COMPSCI 732:

Software Tools and Techniques

Tools for Producing Quality Software

Ewan Tempero e.tempero@cs.auckland.ac.nz

www.cs.auckland.ac.nz/~ewan

Research Interests

. . . to make programmers more productive, that is, help the people who actually produce code to do so faster, with less effort, fewer errors, and with more enjoyment than currently

Research Interests

. . . to make programmers more productive, that is, help the people who actually produce code to do so faster, with less effort, fewer errors, and with more enjoyment than currently

Theme

Software tools are needed for more than just the production of software.

Observation 1

Much of the pain in software development is due to having to deal with kludgey code.

$\Rightarrow$ We need to produce higher quality software than we currently do

Observation 1

Much of the pain in software development is due to having to deal with kludgey code.

$\Rightarrow$ We need to produce higher quality software than we currently do
(whatever “higher quality” means)

Observation 2

We can't know whether or not we have improved the quality of our software if we can't measure its quality

$\Rightarrow$ we need tools that measure quality of software

$\Rightarrow$ we need to know how to measure quality

$\Rightarrow$ we need to know what we mean by quality

$\Rightarrow$ we need to know how to measure software

Research Agenda

- Develop software metrics that might tell us something about software quality
- Develop instruments to measure code according to the identified metrics
- Apply the instruments to existing code
- Analyse the resulting data to identify potential quality problems
- Determine whether or not the metrics actually do tell us something about software quality
- Incorporate the use of the metrics into the software development process

Research Agenda

- Develop software metrics that might tell us something about software quality
  - describe some metrics
- Develop instruments to measure code according to the identified metrics

- Apply the instruments to existing code

- Analyse the resulting data to identify potential quality problems
- Determine whether or not the metrics actually do tell us something about software quality
- Incorporate the use of the metrics into the software development process

Research Agenda

- Develop software metrics that might tell us something about software quality
  - describe some metrics
- Develop instruments to measure code according to the identified metrics
  - discuss issues developing such tools
- Apply the instruments to existing code

- Analyse the resulting data to identify potential quality problems
- Determine whether or not the metrics actually do tell us something about software quality
- Incorporate the use of the metrics into the software development process

Research Agenda

- Develop software metrics that might tell us something about software quality
  - describe some metrics
- Develop instruments to measure code according to the identified metrics
  - discuss issues developing such tools
- Apply the instruments to existing code
  - discuss issues relating to dealing with existing code
- Analyse the resulting data to identify potential quality problems
- Determine whether or not the metrics actually do tell us something about software quality
- Incorporate the use of the metrics into the software development process

Research Agenda

- Develop software metrics that might tell us something about software quality
  - describe some metrics
- Develop instruments to measure code according to the identified metrics
  - discuss issues developing such tools
- Apply the instruments to existing code
  - discuss issues relating to dealing with existing code
- Analyse the resulting data to identify potential quality problems
- Determine whether or not the metrics actually do tell us something about software quality
- Incorporate the use of the metrics into the software development process
  - integrate instruments into IDEs

What is Quality?

Understandability If I want to completely understand a given class, what other classes do I need to understand?

Testability If I want to test a given class, what other classes do I need to test?

Reusability If I want to reuse a given class, what other classes do I need to reuse?

Readability . . .

Comprehensibility . . .

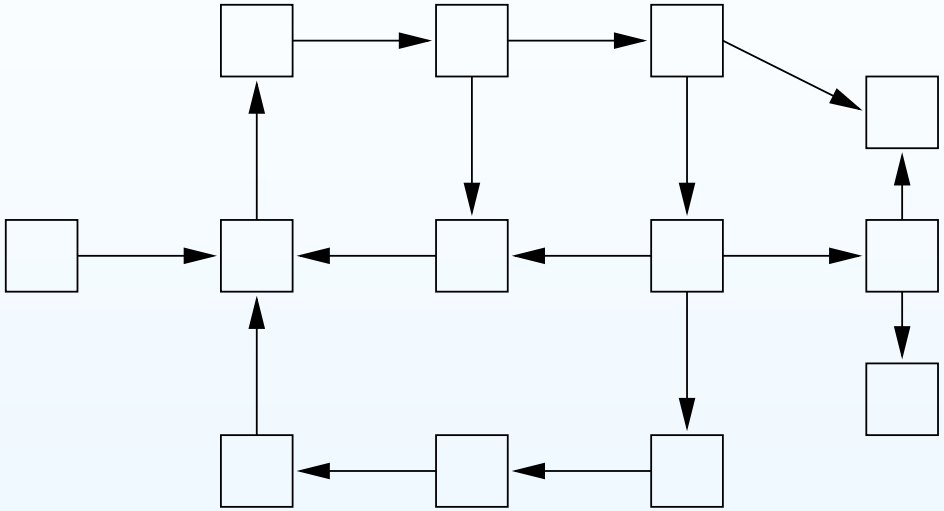
Changeability . . .

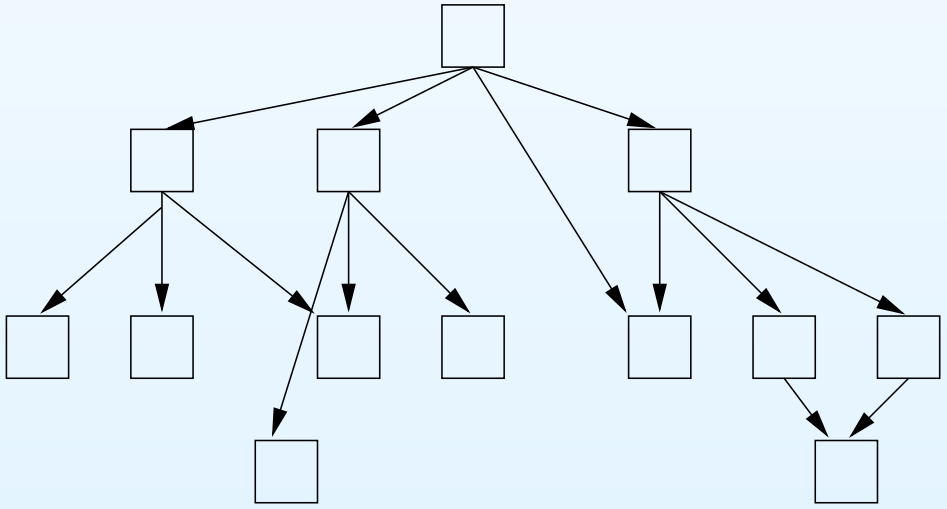
Maintainability . . .

. . .

$\Rightarrow$ possibly multiple metrics

Which is the better design?





Example

```
class A {
  public void aMethod1(B ab) {
    ...
  }
}
```

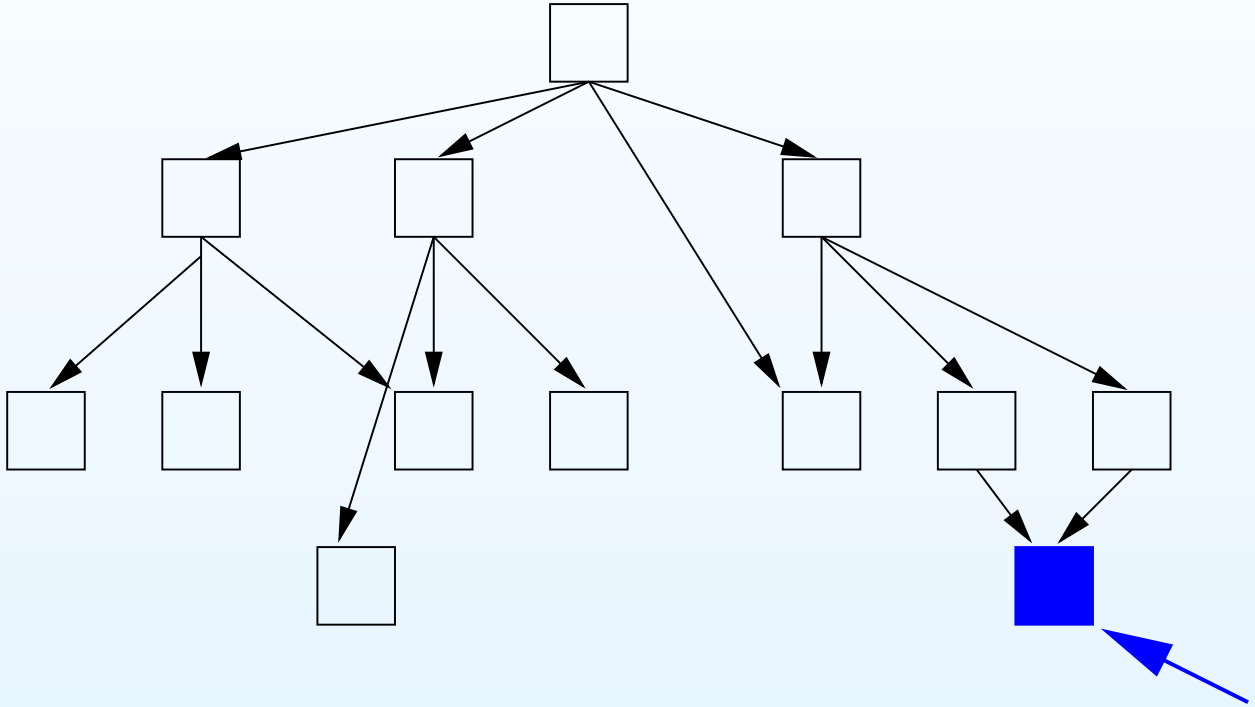
```
class B {
  public void bMethod() {
      C aC = new C();
    ...
  }
}
```

- understandability of A?
- testability of A?
- reusability of A?

