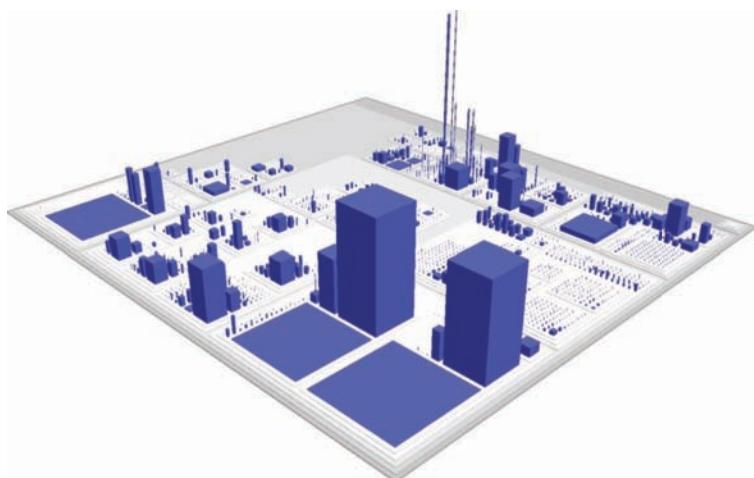


visualizations for code metrics



NEAL FORD software architect / meme wrangler

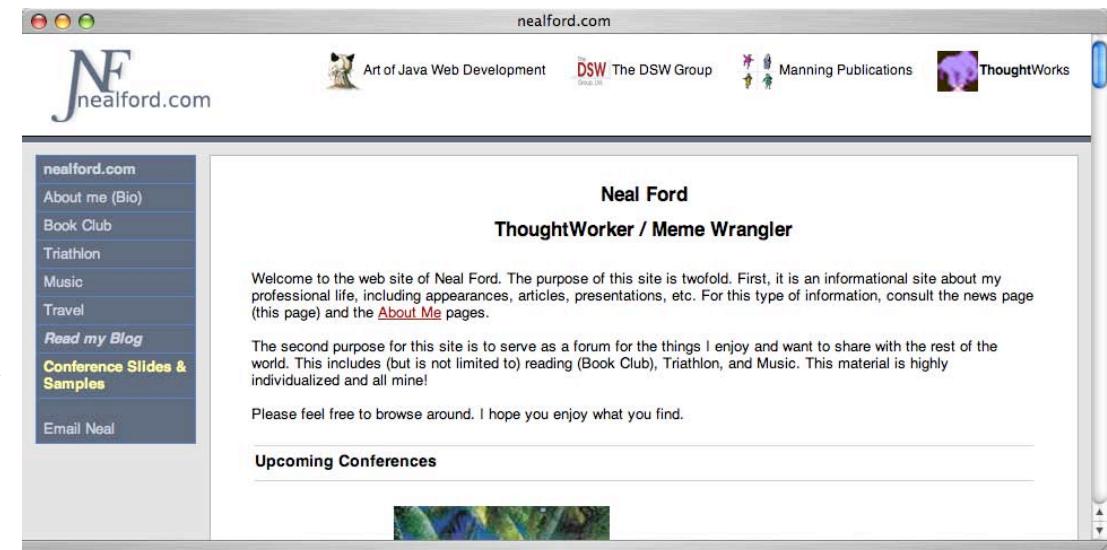
ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
blog: memeagora.blogspot.com
twitter: [@neal4d](https://twitter.com/neal4d)

housekeeping

ask questions anytime

download slides from
nealford.com



download samples from github.com/nealford

what was *that*?

code_swarm

<http://code.google.com/p/codeswarm/>

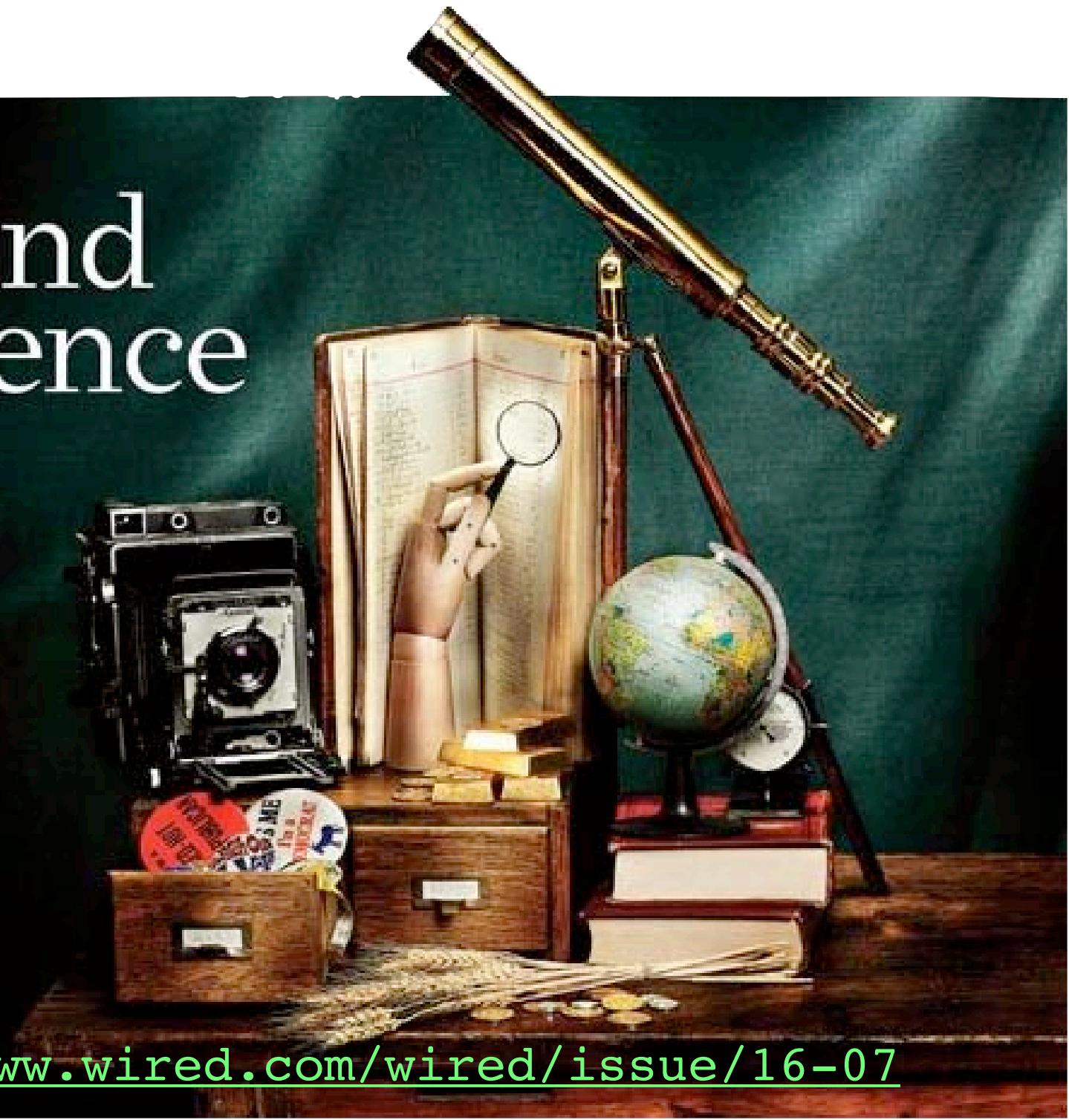
point it to a subversion repository

visualization of check-ins over time

useful? cool!

The End of Science

The quest for knowledge used to begin with grand theories. Now it begins with massive amounts of data. Welcome to the Petabyte Age.



<http://www.wired.com/wired/issue/16-07>

1 TERABYTE

A \$200 HARD DRIVE
THAT HOLDS
260,000 SONGS.

20 TERABYTE

PHOTOS UPLOADED TO
FACEBOOK EACH MONTH

120 TERABYTE

ALL THE DATA
AND IMAGES
COLLECTED BY
THE HUBBLE
SPACE TELESCOPE.

330 TERABYTE

DATA THAT
THE LARGE HADRON
COLLIDER WILL
PRODUCE EACH WEEK.

460 TERABYTE

ALL THE DIGITAL
WEATHER
DATA COMPILED
BY THE NATIONAL
CLIMATIC DATA
CENTER.

530 TERABYTE

ALL THE VIDEOS
ON YOUTUBE.

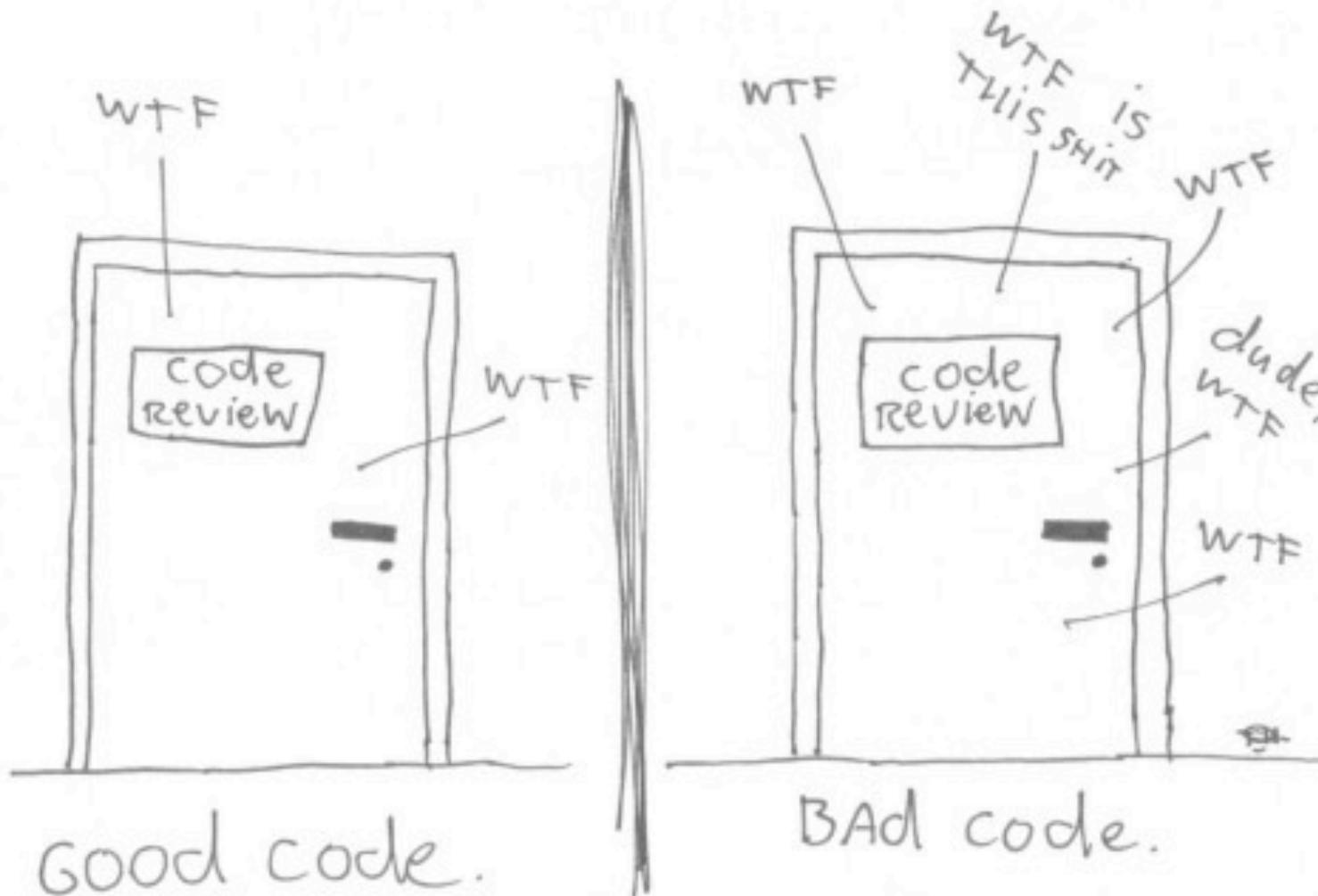
600 TERABYTE

ANCESTRY.COM'S
GENEALOGY
DATABASE (INCLUDES
ALL U.S. CENSUS
RECORDS 1790-2000).

1 PETABYTE

DATA PROCESSED
BY GOOGLE'S
SERVING EVERY
72 MINUTES.

The ONLY VALID MEASUREMENT OF Code QUALITY: WTFs/MINUTE



The background image shows the exterior of a Gothic cathedral, likely Cologne Cathedral, featuring its signature pointed arches, flying buttresses, and detailed stone carvings. The sky is clear and blue.

external perspective

is the software valuable to its users?

internal perspective

how appropriate is the design?

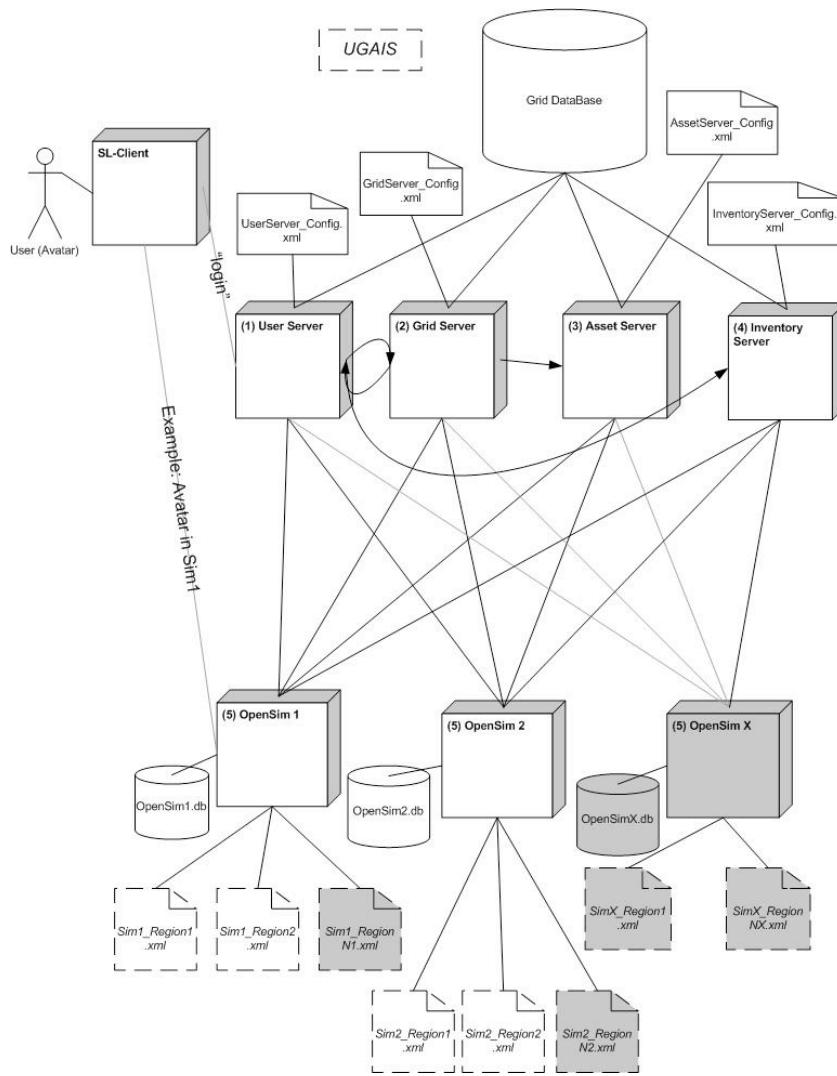
how amenable is emergent design?

how easy is it to understand & extend?

how maintainable is it?

is it salvageable?

30,000 feet



http://opensimulator.org/wiki/Grid_Architecture_Diagram

ground level

```

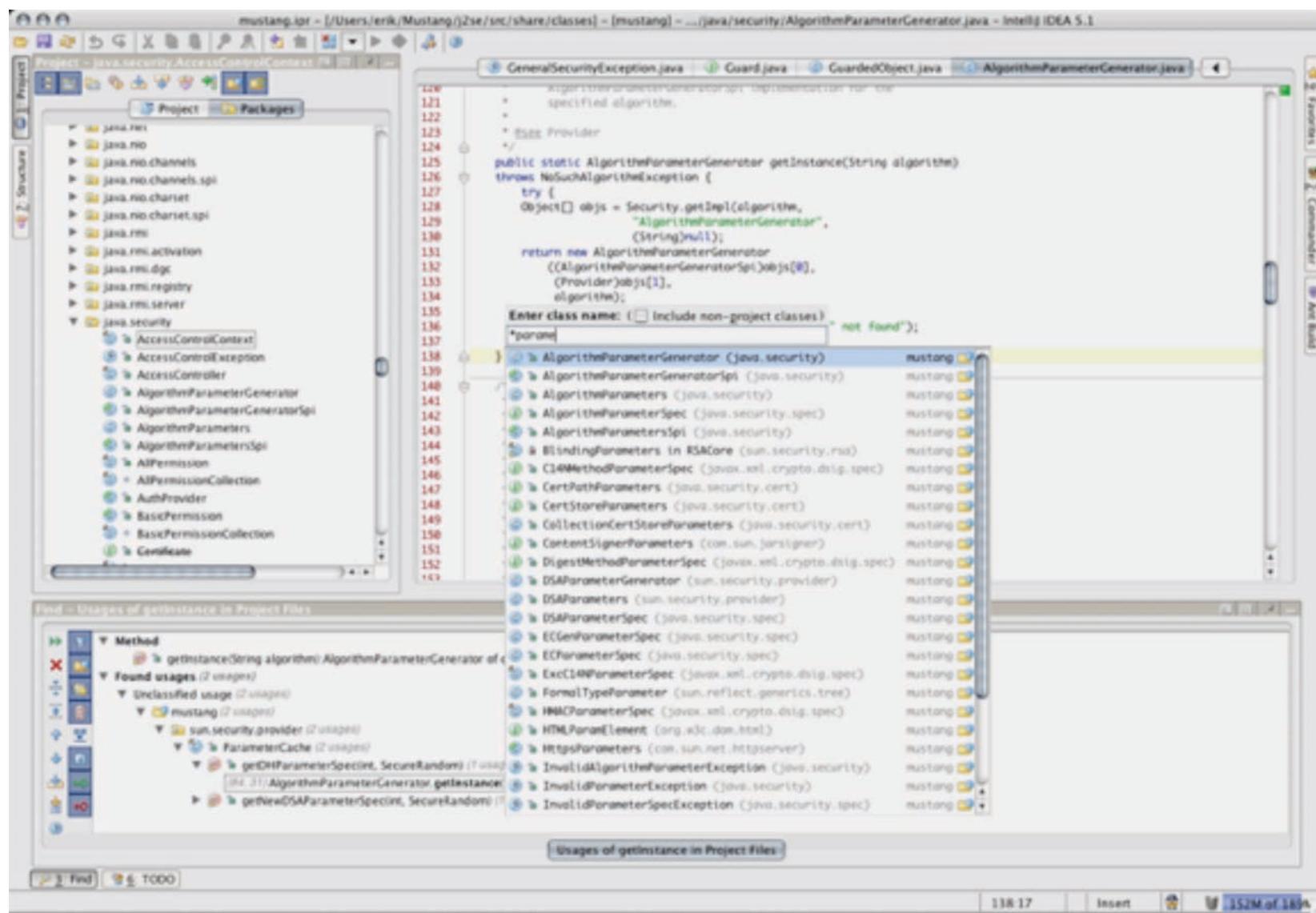
public void mergePluginOutput(BuildDetail build, Map parameters) {
    Iterator iterator = lines().iterator();
    while (iterator.hasNext()) {
        try {
            assemblePlugin(build, parameters, (String) iterator.next());
        } catch (Exception e) {
            logger.error(e);
        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

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

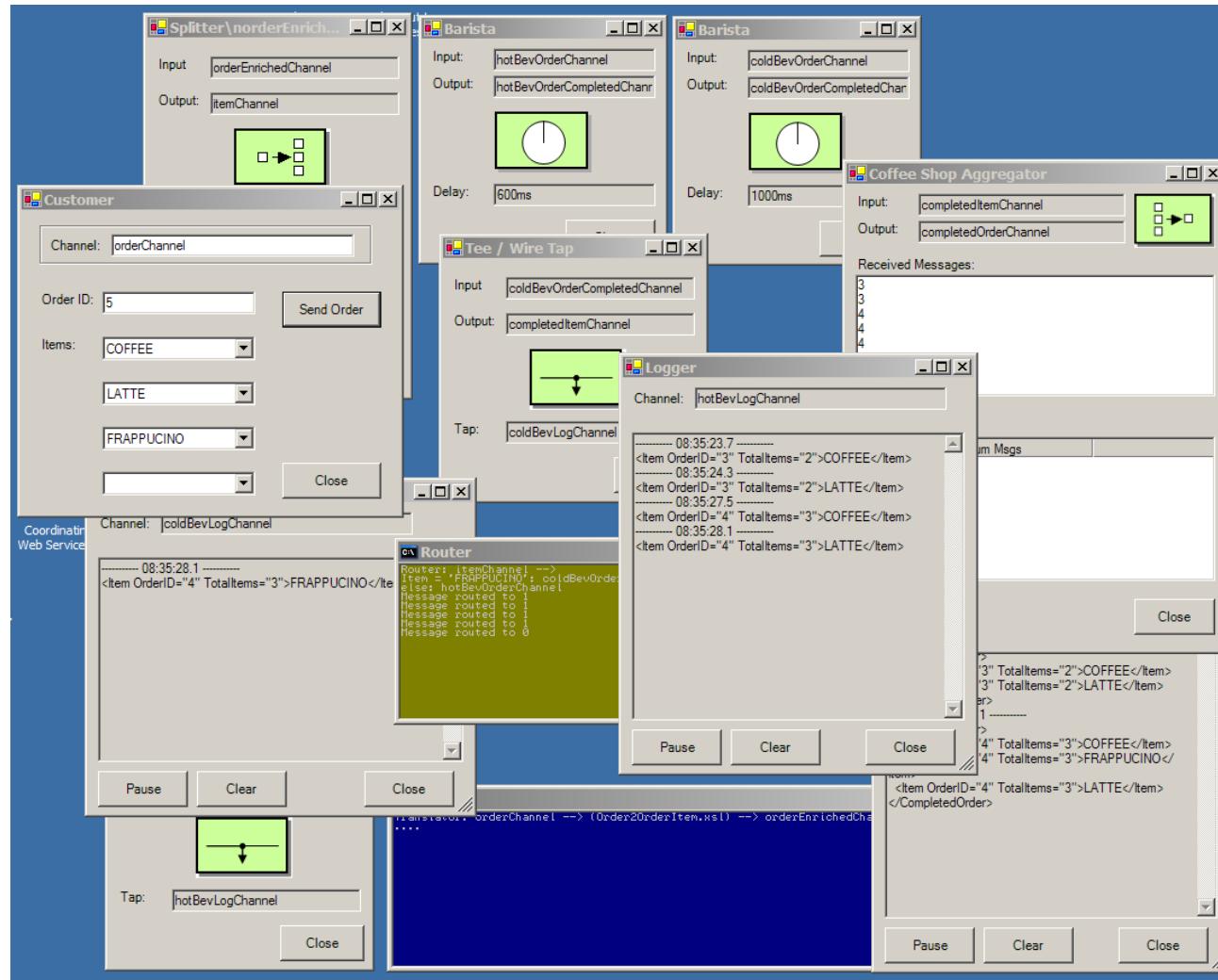
private void mergeParameters(BuildDetail build, Map parameters) {
    parameters.put(Widget.PARAM_CC_ROOT, configuration.getCCRoot());
    parameters.put(CC_PARAM_CC_NAME, build.getPluginName());
}

```

where are the defects?



which way do the messages flow?



where do the pictures come from?

models created up front convey a vision but
usually don't reflect reality

generating a complete model for large systems
is nearly impossible

systems evolve locally, often uncontrolled

the best picture very much depends on the
question you are trying to answer

need tools to create ad-hoc models more
easily

I. select a meta-model

a model that describes a model

example: meta-model for a class diagram

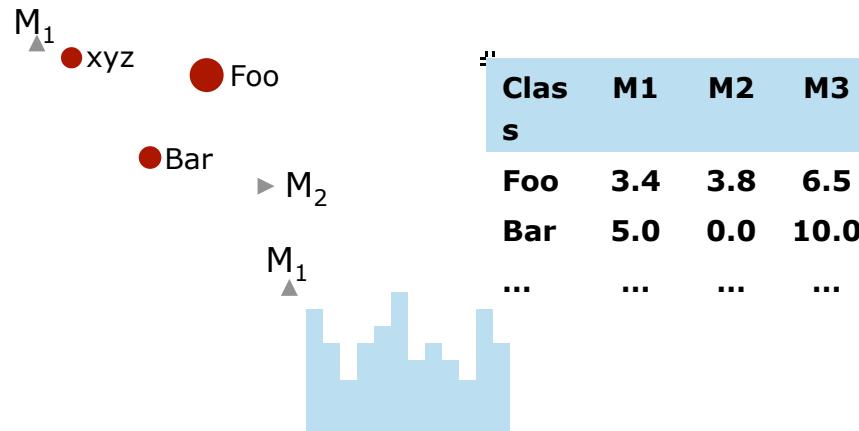
a class is a box with name, methods, fields,...

available connectors: association,
inheritance, aggregation...

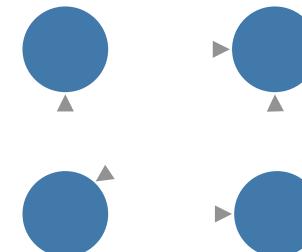
rules: no circles in inheritance, etc.

common meta-models

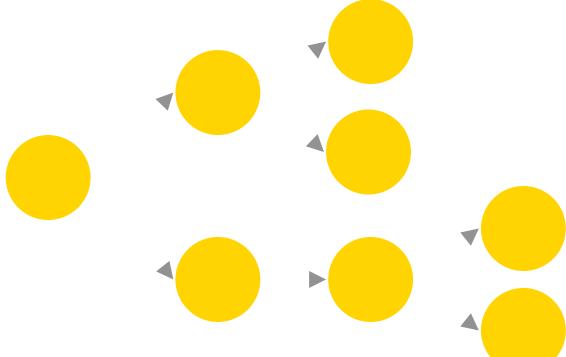
Metrics (Quantitative)



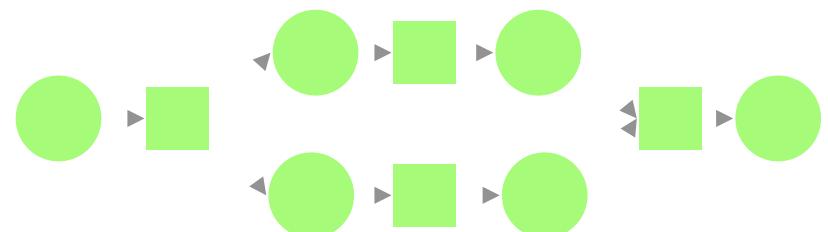
Directed Graph



Tree



Process Model (e.g. Petri Net)



2. inspection / instrumentation

static analysis

source code

byte code

dynamic analysis

profiling, listen to messages, log files,
network sniffer, etc.

