

References

Object Coupling and Object Cohesion, chapter 7 of Essays on Object-Oriented Software Engineering, Vol. 1, Berard, Prentice-Hall, 1993, pp 92-111

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/oodmetric.pdf>

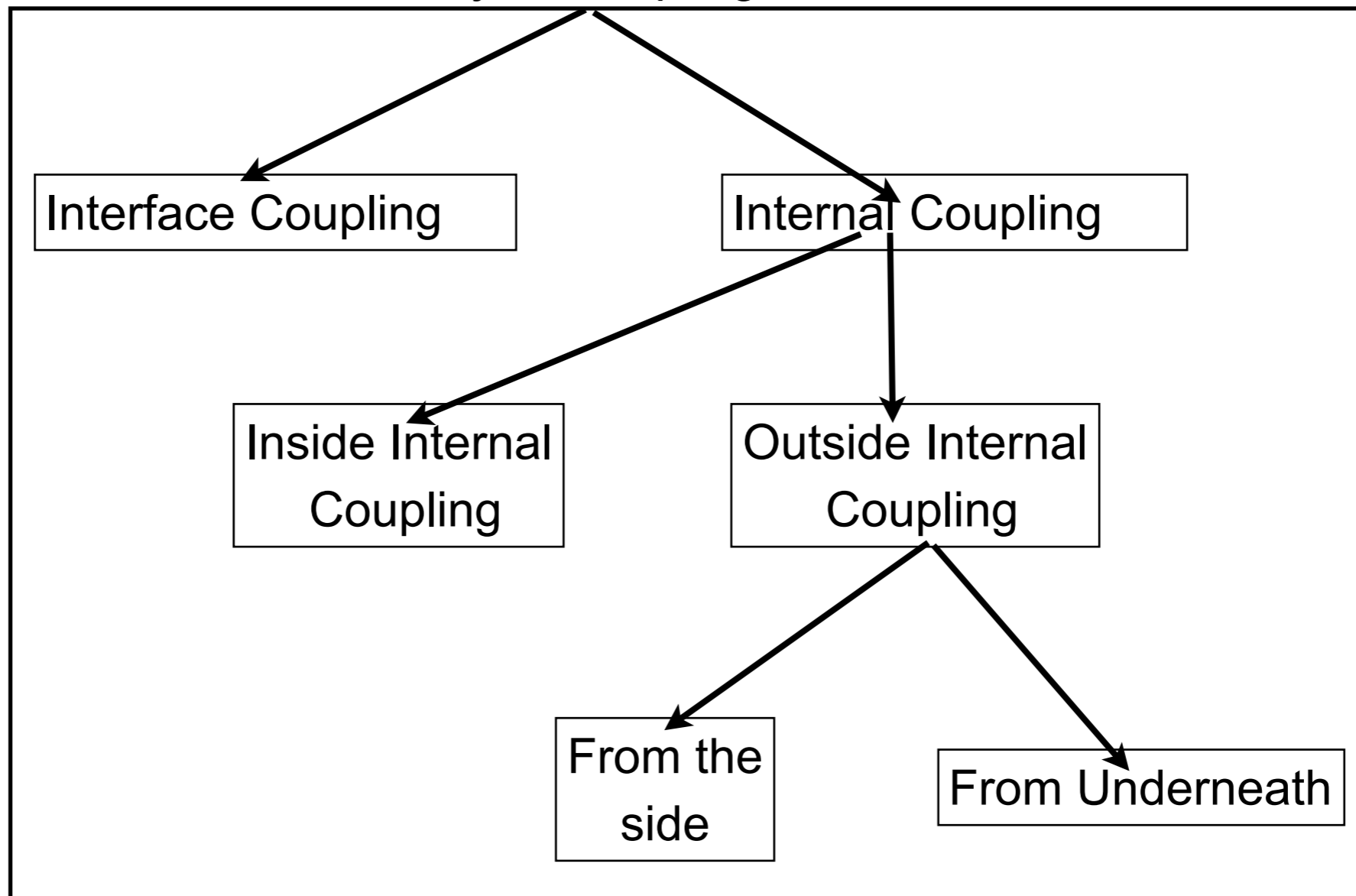
Source code for twitter4j, <http://yusuke.homeip.net/twitter4j/en/index.html>

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

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

Object Coupling

Object Coupling



Internal Coupling & Cohesion

Internal Coupling

Physical relationships among the items that comprise an object

Cohesion

Logical relationships among the items that comprise an object

Interface Coupling

One object refers to another specific object, and the original object makes direct references to one or more items in the specific object's public interface

Includes module coupling already covered

Weakest form of object coupling, but has wide variation

Issues

- Object abstraction decoupling

- Selector decoupling

- Constructor decoupling

- Iterator decoupling

Object Abstraction Decoupling

Assumptions that one object makes about a category of other objects are isolated and used as parameters to instantiate the original object.