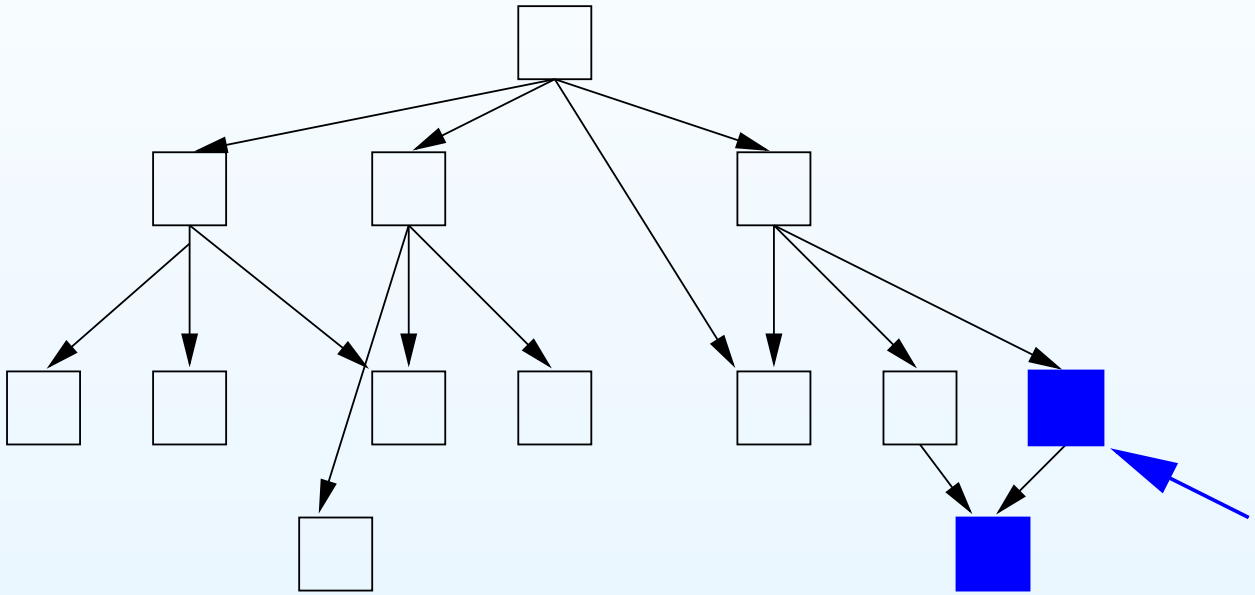
Class Reachability Sets

- Class reachability set for class A = the set of classes that A transitively depends on
- Class reachability set size (CRSS) — metric for transitive (compilation) dependence
- Hayden Melton's PhD research