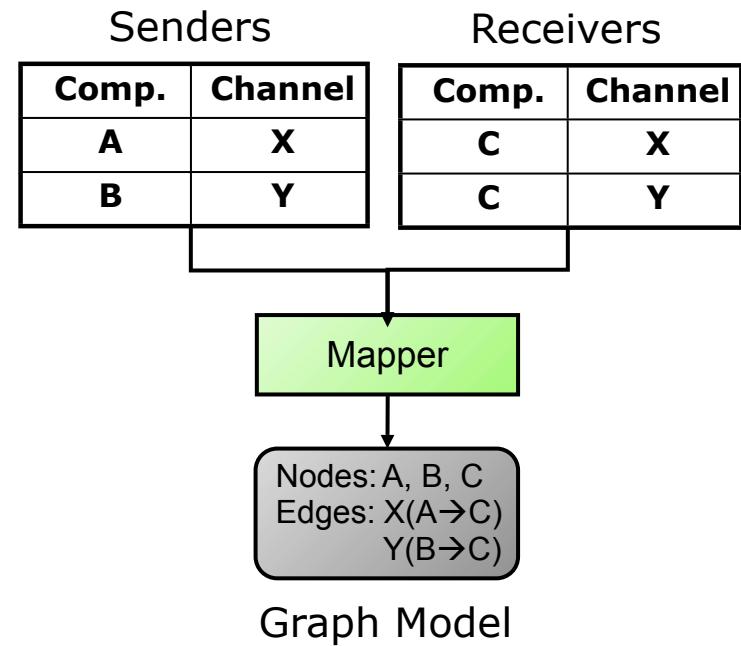


3. mapping to the model

example: messaging system

capture send/receive actions

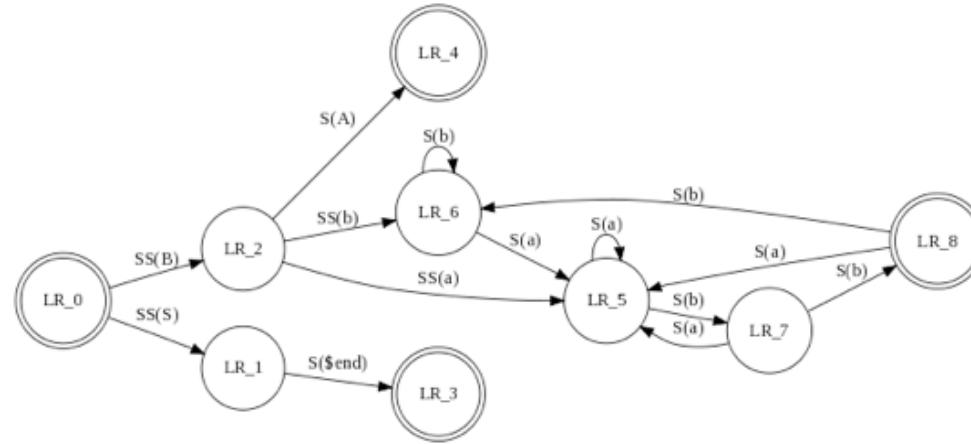
map onto directed graph



4. visualization &...



Graphviz



...validation

don't simply observe

verify & alert

enforce rules or best practices

detect cycles

islands on a dependency graph

the hawthorne effect



the hawthorne effect

*measure and let it be known that
you are measuring*



metrics

cyclomatic complexity

measures complexity of a function

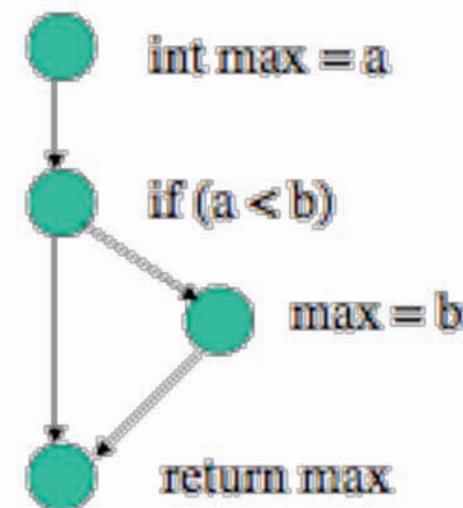
$$V(G) = e - n + 2$$

$V(G)$ = cyclomatic complexity of G

e = # edges

n = # of nodes

```
int max (int a, int b) {  
    int max = a;  
    if (a < b) {  
        max = b;  
    }  
    return max;  
}
```

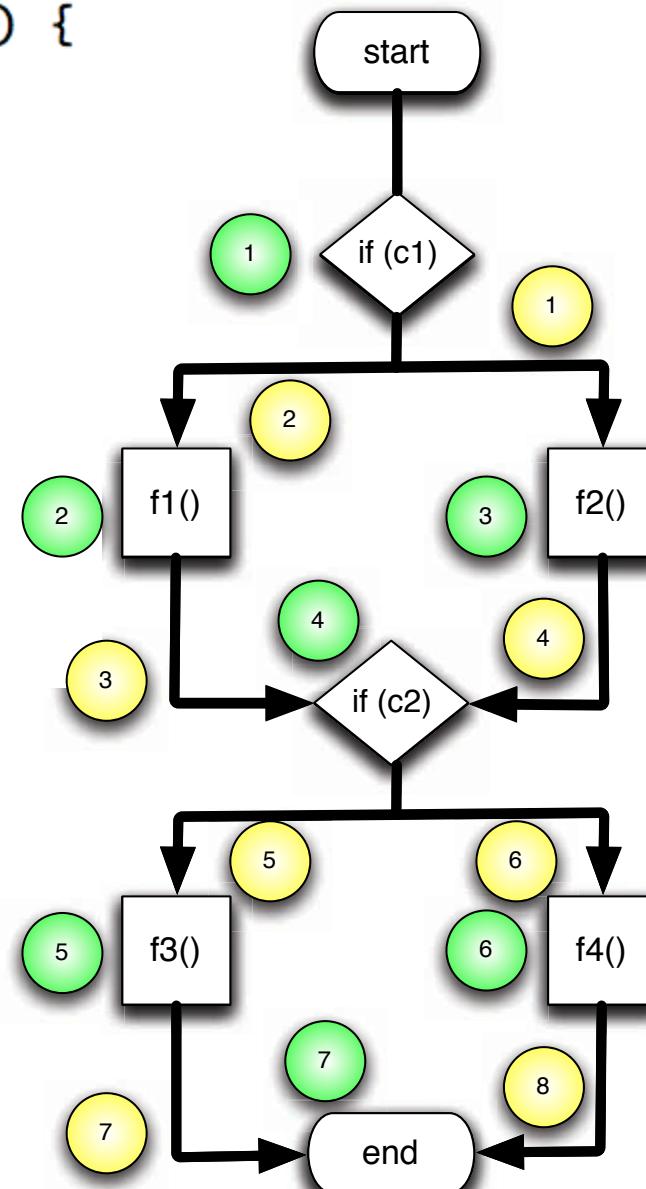


```

public void doIt() {
    if (c1) {
        f1();
    } else {
        f2();
    }
    if (c2) {
        f3();
    } else {
        f4();
    }
}

```

nodes
 edges



chidamber & kemerer object-oriented metrics

shyam r chidamber
chris f kemerer

easy but not terribly useful

very useful

easy (but trivial)

dit	depth of inheritance tree	# levels of inheritance
noc	number of children	# immediate descendants
npm	number of public methods	# public methods in class

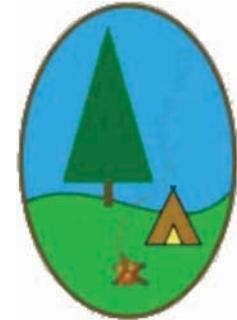
very useful

wmc	weighted methods/ class	Σ of cyclomatic complexity
rfc	response for class	# of methods executed due to method call
lcom	lack of cohesion	Σ of sets of methods not shared via sharing fields
cbo/ ce	efferent couplings	Σ of other classes this class uses (outgoing calls)
ca	afferent couplings	Σ of how many other classes use this class (incoming calls)

visualizations

A black and white graphic featuring a repeating pattern of binary digits (0s and 1s). A large, semi-transparent circular watermark is centered over the pattern. Inside the circle, the word "ERROR" is written in red capital letters. The rest of the text in the circle is in a smaller, gray font.

source monitor



freeware tool for gathering metrics

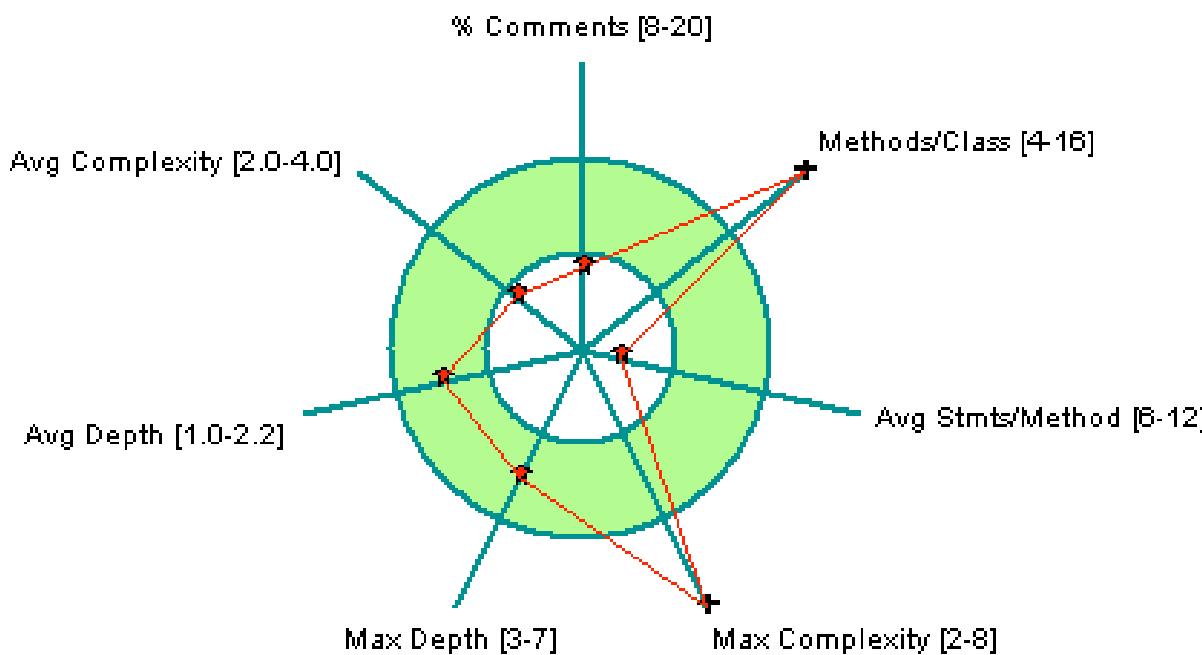
metrics:

lines, statements, % branches, calls, %
comments, classes, methods/class, avg stmts/
method, max complexity, max depth, average
depth, average complexity

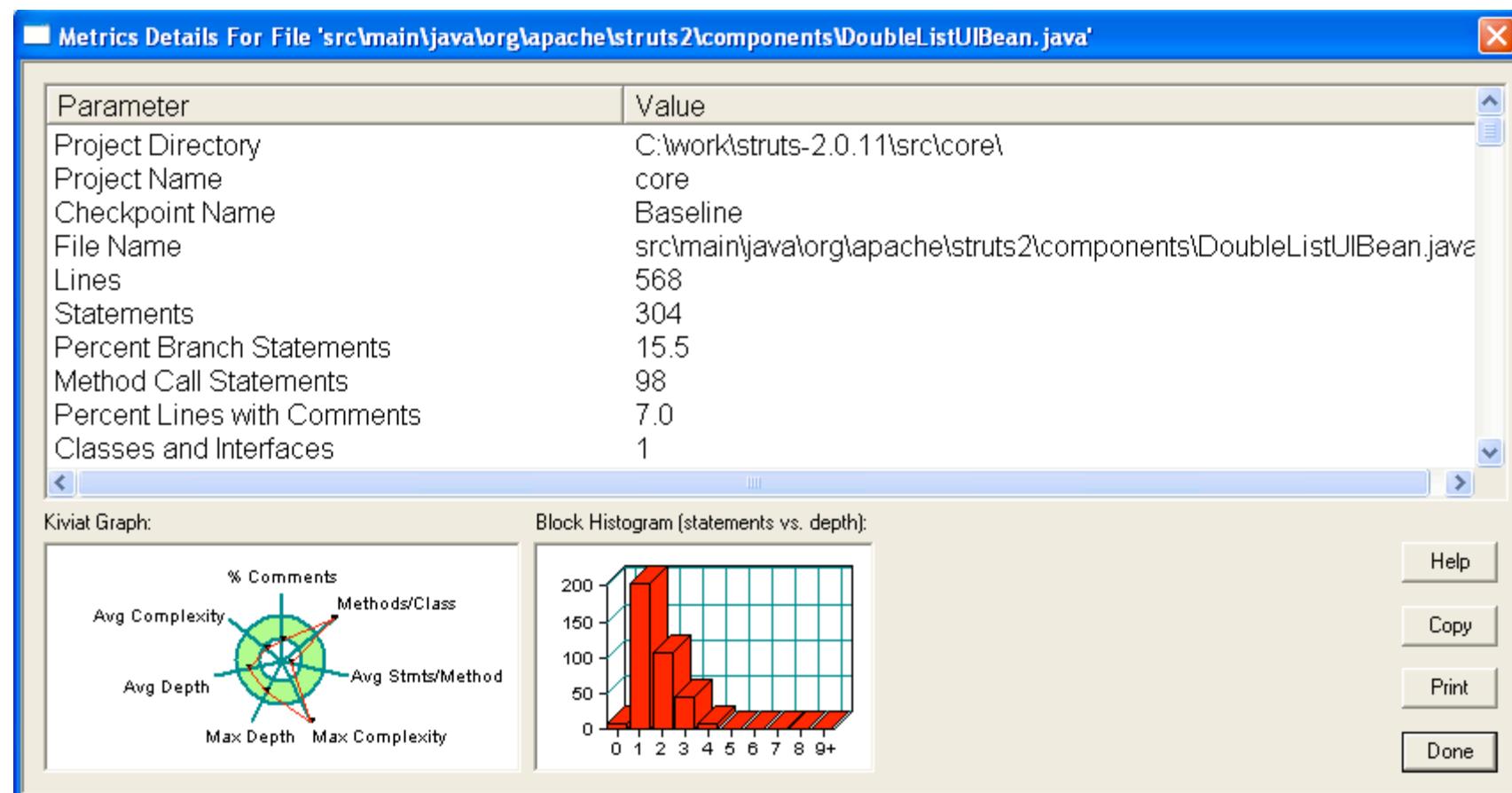
graphical user interface, windows only!

source monitor: kiviat graphs

Kiviat Metrics Graph: Project 'core'
Checkpoint 'Baseline'
File 'src\main\java\org\apache\struts2\components\DoubleListUIBean.java'



source monitor: class summary



source monitor w/ c#

Metrics Details For File 'project\core\util\PathUtil.cs'

Parameter	Value
Project Directory	C:\work\ccnet\
Project Name	ccnet
Checkpoint Name	Baseline
File Name	project\core\util\PathUtil.cs
Lines	462
Statements	191
Percent Comment Lines	10.4
Percent Documentation Lines	9.7
Classes, Interfaces, Structs	1
Methods per Class	4.00
Calls per Method	8.50

Kiviat Graph:

Block Histogram (statements vs. depth):

Depth	Statements
0	5
1	10
2	50
3	55
4	40
5	30
6	25
7	10
8	8
9+	5

Help

Copy

Print

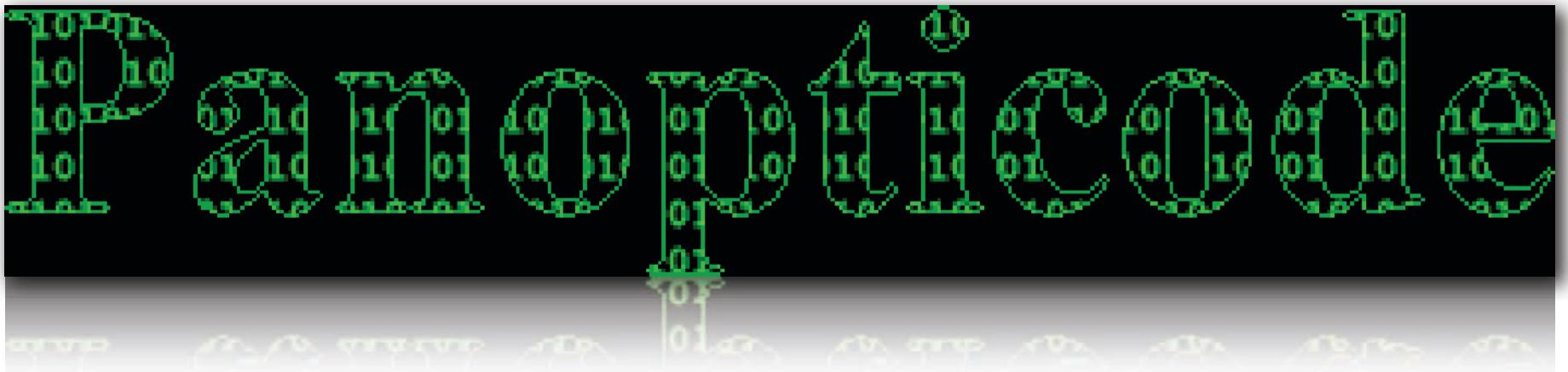
Done



looking for...

classes that violate several kiviat graph ranges

really odd shapes



“A project dedicated to making code metrics so widely understood, valuable, and simple that their use becomes ubiquitous, thus raising the quality of software across the industry.”

panopticode parts

code coverage with emma

1-line change to switch to cobertura

checkstyle

1-line switch for custom rule sets

jdepend

code duplication using simian

javancss

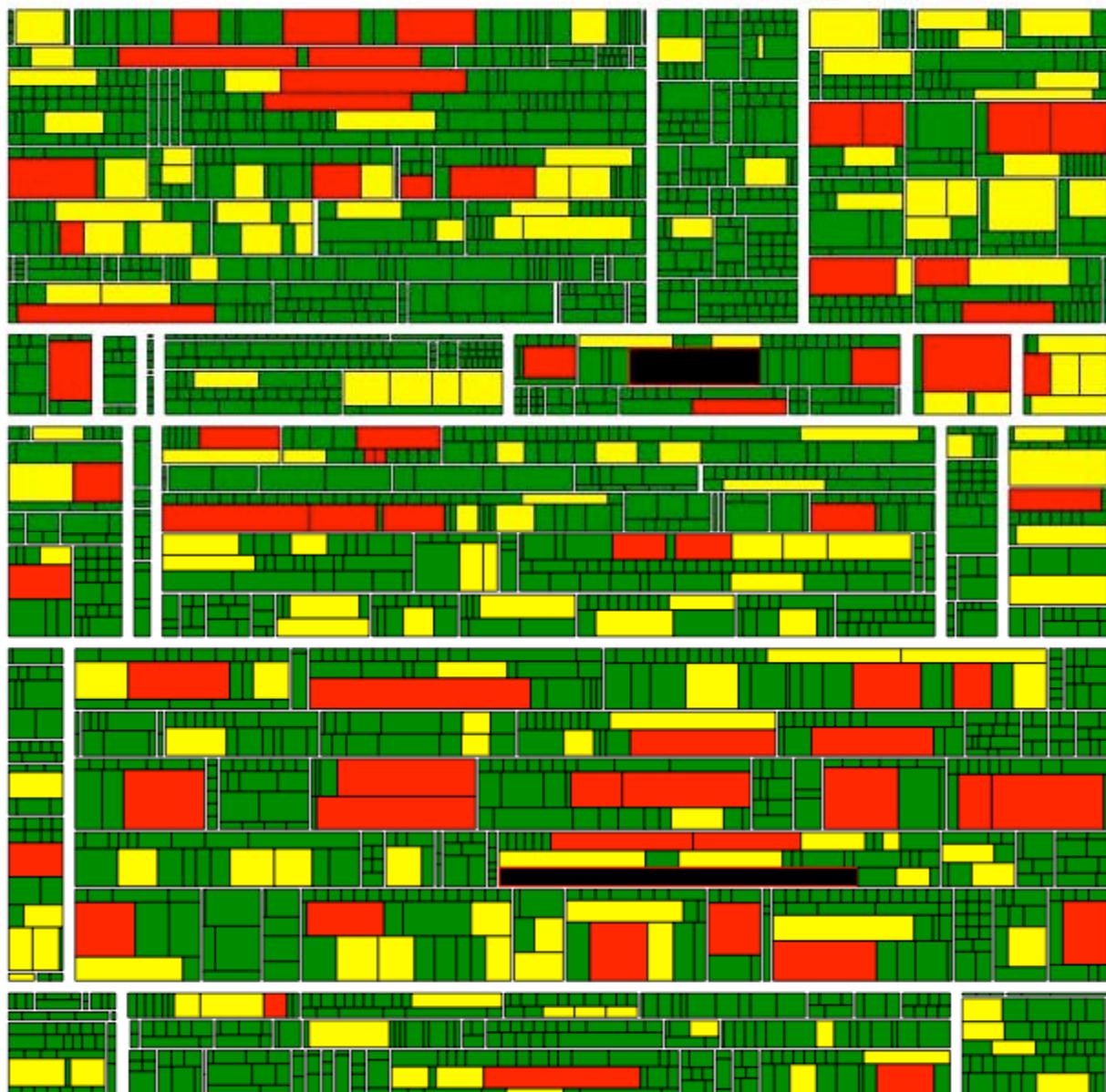
aggregator & reports

volatility

treemaps

interactive-complexity-treemap.svg

CruiseControl Complexity



Details

(Click on a rectangle to view details)

Project:

Package:

File:

Class:

NCSS:

Method Coverage:

Block Coverage:

Line Coverage:

Method:

NCSS:

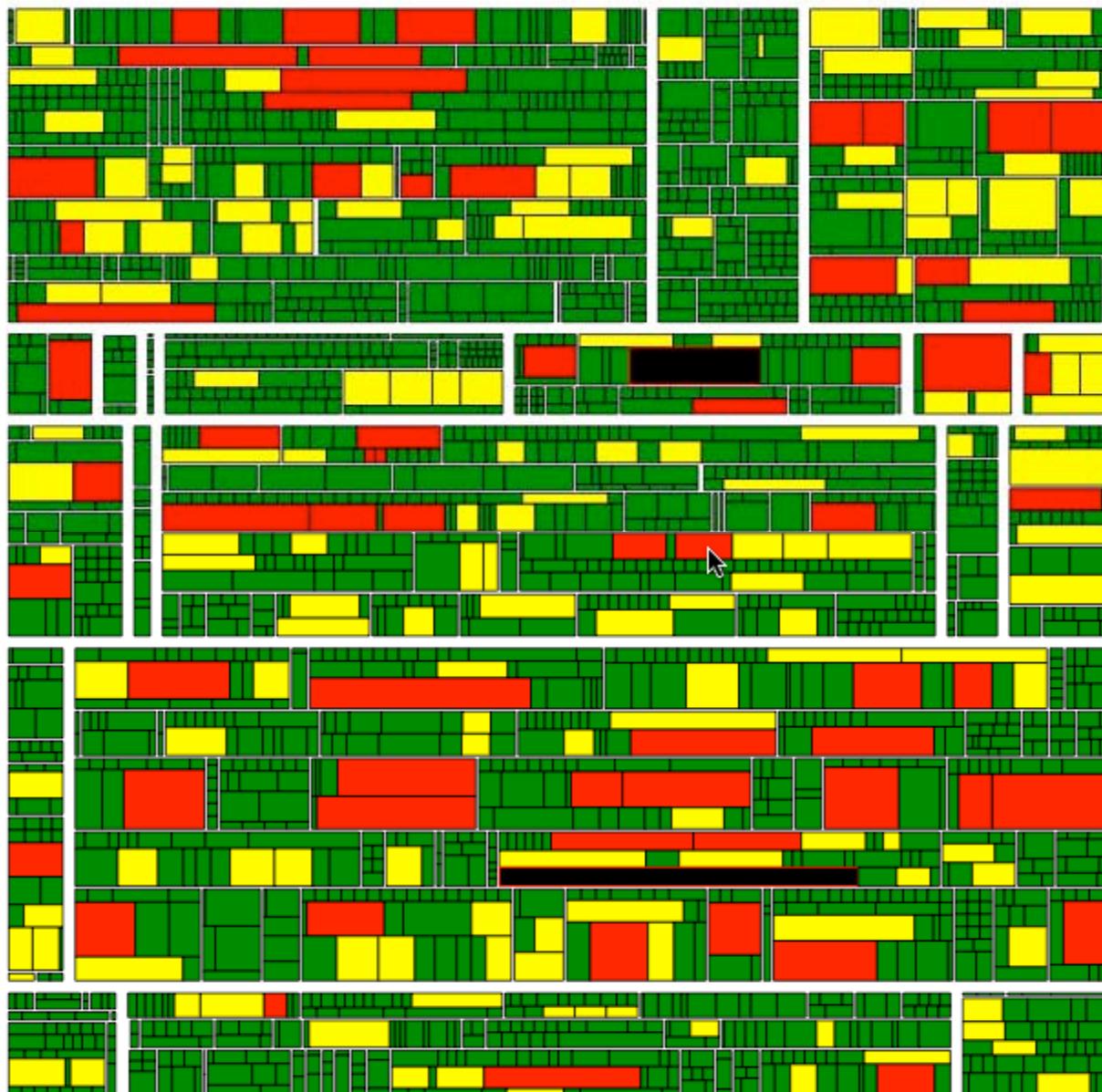
CCN:

Block Coverage:

Line Coverage:

interactive-complexity-treemap.svg

CruiseControl Complexity



Details

(Click on a rectangle to view details)

