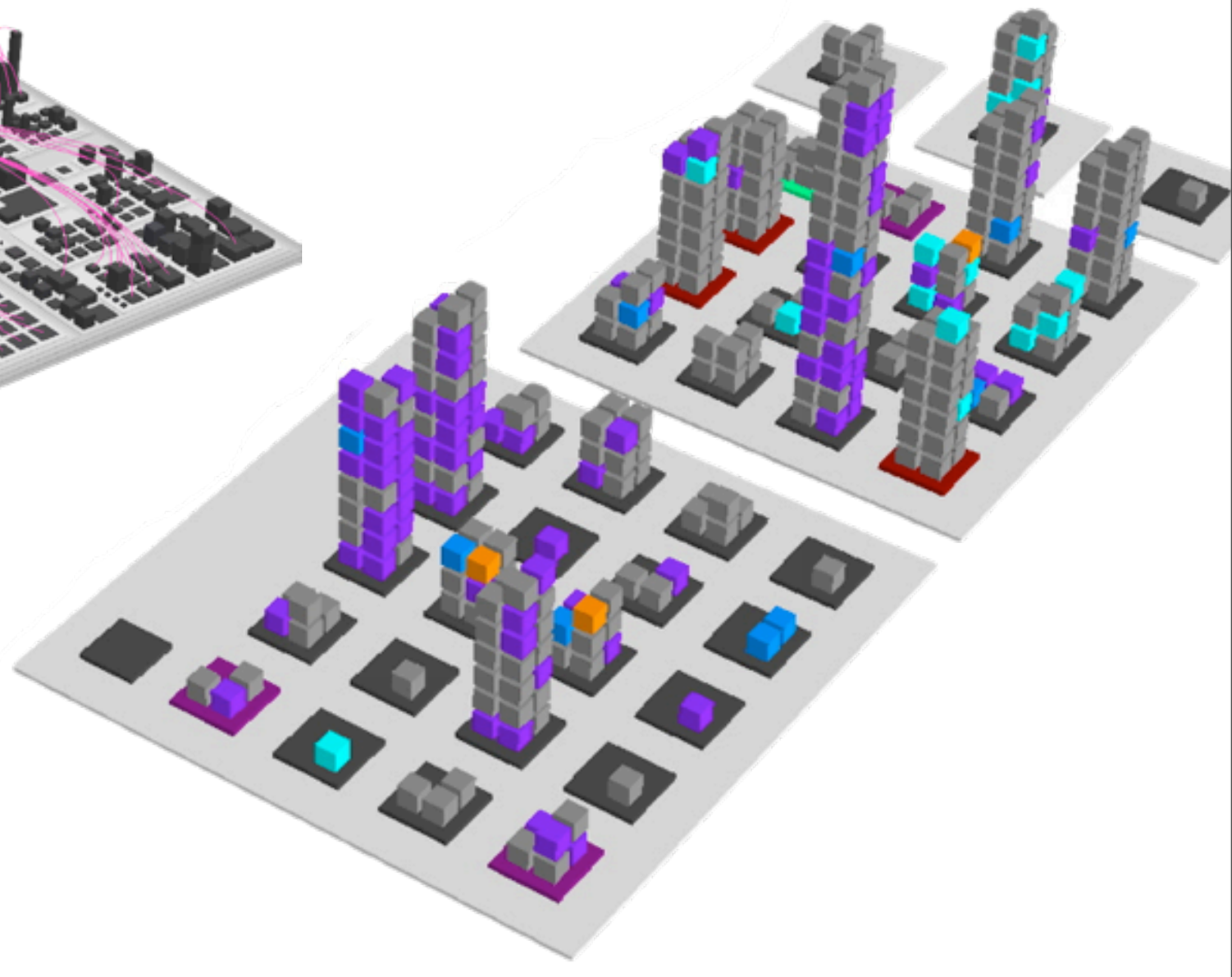
Also part of Moose Technology

Polymetric views in 3D and more

http://www.inf.usi.ch/phd/wettel/codecity-download.html
http://erik.doernenburg.com/2009/07/moose-mse-for-java-and-cs/

