# CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2009
Doc 21 Metrics
28 Apr 2009

# References


Cyclomatic complexity, http://en.wikipedia.org/wiki/Cyclomatic_complexity

Lines of Code, http://en.wikipedia.org/wiki/Source_lines_of_code

Eclipse Metrics, http://metrics.sourceforge.net/

Specialization Index, http://semmle.com/documentation/semmlecode-glossary/specialization-index-of-a-type/

OO Design Quality Metrics: An Analysis of Dependencies, Robert Martin, http://www.objectmentor.com/resources/articles/oodmetrc.pdf

Source code for twitter4j, http://yusuke.homeip.net/twitter4j/en/index.html

Eclipse Metrics Plugin, http://eclipse-metrics.sourceforge.net/

Object-Oriented Metrics: Measures of Complexity, Brian Henderson-Sellers, Prentice Hall, 1996

# Metrics

DeMarco's Principle

Effort moves toward whatever is measured




3

# The Swedish Army Dictum

When the map and the territory don't agree, always believe the territory.



4

# Eclipse Metrics 1.3.6

Eclipse plugin

Docs

http://metrics.sourceforge.net/

Source Forge Site

http://sourceforge.net/projects/metrics

Generates about 20 metrics
Displays result in tables in Eclipse
Generates dependency graphs



5

# Eclipse Metrics Plugin

http://eclipse-metrics.sourceforge.net/

Author: Lance Walton

Generates about same metrics as Metrics 1.3.6
Exports results to html or csv
Generates table and graphs



6

# Lines Of Code

SLOC

Rough measure of size

Effort is highly correlated with SLOC

Physical SLOC

- Code + comments + blank lines
- Not count blank lines over 25% of a section
- Eclipse Metrics - calls this Total Lines of Code (TLOC)

Logical SLOC

- Just lines of actual code
- Eclipse Metrics
  - calls this Method Lines of Code (MLOC)
  - But only code inside method bodies



7

# Basic COCOMO

Software Cost Estimation Model

Effort Applied = $a(KLOC)^b$ [ man-months ]

| Type | a | b |
|---|---|---|
| Organic | 2.4 | 1.05 |
| Semi-detached | 3.0 | 1.12 |
| Embedded | 3.6 | 1.20 |

Organic

Small team, less than rigid requirements

Semi-detached

Medium teams,

Embedded

Tight constraints



8

# Example - 2 KLOC Embedded

Effort Applied = $a(KLOC)^b$ [ man-months ]