Project: CruiseControl

Package: net.sourceforge.cruisecontrol.publishers

File: EmailPublisher.java

Class: EmailPublisher

NCSS: 345

Method Coverage: 72.0% (36.0/50.0)

Block Coverage: 67.4% (819.0/1215.0)

Line Coverage: 68.2% (187.6/275.0)

Method: createUserSet(XMLLogHelper)

NCSS: 19

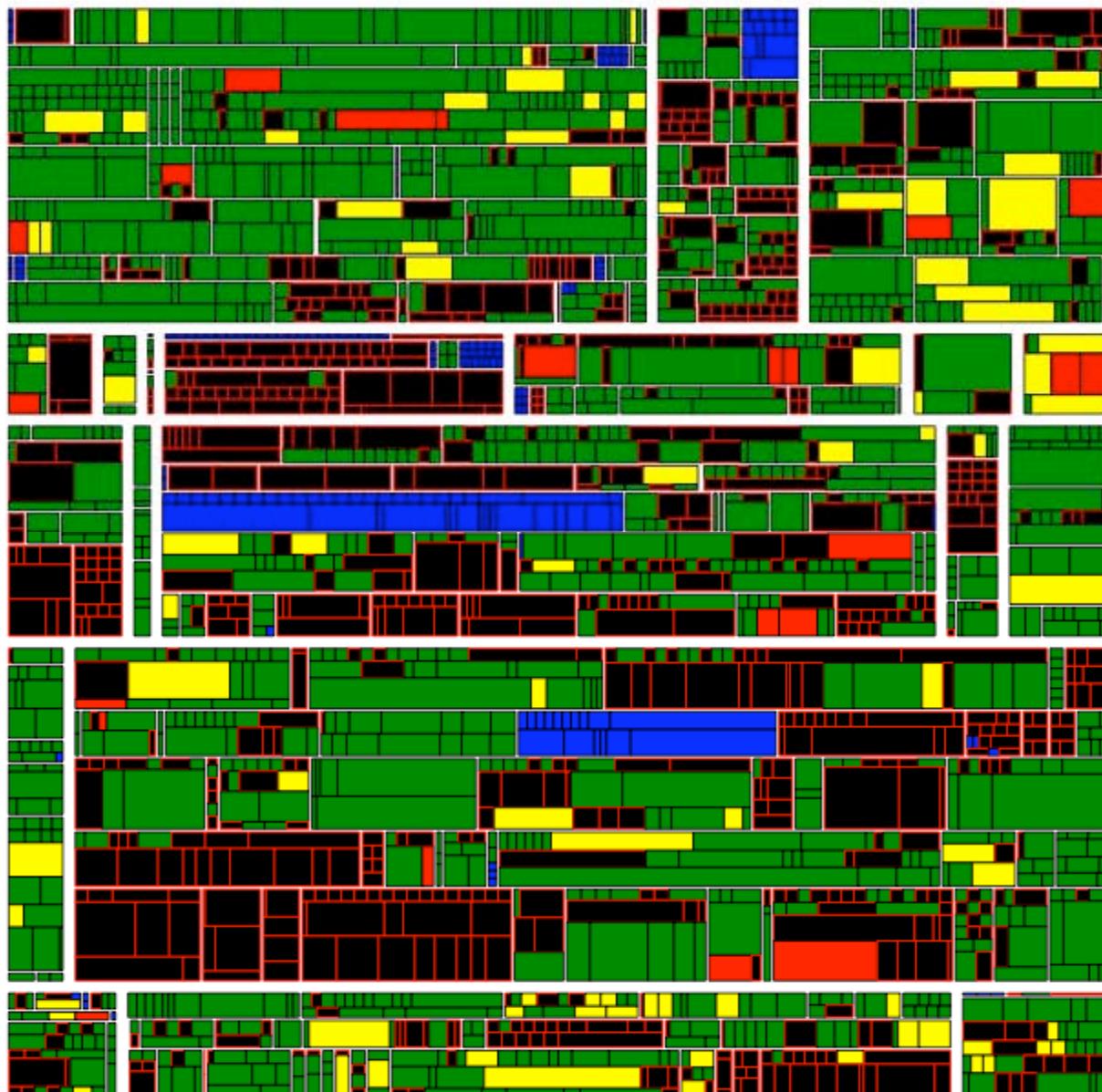
CCN: 11

Block Coverage: 100.0% (122.0/122.0)

Line Coverage: 100.0% (18.0/18.0)

interactive-coverage-treemap.svg

CruiseControl Code Coverage



Details

(Click on a rectangle to view details)

Project:

Package:

File:

Class:

NCSS:

Method Coverage:

Block Coverage:

Line Coverage:

Method:

NCSS:

CCN:

Block Coverage:

Line Coverage:

Coverage >= 75%

Coverage >= 50%

Coverage > 0%

Coverage = 0%

N/A

looking for...

20,000 foot view along a single dimension

simple view of one dimension

information radiators



size & complexity pyramid

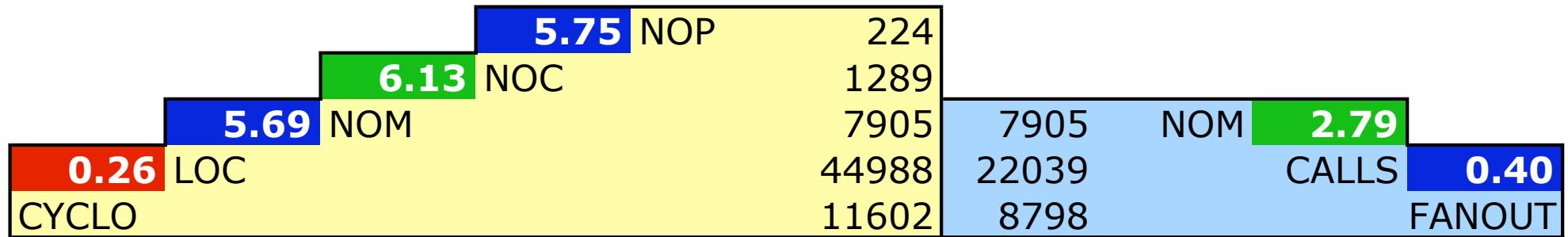
developed at Universities of Berne and Lugano

shows key metrics and their relationships

allows comparison to “industry standards”

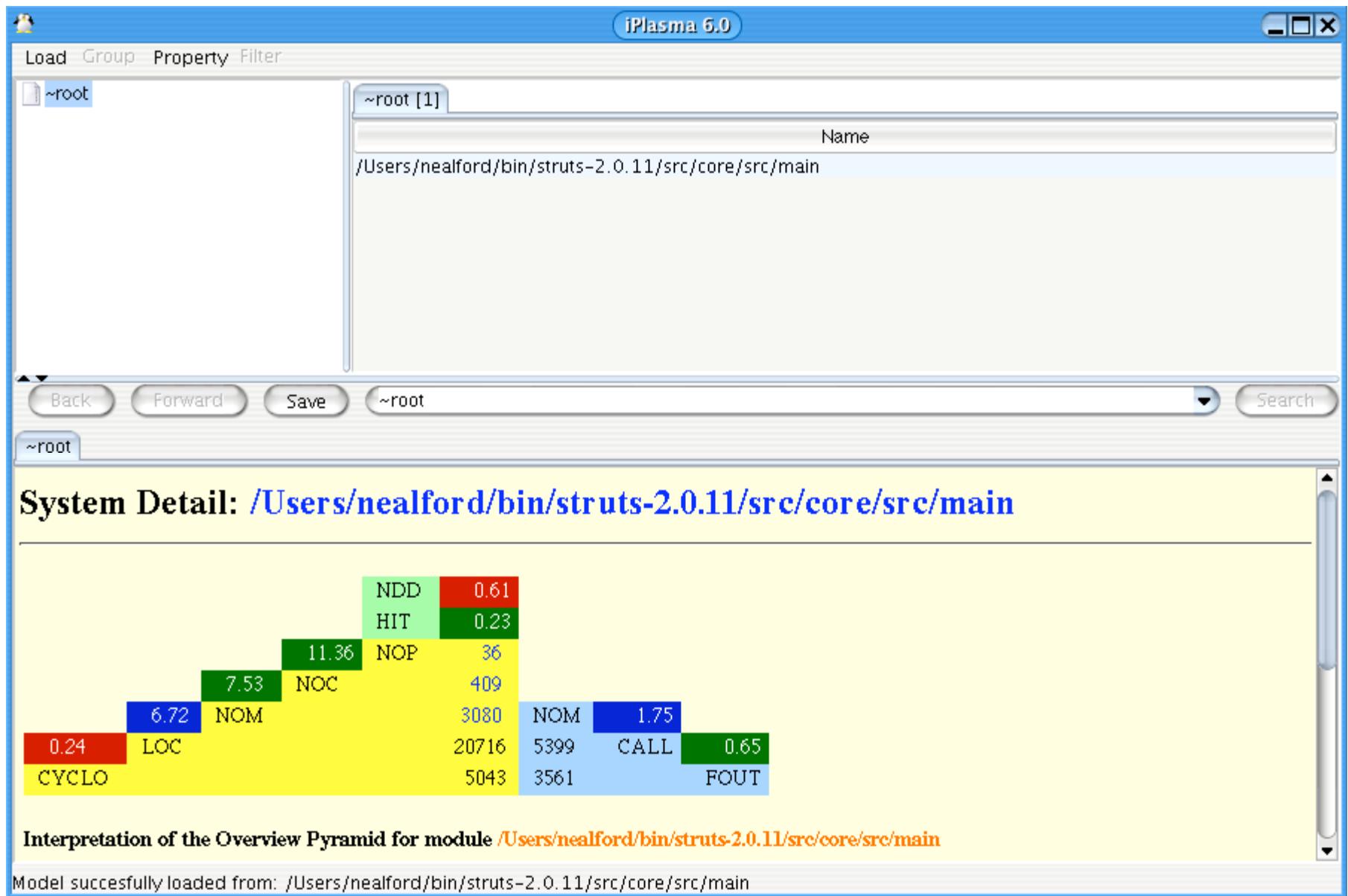
created by iPlasma tool from source code

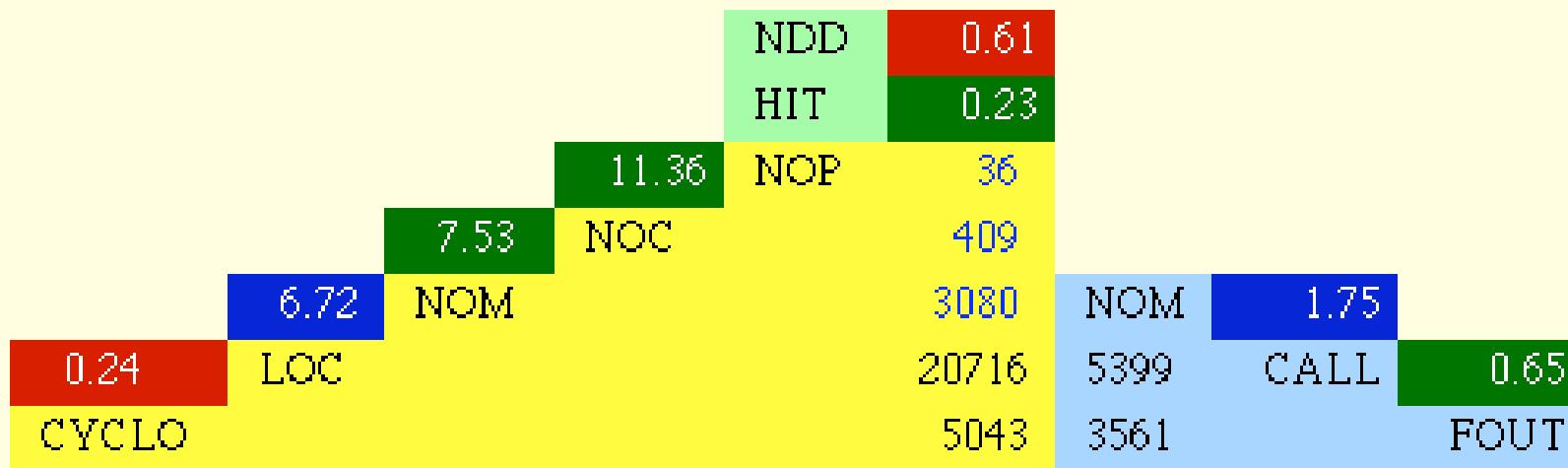
pyramid



	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

iPlasma + Struts





Interpretation of the Overview Pyramid for module [/Users/nealford/bin/struts-2.0.11/src/core/src/main](#)

Class Hierarchies tend to be of **average height** and **wide**

(i.e. inheritance trees tend to have base-classes with many directly derived sub-classes)

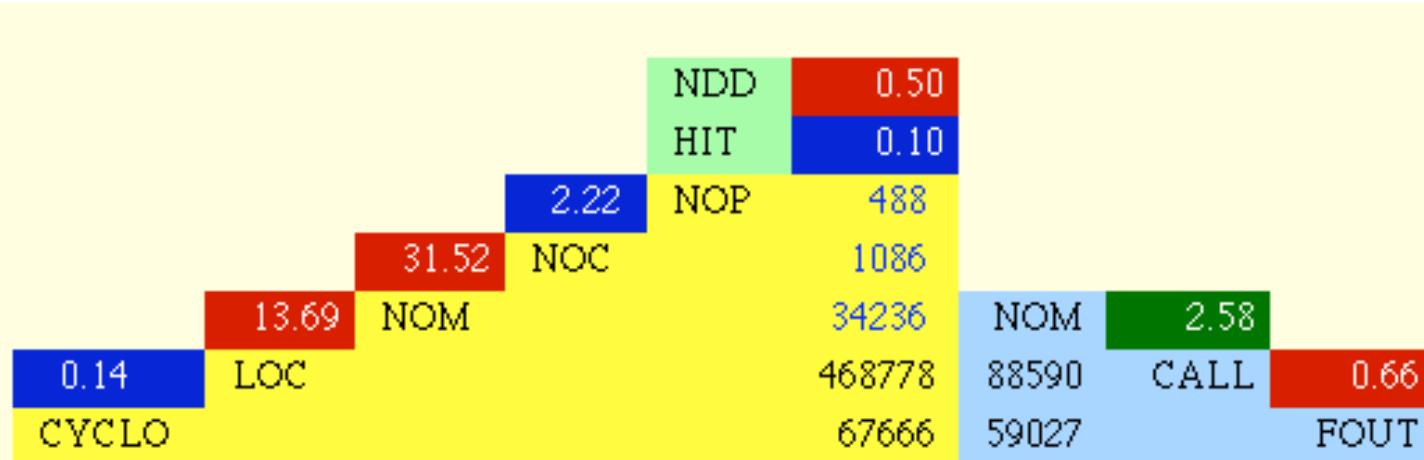
Classes tend to:

- contain an **average** number of methods;
- be organized in **average-sized packages** ;

Methods tend to:

- tend to be rather **short** yet having a rather **complex logic** (i.e. many conditional branches);
- tend to call **few methods** (low coupling intensity) from **several other classes** ;

vuze



Interpretation of the Overview Pyramid for module

/Users/nealford/Downloads/Applications/Vuze_4.2.0.2_source.zip Folder

Class Hierarchies tend to be shallow and wide

(i.e. inheritance trees tend to have only few depth-level(s) and base-classes with many directly derived sub-classes)

Classes tend to:

- be rather **large** (i.e. they define many methods);
 - be organized in rather**fine-grained packages** (i.e. few classes per package);

Methods tend to:

- tend to be rather **long** yet having a rather **simple logic** (i.e. few conditional branches);
 - tend to call an **several methods** from **many other classes** (high coupling dispersion);

looking for...

adherence to industry standards

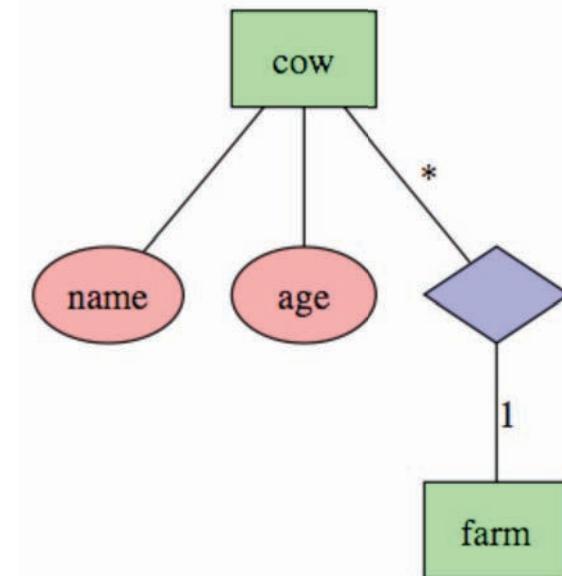
low number of lines / method
(see composed method pattern)

low cyclomatic complexity / line



GraphViz

```
graph ER {
    node [shape=box style=filled fillcolor="#00ff005f"];
    farm;
    cow;
    node [shape=ellipse style=filled fillcolor="#ff00005f"];
    {node [label="name"] cow_name;}
    {node [label="age"] cow_age;}
    node [shape=diamond style=filled fillcolor="#0000ff5f"];
    {node [label=""]
        cow_farm;}
    cow -- cow_name;
    cow -- cow_age;
    cow -- cow_farm [label="*"];
    cow_farm -- farm [label="1"];
}
```

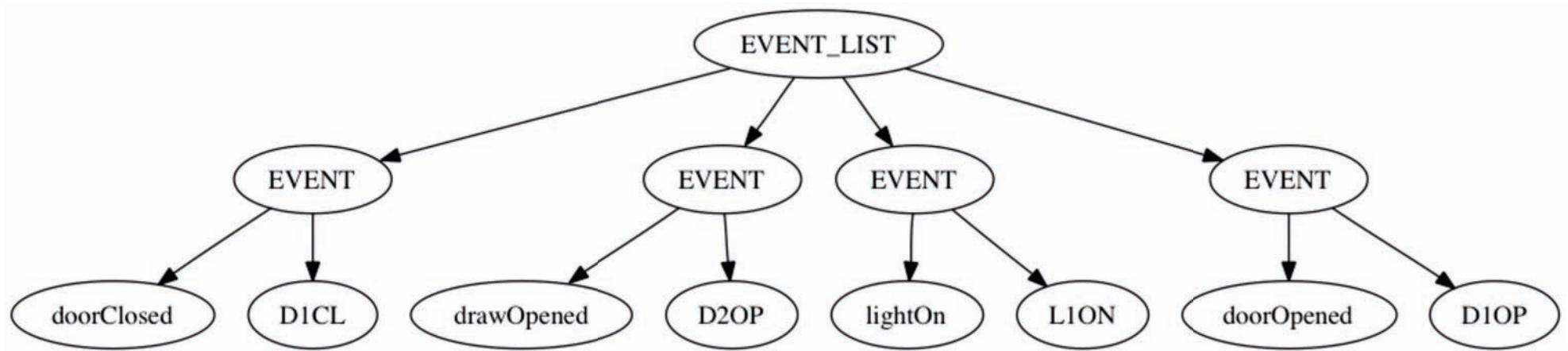


e
v
e
n
t
|
i
s
t

d
o
t
|
f
i
|
e

```
digraph simple {  
    ordering=out  
    e1 [label = "EVENT"]  
    e2 [label = "EVENT"]  
    e3 [label = "EVENT"]  
    e4 [label = "EVENT"]  
    eventList [label = "EVENT_LIST"]  
    eventList -> e1  
    eventList -> e2  
    eventList -> e3  
    eventList -> e4  
    c1 [label = "D1CL"]  
    c2 [label = "D2OP"]  
    c3 [label = "L1ON"]  
    c4 [label = "D1OP"]  
    e1 -> doorClosed  
    e1 -> c1  
    e2 -> drawOpened  
    e2 -> c2  
    e3 -> lightOn  
    e3 -> c3  
    e4 -> doorOpened  
    e4 -> c4  
}
```

output



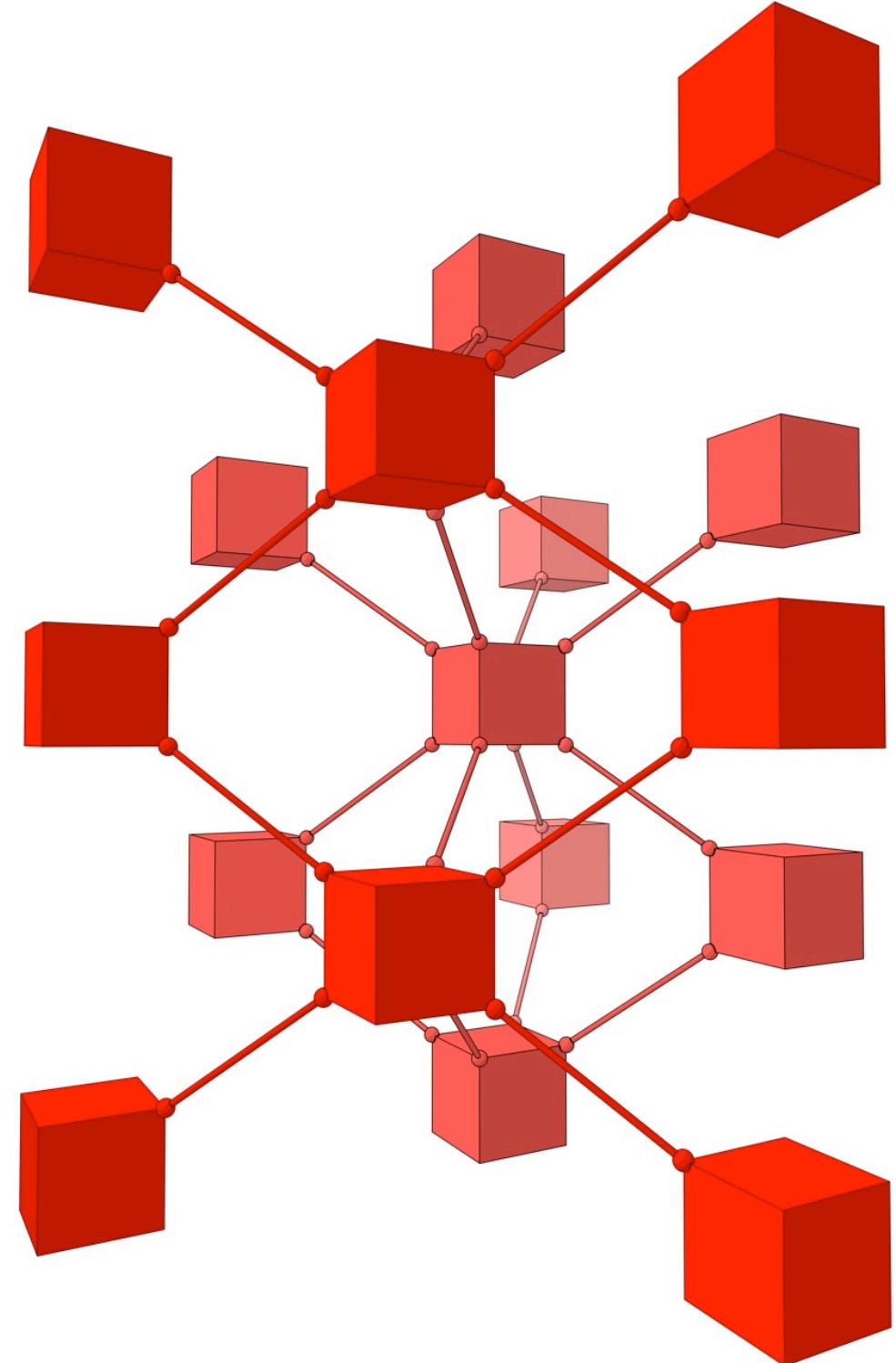
generating dot files

```
public void toDot(StringBuilder result) {
    String dotLabel = String.format("{%s", name);
    if (!commands.isEmpty()) {
        dotLabel += "|";
        for(Command c : commands) dotLabel += String.format("%s\\n", c.getName());
    }
    dotLabel += "}";
    result.append(String.format("%s [shape = Mrecord fontsize = 12 label = \"%s\"]\\n", name, dotLabel));
    for(Map.Entry<Event,State> e : transitions.entrySet()) {
        result.append(String.format("%s -> %s [ label = \"%s\" fontsize = 10 arrowhead = open];\\n",
            name, e.getValue().name, e.getKey().getName()));
    }
}
```

untangling jars

jar analyzer

Kirk Knoernschild
www.kirkk.com



xml output

```
<JarAnalyzer>
  - <Jars>
    - <Jar name="antlr.jar">
      - <Summary>
        - <Statistics>
          <ClassCount>210</ClassCount>
          <AbstractClassCount>48</AbstractClassCount>
          <PackageCount>10</PackageCount>
          <Level>1</Level>
        </Statistics>
      - <Metrics>
        <Abstractness>0.23</Abstractness>
        <Efferent>0</Efferent>
        <Afferent>1</Afferent>
        <Instability>0.00</Instability>
        <Distance>0.77</Distance>
      </Metrics>
      - <Packages>
        <Package>antlr</Package>
        <Package>antlr.build</Package>
        <Package>antlr.collections</Package>
        <Package>antlr.debug</Package>
        <Package>antlr.preprocessor</Package>
        <Package>antlr.actions.cpp</Package>
        <Package>antlr.actions.csharp</Package>
        <Package>antlr.actions.java</Package>
        <Package>antlr.collections.impl</Package>
        <Package>antlr.debug.misc</Package>
      </Packages>
      <OutgoingDependencies> </OutgoingDependencies>
      - <IncomingDependencies>
        <Jar>struts.jar</Jar>
      </IncomingDependencies>
      <Cycles> </Cycles>
      <UnresolvedDependencies> </UnresolvedDependencies>
    </Summary>
  </Jar>
  - <Jar name="commons-beanutils.jar">
    - <Summary>
      - <Statistics>
        <ClassCount>66</ClassCount>
        <AbstractClassCount>7</AbstractClassCount>
        <PackageCount>4</PackageCount>
        <Level>2</Level>
      </Statistics>
    - <Metrics>
      <Abstractness>0.11</Abstractness>
```

JarAnalyzer Analysis

Run with [JarAnalyzer](#) on

Summary

[[summary](#)] [[jars](#)] [[cycles](#)] [[explanations](#)]