C++/Java 1.5 Example

```
class LinkedListCell {
    int cellItem;
    LinkedListCell* next;

    // code can now use fact that cellItem is an int
    if ( cellItem == 5 ) print( "We Win" );
}

template <class type>
class LinkedListCell#2 {
    type cellItem;
    LinkedListCell* next;

    // code does not know the type, it is just a cell item,
    // it becomes an abstraction
}
```

Selectors

Return state information about their encapsulated object and
Do not alter the state of their encapsulated object

```
public void display() {  
    Swing GUI code to display the counter  
}
```

Selector
decoupling

```
public String toString() {return String.valueOf( count );}
```


Selector Decoupling

Counter Example

```
class Counter{
    int count = 0;

    public void increment()    { count++; }
    public void reset()       { count = 0; }
    public void display() {
        Java Swing code to display the counter
        in a slider bar
    }
}
```

Counter



Selector Decoupled

```
class Counter{
    int count = 0;

    public void increment()    { count++; }
    public void reset()       { count = 0; }
    public int count()        {return count;}
    public String toString()  {return String.valueOf( count );}
}
```

Iterator

Allows the user to visit all the nodes in a homogeneous composite object and to perform some user-supplied operation at each node

Primitive Methods

Any method that cannot be implemented simply, efficiently, and reliably without knowledge of the underlying implementation of the object

Functionally cohesive, they perform a single specific function

Small, seldom exceed five "lines of code"

Types

Selectors (get operations)

Constructors (not the same as class constructors)

Iterators

Constructors

Operations that construct a new, or altered version of an object

```
class Calendar {  
    public void getMonth( from where, or what) { blah }  
}
```

```
class Calendar {  
    public static Calendar fromString( String date ) { blah}  
}
```

Primitive Objects

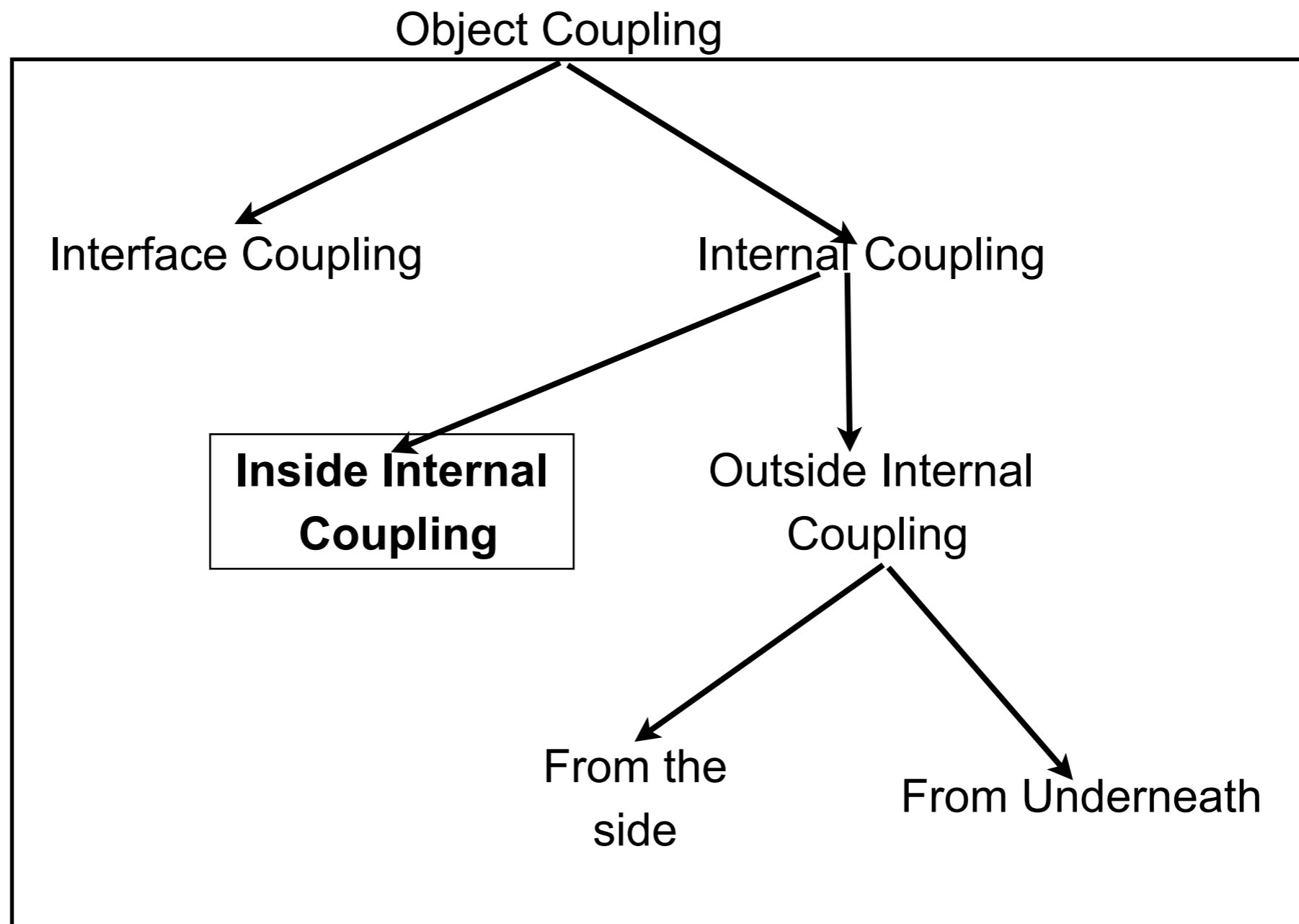
Primitive objects are objects that are both:

- Defined in the standard for the implementation language

- Globally known

Primitive objects don't count in coupling with other objects

Why not?



Inside Internal Object Coupling

Coupling between state and operations of an object

The big issue: Accessing state

Changing the structure of the state of an object requires changing all operations that access the state including operations in subclasses

Solution: Access state via access operations

C++ implementation

Provide private functions to access and change each data member

Outside Internal Coupling from Underneath

Coupling between a class and subclass involving private state and private operations

Major Issues

Access to inherited state

- Direct access to inherited state

- Access via operations

Unwanted Inheritance

- Parent class may have operations and state not needed by subclass

Outside Internal Coupling from the Side

Class A accesses private state or private operations of class B

Class A and B are not related via inheritance

Main causes

Using non-object-oriented languages

Special language "features"

C++ friends

Metrics

Metrics

DeMarco's Principle

Effort moves toward whatever is measured



The Swedish Army Dictum

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

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

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

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

Basic COCOMO

Software Cost Estimation Model

$$\text{Effort Applied} = a(\text{KLOC})^b \quad [\text{man-months}]$$

| 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

Example - 2 KLOC Embedded

$$\text{Effort Applied} = a(\text{KLOC})^b \quad [\text{man-months}]$$

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

Problems with LOC

Language differences

Hand written code verses autogenerated code

Programmer variation

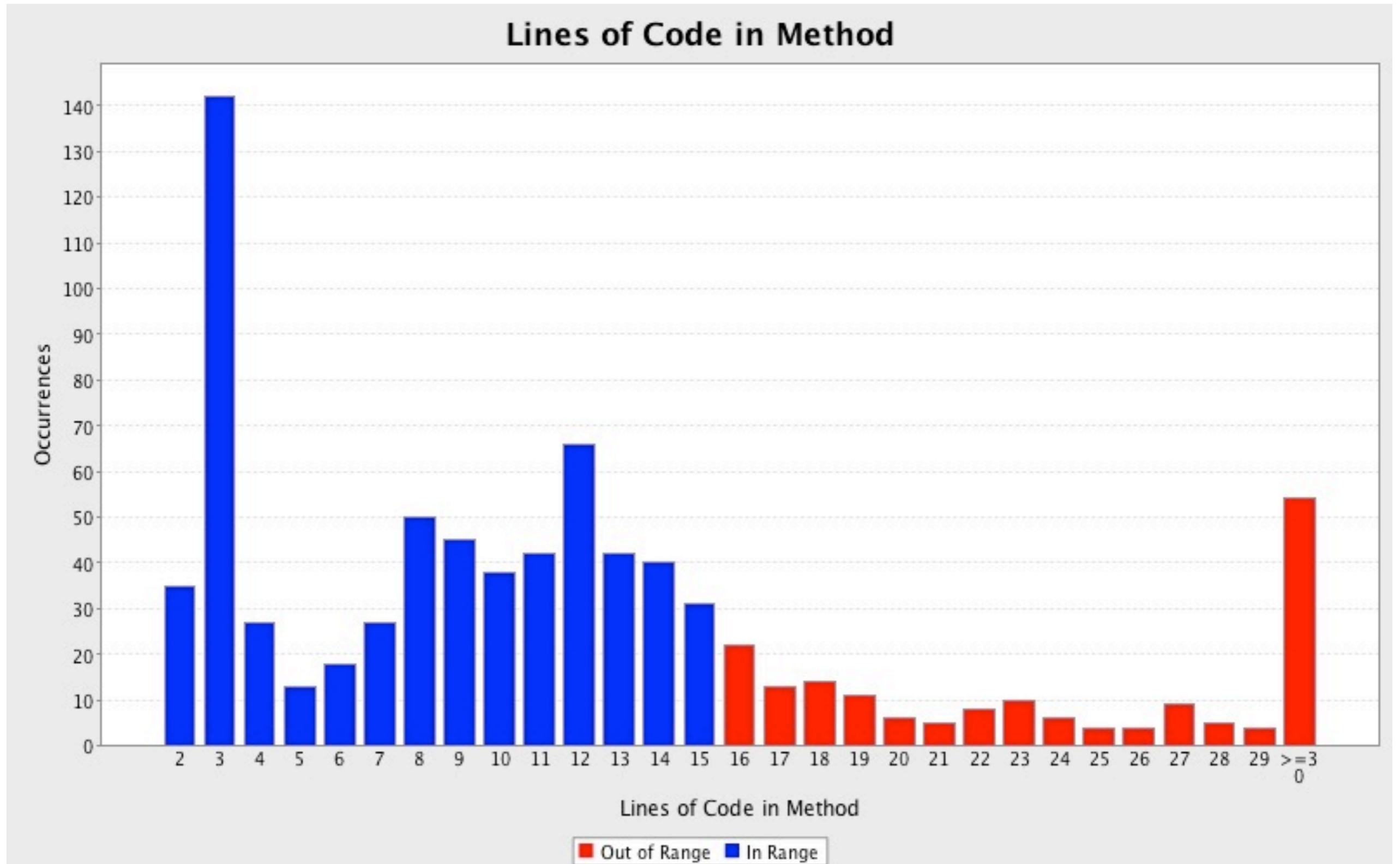
Defining and counting LOC

Coding accounts for about 35% of overall effort

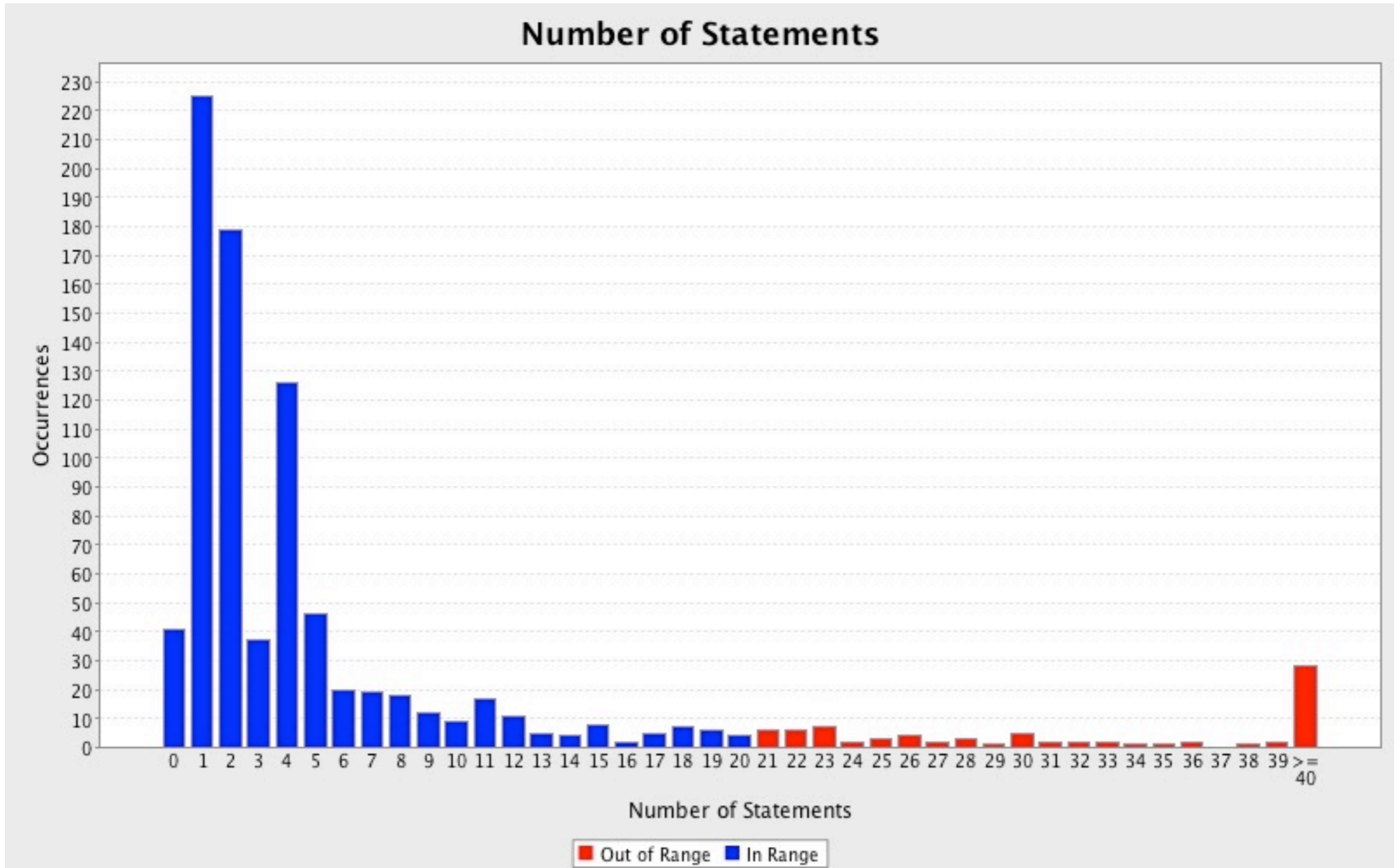
Twitter4j Example

| 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



Eclipse Metrics Plugin



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 |

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

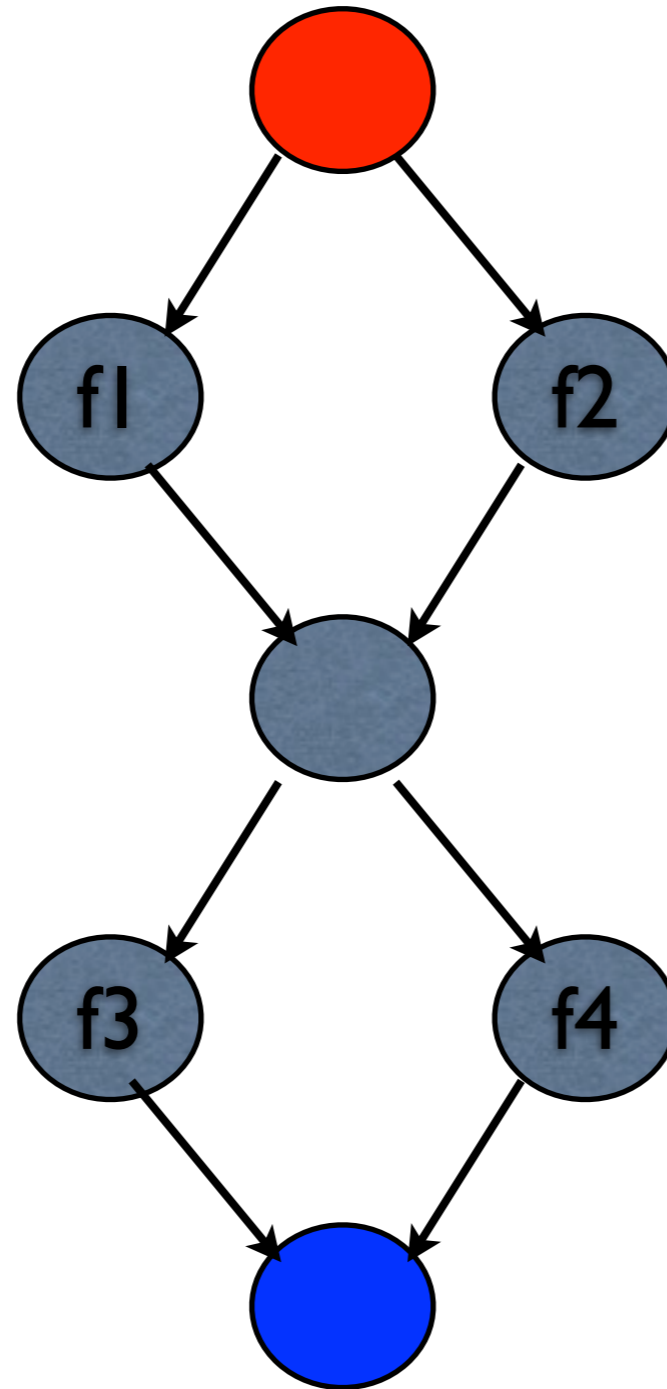
N = the number of nodes of the graph

P = the number of connected components.

Example

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

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



$$N = 7$$

$$E = 8$$

$$M = 8 - 7 + 2 * 1 = 3$$

What does it tell us?

branch coverage \leq cyclomatic complexity \leq 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