CodeCity continued

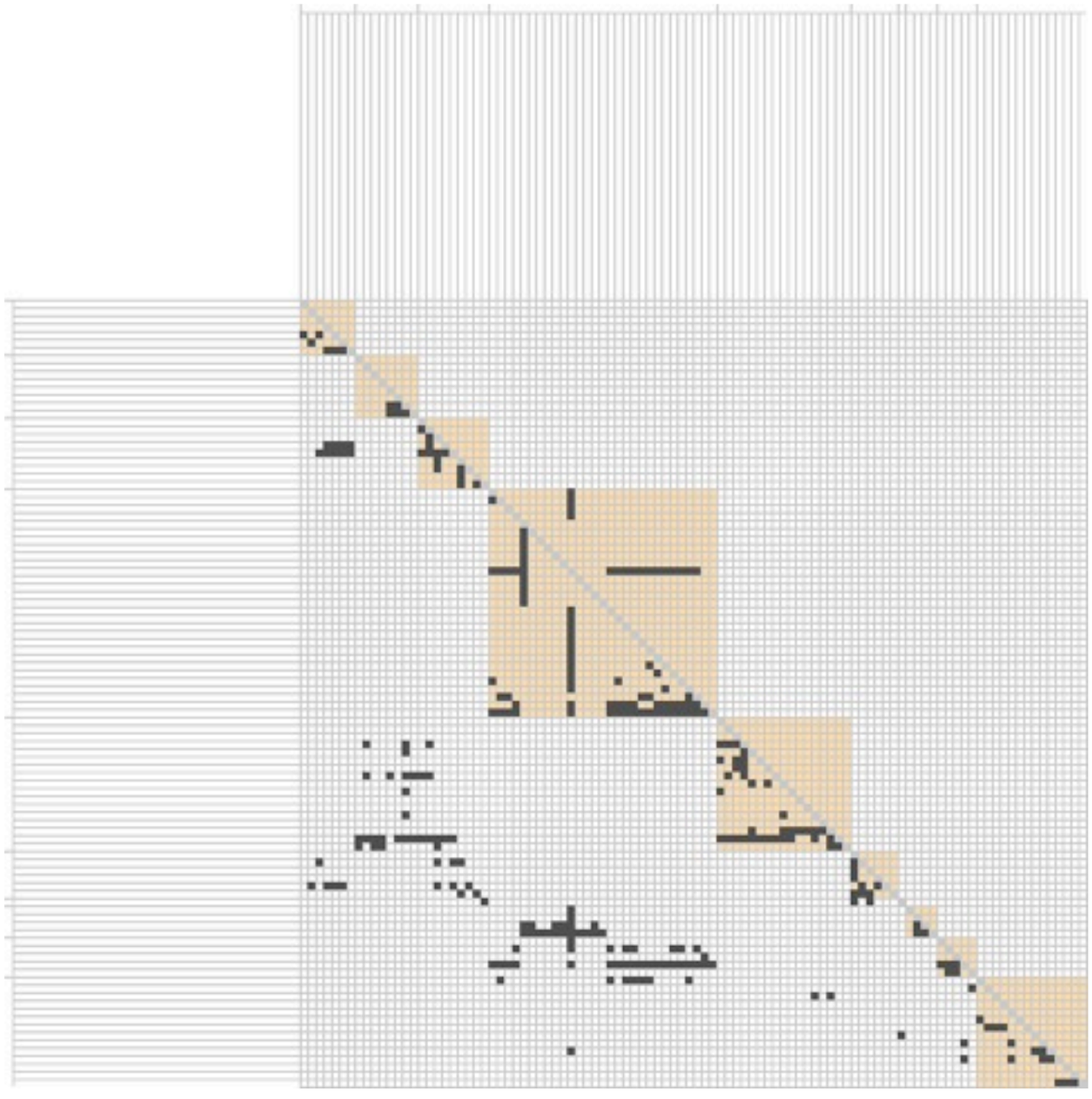




⚠ Might need a license for VisualWorks Smalltalk

Dependency Structure Matrix

		.beans												.core							util
		access	annotation	xml	parsing	support	wiring	serviceloader	config	support	annotation	factory	beans	propertyeditors	style	task	enums	type	io	annotation	core	util
.beans	access																					
	annotation																					
	xml	1																				
	parsing			24																		
	support	2	11	31																		
	wiring		2																			
	serviceloader																					
	config	2	12	30	7	60	1	1														
	support								1													
	annotation																					
	factory	7	8	6	1	75	6	4	60													
	beans	5	12	10	7	54			50	8	2	14										
	propertyeditors									5			19									
.core	style																					
	task																					
	enums																					
	type		3																			
	io	4		18	7	11			2	5			2	7				6				
	annotation		2			2												1				
	core		6	4		10			10			1	17						1	3		
	util	1	12	22	8	44	5	2	34	5	1	3	25	23	6	7	8	4	34	1	20	