Jar Name	Total Classes	Abstract Classes	Packages	Level	Abstractness	Efferent	Afferent	Instability	Distance
antlr.jar	210	48	10	1	0.23	0	1	0.00	0.77
commons-beanutils.jar	66	7	4	2	0.11	2	3	0.40	0.49
commons-collections.jar	187	15	3	1	0.08	0	4	0.00	0.92
commons-digester.jar	55	9	3	3	0.16	3	2	0.60	0.24
commons-fileupload.jar	16	4	1	1	0.25	0	1	0.00	0.75
commons-logging.jar	18	2	2	1	0.11	0	4	0.00	0.89
commons-validator.jar	30	1	2	4	0.03	5	1	0.83	0.14
jakarta-oro.jar	62	13	6	1	0.21	0	1	0.00	0.79
struts.jar	289	33	25	5	0.11	7	0	1.00	0.11

Jars

[[summary](#)] [[jars](#)] [[cycles](#)] [[explanations](#)]

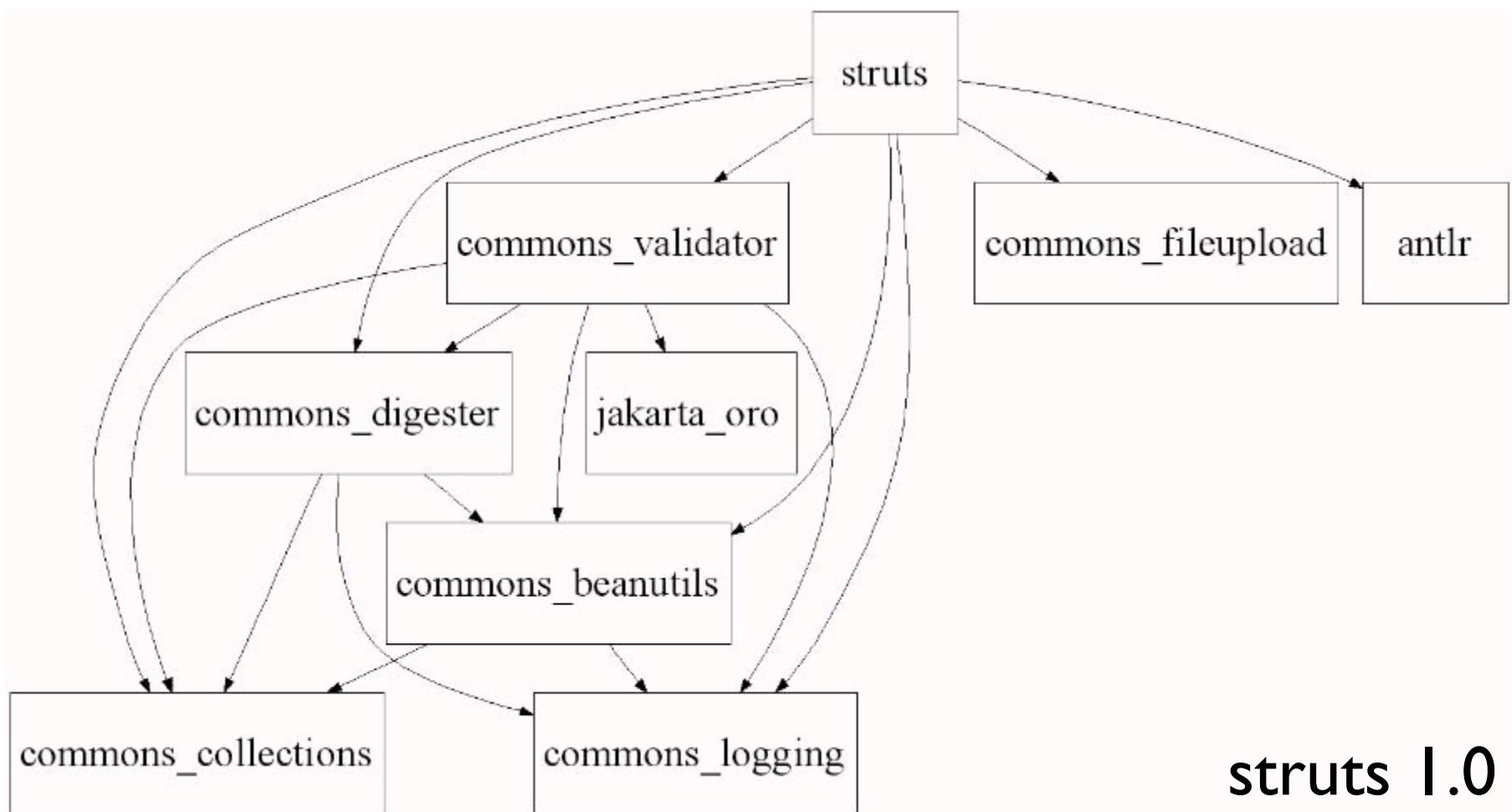
antlr.jar

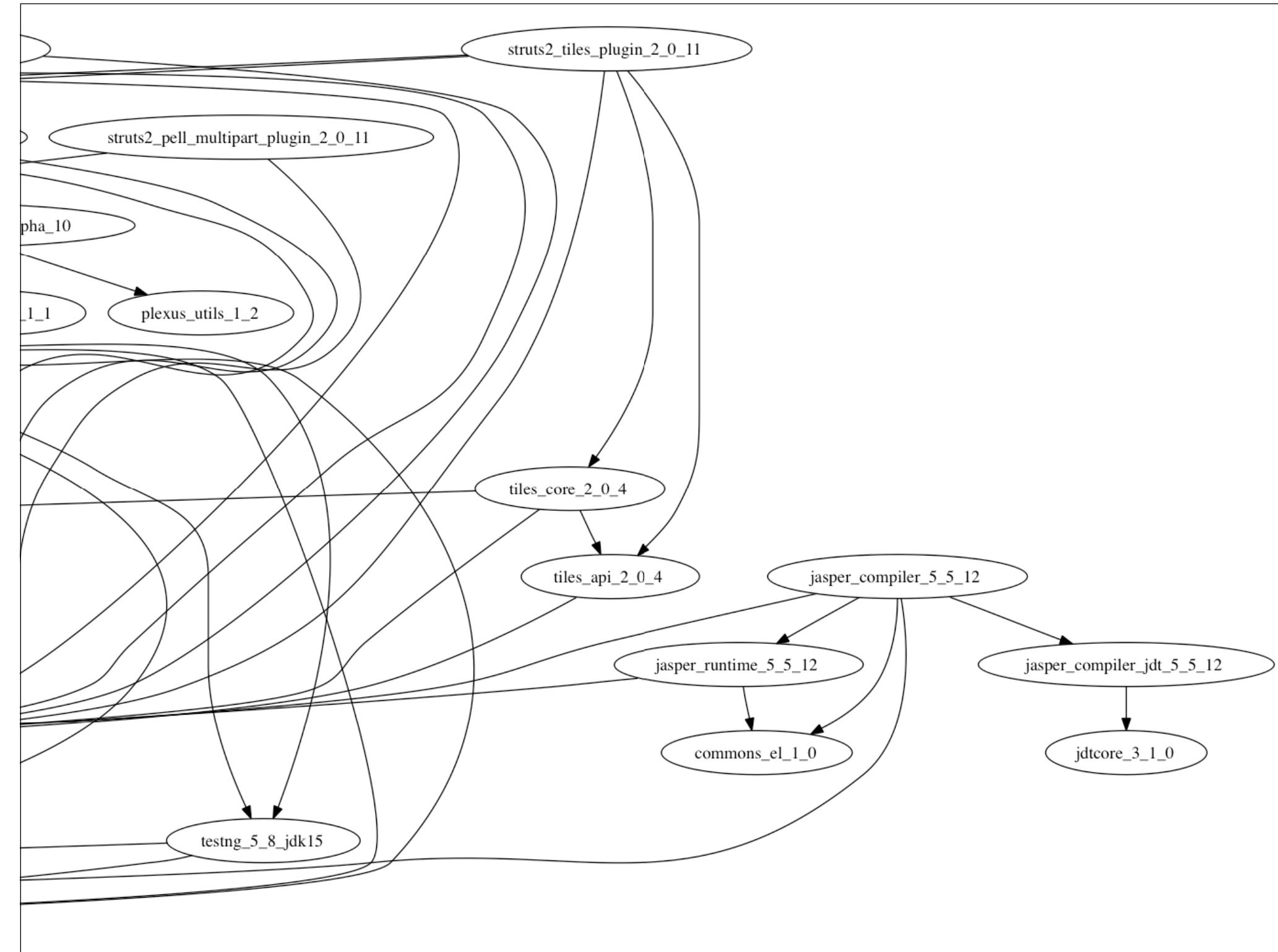
Level: 1	Afferent Couplings: 1	Efferent Couplings: 0	Abstractness: 0.23	Instability: 0.00	Distance: 0.77		
Uses Jars		Used by Jars		Cycles With			
None		struts.jar		None			
Packages within jar			Unresolved Packages				
antlr antlr.build antlr.collections antlr.debug antlr.preprocessor antlr.actions.cpp antlr.actions.csharp antlr.actions.java antlr.collections.impl antlr.debug.misc			None				

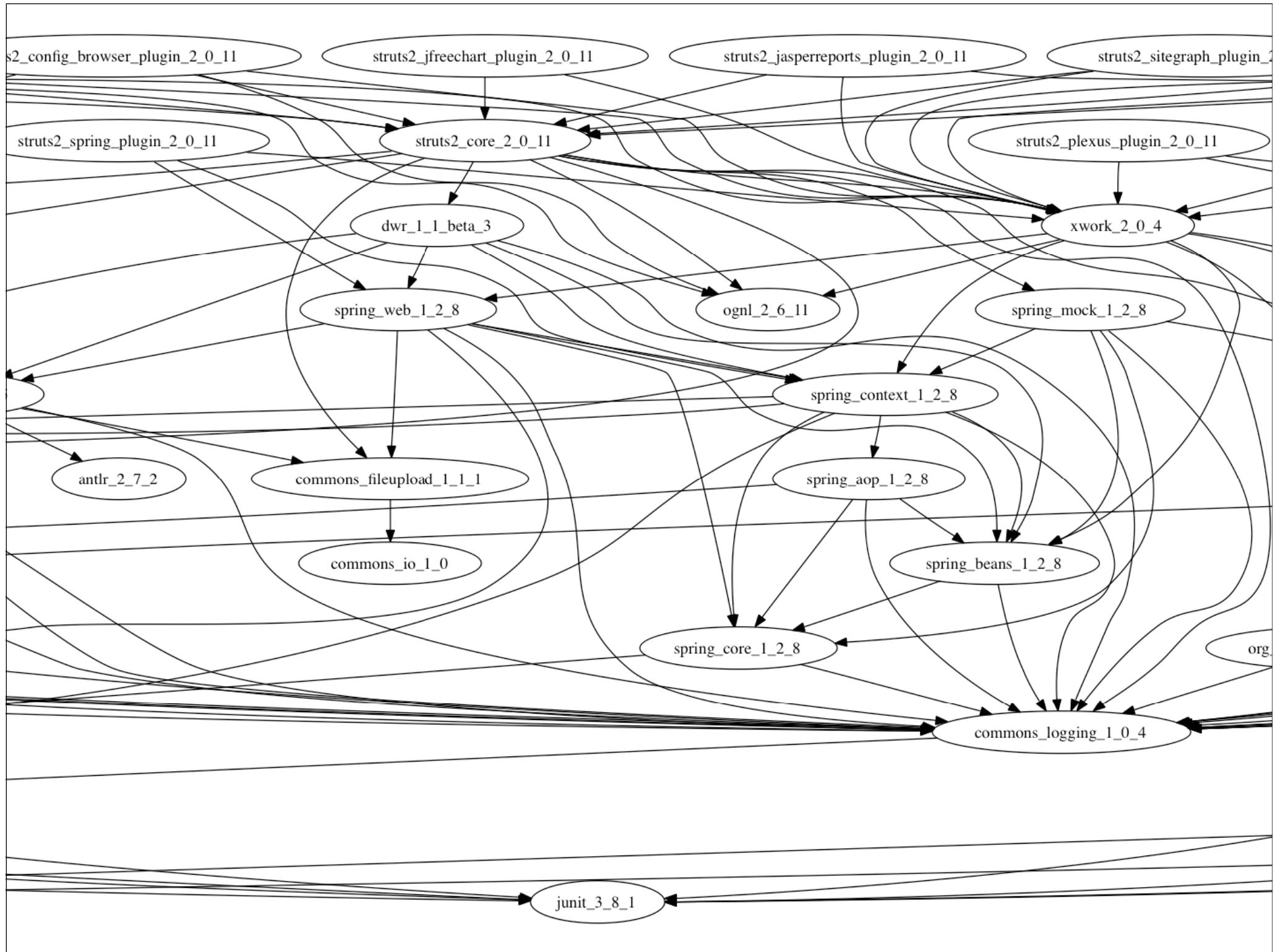
commons-beanutils.jar

Level: 2	Afferent Couplings: 3	Efferent Couplings: 2	Abstractness: 0.11	Instability: 0.40	Distance: 0.49		
Uses Jars		Used by Jars		Cycles With			
commons-collections.jar commons-logging.jar		commons-digester.jar commons-validator.jar struts.jar		None			
Packages within jar			Unresolved Packages				
org.apache.commons.beanutils.converters org.apache.commons.beanutils org.apache.commons.beanutils.locale.converters org.apache.commons.beanutils.locale			None				

graphical







looking for...

not that!

small number of one-way dependencies

no “rats’ nests”

no cycles





Vizant

ant task to create a GraphViz
DOT file from an ant build file

<http://vizant.sourceforge.net/>

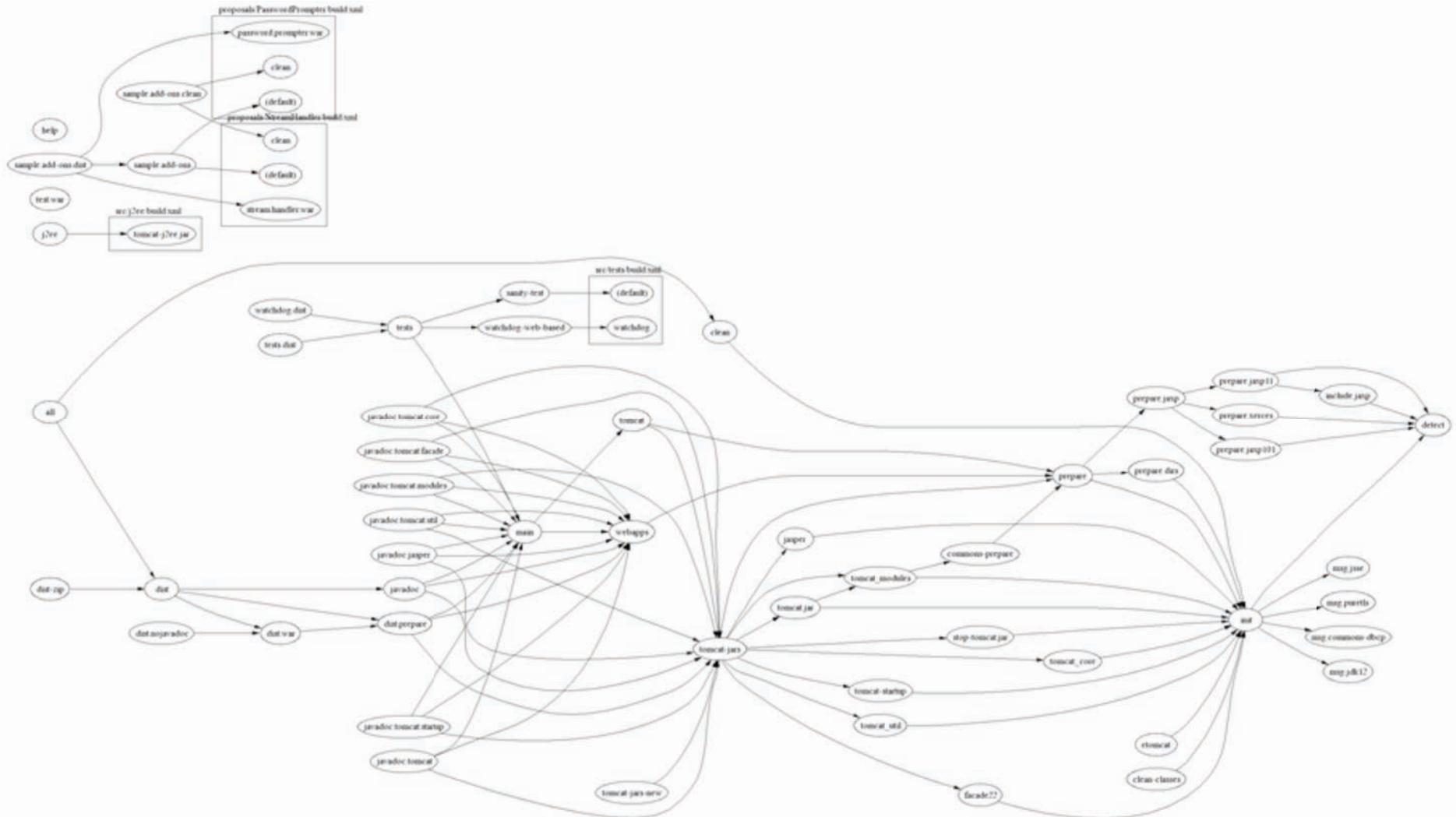
ant 1.5



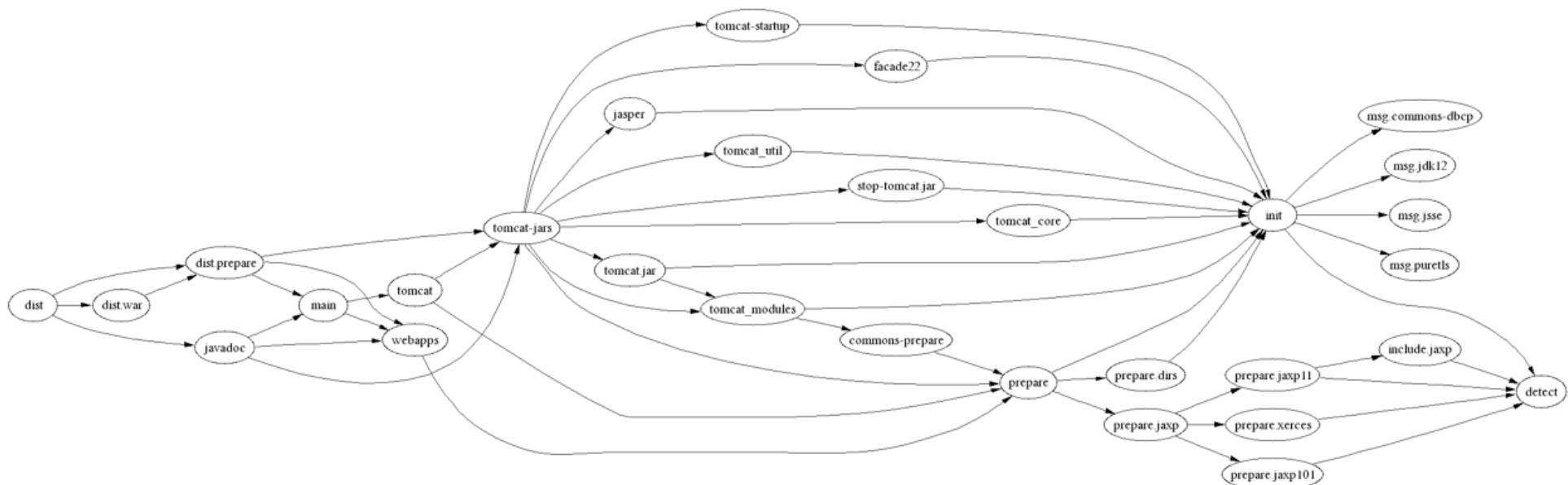
log4j



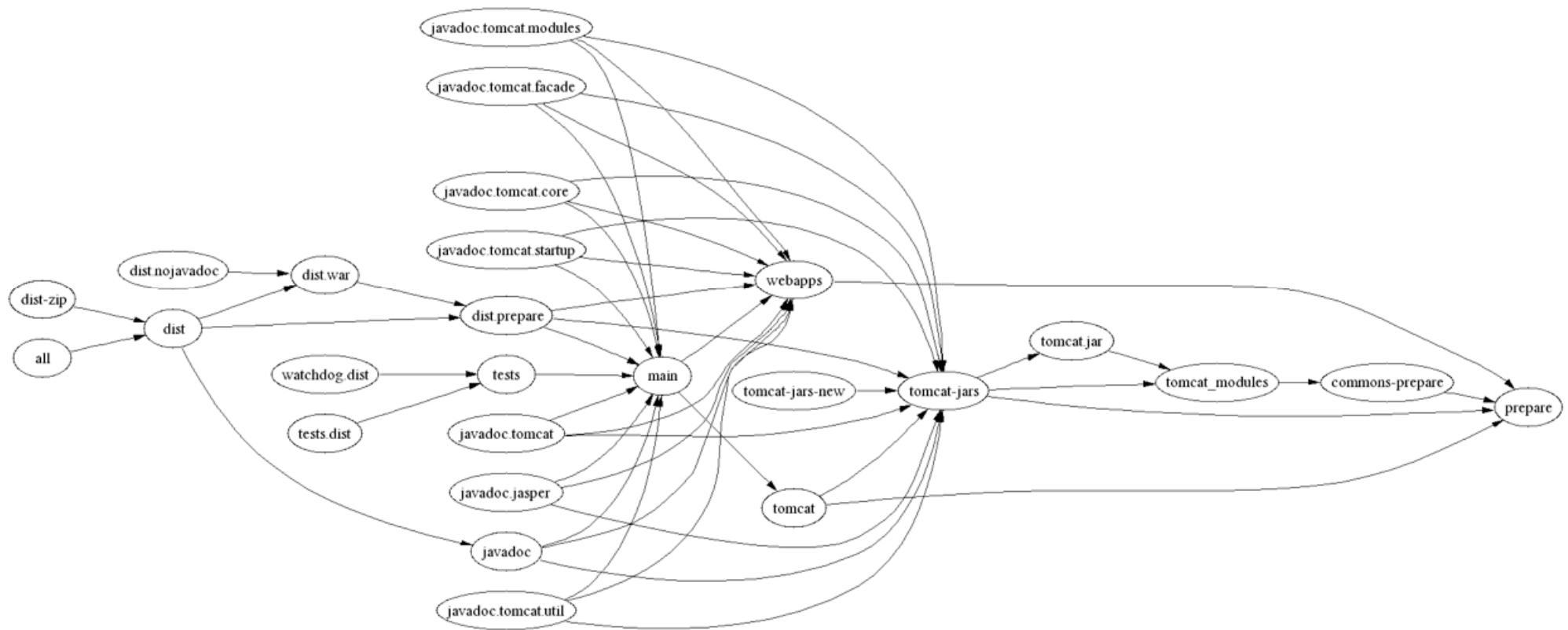
tomcat 3.3.1



from=“dist”



to=“prepare”



look for...