Effort Applied = $3.6*(2)^{1.20}$ = 8.3 man-months



9

# Problems with LOC

Language differences

Hand written code verses autogenerated code

Programmer variation

Defining and counting LOC

Coding accounts for about 35% of overall effort

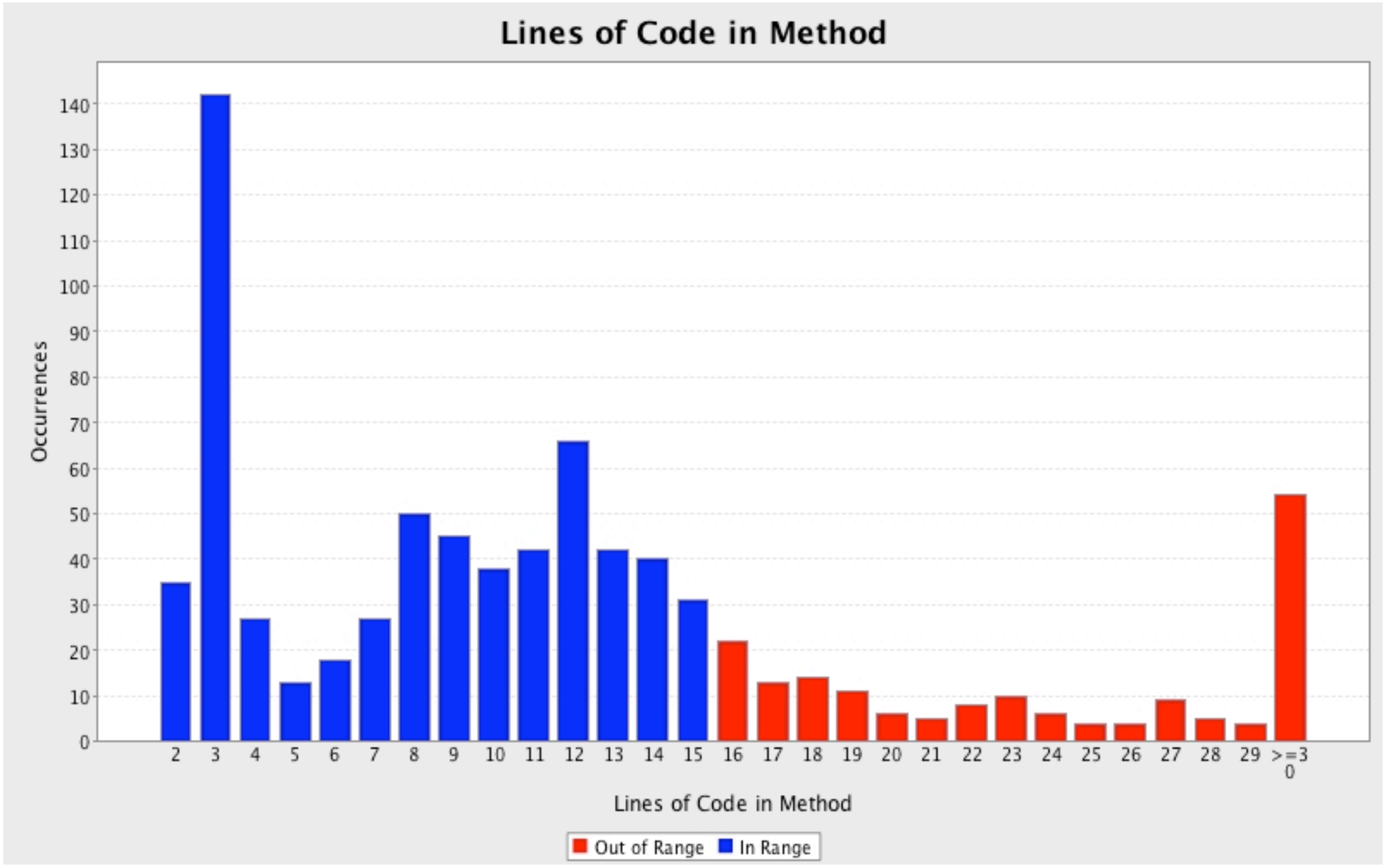


10

# Twitter4j Example

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| Total Lines of Code | 8161 | | | |
| java | 6908 | | | |
| twitter4j.org.json | 3193 | | | |
| twitter4j | 2489 | | | |
| twitter4j.http | 894 | | | |
| twitter4j.examples | 332 | | | |
| java | 1253 | | | |
| twitter4j | 1115 | | | |
| twitter4j.http | 138 | | | |
| Method Lines of Code (avg/max per method) | 5854 | 7.254 | 22.032 | 518 |
| java | 4899 | 6.949 | 22.726 | 518 |
| twitter4j.org.json | 2759 | 14.295 | 41.037 | 518 |
| twitter4j.http | 557 | 5.626 | 10.117 | 76 |
| twitter4j.examples | 240 | 26.667 | 14.877 | 57 |
| twitter4j | 1343 | 3.324 | 4.324 | 29 |
| java | 955 | 9.363 | 16.298 | 123 |
| twitter4j | 853 | 9.374 | 16.949 | 123 |
| twitter4j.http | 102 | 9.273 | 9.304 | 33 |