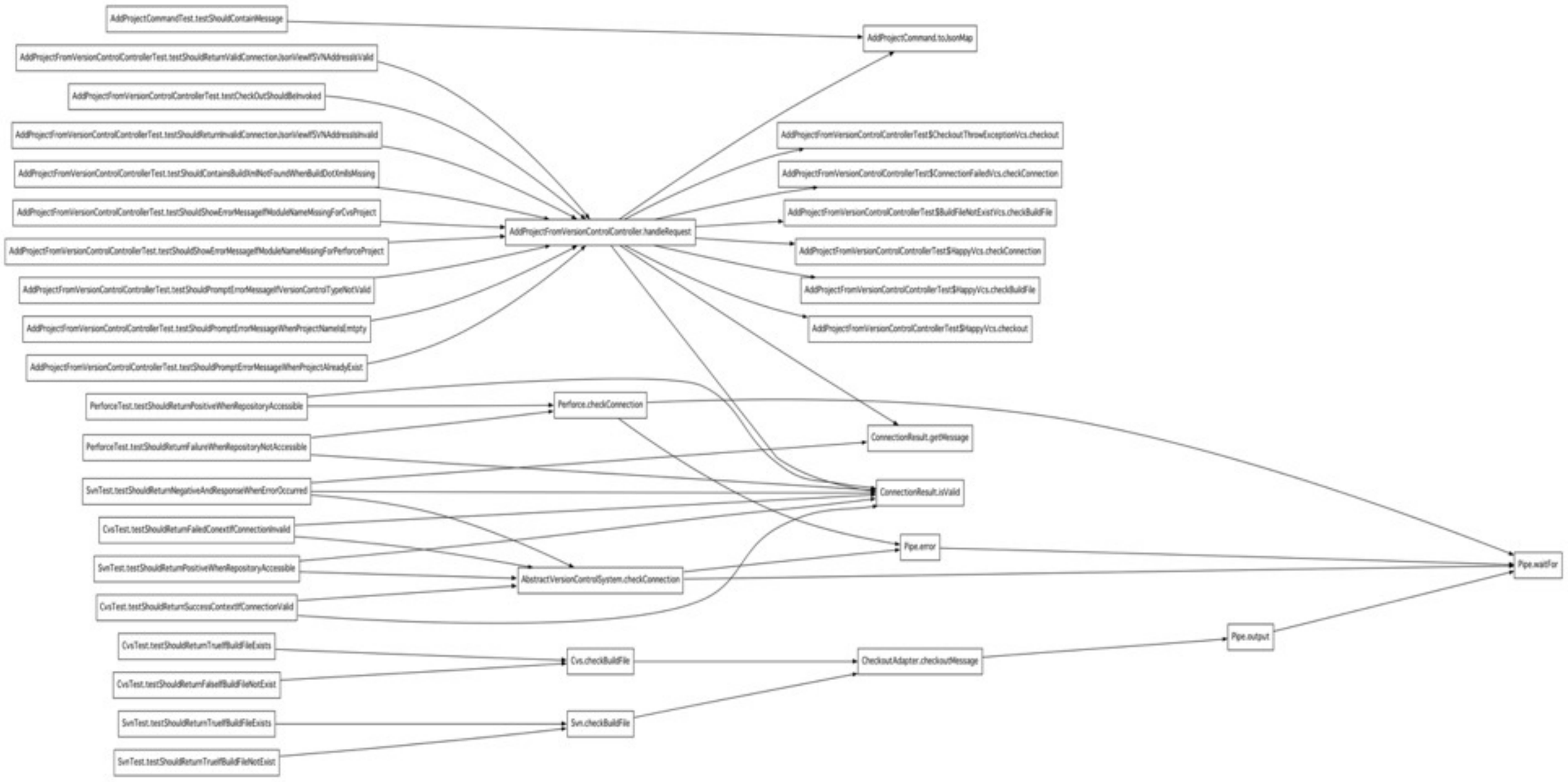


Not metrics based

Looks at “complex complexity”

http://erik.doernenburg.com/2010/04/dependency-structure-matrix/

Test coverage



Not as a percentage, shows call graph

AspectJ + GraphViz

http://erik.doernenburg.com/2008/09/call-graph-visualisation-with-aspectj-and-dot/

DIY

1. **Get metrics**
    - SourceMonitor, checkstyle, text tools, etc
    - iPlasma, et al
2. **Aggregate data**
    - Ruby scripts, unix tools, etc
    - VBA and pivot tables
3. **Render graphics**
    - Excel is a powerful graphing tool
    - Gnuplot and InfoViz are easy to use

How do you see quality?

Comparisons

- industry standards
- different revisions: trends
- different parts: outliers

Aesthetics

- symmetry
- balance/harmony

What next?

Measure

- tech debt
- effectiveness of training

Guide

- refactoring
- clean-up

Celebrate

Thank you

http://erik.doernenburg.com/topics/softviz

http://97things.oreilly.com/wiki/index.php/Get_the_1000ft_view

http://www.moosetechnology.org

ThoughtWorks®