hot spots

more even distribution

networks around common dependent elements

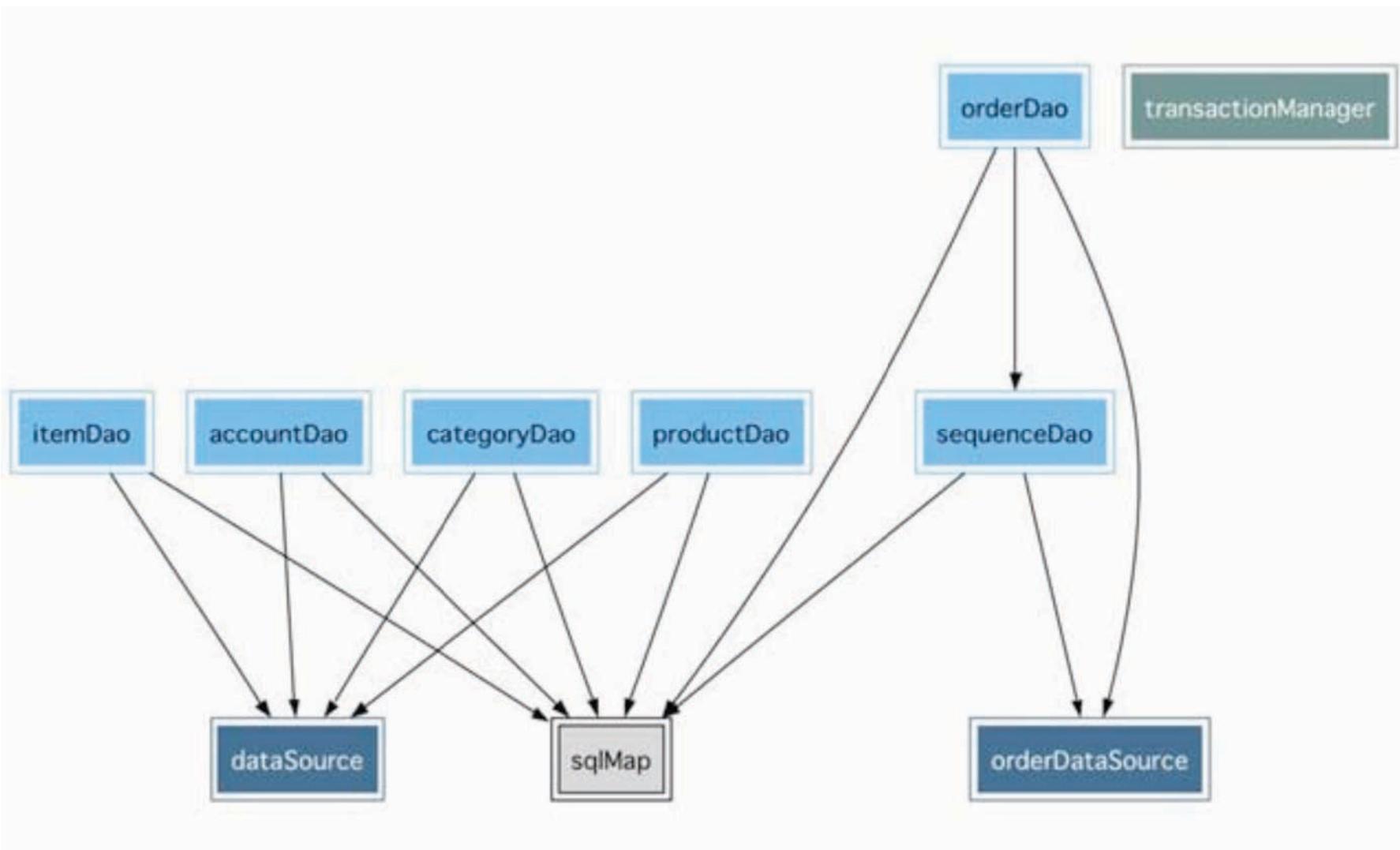
think about extracting
via macrodef

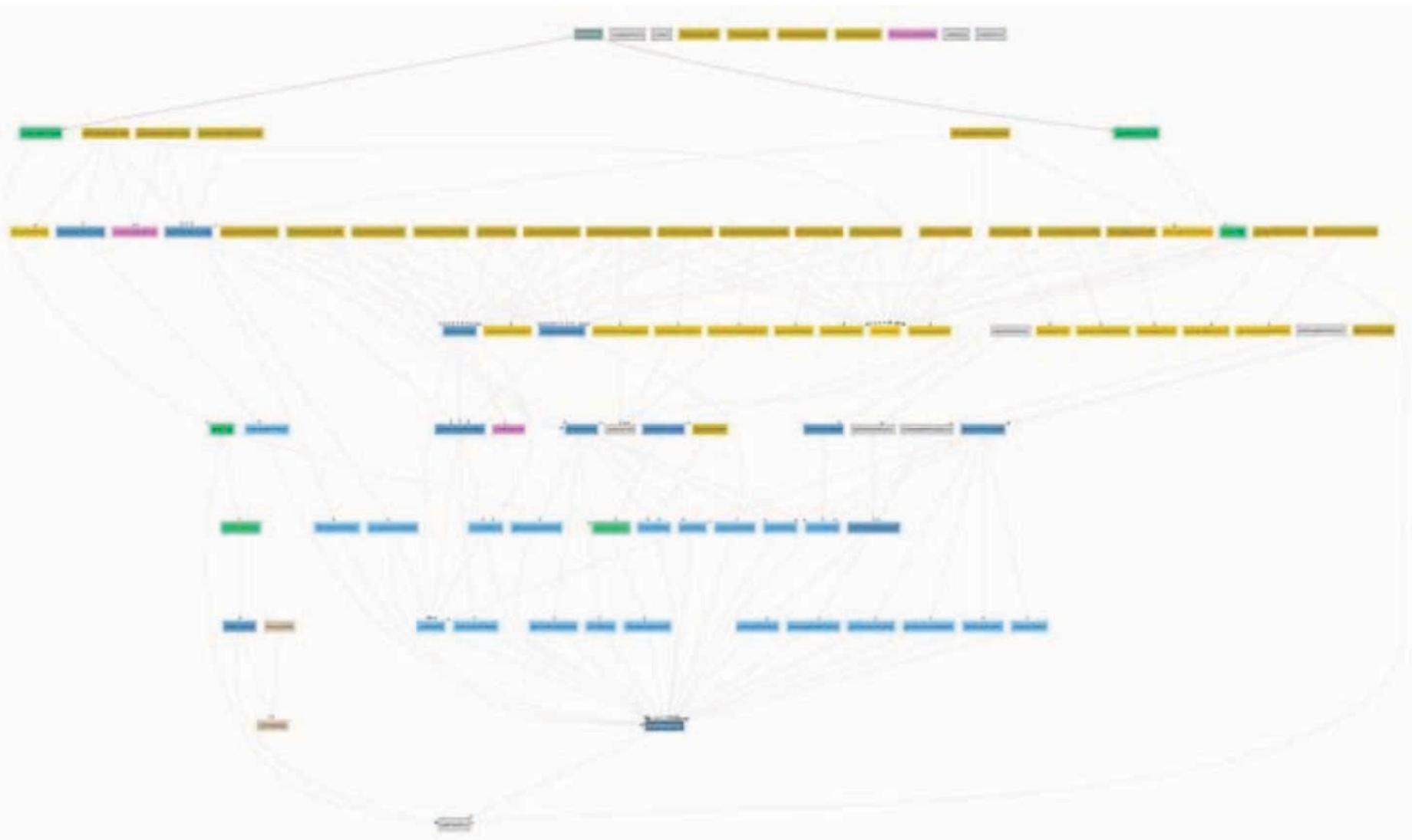


SpringViz

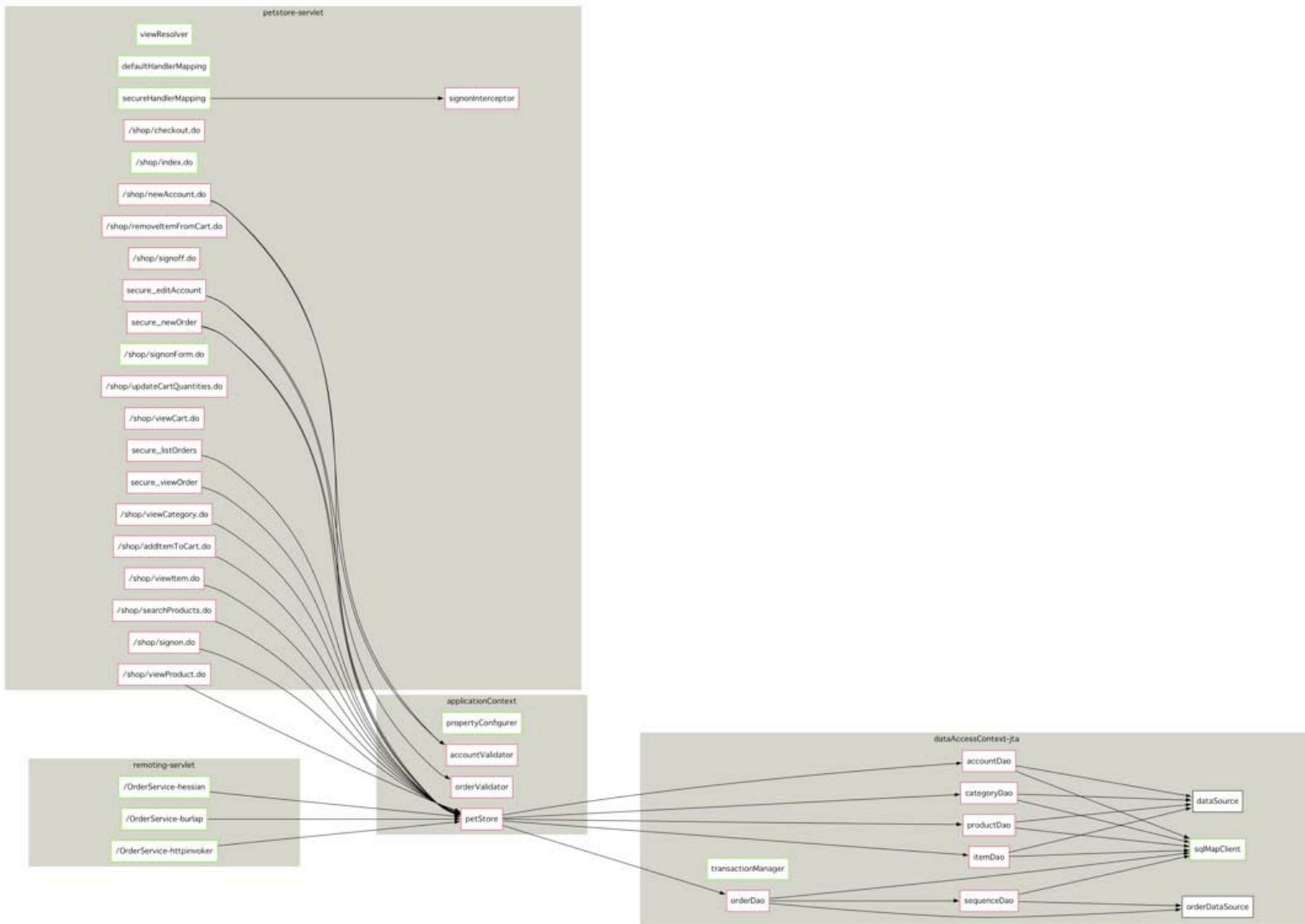
XSLT => DOT for spring
dependencies

<http://www.samoht.com/wiki/wiki.pl?SpringViz>





SpringViz



look for...



regularity

symmetry

overloaded dependencies

isolated pockets

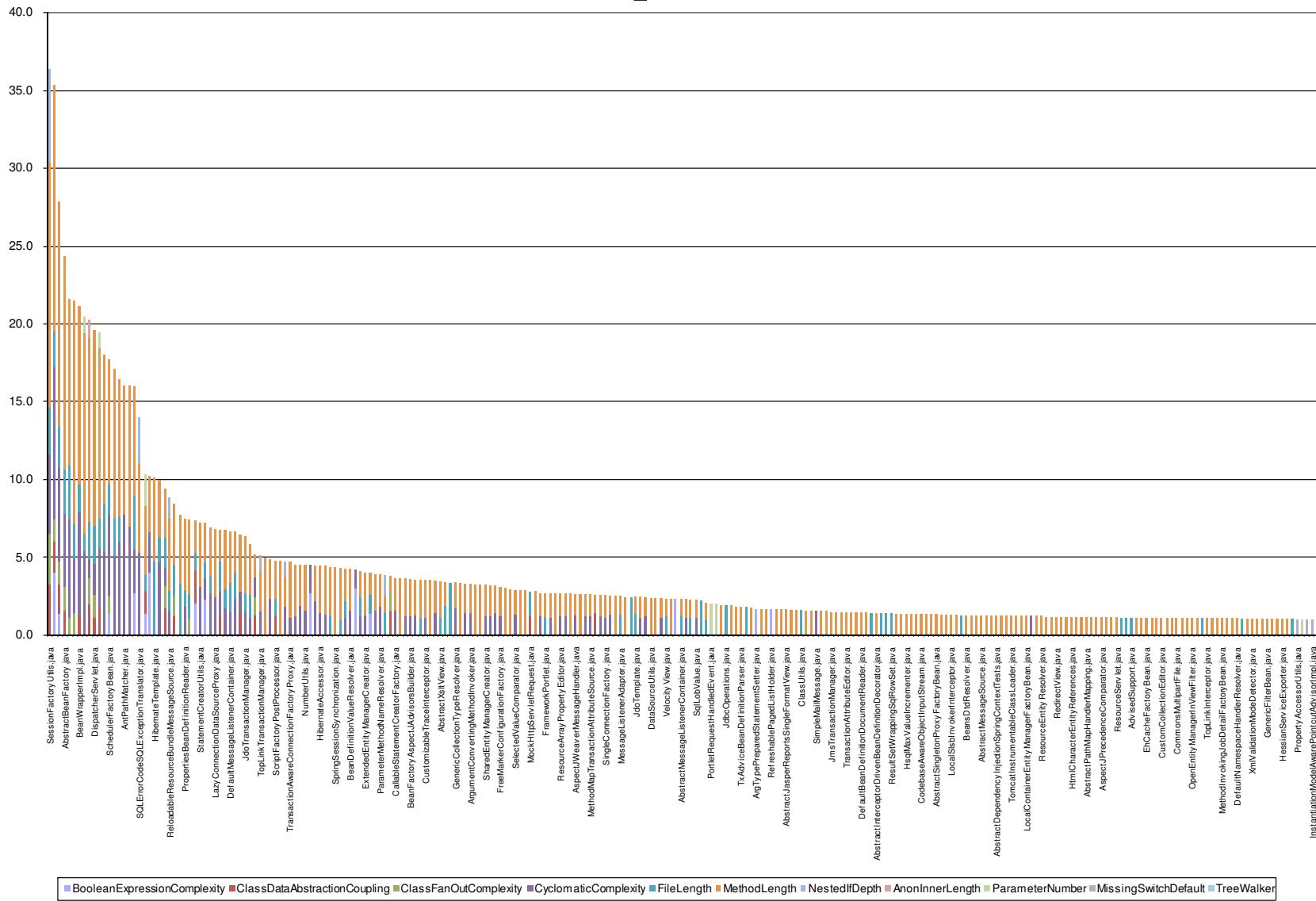
toxicity chart

**provides easy to compare
overview of quality**

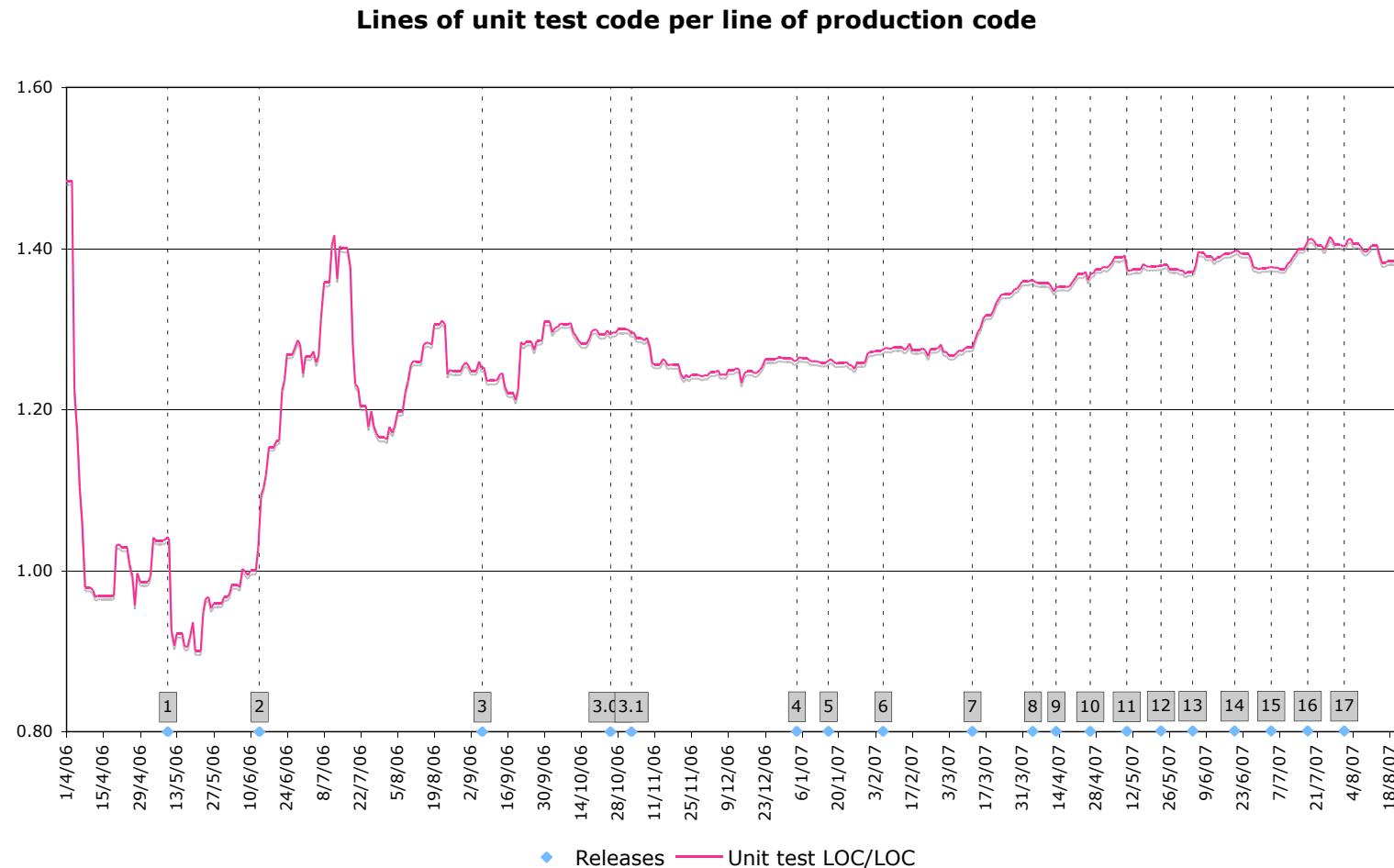
**created with checkstyle +
excel**



toxicity chart

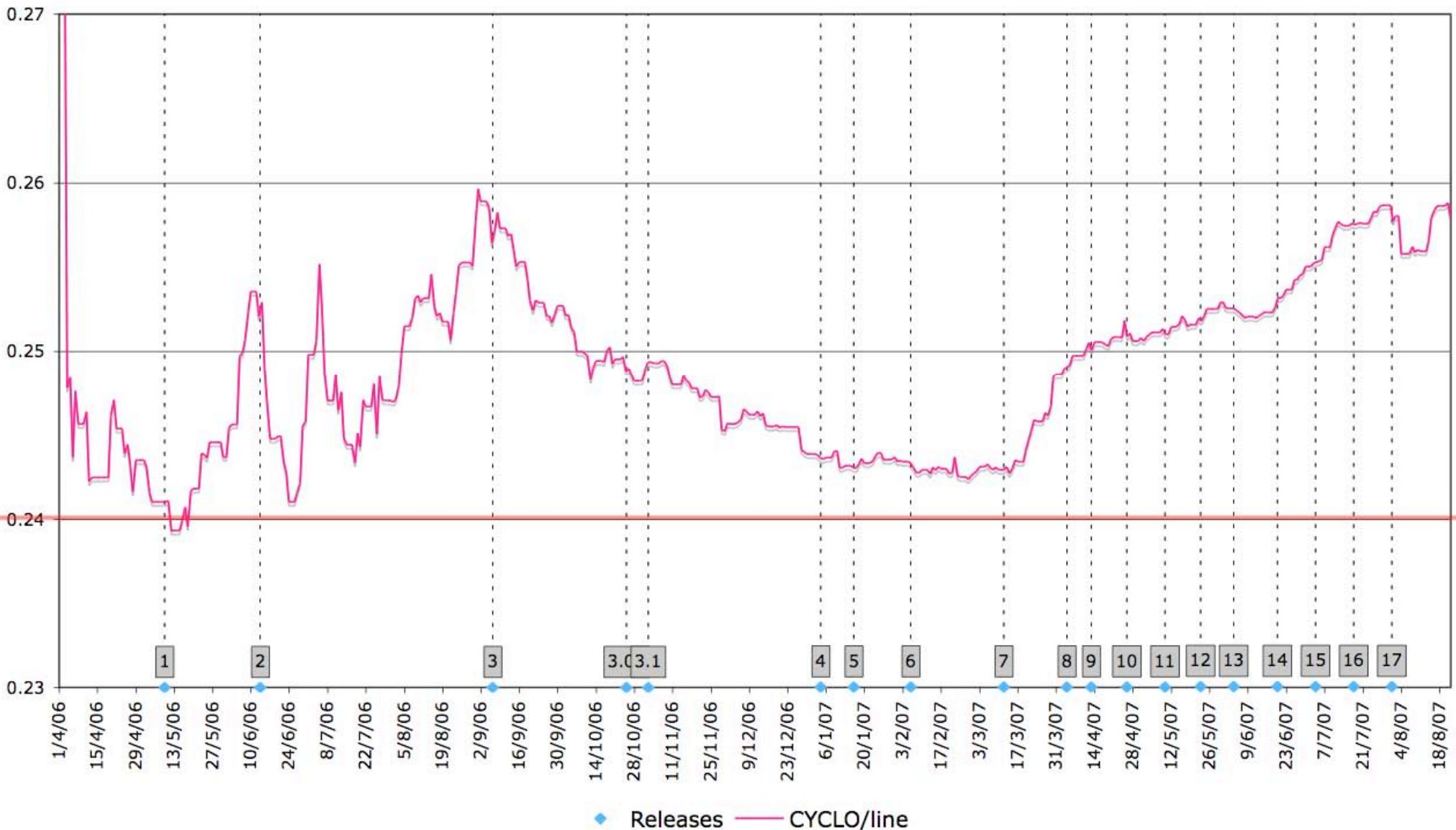


test to code ratio



created using unix tools + excel

Operational Complexity (branching point density)



look for...

notifications along your key dimension

toxicity: high-spikes

test-to-code ratio: higher is better

complexity / loc: trends

deltas more interesting than
raw numbers



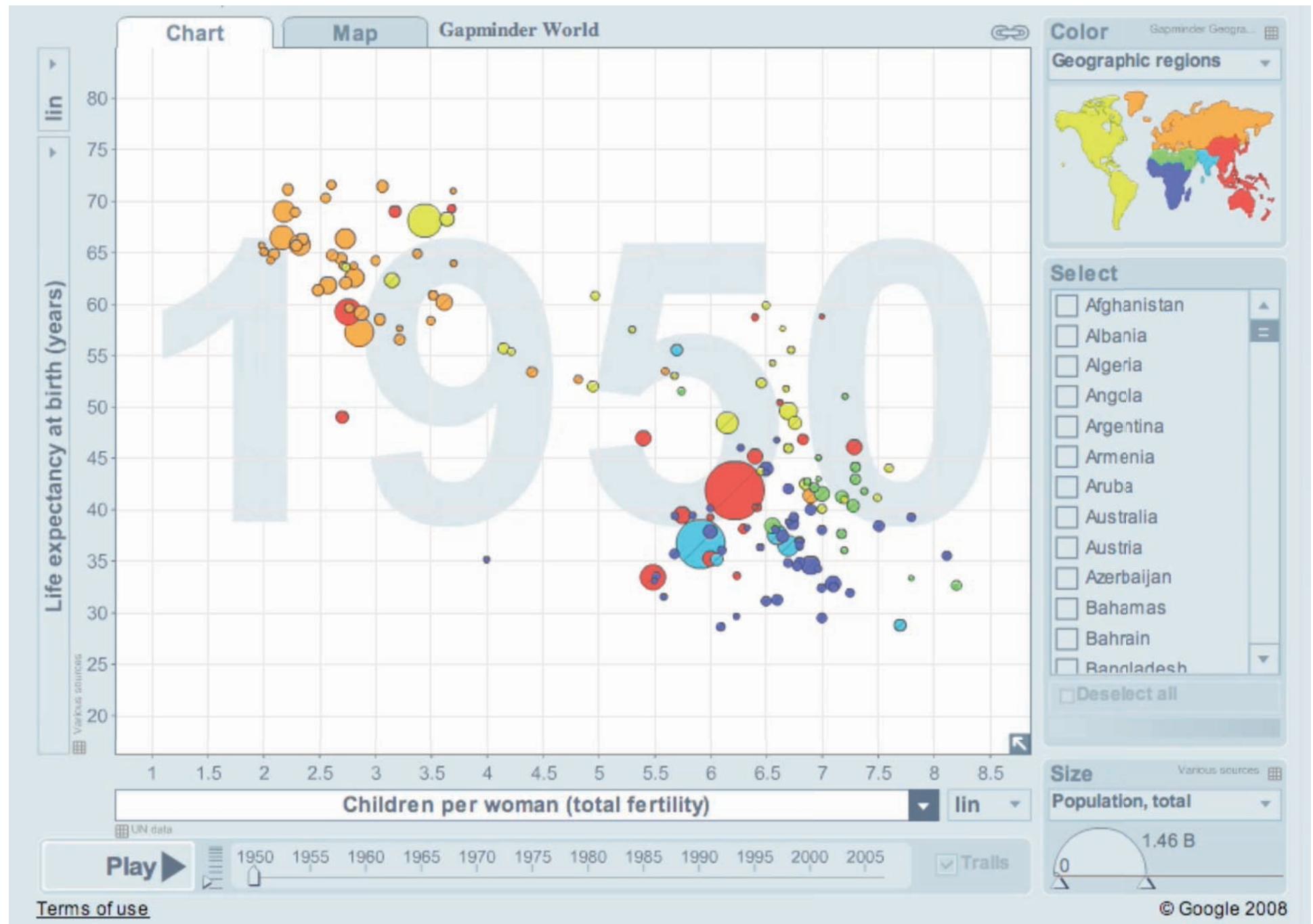


“Unveiling the beauty of statistics for a fact based world view.”

time-based statistical view of chart data

founded in Stockholm by Ola Rosling, Anna Rosling Rönnlund and Hans Rosling

now realized in the google spreadsheet motion gadget



the data

	Version		loc/cc	LOC	CC
1	v1	1999	6.02941176470588	12915	2142
3	v2	2000	6.17486583184258	13807	2236
4	v3	2001	6.17706949977866	13954	2259
5	v4	2002	6.16593886462882	14120	2290
6	v5	2003	6.29637618636756	14595	2318
7	v6	2004	6.22636327971754	15871	2549
8	v7	2005	6.22466281310212	16153	2595
9	v8	2006	6.20398773006135	16180	2608
10	v9	2007	6.17825921702775	16255	2631
11	v10	2008	6.21148725751236	16330	2629

motion chart gadget



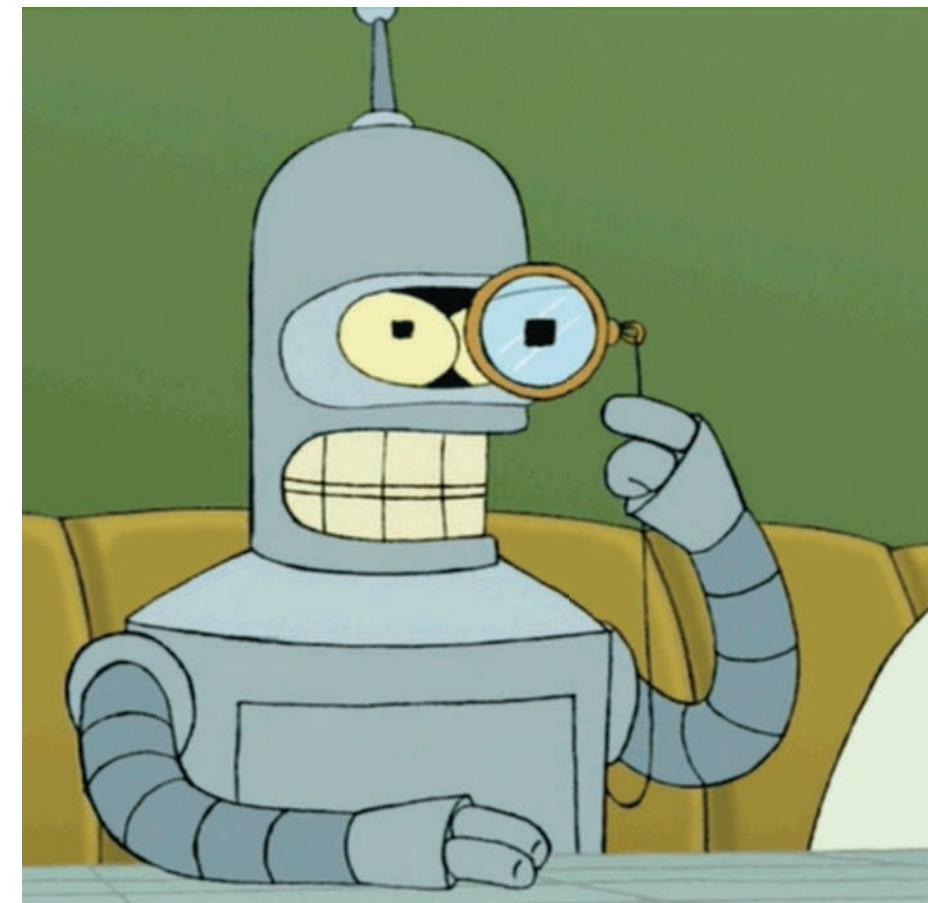
look for...