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

NIST Structured Testing methodology

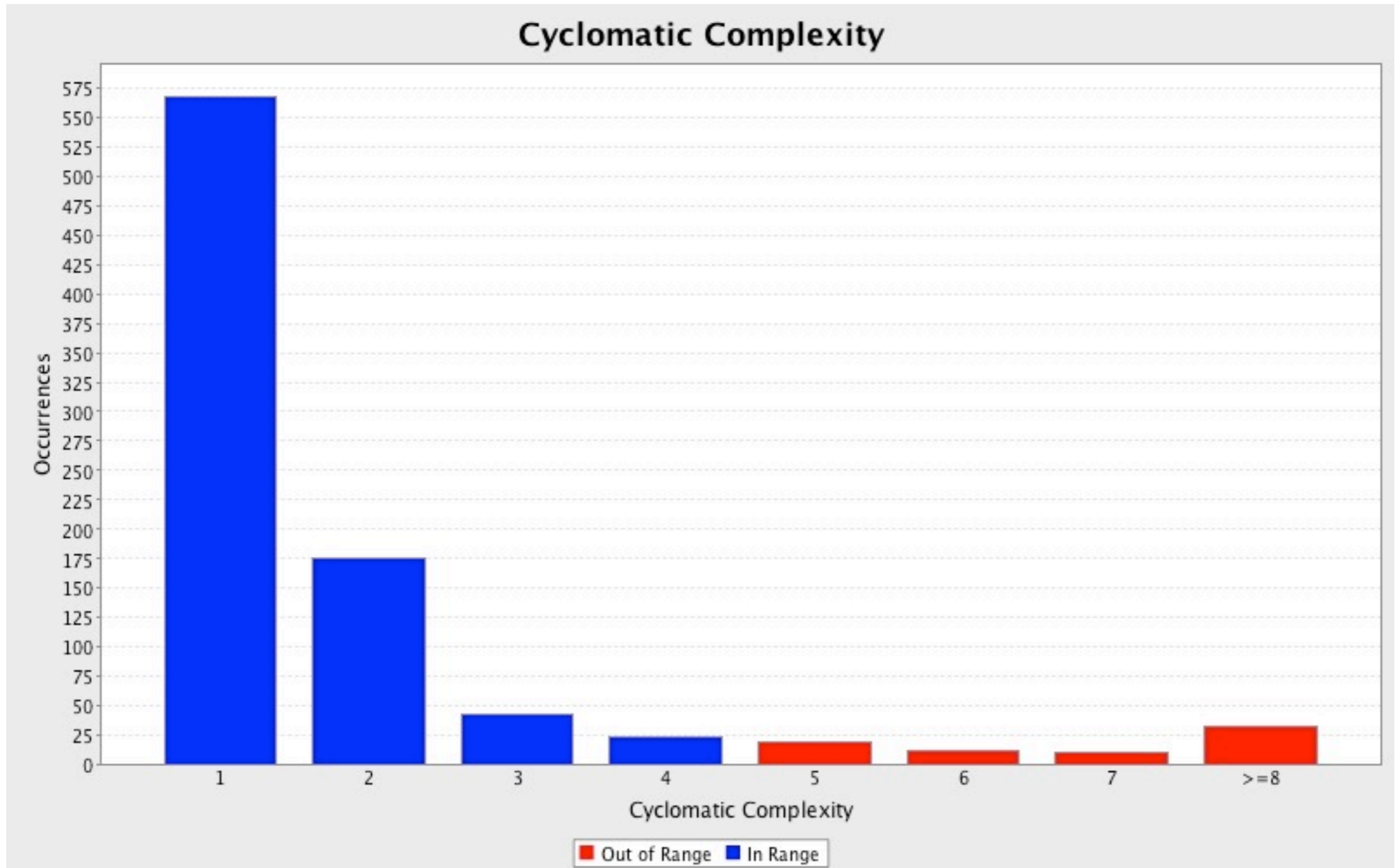
Split modules with cyclomatic complexity greater than 10

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

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 |
| ▶ JSOStringer.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 |