# Eclipse Metrics Plugin

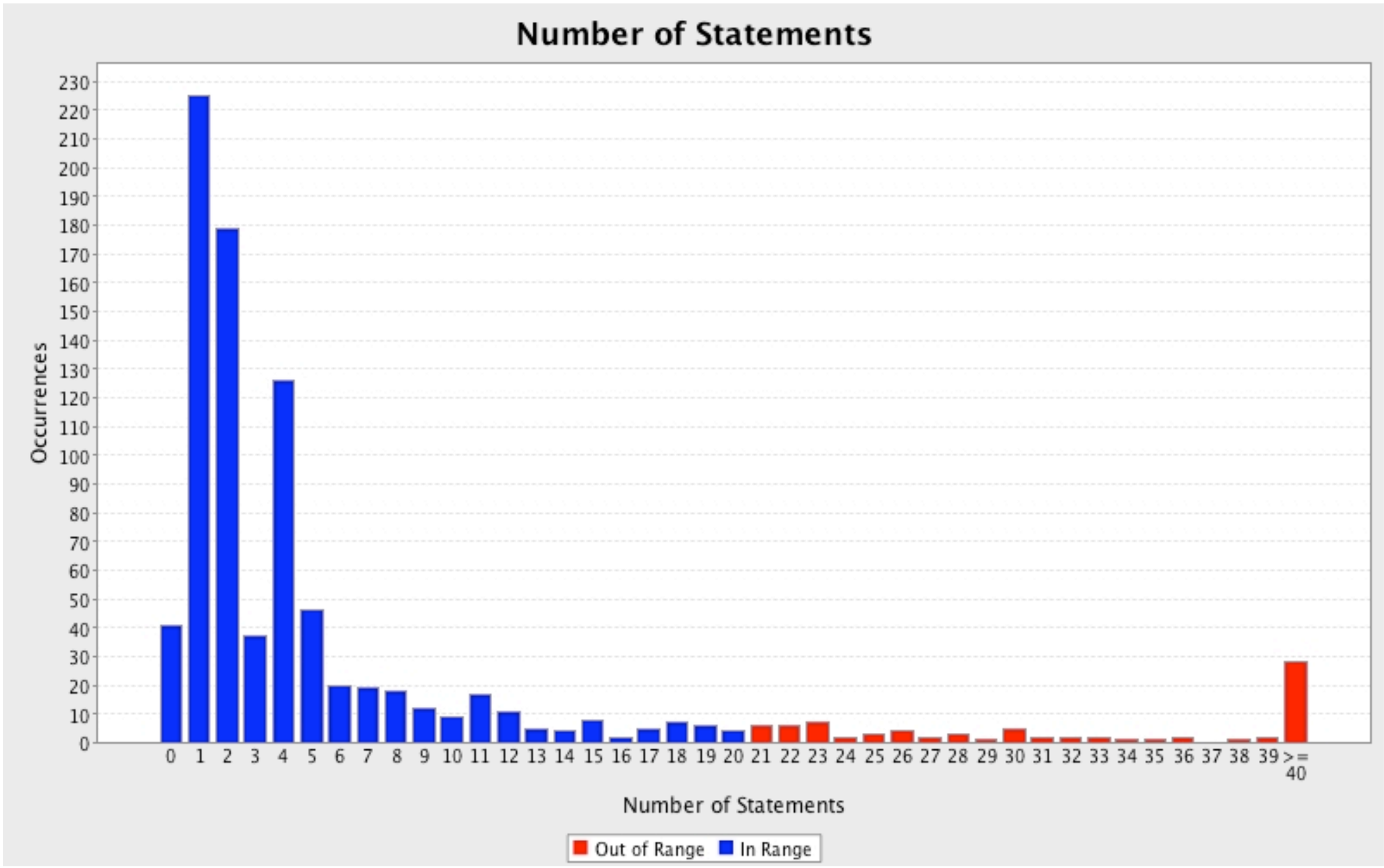





12

Metrics 1.3.6 finds 8161 total lines of code. Eclipse Metrics Plugin finds 11113 total lines of code as it does not remove large segments of white space

# Eclipse Metrics Plugin








Number of statements = Logical LOC. Not the difference from the graph on the previous slide.

# More Size Metrics

Number of Packages

Number of Interfaces

Number of classes  per Package

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Number of Classes (avg/max per packageFragment | 58 | 9.667 | 5.558 | 18 |
| ▶ java | 49 | 12.25 | 4.815 | 18 |
| ▶ java | 9 | 4.5 | 2.5 | 7 |



14

The number of interfaces might be considered a metric related to abstraction. Although (number of interfaces / number of class) per package might be a better metric. At least in Java. Abstraction is  a better metric for this.

# McCabe Cyclomatic Complexity

Number of linearly independent paths through a program

From graph theory

$$M = E - N + 2P$$

M = cyclomatic complexity
E = the number of edges of the graph
N = the number of nodes of the graph
P = the number of connected components.

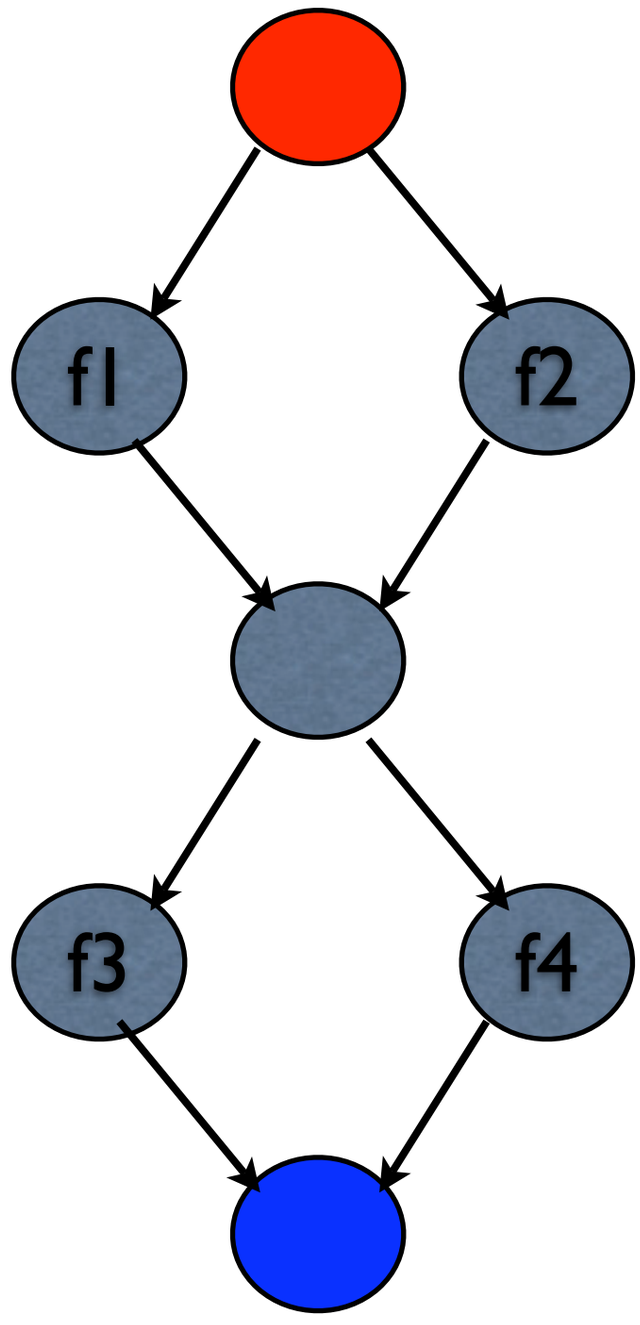


15

# Example

```
if( c1() )
   f1();
else
   f2();

if( c2() )
   f3();
else
   f4();
```




N = 7

E = 8

M = 8 - 7 + 2*1 = 3





Example From http://en.wikipedia.org/wiki/Cyclomatic_complexity

# What does it tell us?

branch coverage ≤ cyclomatic complexity ≤ number of paths

Cyclomatic Complexity

Is an upper bound for the number of test cases that are necessary to achieve a complete branch coverage