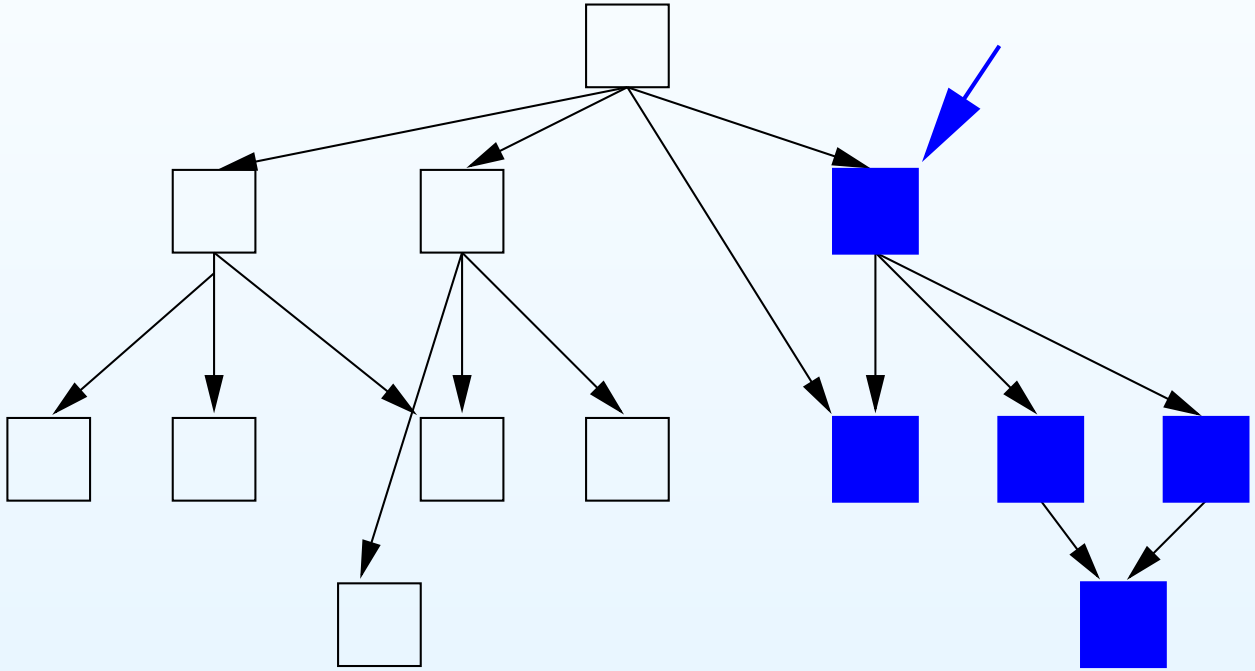
Class Reachability Sets



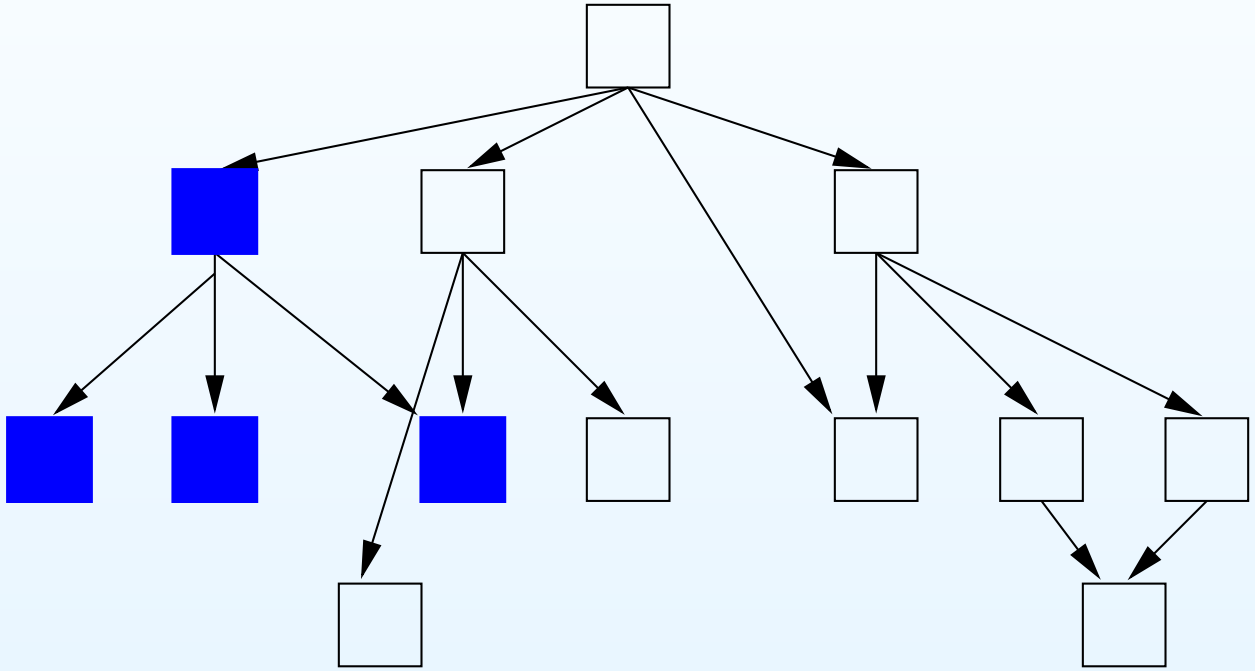
Class Reachability Sets



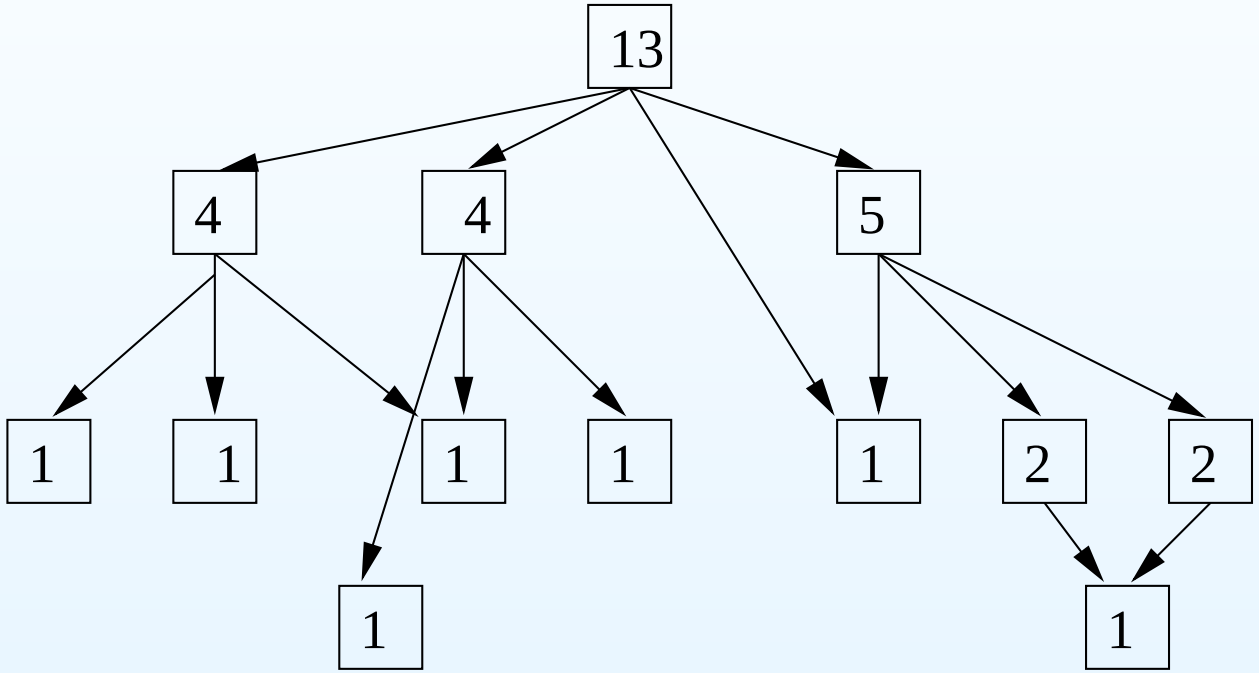
Class Reachability Sets



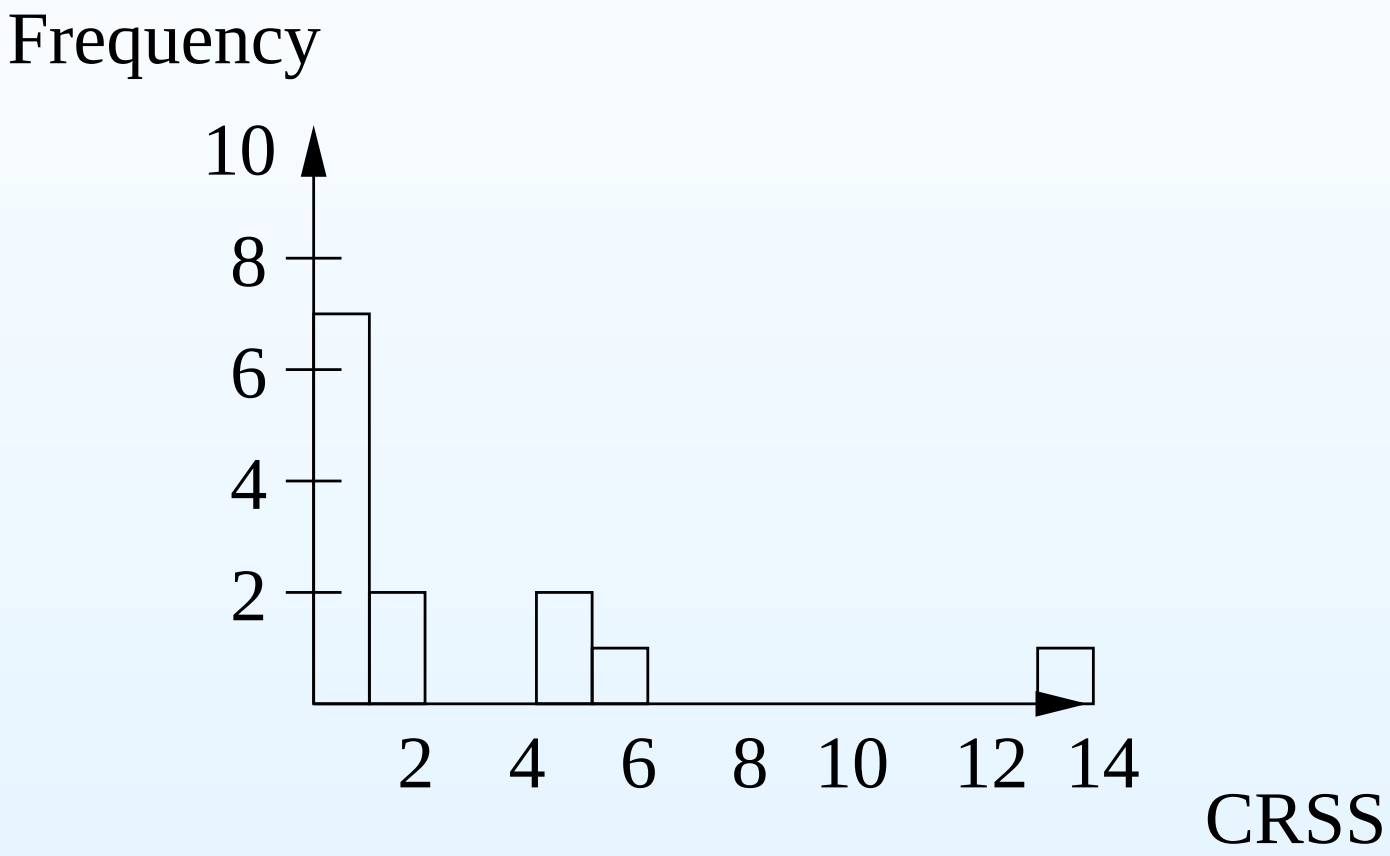
Class Reachability Sets



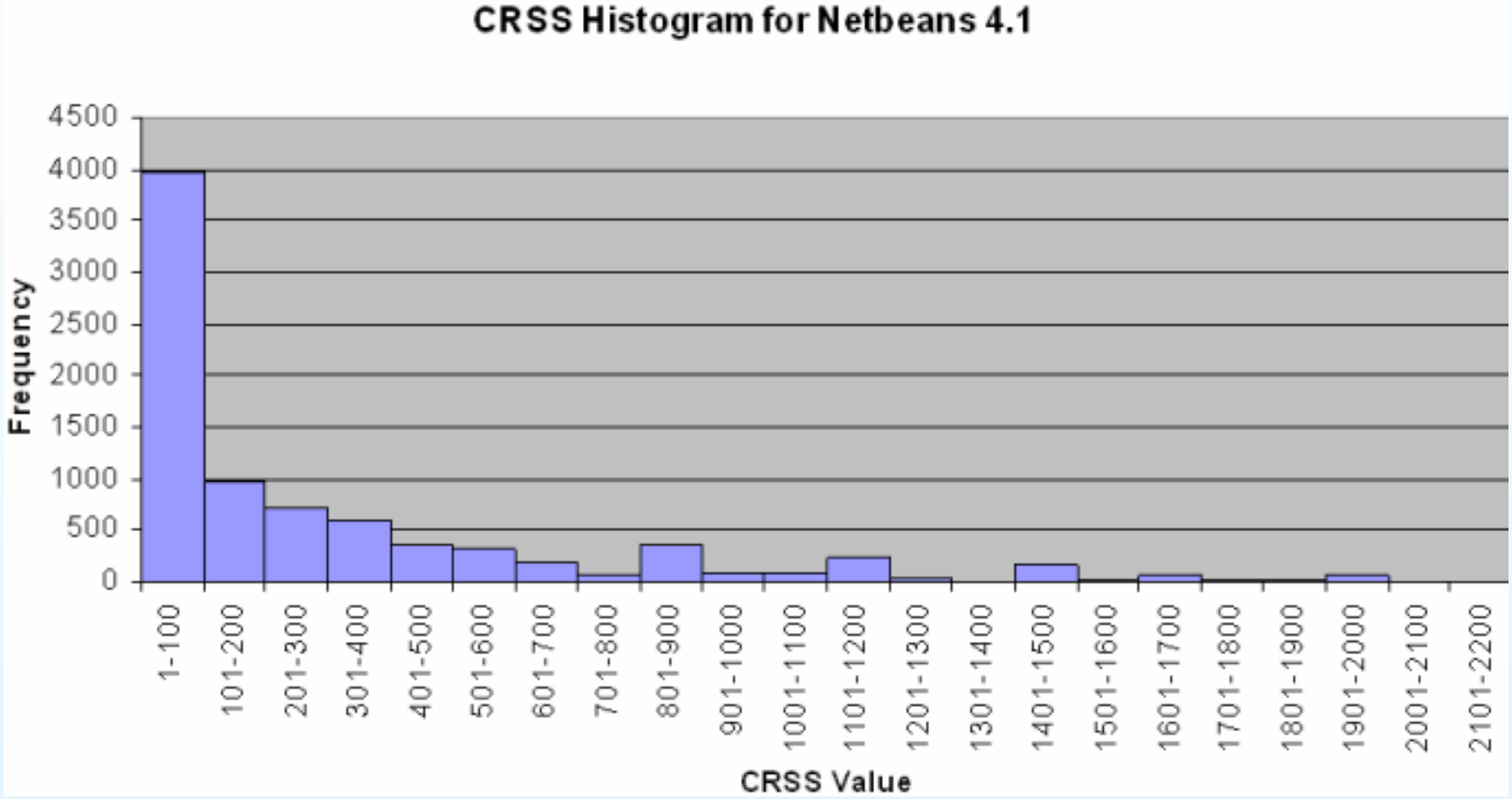