Eclipse Metrics Plugin



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 |

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

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 |

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;
}
```

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 |

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

$\text{NORM} * \text{DIT} / \text{number of methods}$

If greater than 5 likely that superclass abstraction has a problem

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

$\langle r \rangle$ be the mean of r(f) over F.

High Cohesion

When each method accesses all fields

$$\langle r \rangle = |M|$$

$$\text{LCOM} = 0$$

Low Cohesion

When each method accesses one fields

$$\langle r \rangle = 1$$

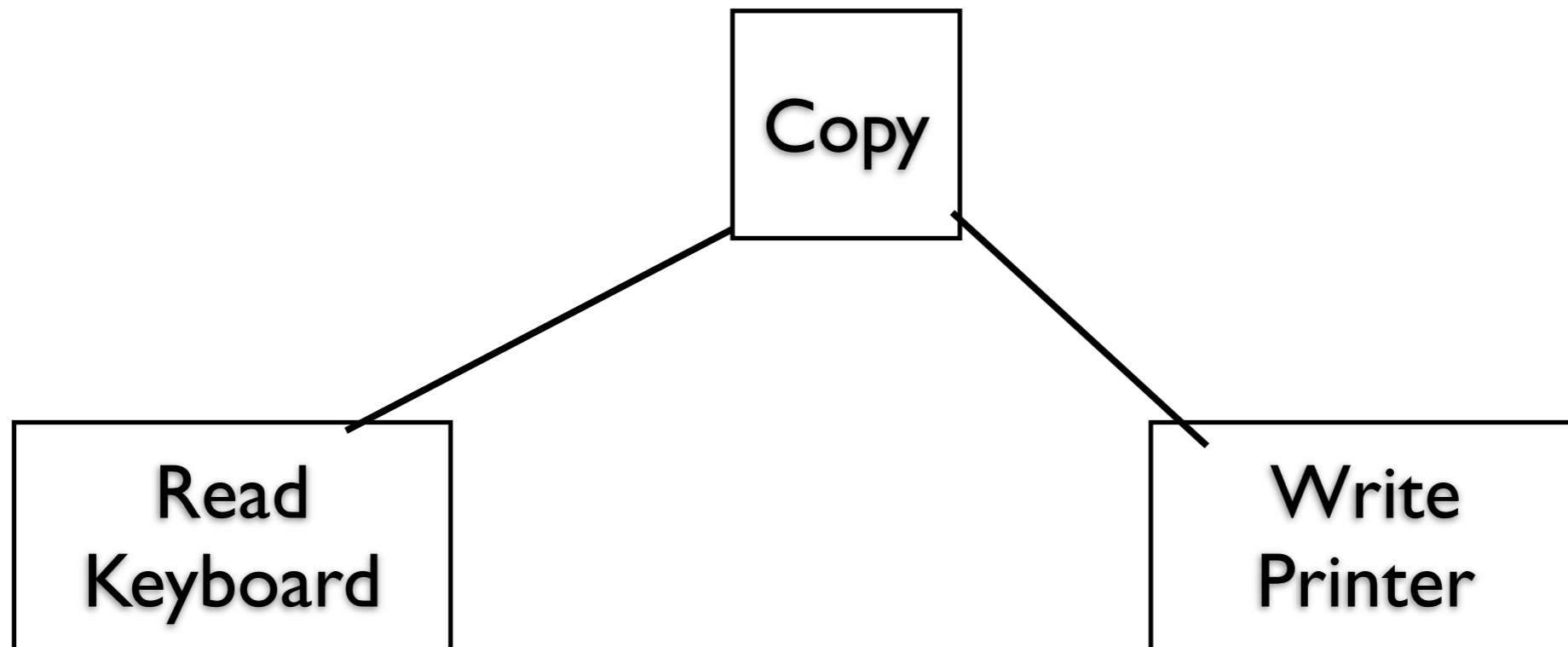
$$\text{LCOM} = 1$$

Lack of Cohesion of Methods

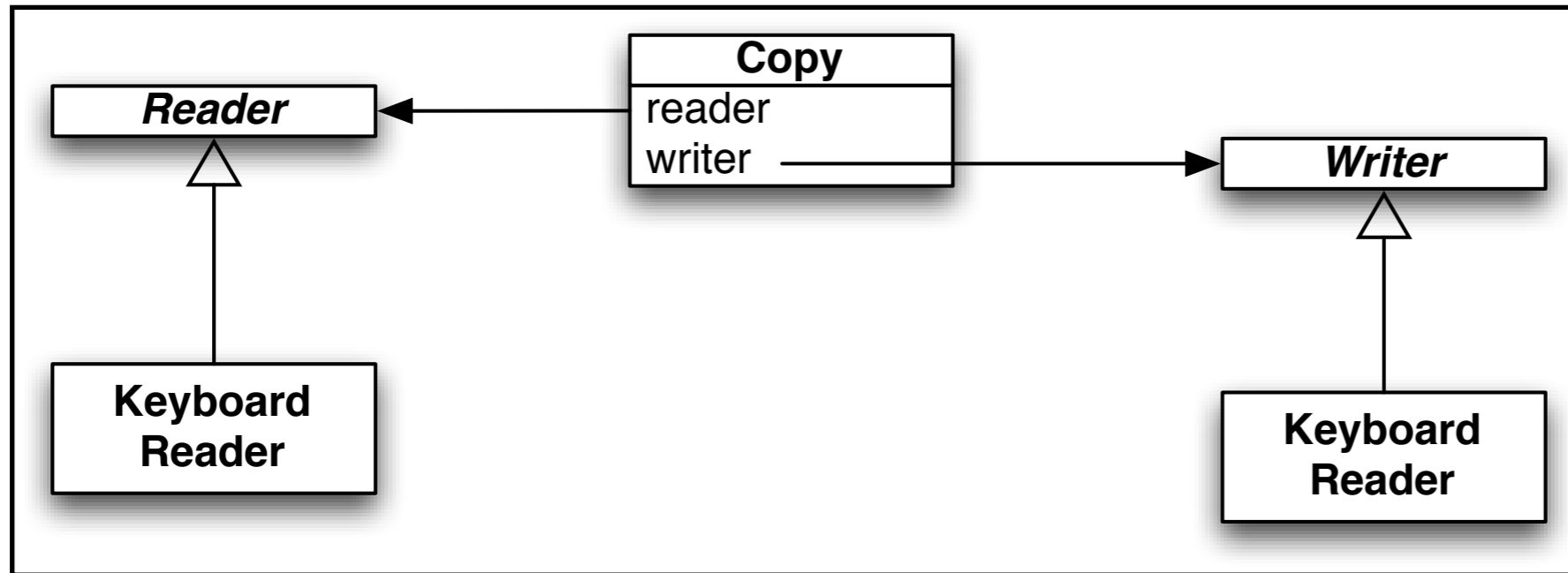
| Metric | Total | Mean | Std. Dev. | Maximum |
|--|-------|-------|-----------|---------|
| ▼ Lack of Cohesion of Methods (avg/max per type) | | 0.26 | 0.342 | 0.938 |
| ▼ java | | 0.25 | 0.336 | 0.938 |
| ▶ twitter4j.http | | 0.358 | 0.348 | 0.938 |
| ▶ twitter4j | | 0.461 | 0.359 | 0.902 |