Is a lower bound for the number of paths through the code



17

# Cyclomatic Complexity & Quality

Higher Cyclomatic Complexity might indicate lower cohesion

- One study indicated it is better indicator than metrics designed for cohesion

Some evidence that higher Cyclomatic Complexity implies more bugs



18

# NIST Structured Testing methodology

Split modules with cyclomatic complexity greater than 10

It may be appropriate in some circumstances to permit modules with a complexity as high as 15



19

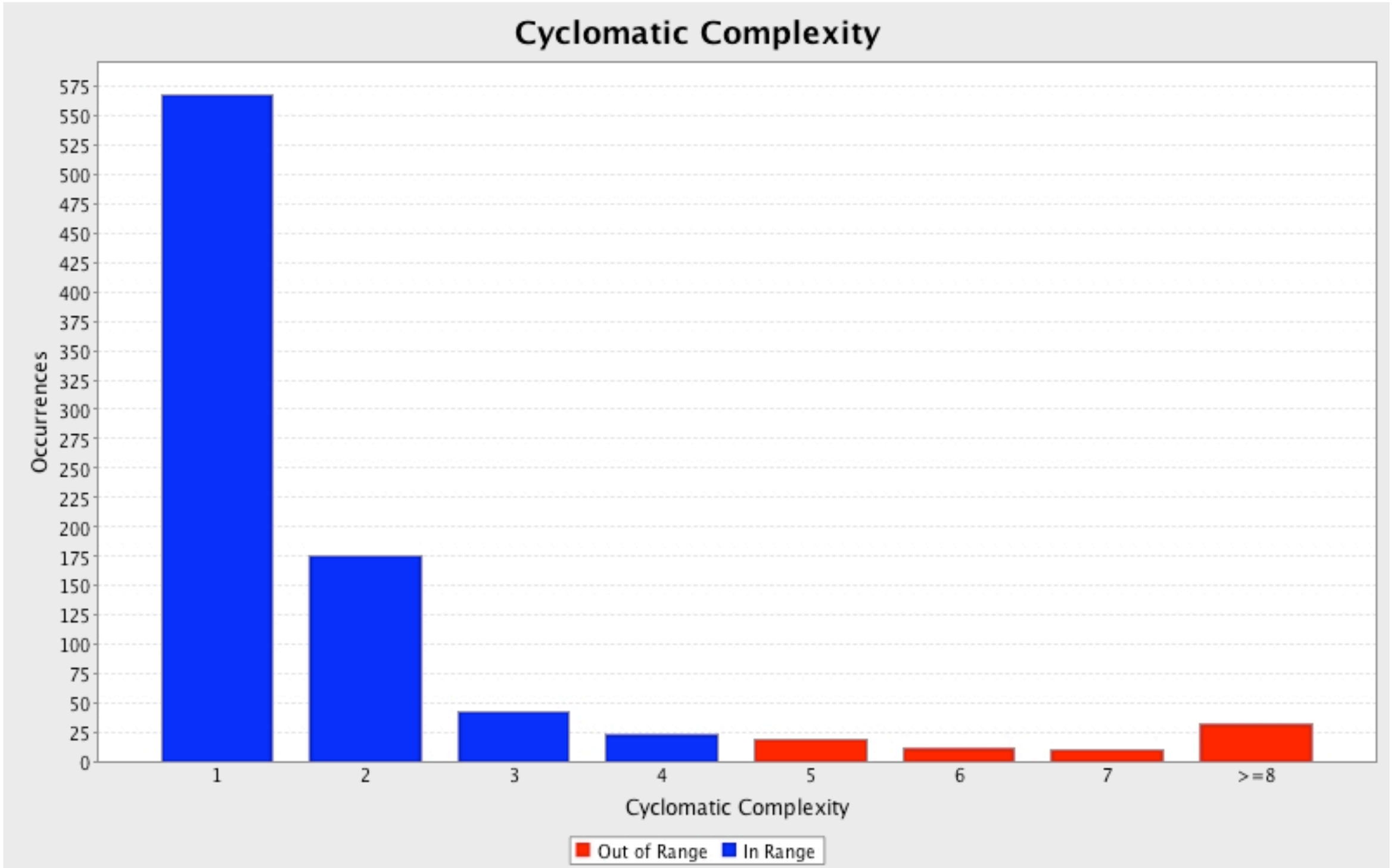
# Eclipse Metrics 1.3.6

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ McCabe Cyclomatic Complexity (avg/max per method) | | 2.15 | 3.569 | 46 |
| ▼ java | | 2.288 | 3.787 | 46 |
| ▼ twitter4j.org.json | | 4.212 | 5.9 | 46 |
| ▶ JSONML.java | | 10.286 | 15.229 | 46 |
| ▶ XML.java | | 11.5 | 12.42 | 36 |
| ▶ XMLTokener.java | | 12.143 | 9.463 | 28 |
| ▶ Test.java | | 21 | 0 | 21 |
| ▶ JSONObject.java | | 3.552 | 4.306 | 19 |
| ▶ JSONTokener.java | | 4.688 | 3.531 | 14 |
| ▶ HTTP.java | | 7.5 | 4.5 | 12 |
| ▶ JSONArray.java | | 2.2 | 2.04 | 12 |
| ▶ CDL.java | | 4.3 | 3.132 | 11 |
| ▶ HTTPTokener.java | | 5.5 | 4.5 | 10 |
| ▶ JSONWriter.java | | 2.786 | 2.042 | 8 |
| ▶ Cookie.java | | 5.75 | 1.299 | 7 |
| ▶ CookieList.java | | 3 | 1 | 4 |
| ▶ JSONStringer.java | | 1.5 | 0.5 | 2 |
| ▶ JSONException.java | | 1 | 0 | 1 |
| JSONString.java | | 0 | 0 | |
| ▶ twitter4j | | 1.408 | 2.099 | 29 |
| ▶ twitter4j.http | | 2.03 | 2.359 | 16 |
| ▶ twitter4j.examples | | 3.333 | 1.333 | 6 |
| ▶ java | | 1.196 | 0.805 | 7 |



20

# Eclipse Metrics Plugin






21

# Weighted Methods per Class (WMC)

Sum of the McCabe Cyclomatic Complexity for all methods in a class

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Weighted methods per Class (avg/max per type) | 1735 | 29.914 | 41.206 | 235 |