CRSS values



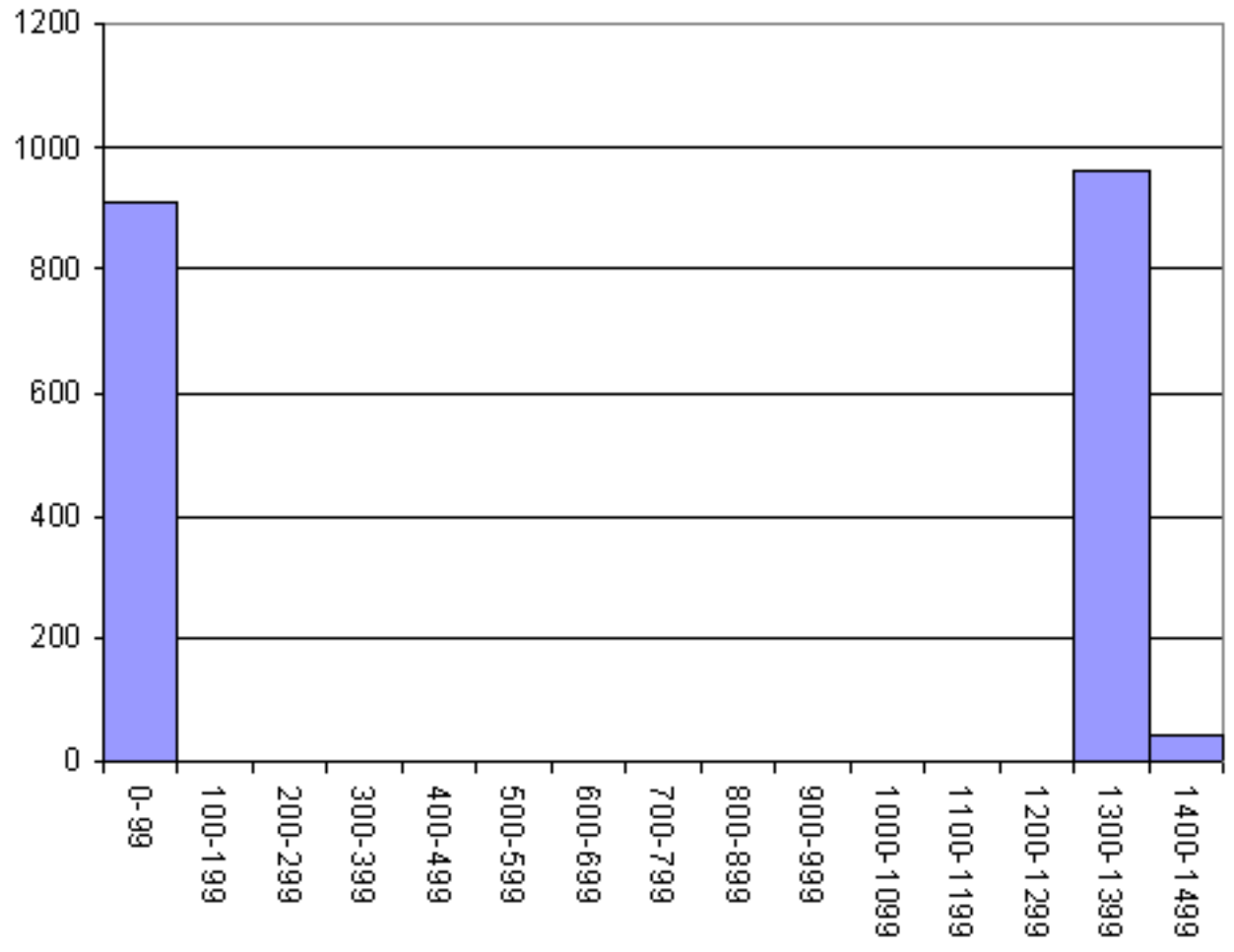
Presenting CRSS data



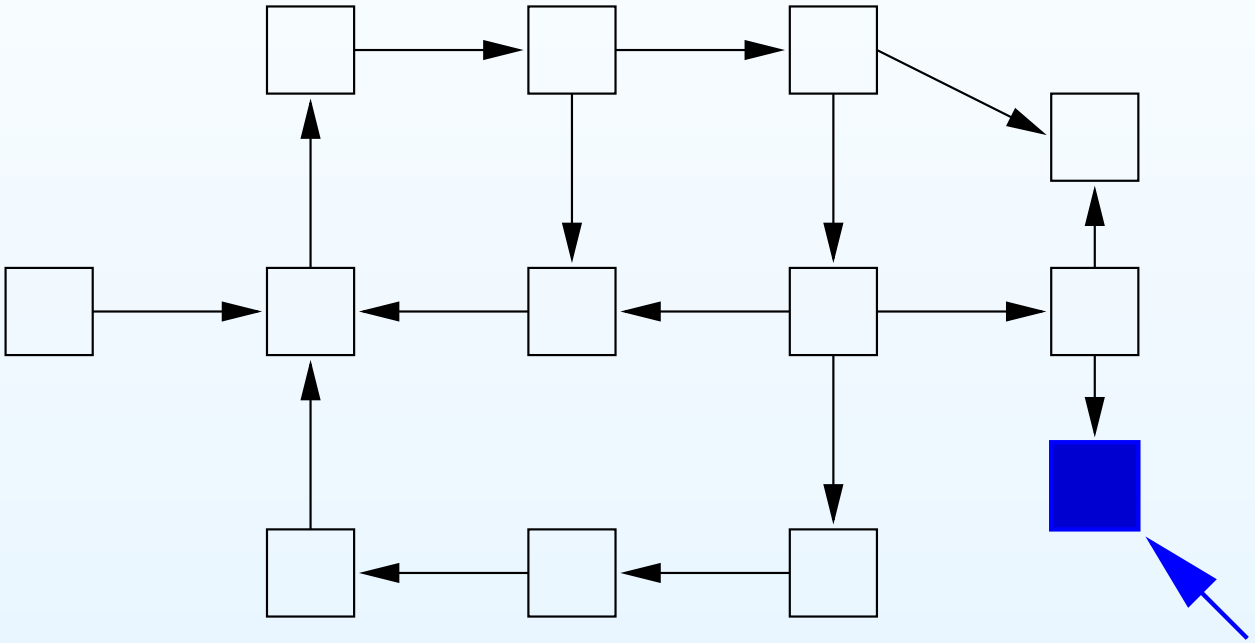
Netbeans CRSS



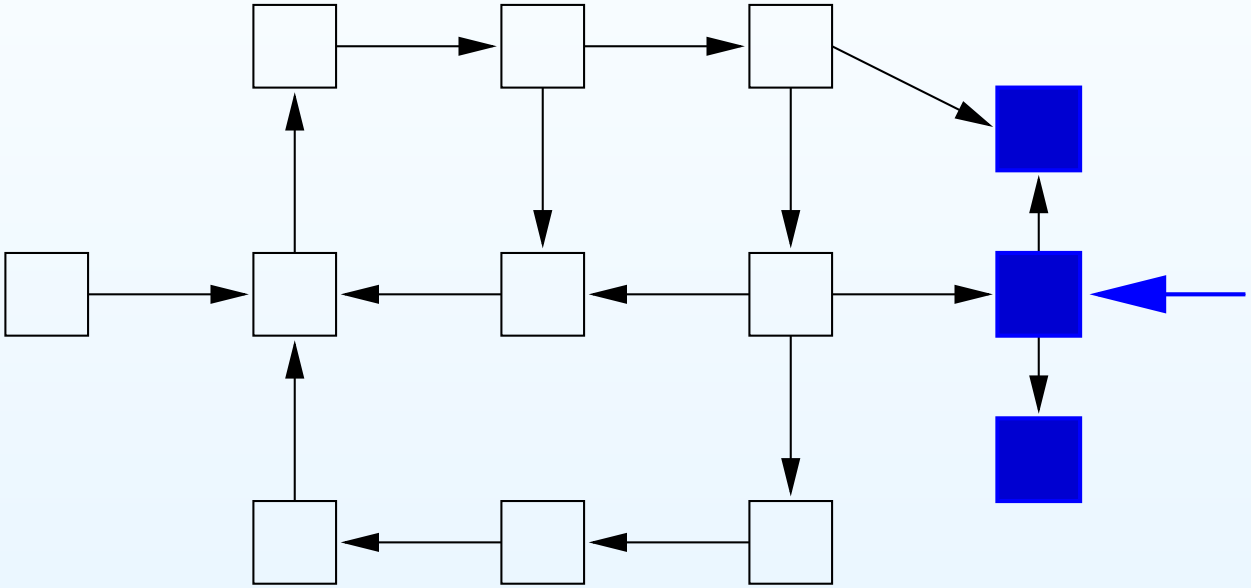
Azureus CRSS distribution



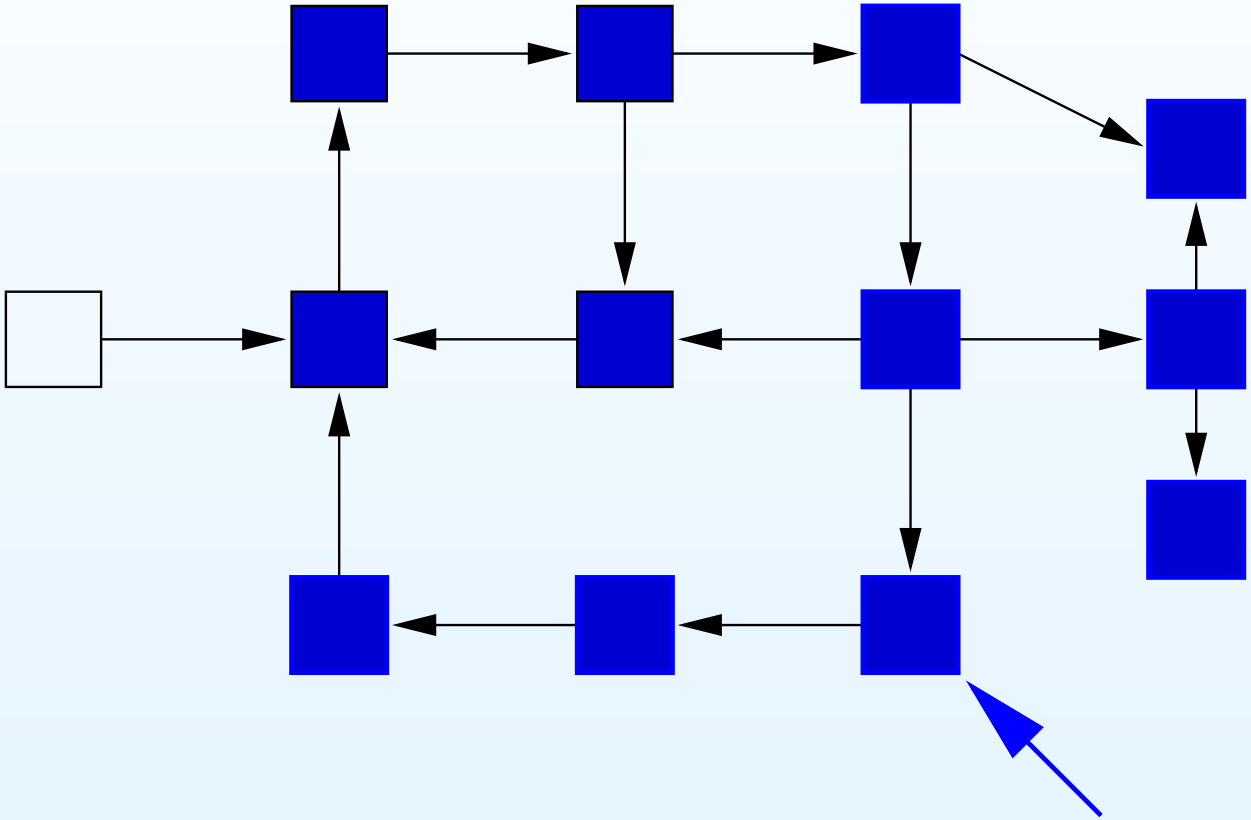
Class Reachability Sets



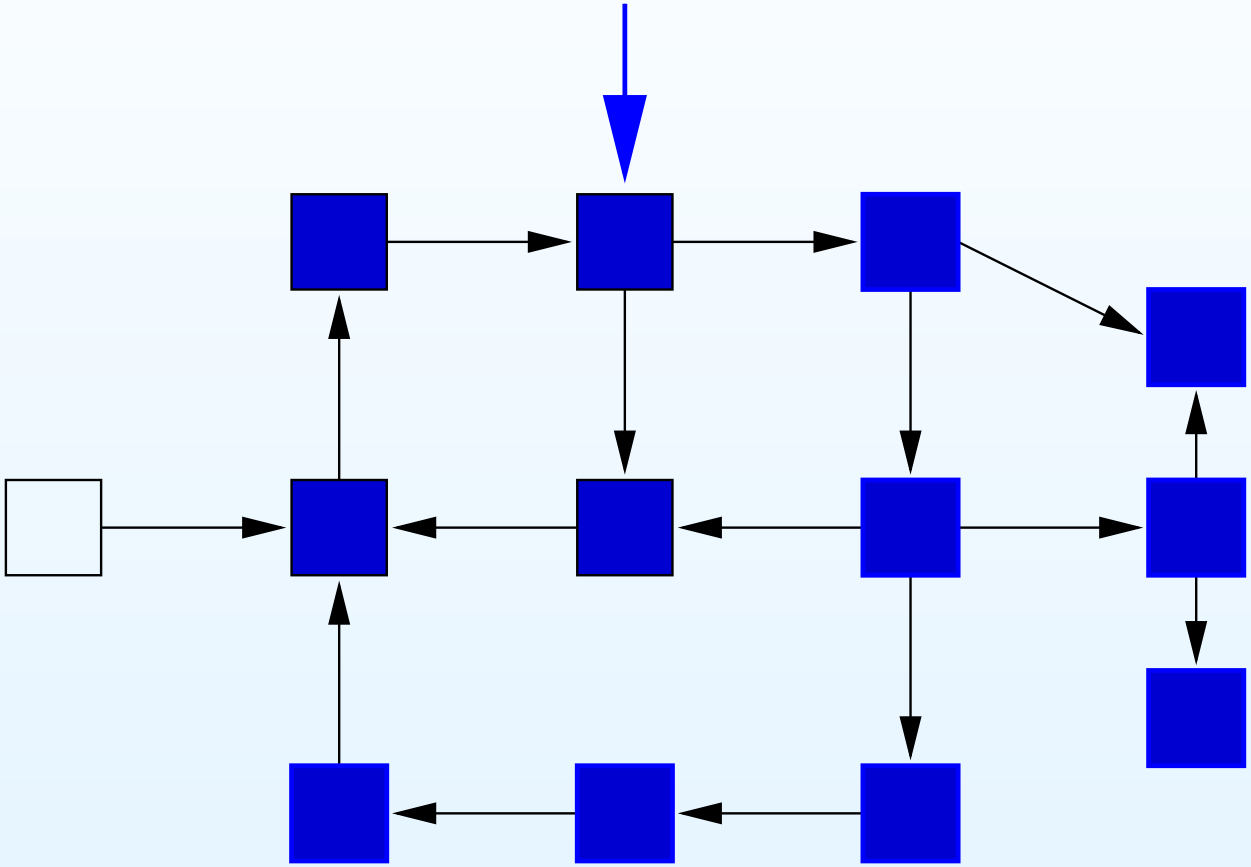