| ▶ twitter4j.org.json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

Metrics for Stable Code

Dependencies make code rigid, fragile and difficult to reuse



Flexible version



Have dependencies on Reader/Writer classes
But these classes are stable

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

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

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 category

Instability (I)

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

$I = 0$ means a category is maximally stable

$I = 1$ means a category is maximally instable

Instability 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 | | | |

How to be flexible and stable?

Use abstract classes

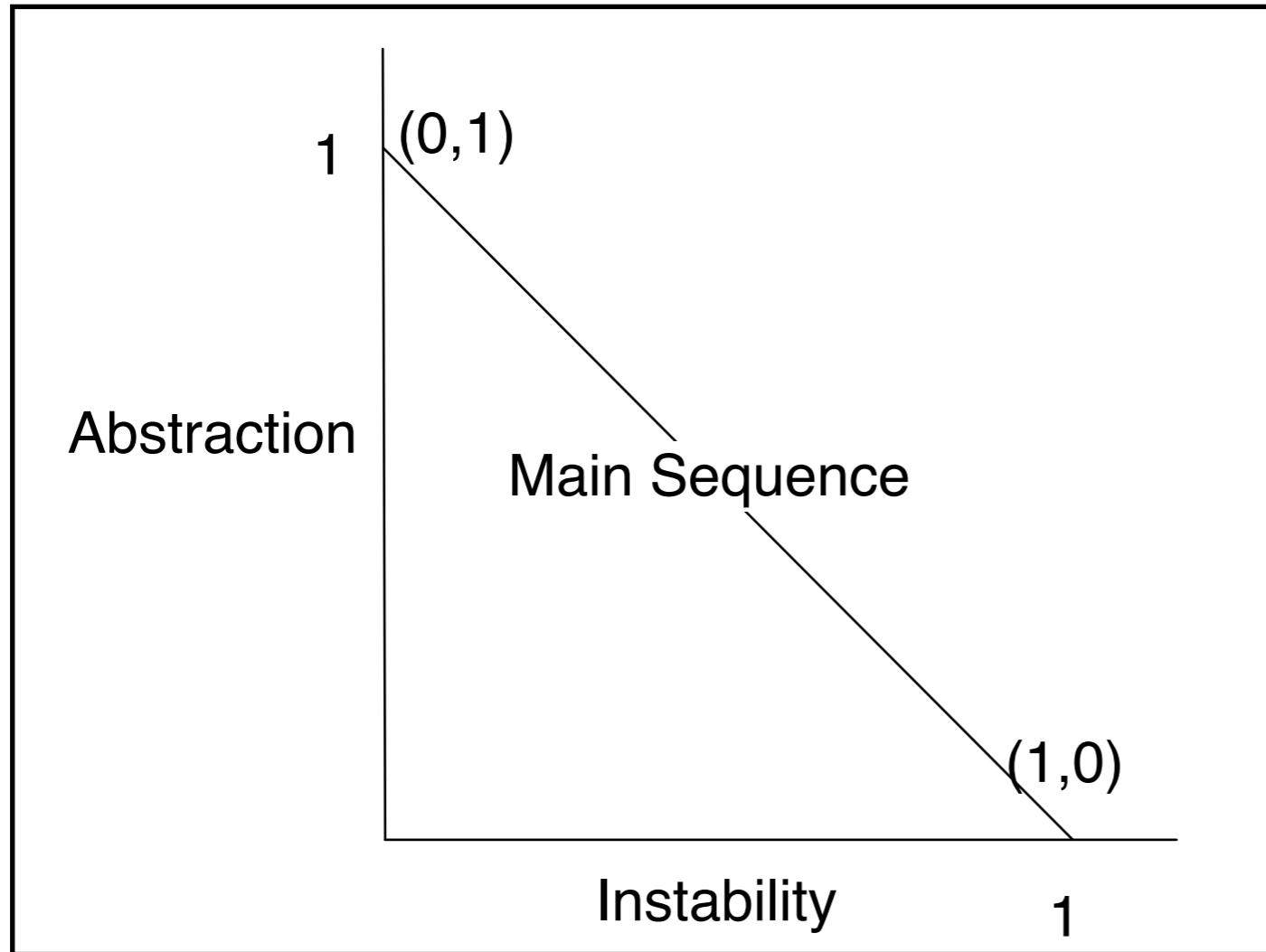
Abstractness (A)

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

A = 1, all classes are abstract

A = 0, all classes are concrete

Main Sequence



Distance From Main Sequence

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

$D_n = 0$, category is on the main sequence

$D_n = 1$, category is far from main sequence

Values not near zero suggest restructuring the category

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 | | | |