| ▼ java | 1613 | 32.918 | 43.423 | 235 |
| ▶ twitter4j.org.json | 813 | 50.812 | 57.857 | 235 |
| ▶ twitter4j | 569 | 31.611 | 33.705 | 140 |
| ▶ twitter4j.http | 201 | 25.125 | 29.464 | 100 |
| ▶ twitter4j.examples | 30 | 4.286 | 3.01 | 11 |
| ▼ java | 122 | 13.556 | 18.963 | 56 |
| ▶ twitter4j | 110 | 15.714 | 20.899 | 56 |
| ▶ twitter4j.http | 12 | 6 | 4 | 10 |



22

# Basic Class Metrics

Number of methods per class
Number of static methods per class
Number of attributes(fields) per class
Number of static attributes per class

Number of parameters per method



23

# Twitter4j Example

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Number of Methods (avg/max per type) | 742 | 12.793 | 21.461 | 111 |
| ▼ java | 641 | 13.082 | 22.274 | 111 |
| ▶ twitter4j | 398 | 22.111 | 29.04 | 111 |
| ▶ twitter4j.org.json | 151 | 9.438 | 16.948 | 55 |
| ▶ twitter4j.http | 90 | 11.25 | 14.481 | 49 |
| ▶ twitter4j.examples | 2 | 0.286 | 0.7 | 2 |
| ▶ java | 101 | 11.222 | 16.253 | 52 |
| ▼ Number of Parameters (avg/max per method) | | 0.954 | 0.901 | 6 |
| ▼ java | | 1.033 | 0.918 | 6 |
| ▶ twitter4j | | 1.017 | 0.999 | 6 |
| ▶ twitter4j.http | | 0.97 | 1.039 | 6 |
| ▶ twitter4j.org.json | | 1.104 | 0.652 | 3 |
| ▶ twitter4j.examples | | 0.889 | 0.314 | 1 |
| ▶ java | | 0.412 | 0.512 | 2 |



24

# Nested Block Depth

The depth of nested blocks of code

Depth = 2

```
public static JSONObject toJSONObject(String string) throws JSONException {
    JSONObject o = new JSONObject();
    JSONTokener x = new JSONTokener(string);
    while (x.more()) {
        String name = Cookie.unescape(x.nextTo('='));
        x.next('=');
        o.put(name, Cookie.unescape(x.nextTo(';')));
        x.next();
    }
    return o;
}
```