Class Reachability Sets



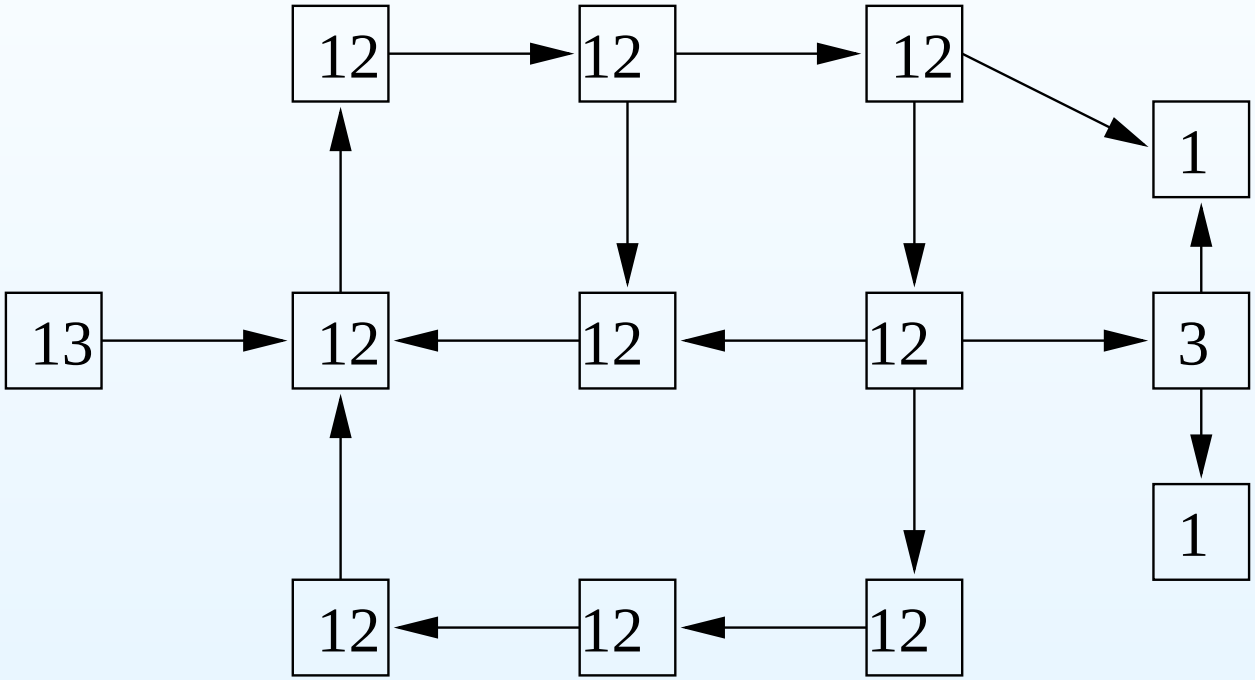
Class Reachability Sets



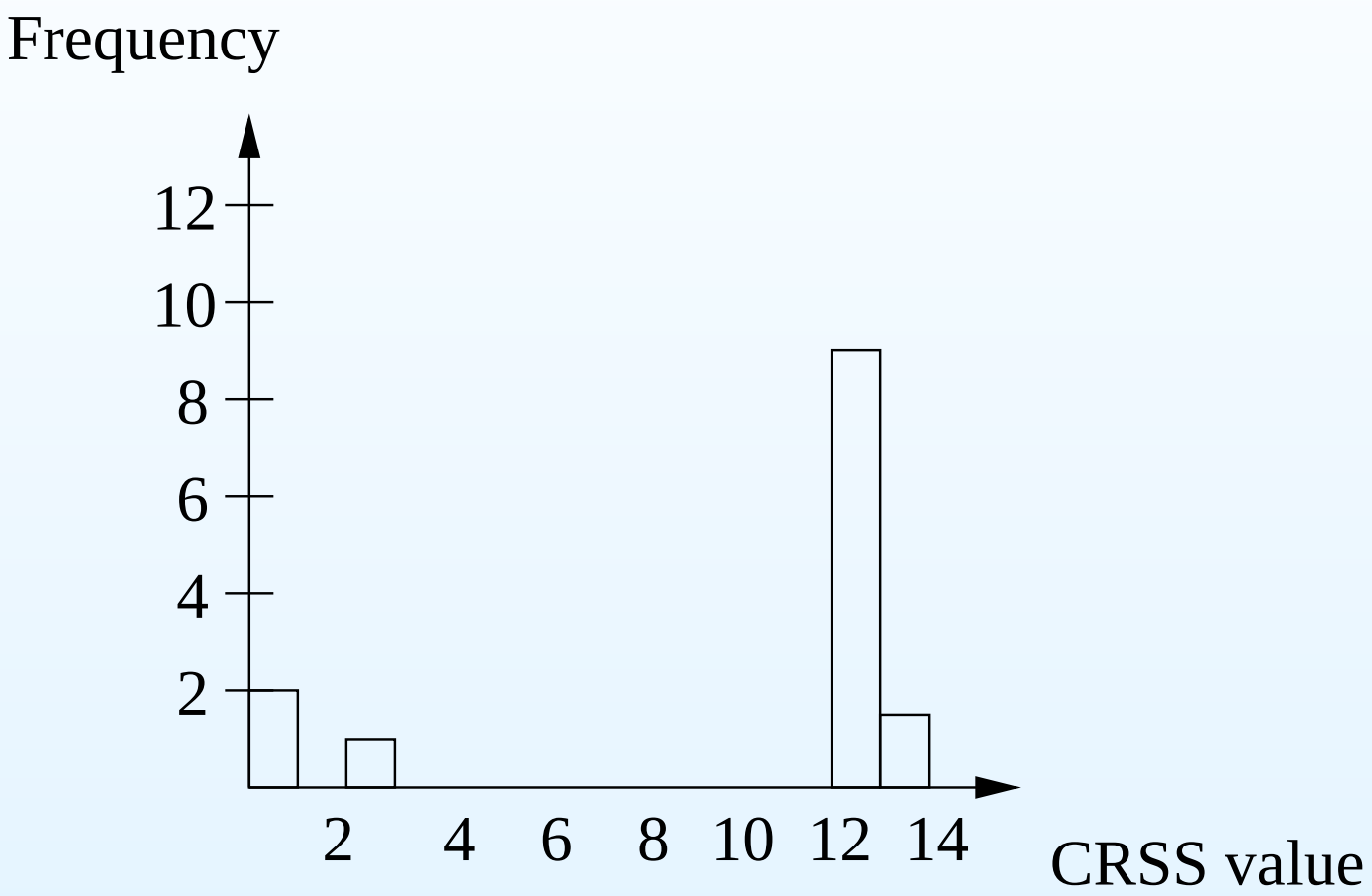
Class Reachability Sets



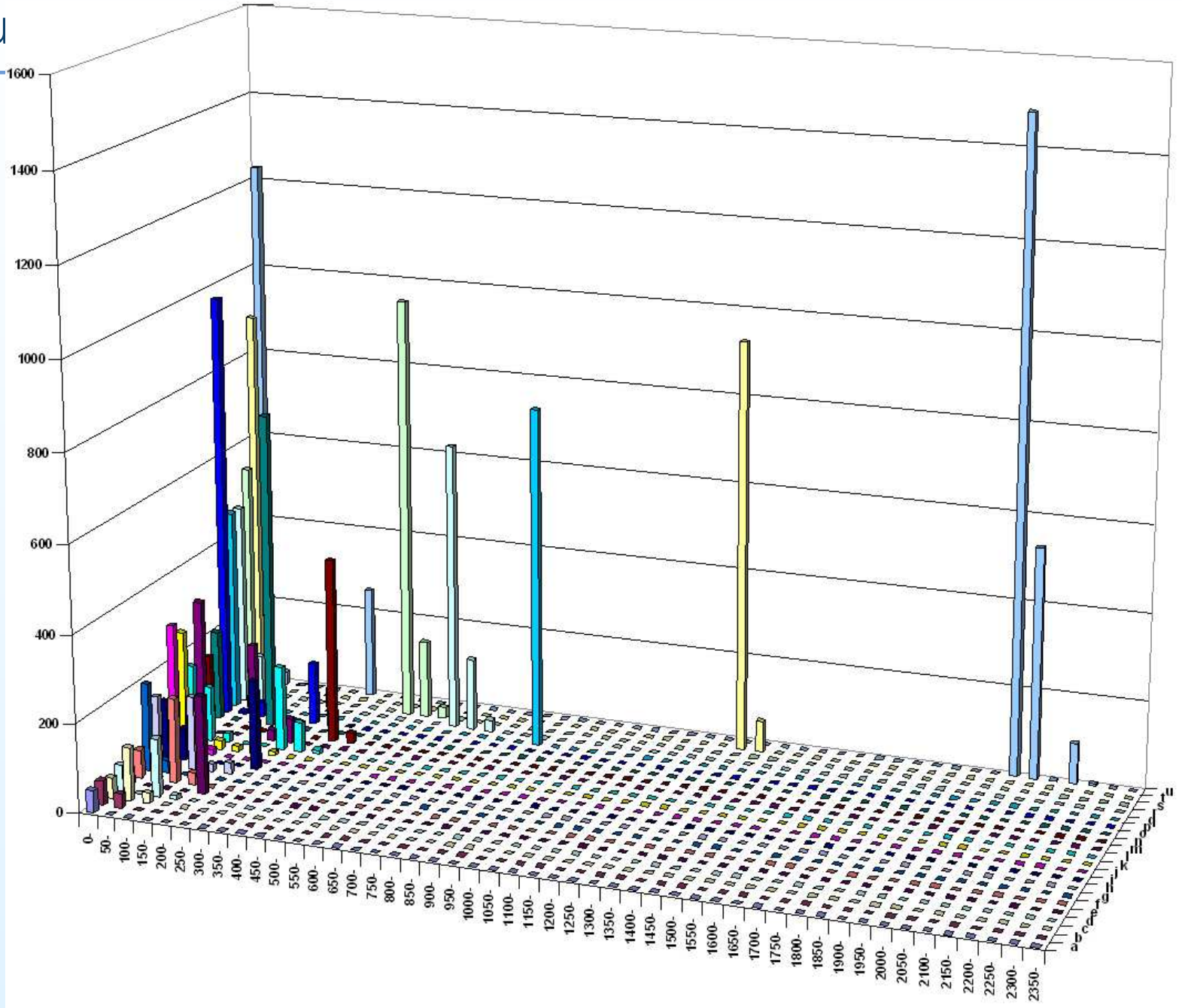
CRSS values



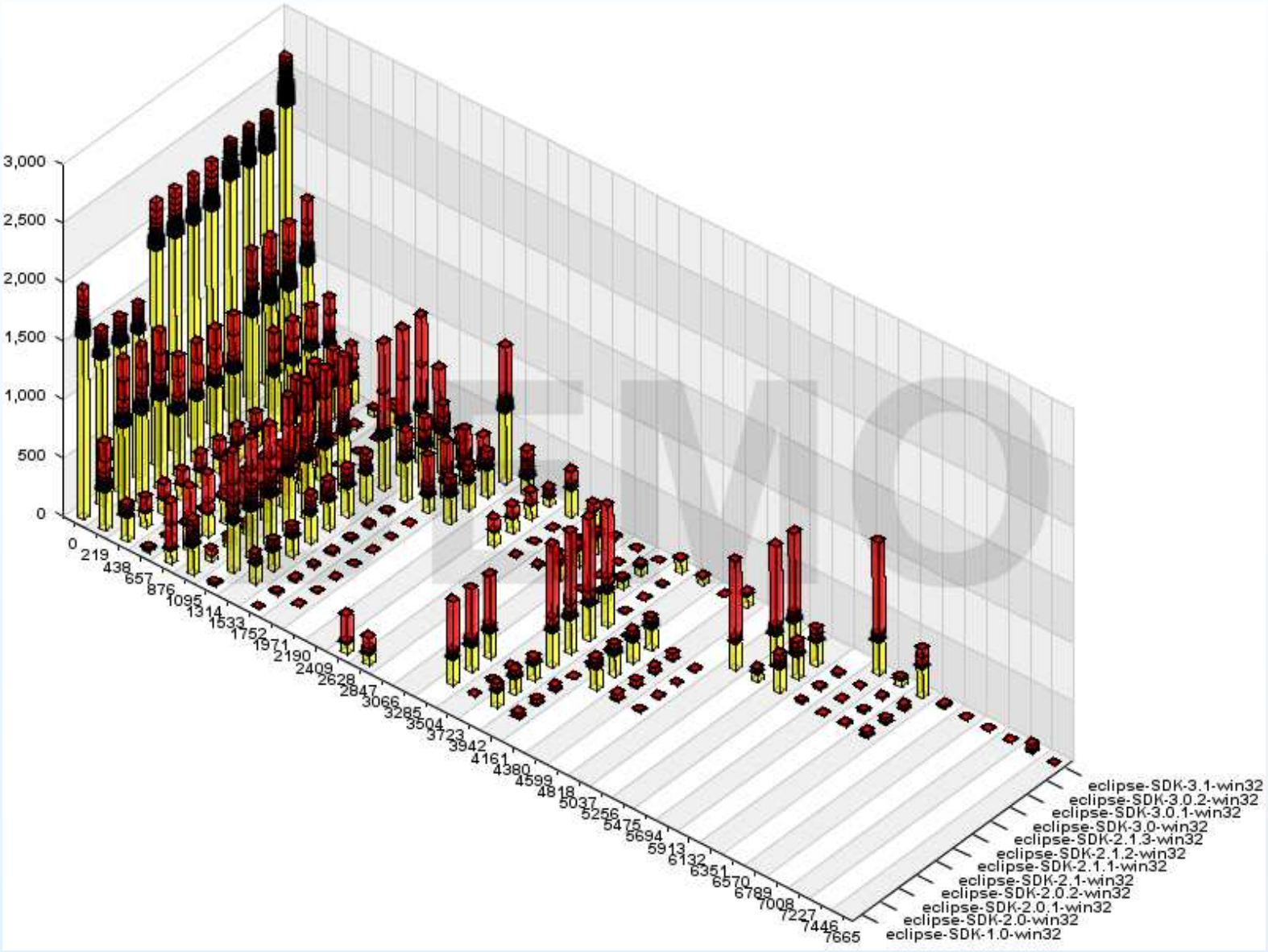
Presenting CRSS data



Corpu


1600
1400
1200
1000
800
600
400
200
0
0-
50-
100-
150-
200-
250-
300-
350-
400-
450-
500-
550-
600-
650-
700-
750-
800-
850-
900-
950-
1000-
1050-
1100-
1150-
1200-
1250-
1300-
1350-
1400-
1450-
1500-
1550-
1600-
1650-
1700-
1750-
1800-
1850-
1900-
1950-
2000-
2050-
2100-
2150-
2200-
2250-
2300-
2350-