time-based trends

odd outliers along
dimensions

symmetry

fluidity

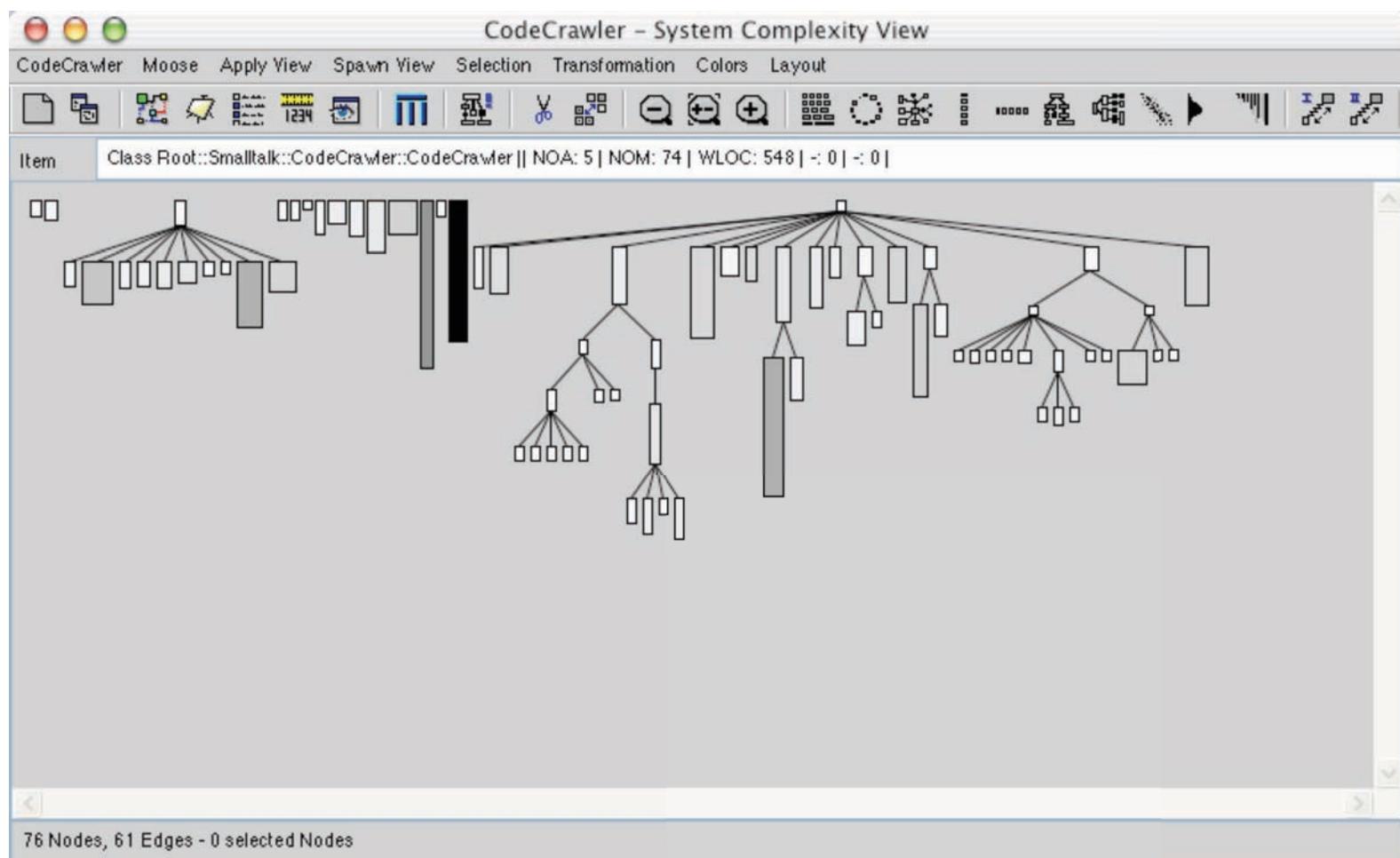


CodeCrawler

www.inf.unisi.ch/faculty/lanza/codecrawler.html



CodeCrawler



CodeCrawler

academic graphical metrics tool

language independent

written in VisualAge Smalltalk

based on the Moose platform

quirky but powerful

x-ray

graduate student project written by Jacopo
Malnati (<http://atelier.inf.unisi.ch/~malnatij/xray.php>)

open-source software visualization plug-in for
eclipse

provides system complexity view, class &
package dependency view

model of the underlying Java project can be
triggered and used by other plug-ins

Java - X-Ray/src/org/malnati/xvplugin/views/ViewFacade.java - Eclipse SDK

```
private Composite parent = null;
// viewPanel
private ScrollPane panel = null;
// tree
private LightweightSystem tree = null;

// constants
public static final String SYSTEM_COMPLEXITY = "system complexity";
public static final String CLASS_DEPENDENCY = "class dependency";
public static final String PACKAGE_DEPENDENCY = "package dependency";
public static final String CURRENT_VISUALIZATION = SYSTEM_COMPLEXITY;

//Handlers
private SystemComplexityHandler systemComplexityHandler = null;
private ClassDependencyHandler classDependencyHandler = null;
```

Javadoc Declaration Search Progress X-Ray [X-Ray] P:19 C:114 M:799 L:9095

x-ray visualizing itself through via system complexity view

using x-ray

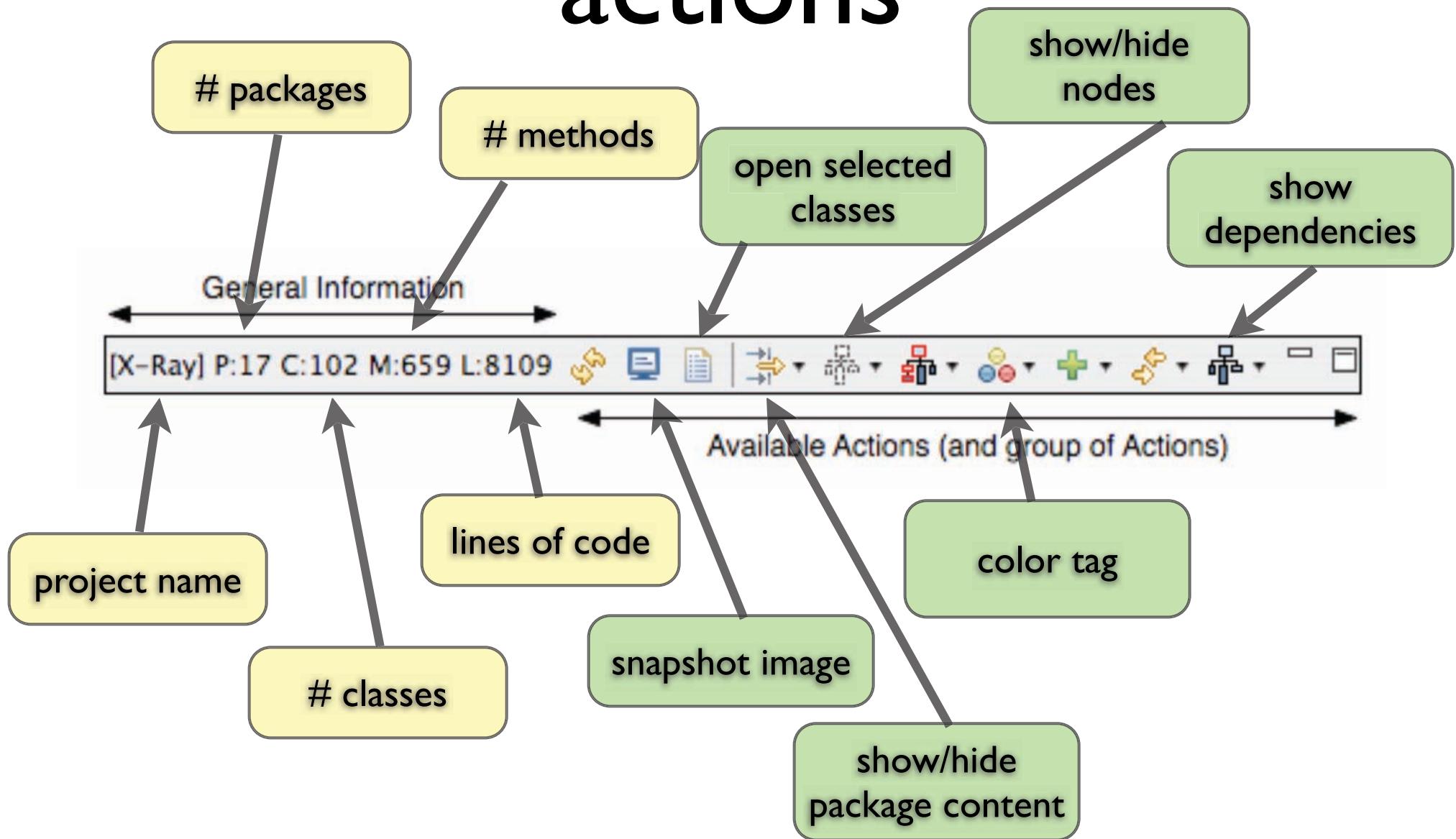
install the plug-in

choose the Analyze Current Project action
from the package explorer

x-ray creates *textual information* and *actions*

visualizations characterized by entities
positioned according to layouts and criterions

actions



polymetric views



system complexity view

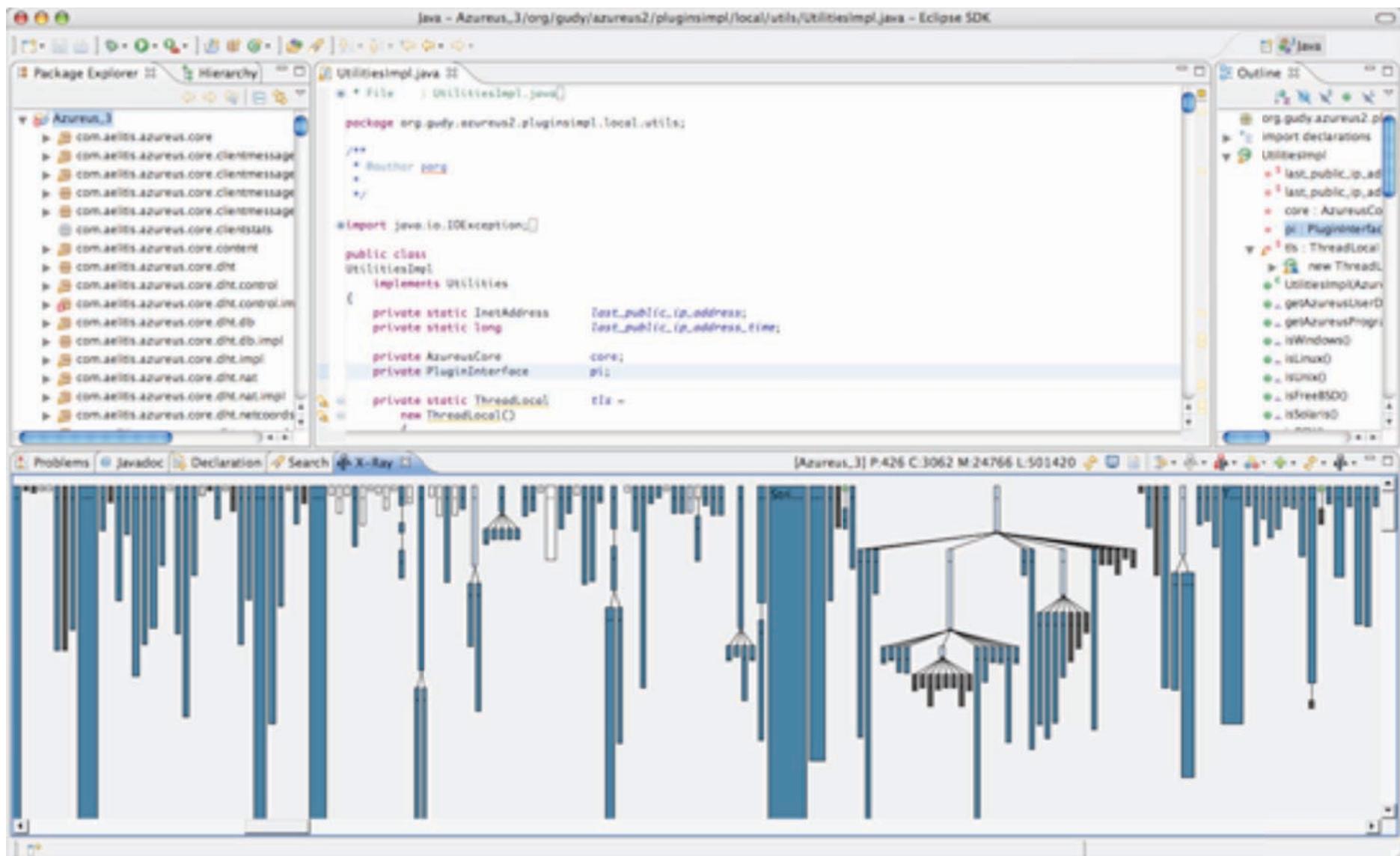


class dependency view



package dependency view

system complexity view



system complexity view

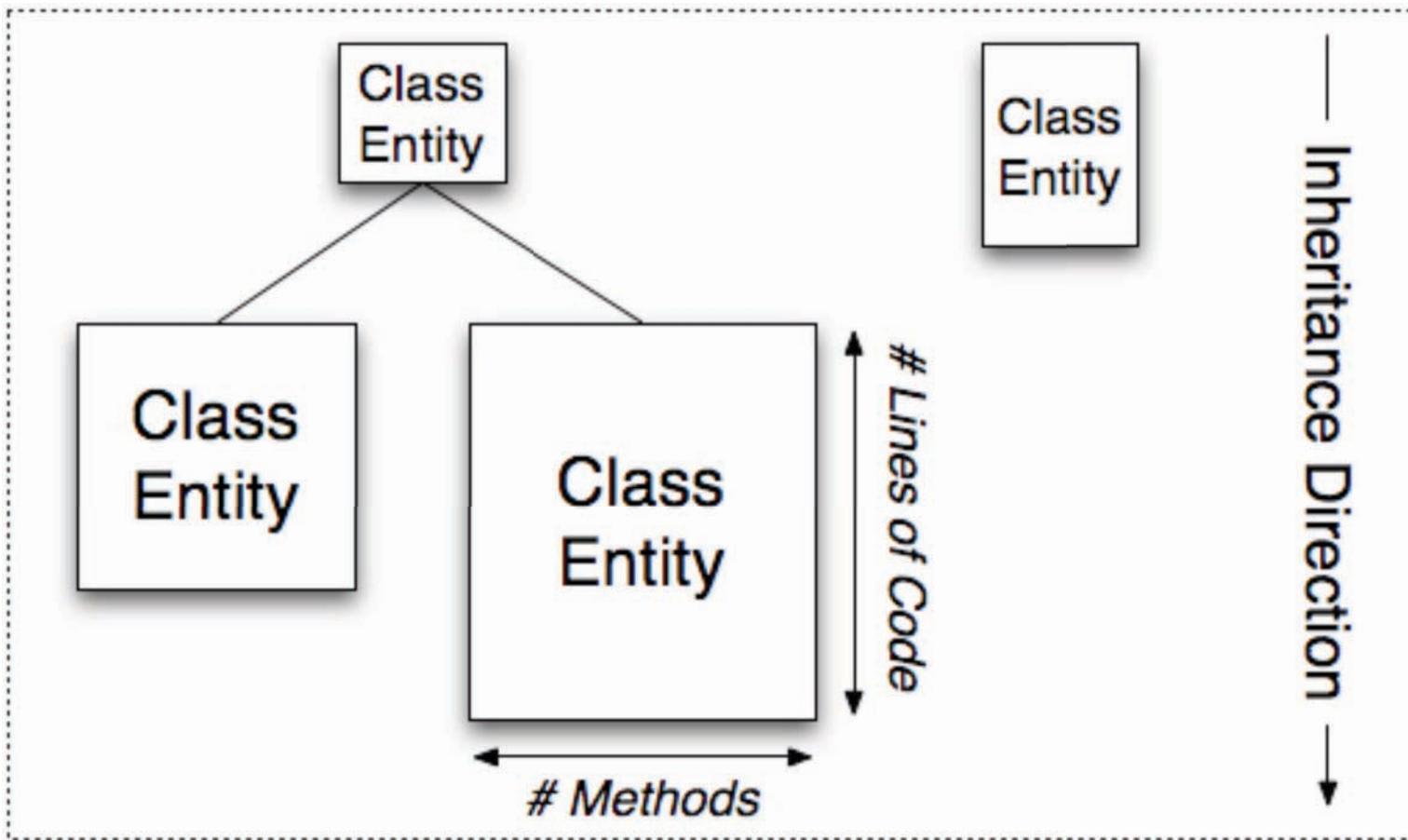
tries to illustrate disharmonies in the design and implementation of a system.

identify big nodes (compared to the others)

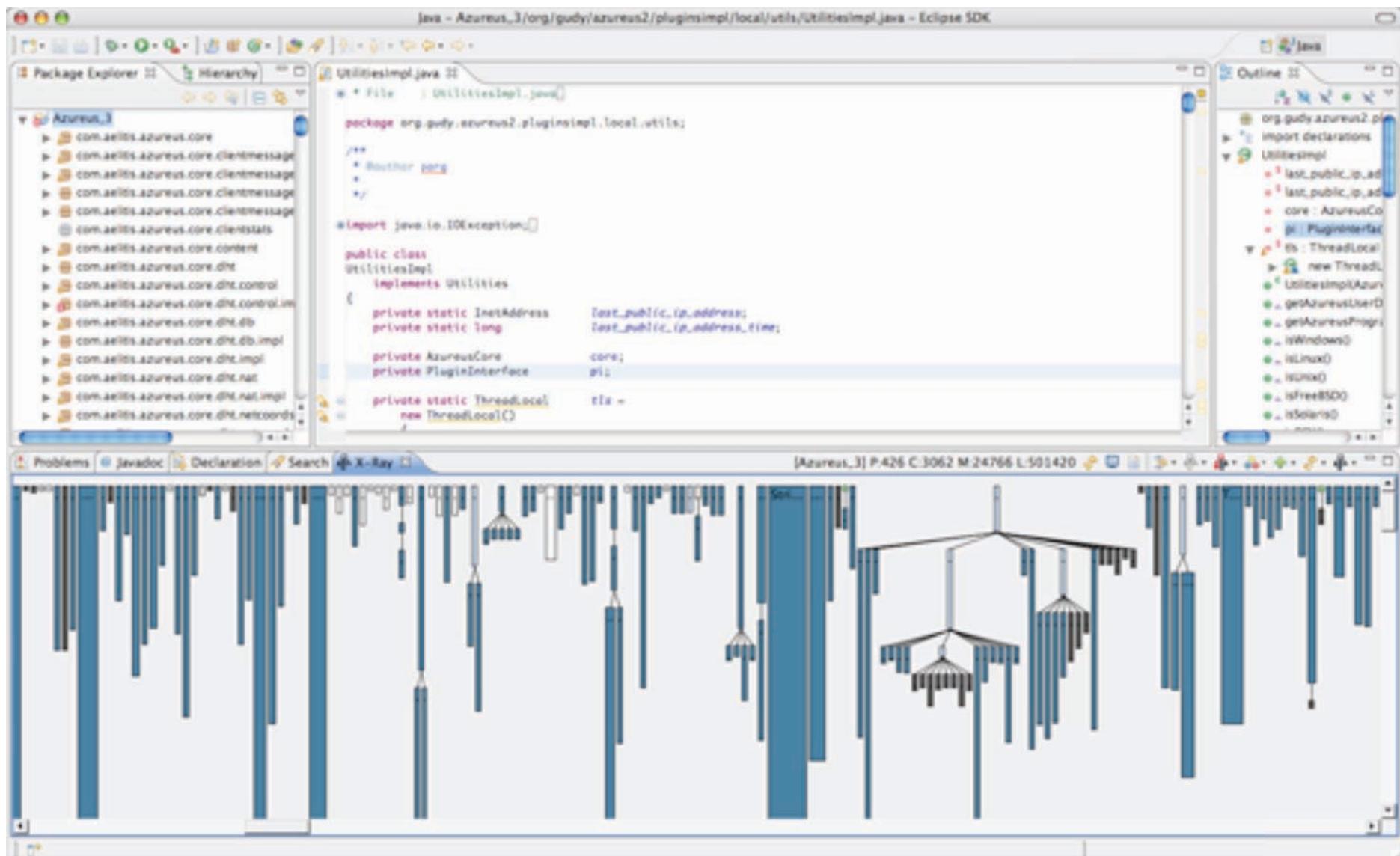
anomalies of shape (provided by the inheritance tree)

view provides several different dimensions of metrics

positional metrics



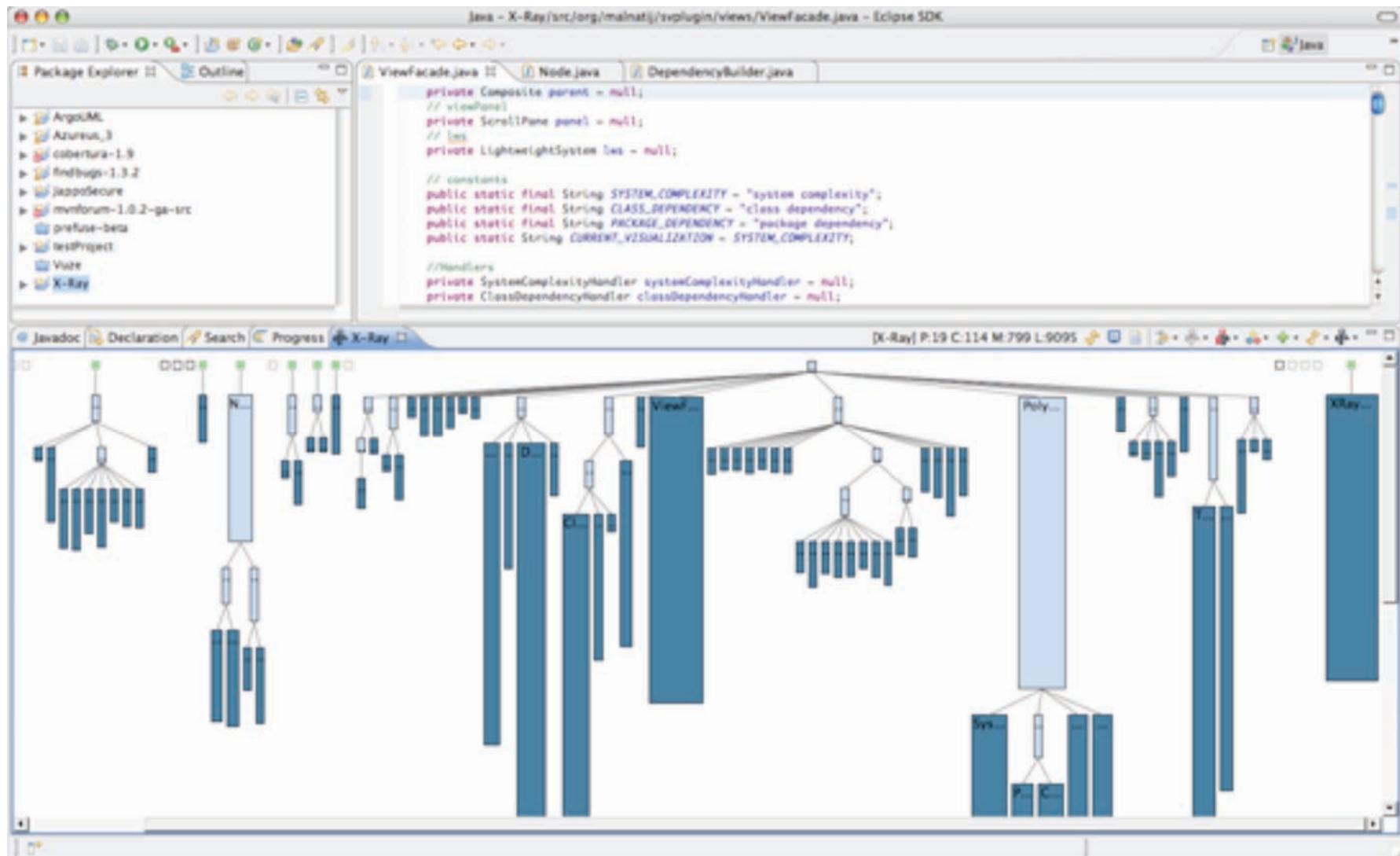
system complexity view



color metrics

- **Dark Blue** implies a concrete class.
- **Light Blue** implies an abstract class.
- **White** implies an interface.
- **Green** implies an external class. By external class we mean a class that is external to the project, while some internal classes are inheriting from it.
- **Light Gray** implies an abstract inner class.
- **Dark Gray** implies a concrete inner class.