25

# Twitter4j Example

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Nested Block Depth (avg/max per method) | | 1.489 | 0.938 | 8 |
| ▼ java | | 1.549 | 0.984 | 8 |
| ▼ twitter4j.org.json | | 2.047 | 1.348 | 8 |
| ▶ JSONML.java | | 3.143 | 2.642 | 8 |
| ▶ XML.java | | 3.833 | 2.672 | 8 |
| ▶ JSONObject.java | | 1.881 | 1.153 | 6 |
| ▶ CDL.java | | 2.5 | 1.5 | 5 |
| ▶ Cookie.java | | 3.25 | 0.829 | 4 |
| ▶ JSONTokener.java | | 2.375 | 1.053 | 4 |
| ▶ CookieList.java | | 3 | 1 | 4 |
| ▶ HTTPTokener.java | | 2.5 | 1.5 | 4 |
| ▶ XMLTokener.java | | 2.857 | 0.833 | 4 |
| ▶ JSONArray.java | | 1.58 | 0.851 | 4 |
| ▶ JSONWriter.java | | 1.786 | 1.013 | 4 |
| ▶ Test.java | | 3 | 0 | 3 |
| ▶ HTTP.java | | 2.5 | 0.5 | 3 |
| ▶ JSONException.java | | 1 | 0 | 1 |
| ▶ JSONStringer.java | | 1 | 0 | 1 |
| JSONString.java | | 0 | 0 | |
| ▶ twitter4j.examples | | 3 | 1.054 | 5 |
| ▶ twitter4j.http | | 1.465 | 0.868 | 5 |
| ▶ twitter4j | | 1.3 | 0.619 | 4 |
| ▶ java | | 1.078 | 0.269 | 2 |



26

# Some Inheritance Metrics

Depth of Inheritance Tree (DIT)

Distance from class Object in the inheritance hierarchy

Number of Children

Total number of direct subclasses of a class

Number of Overridden Methods (NORM)

Specialization Index

NORM * DIT / number of methods

If greater than 5 likely that superclass abstraction has a problem



27

# Lack of Cohesion in Methods (LCOM)

$$\frac{\langle r \rangle - |M|}{1 - |M|}$$

M be the set of methods defined by the class

F be the set of fields defined by the class

r(f) be the number of methods that access field f, where f is a member of F

<r> be the mean of r(f) over F.

## High Cohesion

When each method accesses all fields

<r> = |M|

LCOM = 0

## Low Cohesion

When each method accesses one fields

<r> = 1

LCOM = 1



28

Some people recommend that you create an accessor for each field. When you need to access the field in the class you use the accessor, rather than access the field directly. This will give you a LCOM of 1. One has to be careful with metrics.

# Lack of Cohesion of Methods

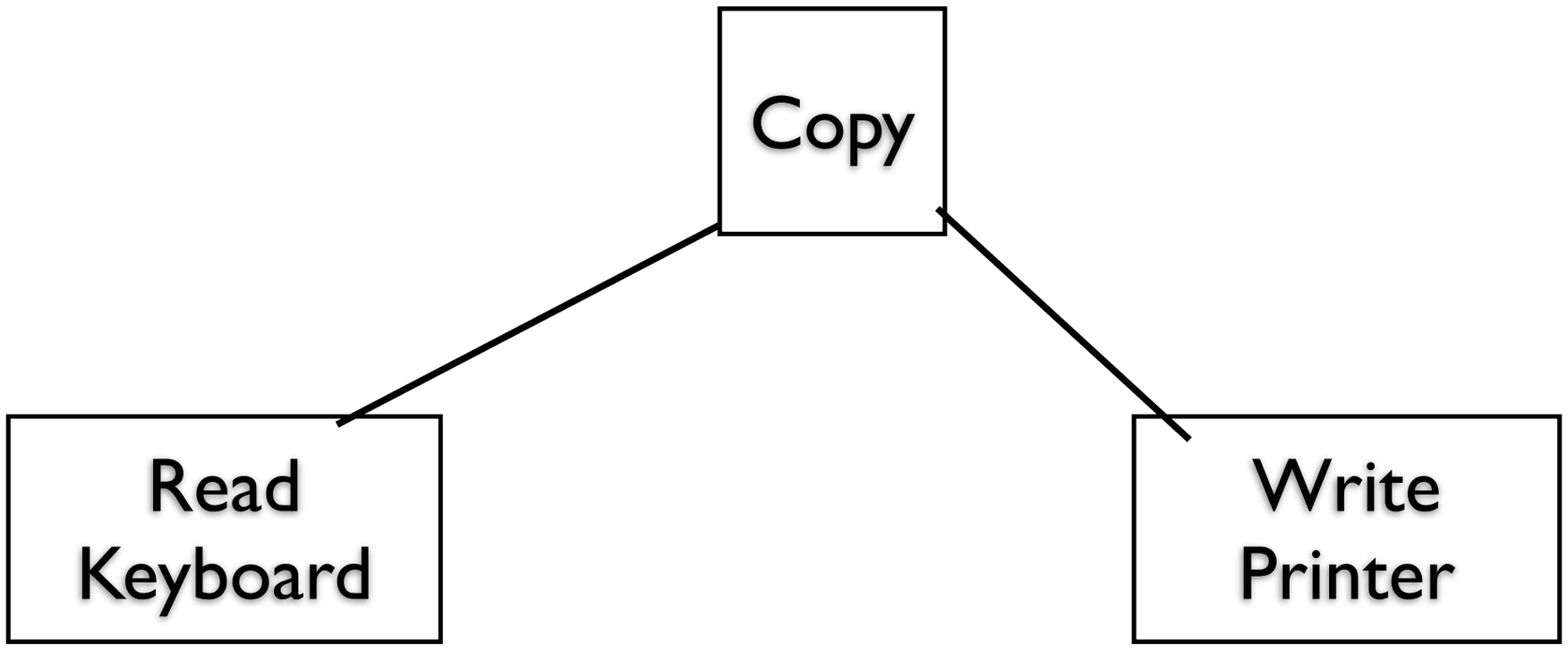
| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Lack of Cohesion of Methods (avg/max per type) | | 0.26 | 0.342 | 0.938 |
| ▼ java | | 0.25 | 0.336 | 0.938 |
| ▶ twitter4j.http | | 0.358 | 0.348 | 0.938 |
| ▶ twitter4j | | 0.461 | 0.359 | 0.902 |
| ▶ twitter4j.org.json | | 0.056 | 0.15 | 0.5 |
| ▶ twitter4j.examples | | 0.024 | 0.058 | 0.167 |
| ▶ java | | 0.319 | 0.37 | 0.905 |