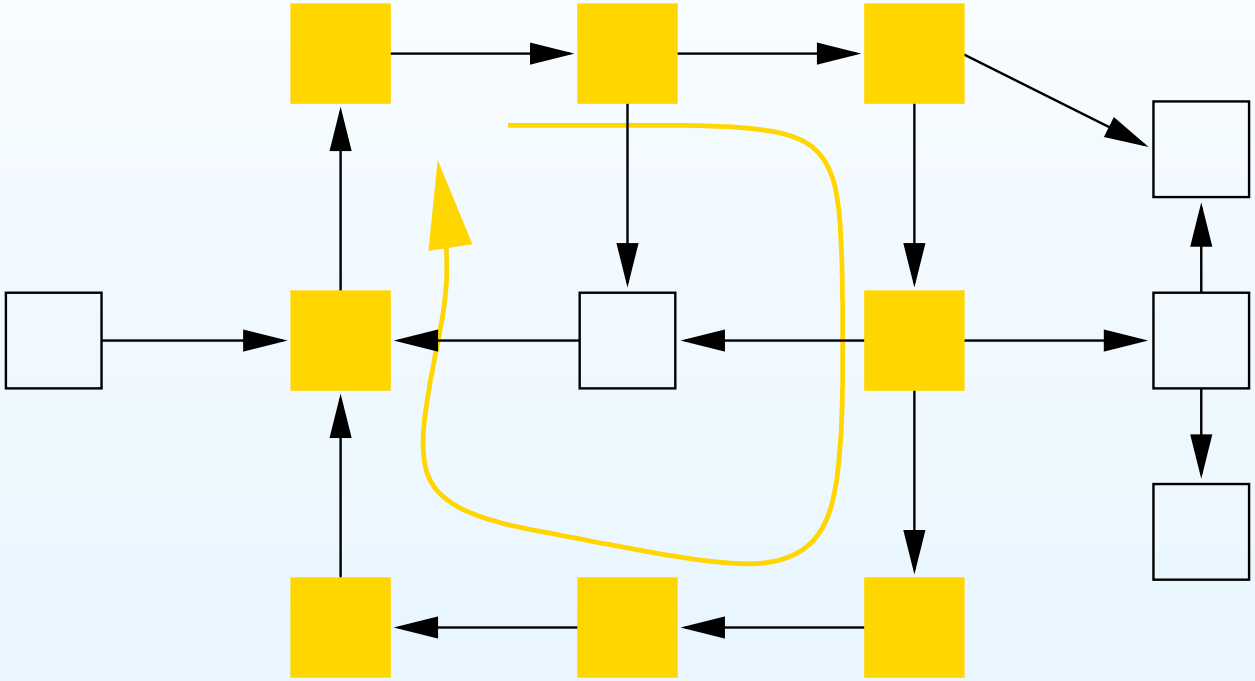
Eclipse CRSS progression



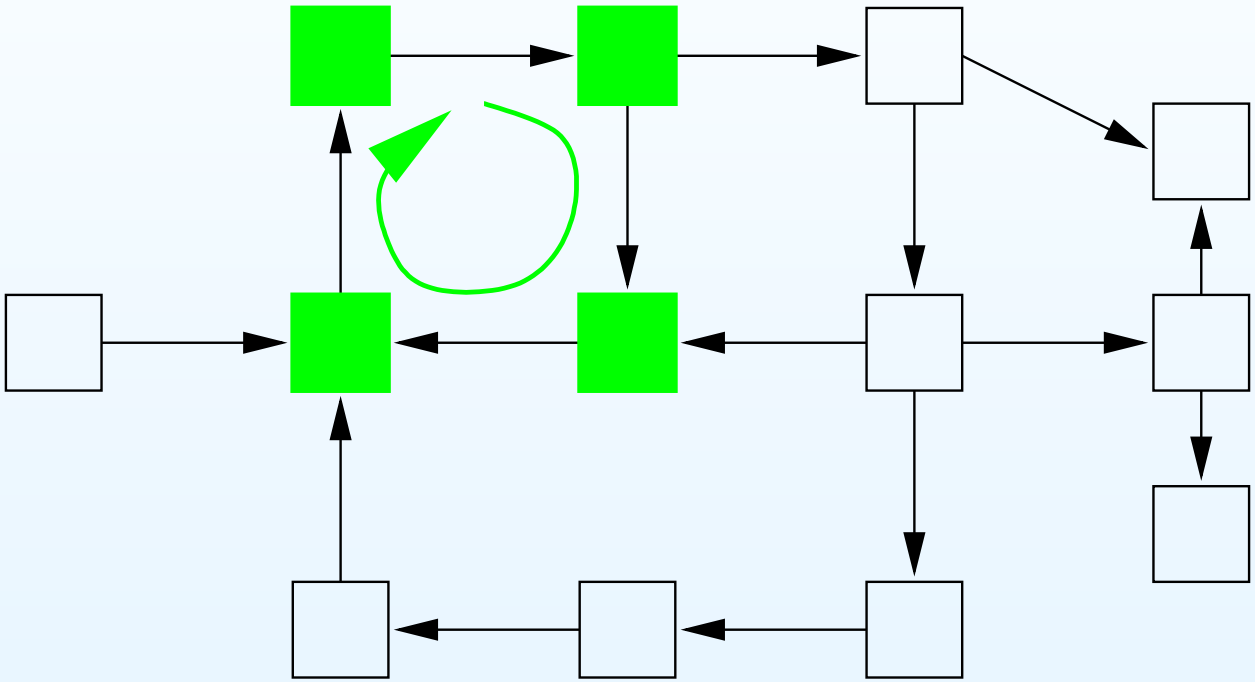
Why Cycles are bad

- Understandability — where to start?
- Testability — where to start?
- Reusability — have to take everything

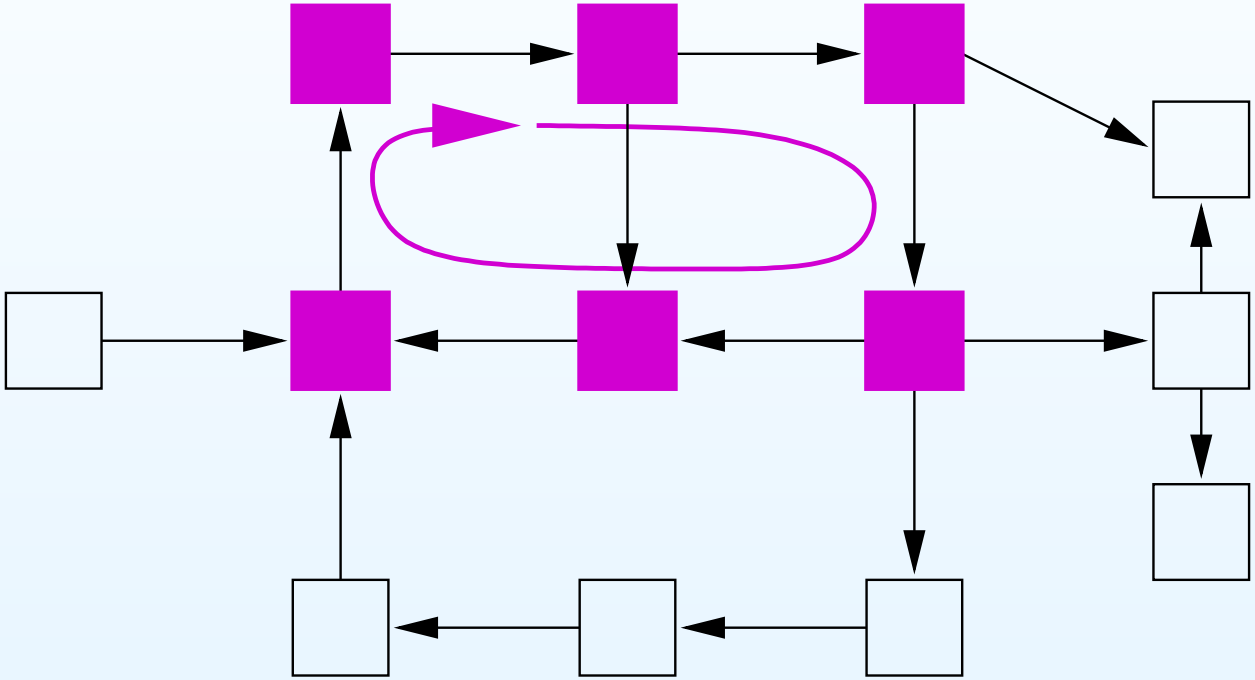
“Measuring Cycles”



“Measuring Cycles”



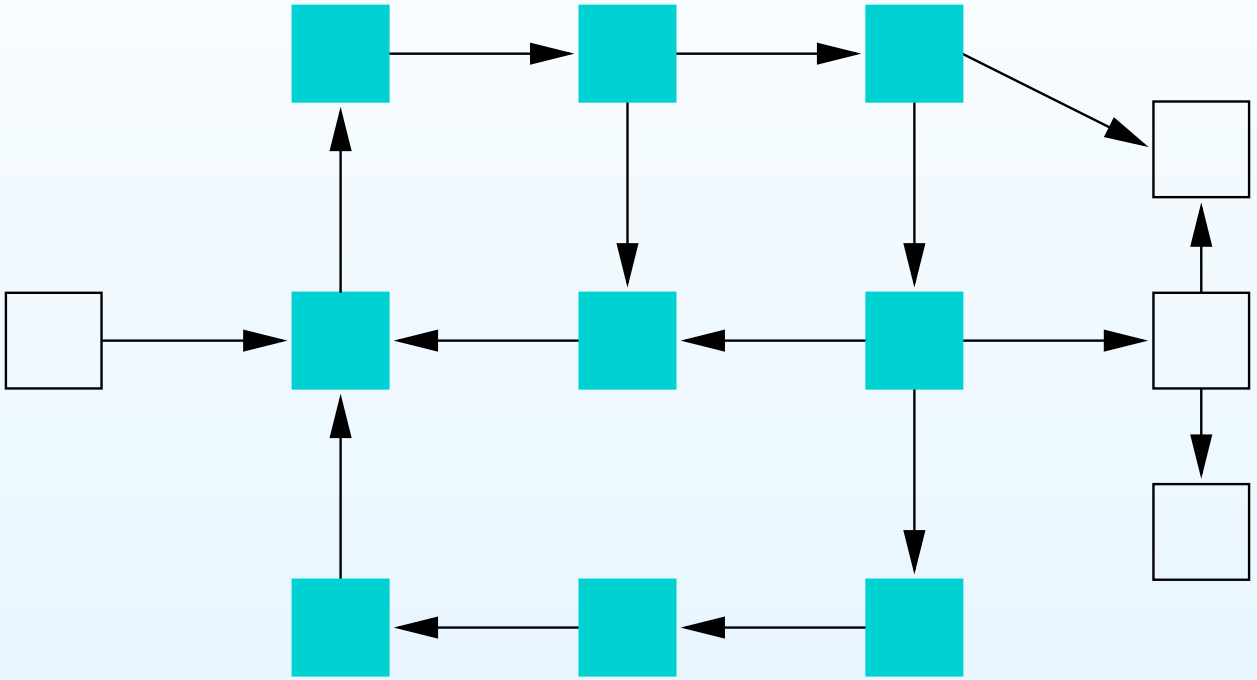
“Measuring Cycles”



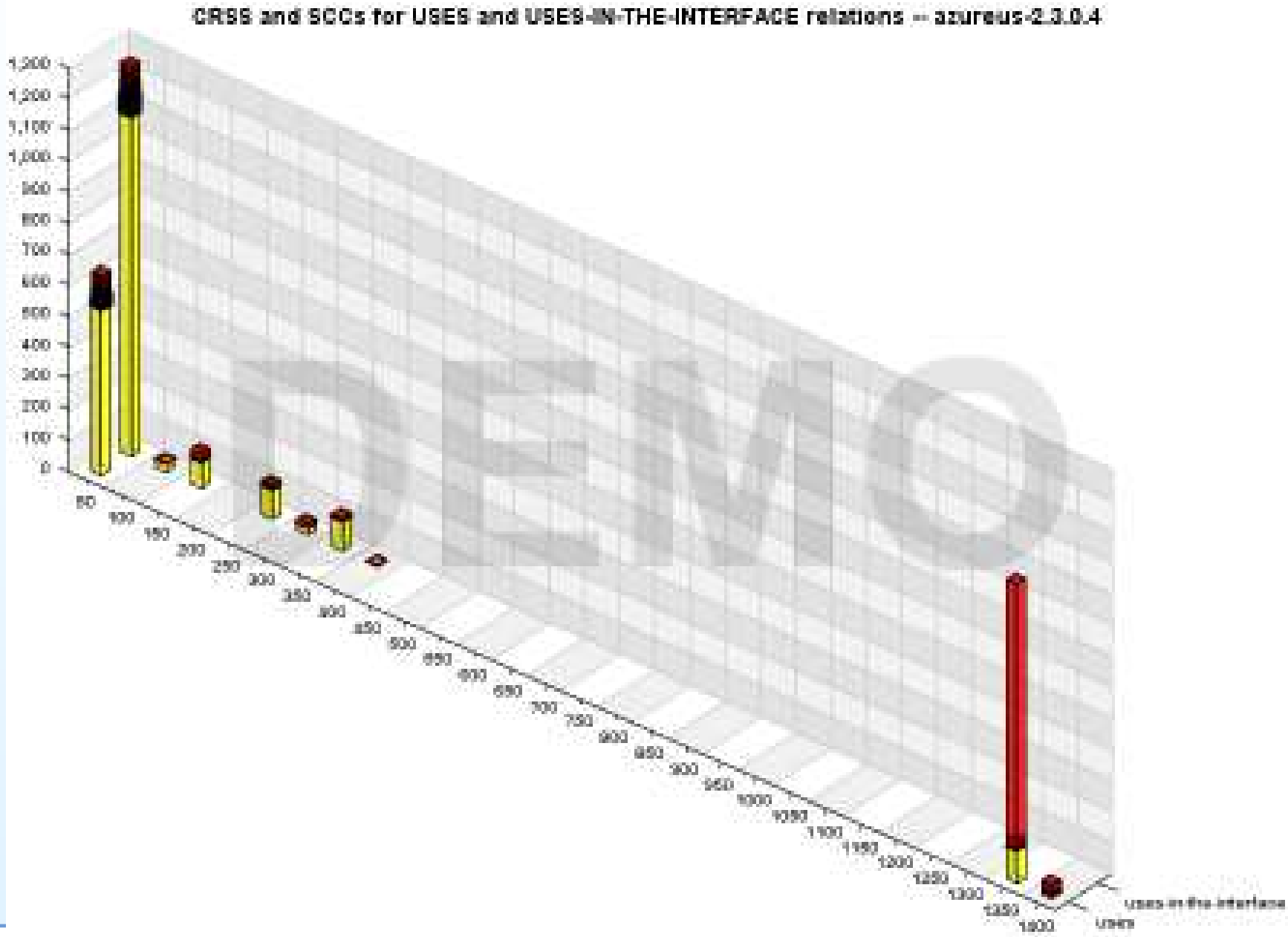
SCCS

- Strongly Connected Components (SCC) — subgraph in which every vertex is *reachable* from every other vertex
- Largest “cycle” for a given set of vertices
- SCCS — Strongly Connected Component Size