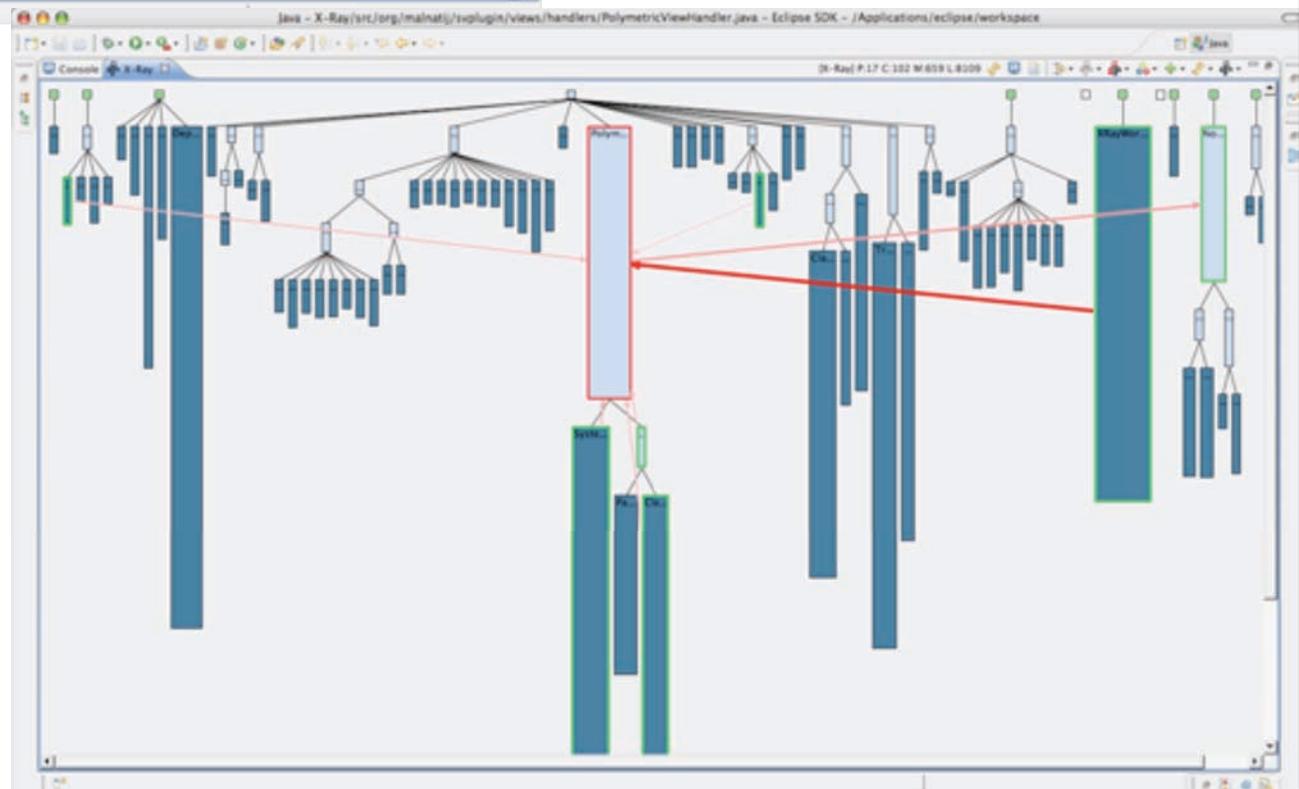
system complexity view



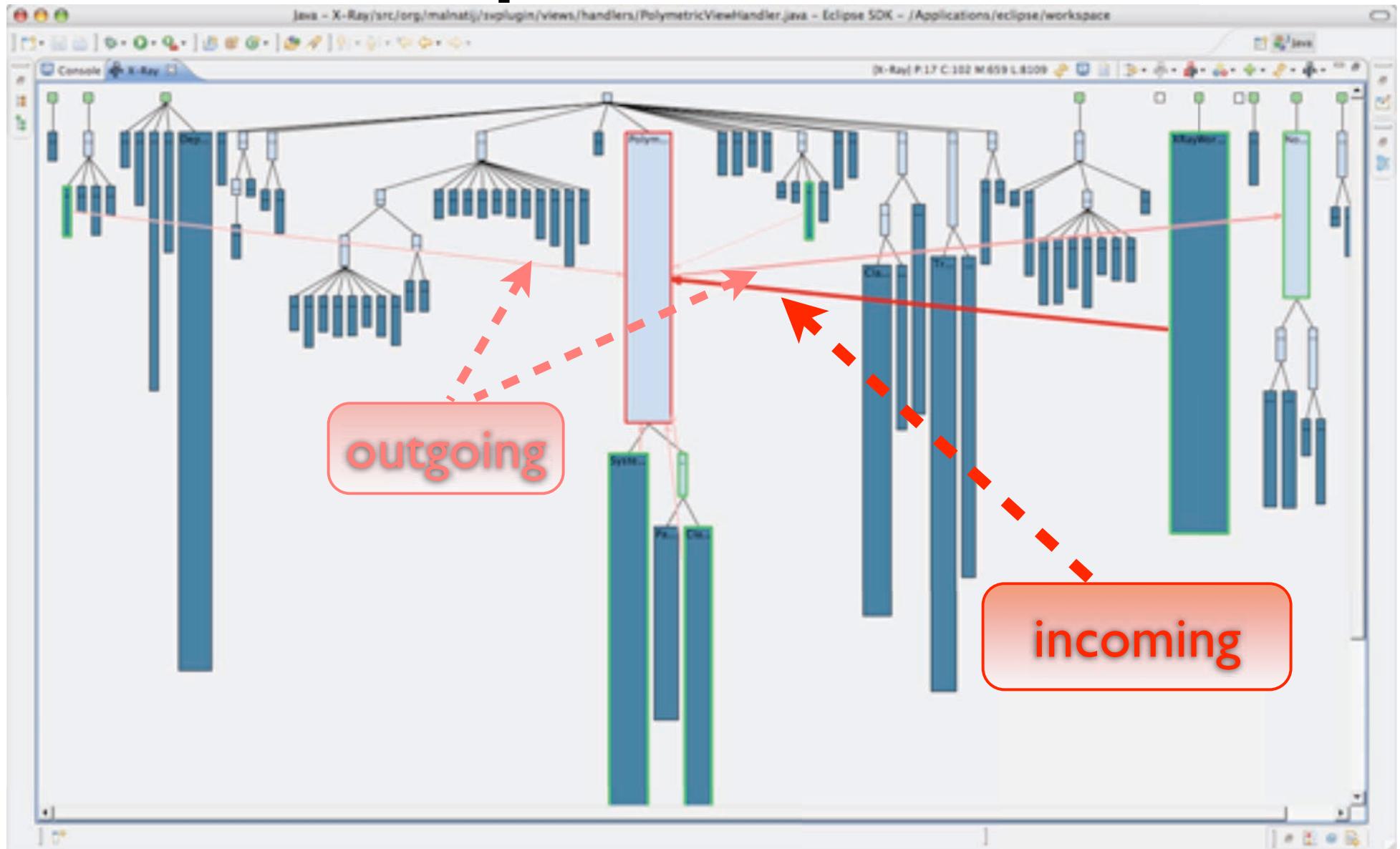
Azureus 3.0 (more than 500,000 lines of code)

The screenshot shows the Eclipse IDE interface with the Azureus 3.0 codebase loaded. The Package Explorer view on the left shows the project structure with packages like com.aelitis.azureus.core, com.aelitis.azureus.core.clientmessage, and com.aelitis.azureus.core.dht. The UtilitiesImpl.java file is open in the editor, showing Java code implementing the Utilities interface. The Outline view on the right shows the class hierarchy and member variables. The X-Ray view at the bottom displays a complex call graph with many nodes and connections, representing the internal structure of the application.

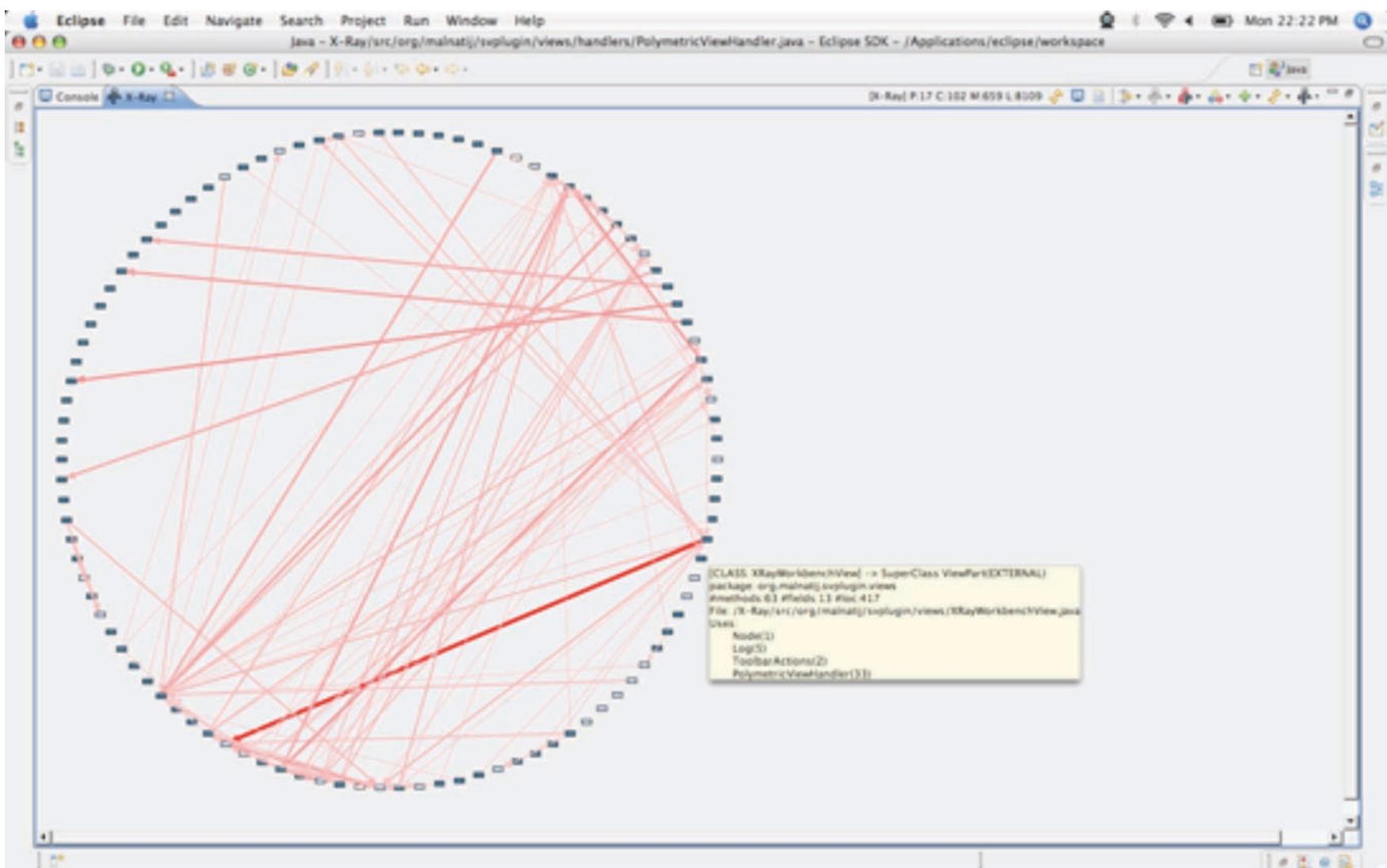
x-ray itself



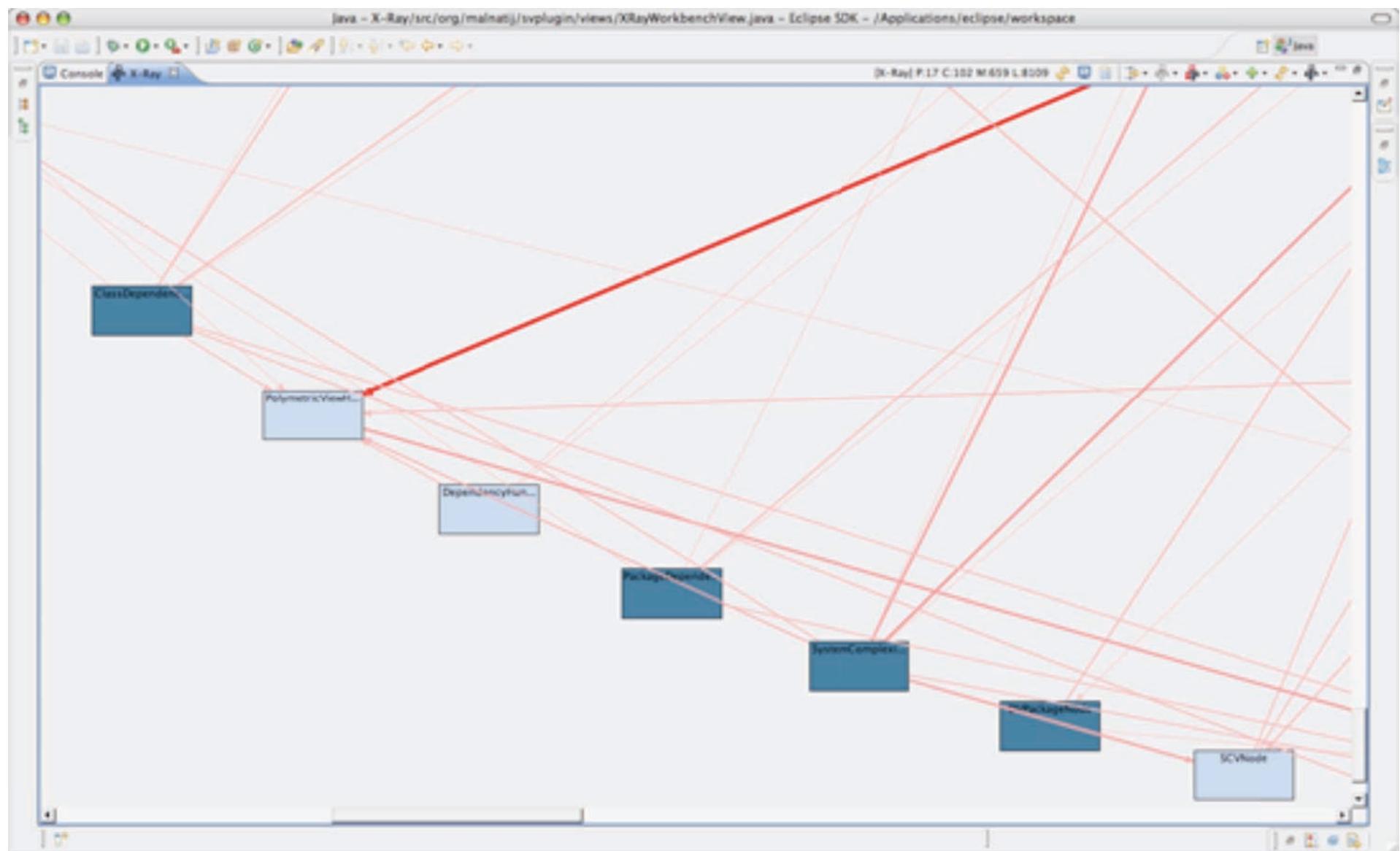
dependencies



class & package dependency views



class dependency view



package dependency

Java - X-Ray/src/org/malinatj/svplugin/SVPluginActivator.java - Eclipse SDK

Package Explorer Hierarchy

SVPluginActivator.java

```
package org.malinatj.svplugin;

import org.eclipse.jface.resource.ImageDescriptor;

/**
 * The activator class controls the plug-in life cycle
 */
public class SVPluginActivator extends AbstractUIPlugin {

    // The plugin ID
    public static final String PLUGIN_ID = "org.malinatj.svplugin";

    // The shared instance
    private static SVPluginActivator plugin;
```

Problems Javadoc Declaration Search X-Ray [X-Ray] P:17 C:102 M:659 L:8109

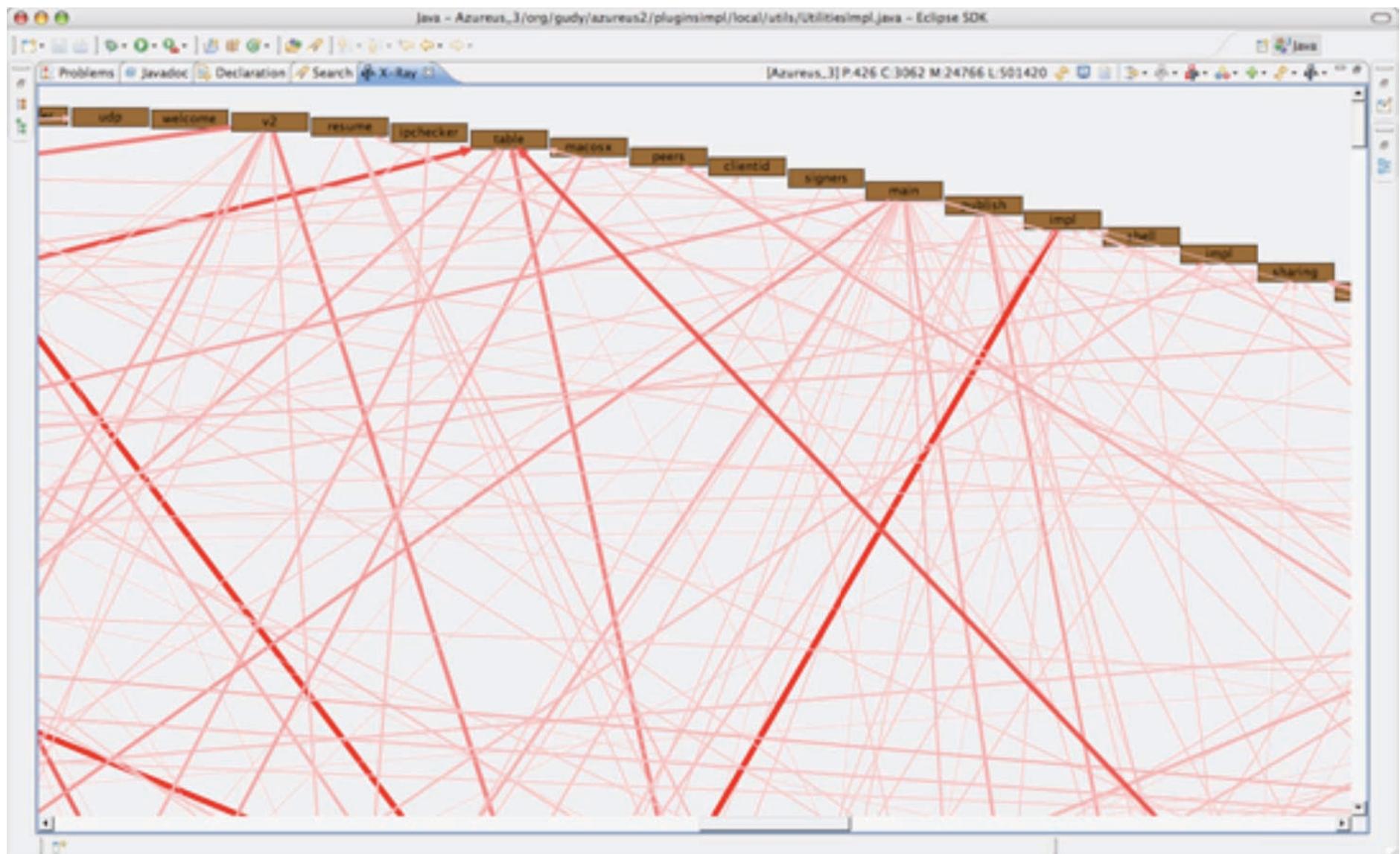
Outline

actions ICR cascade layouts filters svplugin views mapping

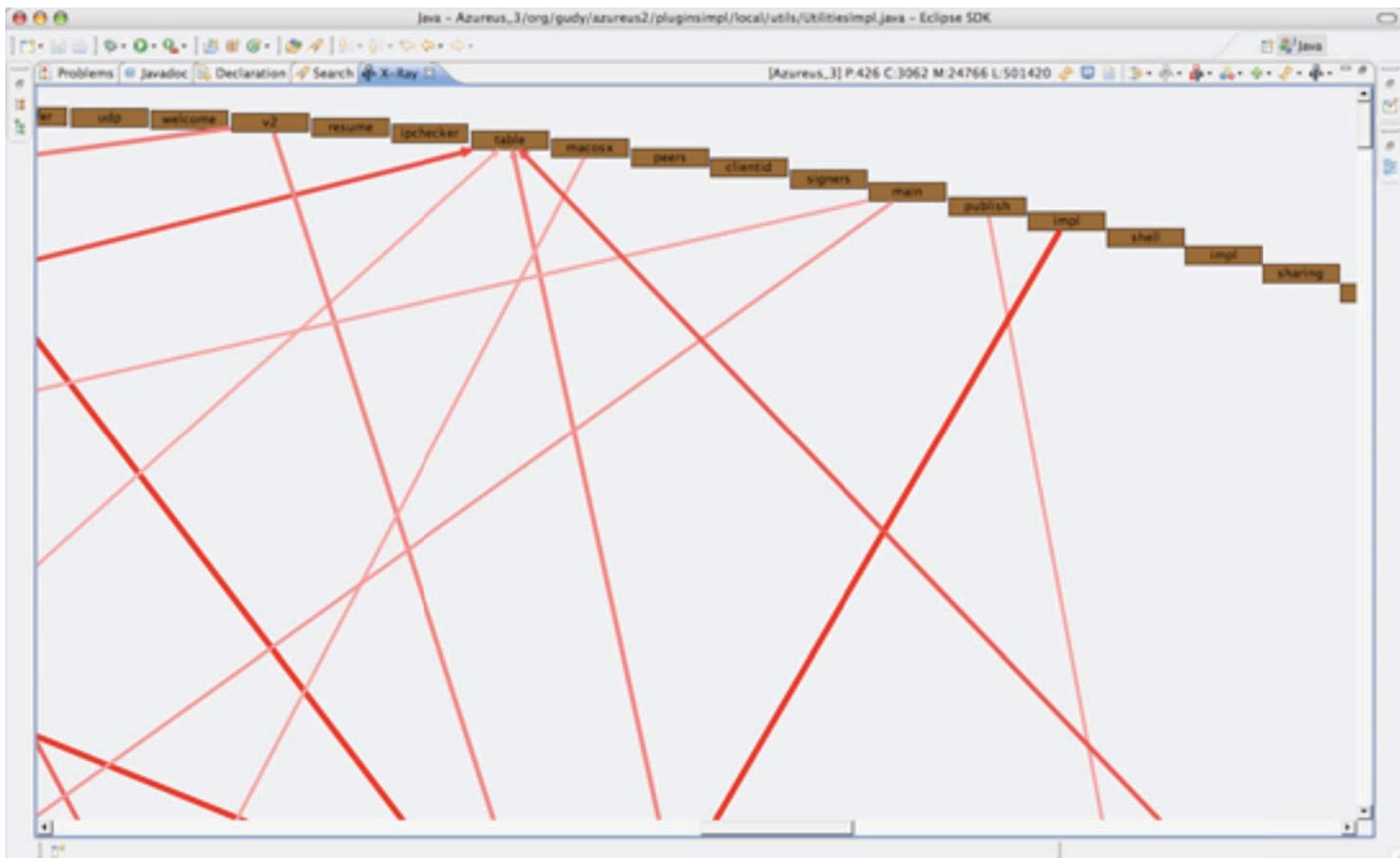
links providers validators util handlers nodes dialogs menuitems actions

org.malinatj.svplugin.SVPluginActivator.java/org.malinatj.svplugin - X-Ray/src

Azureus packages



filtering (< 30 weight)



proximity alert

The screenshot shows the X-Ray Eclipse plugin interface. The top part displays the Java code for `ClassRepresentation.java` in the Package Explorer. The right side shows the General Information and Class Analysis Settings panes. The bottom part is the Classes Viewer, which lists various classes with their proximity alert scores and other metrics.

General Information

Category	Value
Project	X-Ray
Packages	19 of 19
Classes	134 of 134
Interfaces	4 of 4
Fields	213 of 213
Methods	761 of 761
Dependencies	234 of 234
Lines of code	8876 of 8876

Class Analysis Settings

Classes Viewer

Class Name	Package	Proximity Alert	Fields	i-Fields	Methods	I-M. odds	Constr.	i-Constr.	Uses	MsgSent	Used	MsgRec	Loc
ZoomItemMenuProvider	org.malinatj.vplugin.views.actions.menuitems.providers	27.0	0	21	2	0	1	1	0	0	0	0	38
ZoomAction	org.malinatj.vplugin.views.actions	21.0	0	0	2	1	1	1	0	0	0	0	45
XRayWorkbenchView	org.malinatj.vplugin.views	299.0	6	0	61	0	1	1	5	57	12	18	117
XRay	org.malinatj.vplugin	7.0	1	0	0	0	1	0	0	0	0	0	12
ViewFilter	org.malinatj.vplugin.model.view.communication	14.0	2	1	2	0	1	1	1	1	0	0	20
ViewFacade	org.malinatj.vplugin.views	366.0	18	1	33	0	1	1	6	12	1	22	246
ViewActionDelegate	org.malinatj.vplugin.actions	23.0	1	1	1	0	1	1	1	3	0	0	38
UpAnchor	org.malinatj.vplugin.graph.links	11.0	0	0	2	1	1	1	0	0	0	0	18
UntagMenuItemProvider	org.malinatj.vplugin.views.actions.menuitems.providers	17.0	0	2	2	1	1	0	0	0	0	0	27
UniquelyIdentifiableObject	org.malinatj.vplugin.model	5.0	0	0	1	0	1	0	0	0	0	0	8
TreeLayout	org.malinatj.vplugin.layouts	268.0	5	7	27	6	1	1	5	21	1	10	457
ToolbarActions	org.malinatj.vplugin.views.actions	42.0	3	1	5	0	1	1	1	1	2	3	66
TicksPredictor	org.malinatj.vplugin.model.core	15.0	1	0	3	0	1	1	0	0	1	1	62
SystemHotspotHandler	org.malinatj.vplugin.views.handlers	381.0	0	9	23	49	1	1	0	0	0	0	143
SystemComplexityHandler	org.malinatj.vplugin.views.handlers	337.0	4	9	42	49	1	1	8	35	1	1	547
StringValidator	org.malinatj.vplugin.views.actions	13.0	0	1	0	1	1	1	0	0	0	0	22
SnapshotWarning	org.malinatj.vplugin.views.actions.dialogs	18.0	0	0	1	0	1	1	1	1	0	0	31
SnapshotAction	org.malinatj.vplugin.views.actions	53.0	1	1	7	1	1	1	2	3	0	0	91
SizedArrowLink	org.malinatj.vplugin.graph.links	17.0	0	6	2	5	1	1	2	2	0	0	21

look for...