29

# Metrics for Stable Code

Dependencies make code rigid, fragile and difficult to reuse

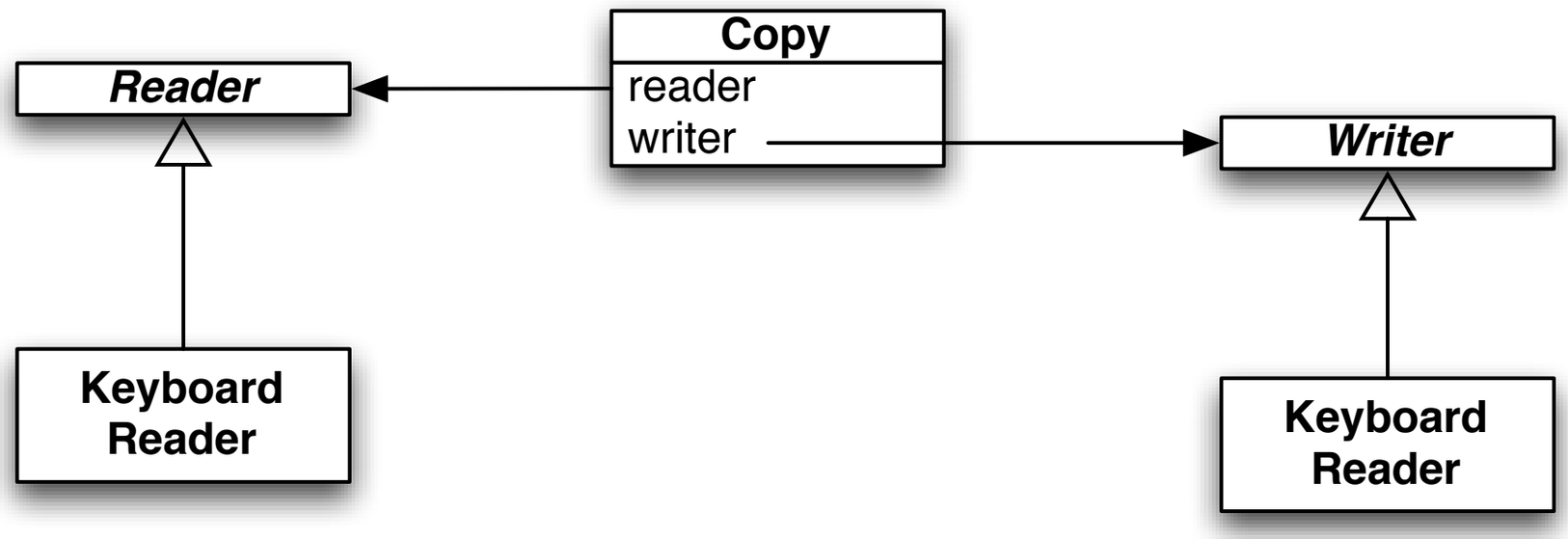





30

Consider a program that copies characters typed on a keyboard to a printer. Rest of lecture is from OO Design Quality Metrics: An Analysis of Dependencies, Robert Martin, http://www.objectmentor.com/resources/articles/oodmetrc.pdf

# Flexible version




Have dependencies on Reader/Writer classes

But these classes are stable



31

# Main Idea

When code depends on other classes, changes to those classes can force the code to change

The fewer classes code depends on the stabler the code is



32

# Class Categories

Group of highly cohesive classes that

1. The classes within a category are closed together against any force of change

   If one class must change, all classes are likely to change

2. The classes within a category are reused together

3. The classes within a category share some common function or achieve some common goal



33

# Dependency Metrics

Afferent Couplings (Ca)

The number of classes outside this category that depend upon classes within this category

Efferent Couplings (Ce)

The number of classes inside this category that depend upon classes outside this categories

Instability (I)

$$\frac{Ce}{Ca+Ce}$$

I = 0 means a category is maximally stable

I = 1 means a category is maximally instable



34

# Instabilty Twitter4j Example

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Instability (avg/max per packageFragment) | | 0.645 | 0.35 | 1 |
| ▼ java | | 0.51 | 0.354 | 1 |
| twitter4j.examples | 1 | | | |
| twitter4j | 0.538 | | | |
| twitter4j.http | 0.5 | | | |
| twitter4j.org.json | 0 | | | |
| ▼ java | | 0.917 | 0.083 | 1 |
| twitter4j.http | 1 | | | |
| twitter4j | 0.833 | | | |



35

# How to be flexible and stable?

Use abstract classes



36

# Abstractness (A)

$$\frac{\text{\# of abstract classes in category}}{\text{total \# of classes in category}}$$

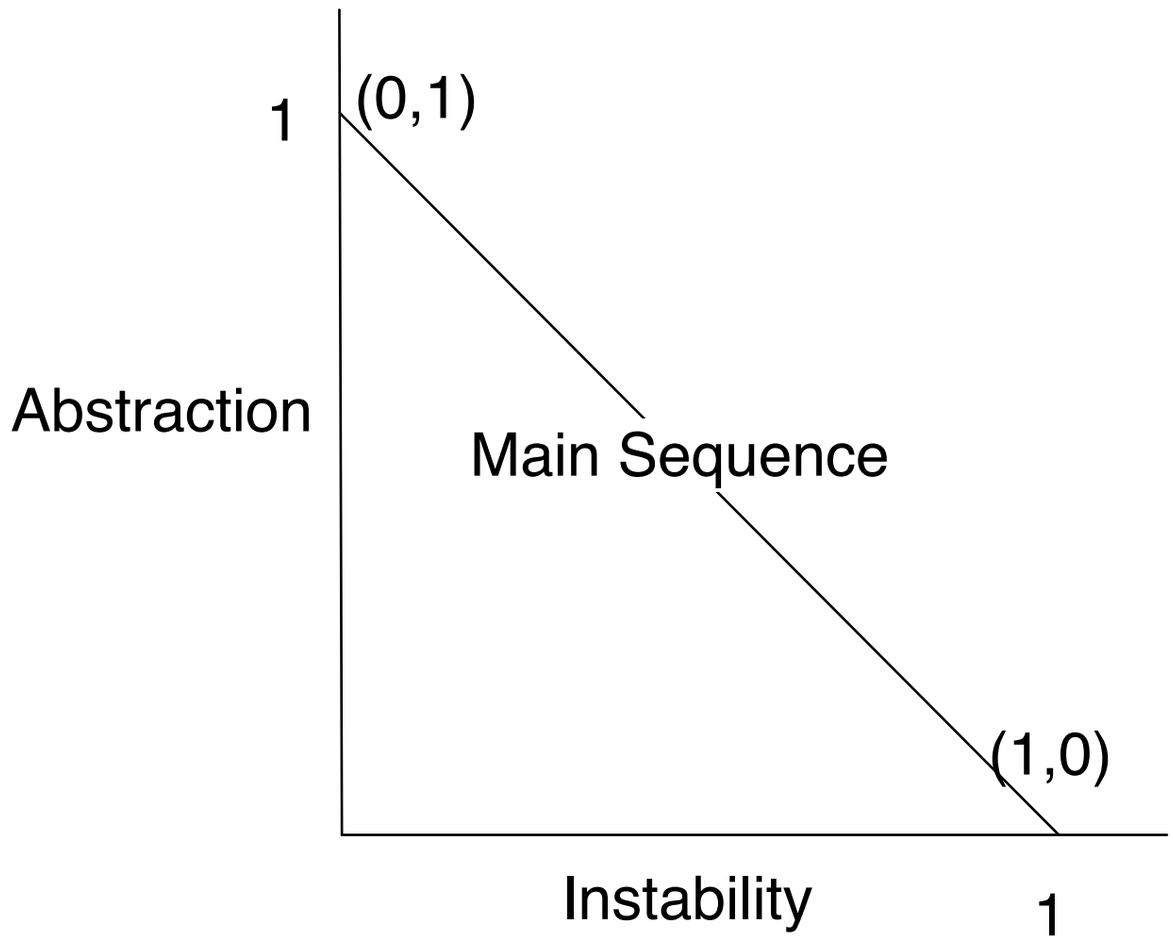
A = 1, all classes are abstract

A = 0, all classes are concrete



37

# Main Sequence






38

# Distance From Main Sequence

$$Dn = | A + I - 1 |$$

Dn = 0 , category is on the main sequence

Dn = 1, category is far from main sequence

Values not near zero suggest restructuring the category



39

# Twitter4j Example

| Metric | Total | Mean | Std. Dev. | Maximum |
|---|---|---|---|---|
| ▼ Normalized Distance (avg/max per packageFragment) | | 0.327 | 0.329 | 0.941 |
| ▼ java | | 0.449 | 0.337 | 0.941 |
| twitter4j.org.json | 0.941 | | | |
| twitter4j.http | 0.5 | | | |
| twitter4j | 0.356 | | | |
| twitter4j.examples | 0 | | | |
| ▼ java | | 0.083 | 0.083 | 0.167 |
| twitter4j | 0.167 | | | |
| twitter4j.http | 0 | | | |



40