towers (=> lots of code, lots of methods)

tangled dependencies

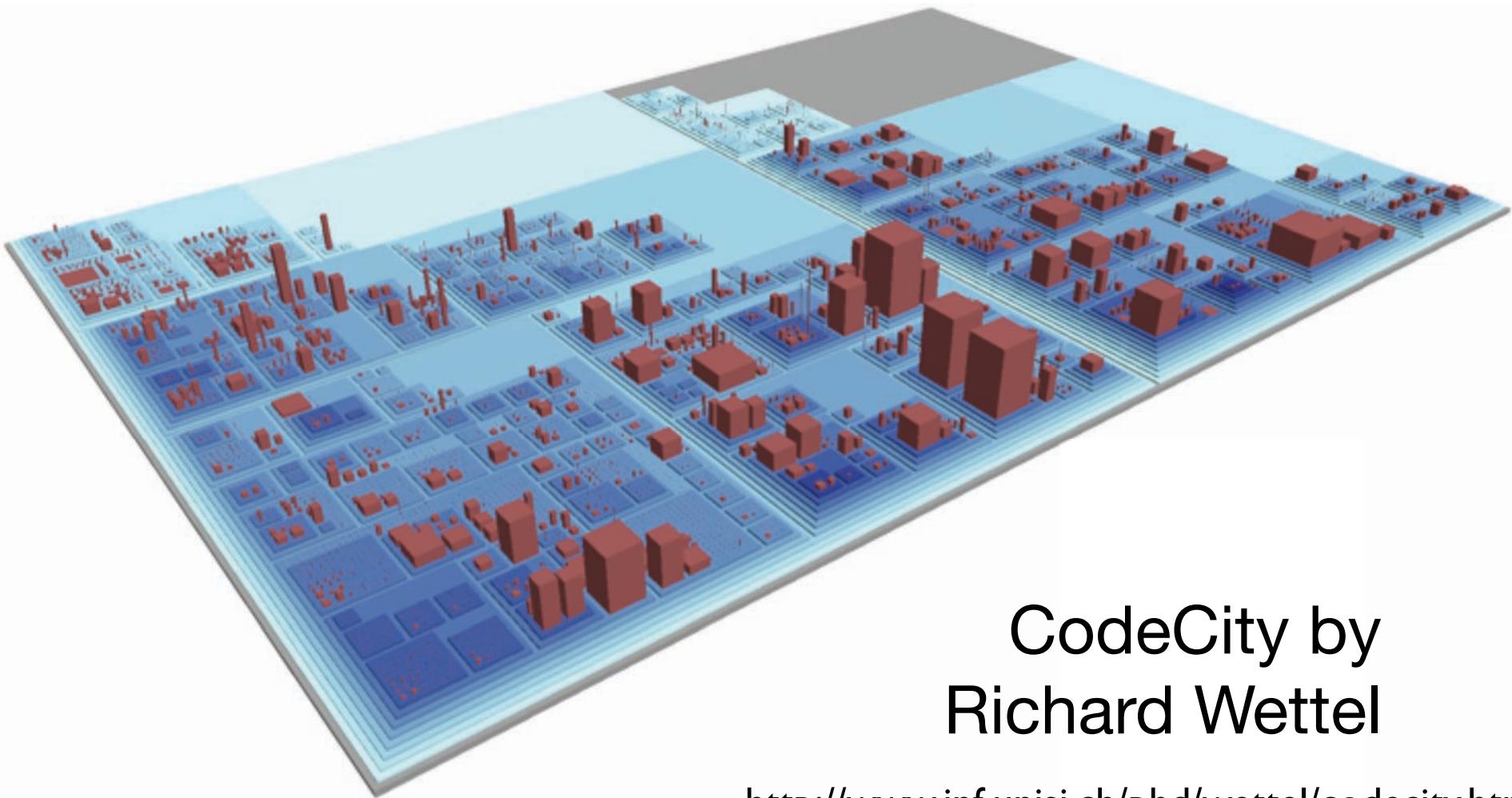
exuberant responsibility

natural partitions

balance



10,000 ft view (literally)



CodeCity by
Richard Wettel

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

CodeCity

integrated environment for software analysis

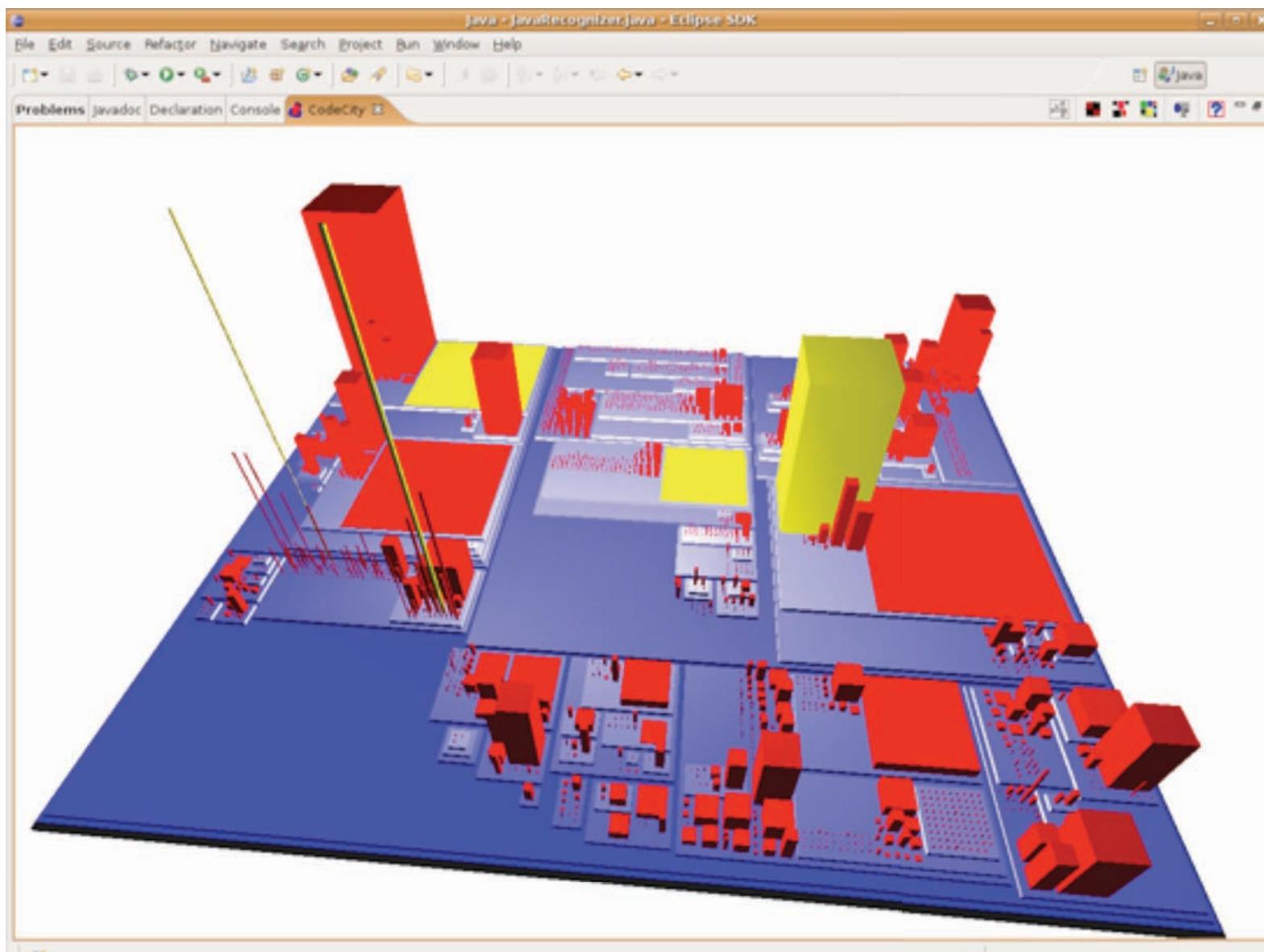
software systems are visualized as interactive,
navigable 3D cities

written in VisualWorks Smalltalk, atop the
Moose platform

classes => buildings

packages => districts

citylyzer



Citylyzer

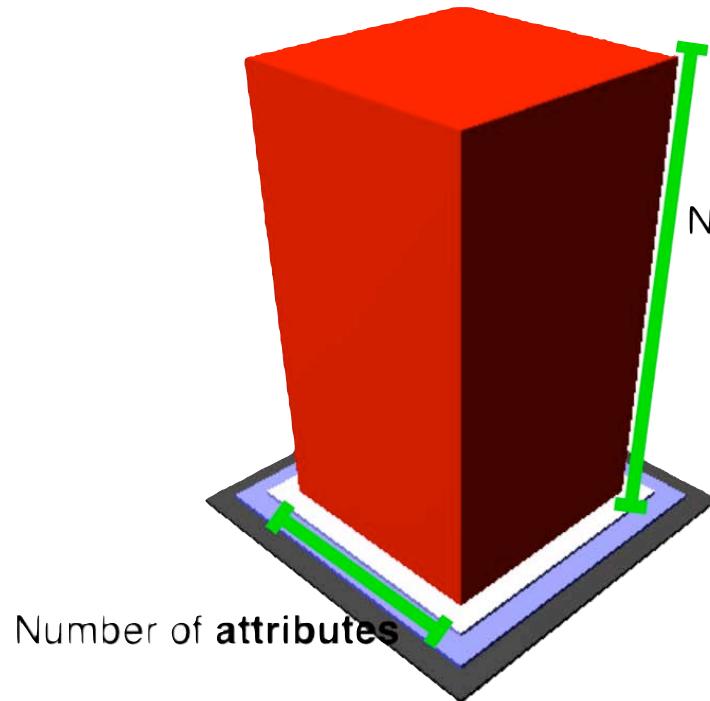
written atop x-ray

inspired by CodeCity

building height => number of methods

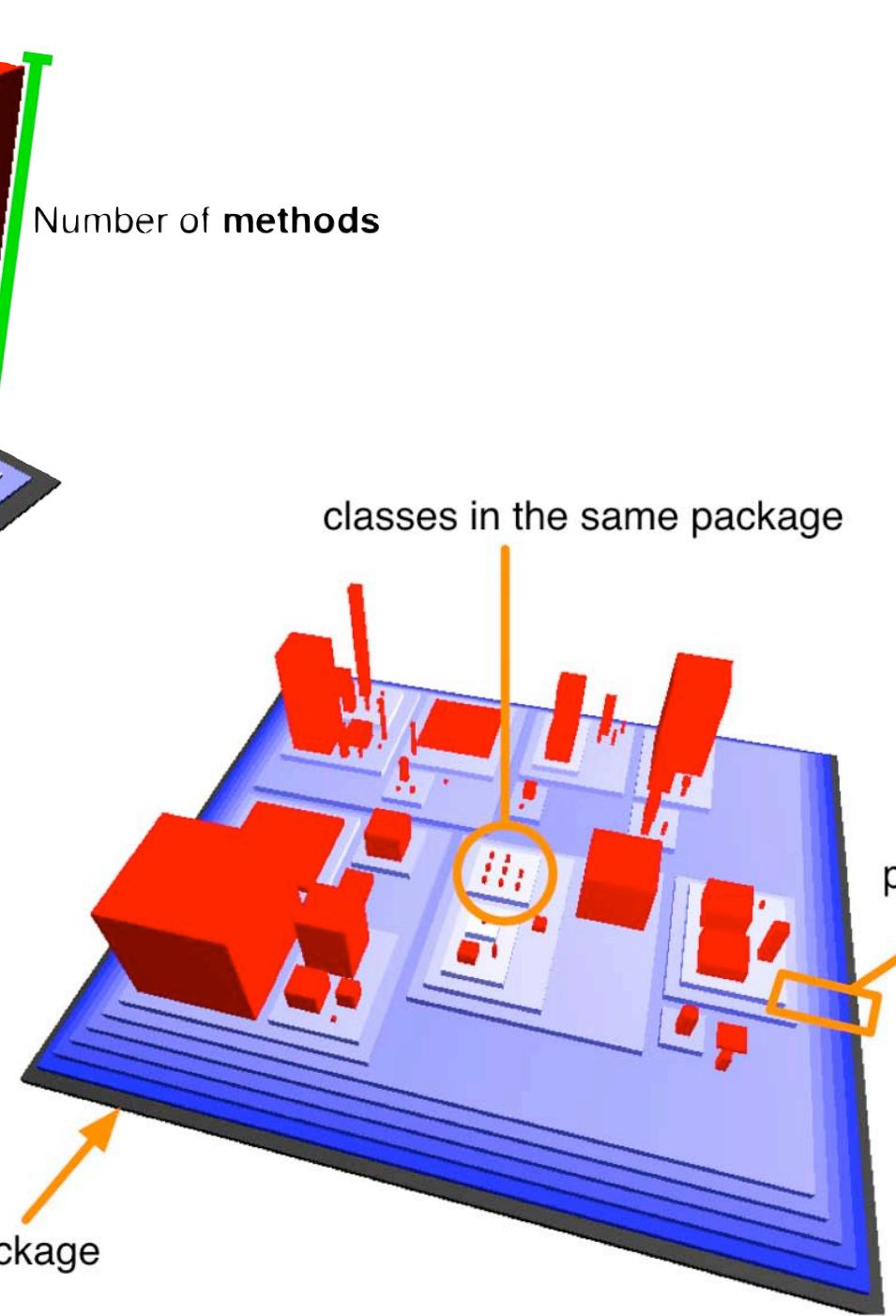
width/length => number of attributes

packages => districts



Number of **methods**

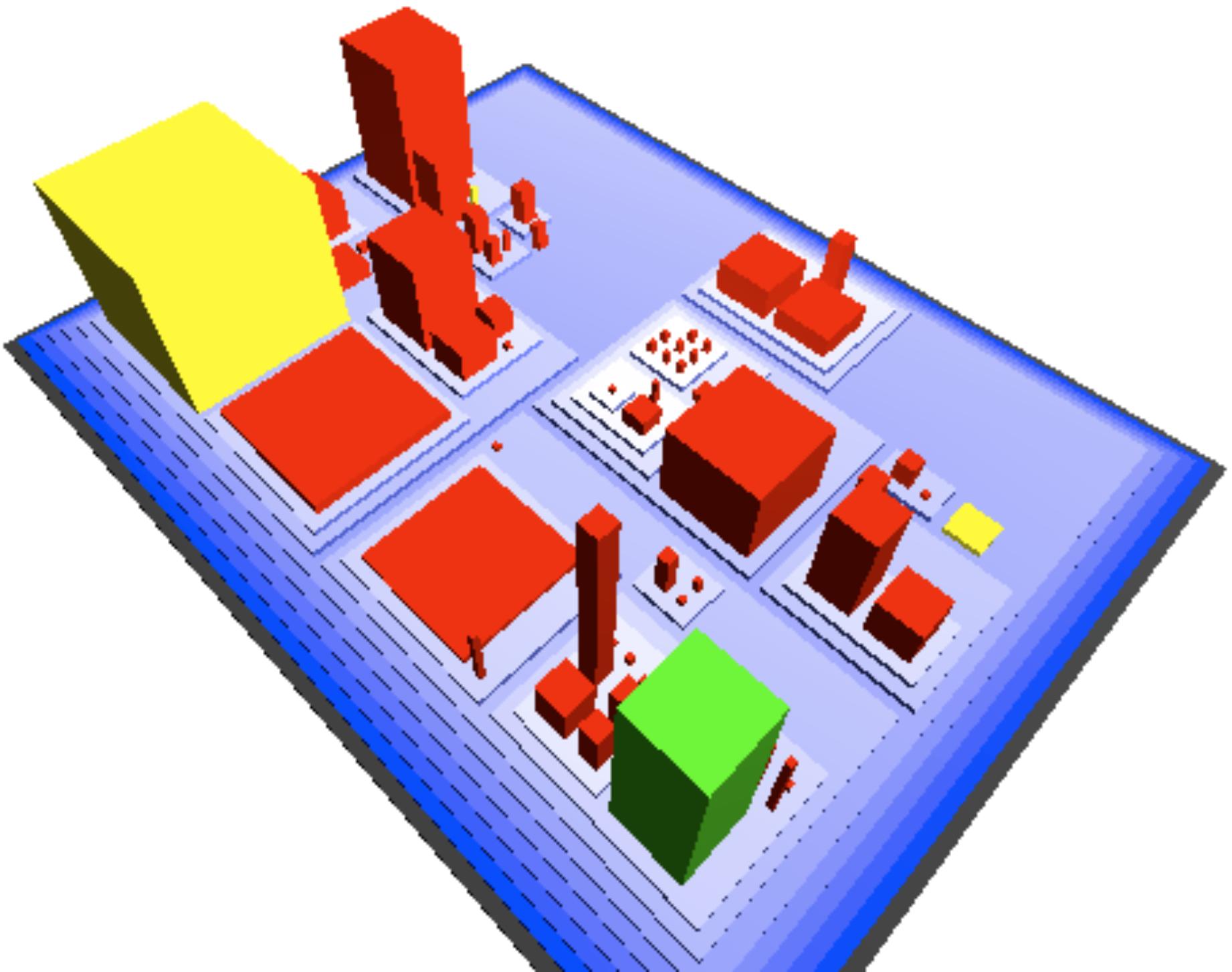
Number of **attributes**

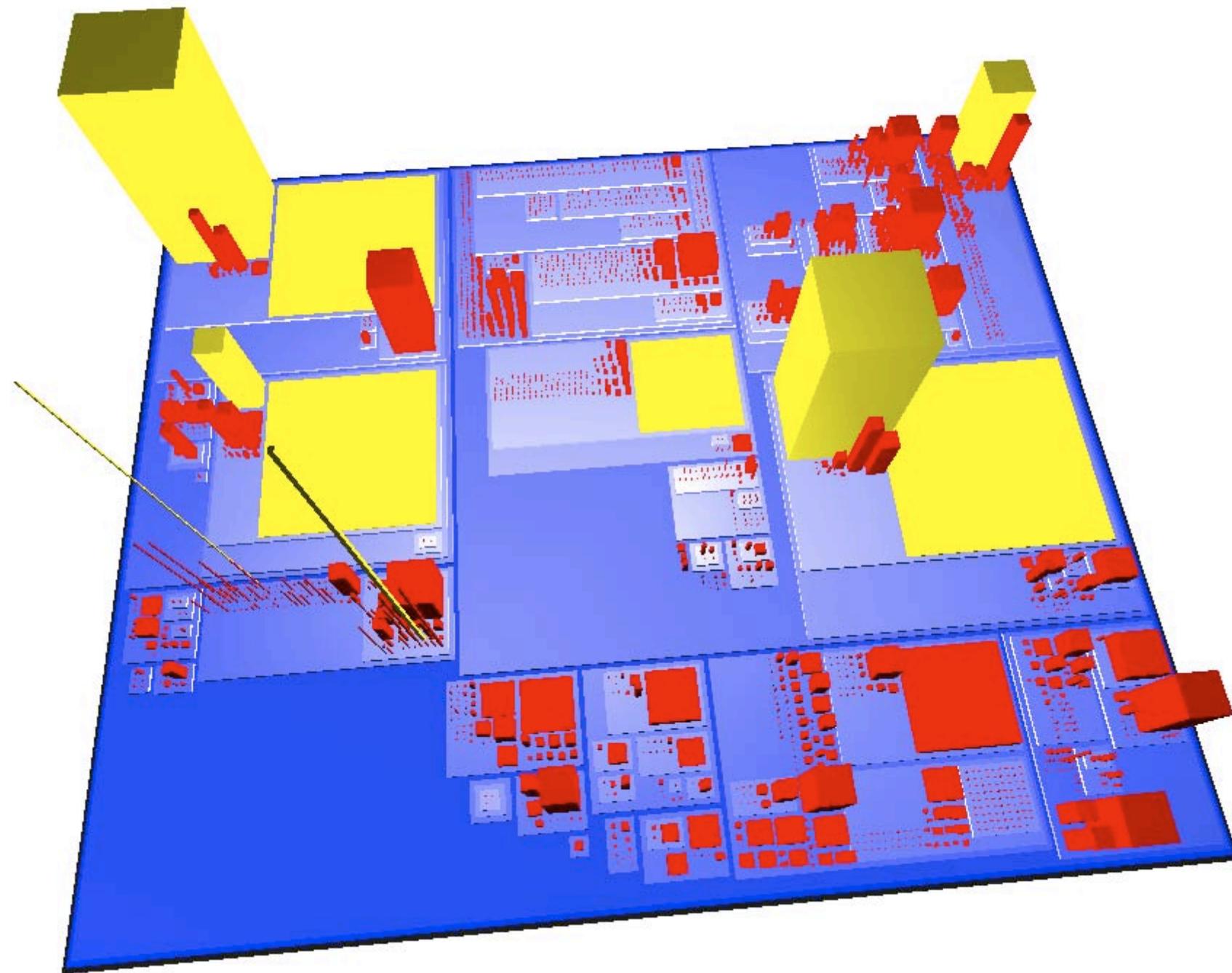


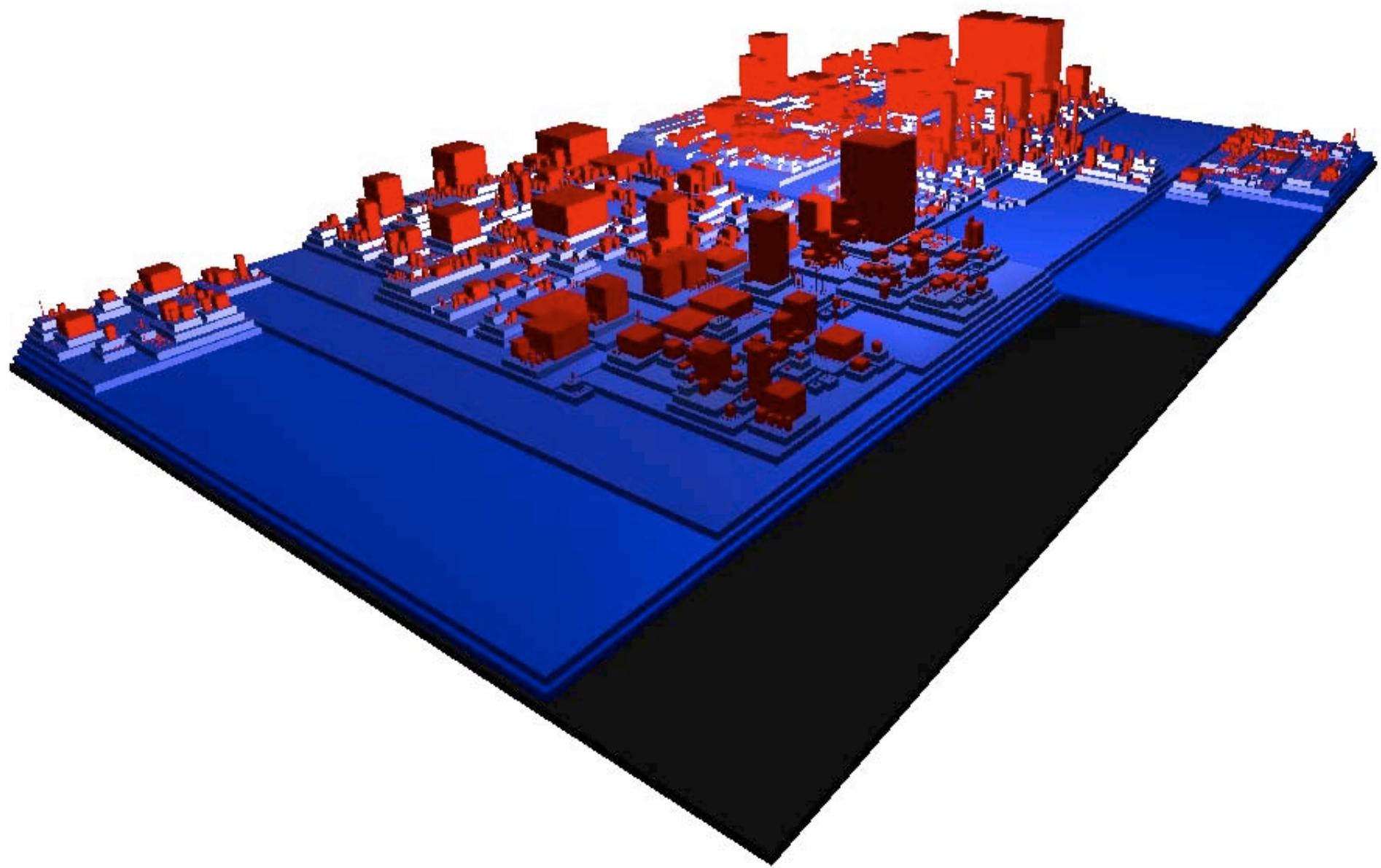
classes in the same package

packages hierarchy

Root package







look for...

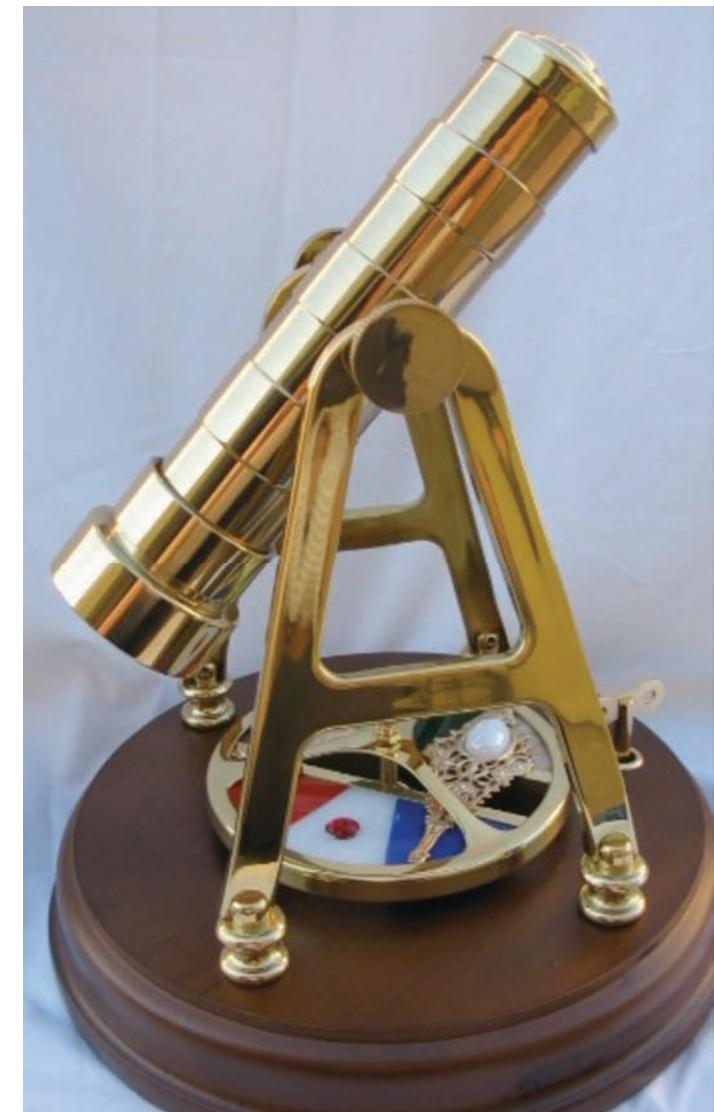
“real” cityscape

buildings that wouldn’t
exist in the real world

overly crowded neighborhoods

abandoned parts of town

would you live there?



metrics + agility

wire metrics & analysis into
continuous integration

manually check often

discuss on iteration boundaries

fail the build with:

xpath expressions & plugins
jdepend-like unit tests



summary

intermediate altitude

information radiators

single dimensions

compelling evidence

cool!

? , S

please fill out the session evaluations
samples at github.com/nealford



This work is licensed under the Creative Commons
Attribution-Share Alike 3.0 License.

<http://creativecommons.org/licenses/by-sa/3.0/us/>

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
blog: memeagora.blogspot.com
twitter: neal4d