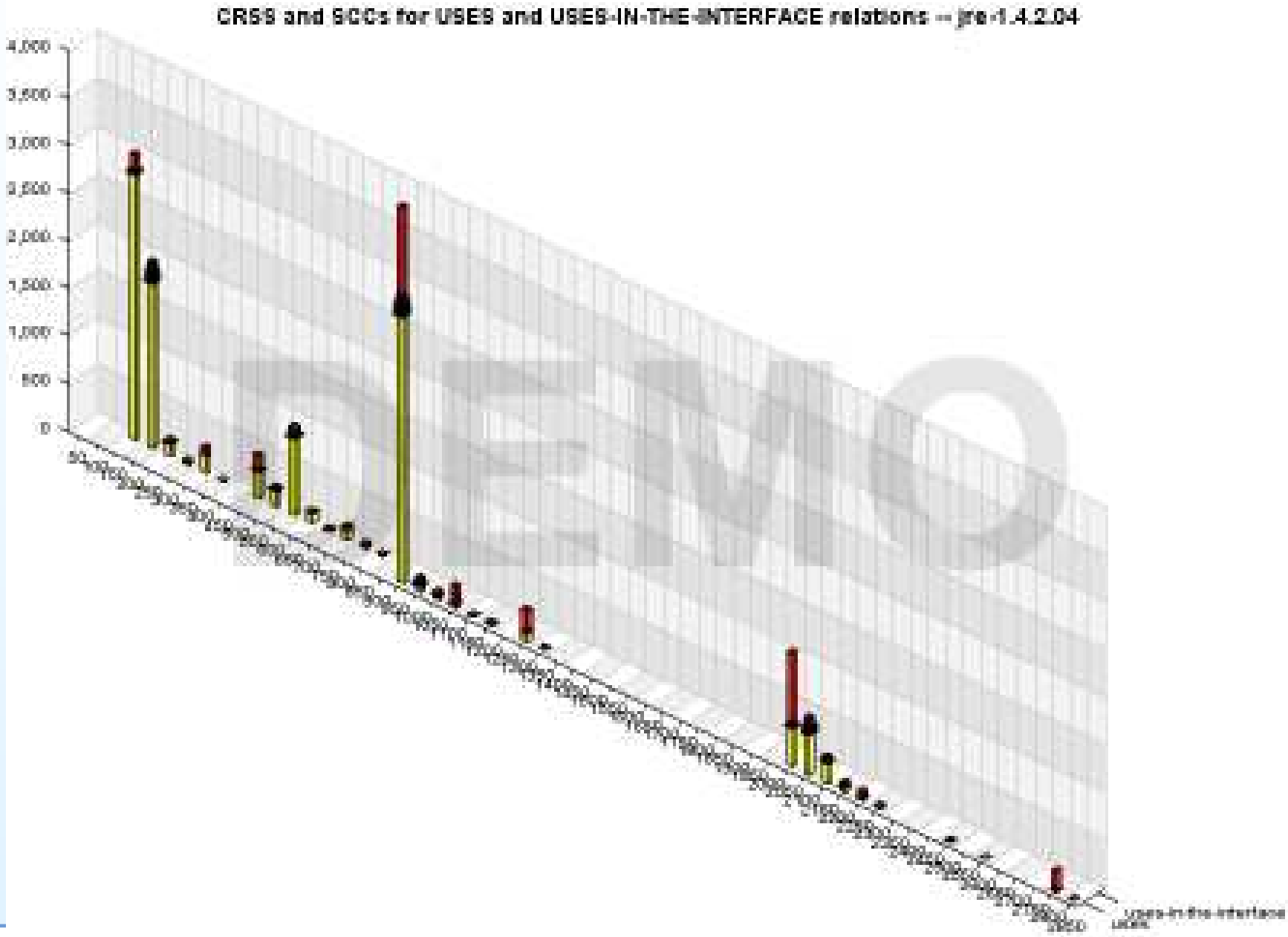
SCCS

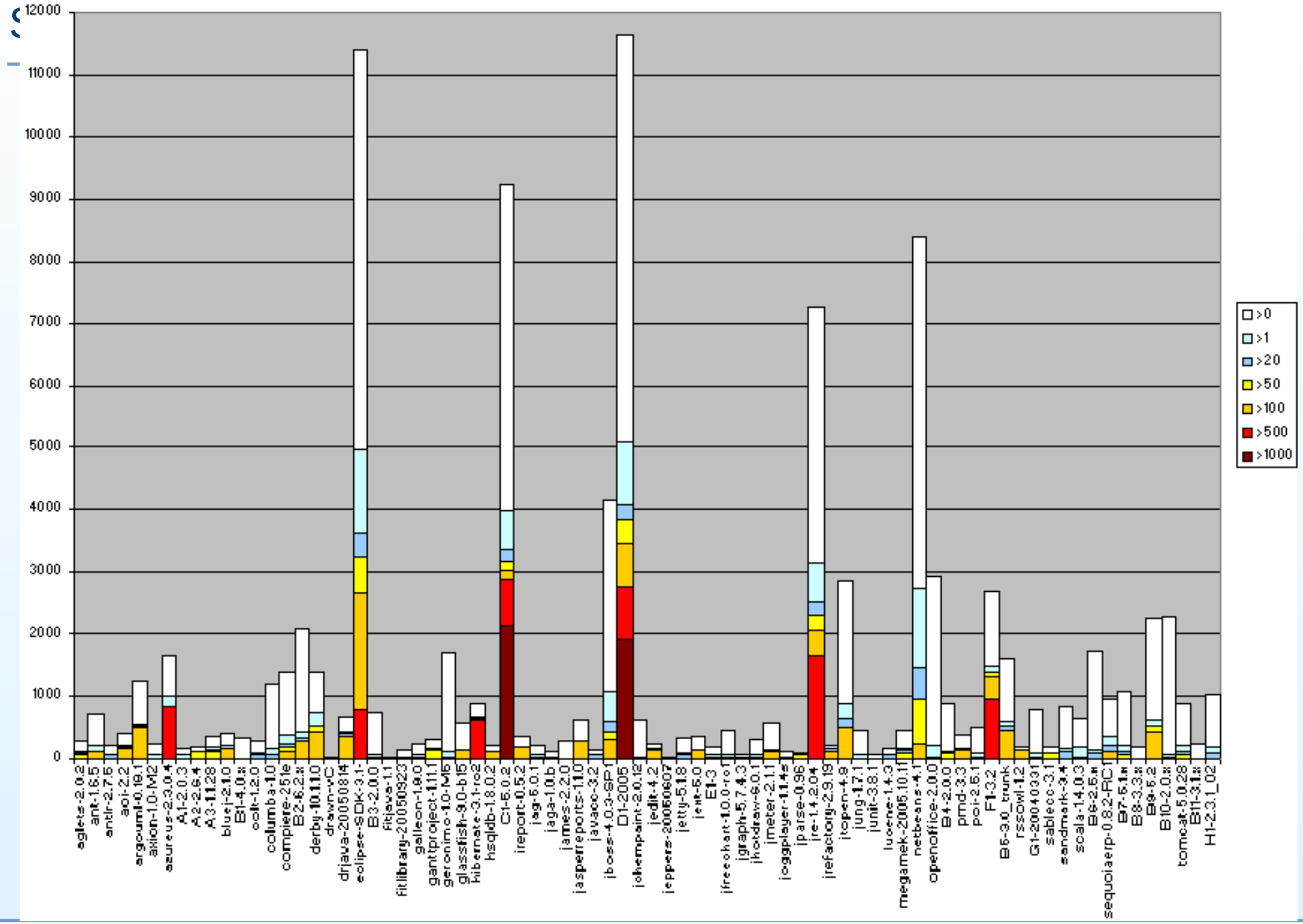


Azureus-2.3.0.4 SCCS



JRE-1.4.2.04 SCCS




12000
11000
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
>0
>1
>20
>50
>100
>500
>1000
sglets-2.0.2
ant-1.6.5
antlr-2.7.5
aoi-2.2
argouml-0.16.1
axion-1.0-M2
azureus-2.3.0.4
A1-2.0.3
A2-2.6.4
A3-1.1.28
bluej-2.1.0
B1-4.0.x
colt-1.2.0
columba-1.0
compiere-251e
B2-6.2.x
derby-10.1.1.0
drawswf-1.1.0
drjava-20050814
eclipse-SDK-3.1-
B3-2.0.0
fitjava-1.1
fitlibrary-20050923
galleon-1.9.0
ganttproject-1.11.1
geronimo-1.0-M6
glassfish-9.0-b15
hibernate-3.1-rc2
hsqldb-1.8.0.2
C1-5.0.2
ireport-0.5.2
jag-5.0.1
jaga-1.0.b
james-2.2.0
jasperreports-1.1.0
javacc-3.2
jboss-4.0.3-SP1
D1-2005
jchempaint-2.0.12
jedit-4.2
jeppers-20060607
jetty-5.1.8
jext-5.0
E1-3
jfreechart-1.0.0-rc1
jgraph-5.7.4.3
jhotdraw-6.0.1
jmeter-2.1.1
joggplayer-1.1.4.s
jparse-0.96
jre-1.4.2.04
jrefactory-2.9.19
jtopen-4.9
jung-1.7.1
junit-3.8.1
lucene-1.4.3
megamek-2005.10.11
netbeans-4.1
openoffice-2.0.0
B4-2.0.0
pmd-3.3
poi-2.5.1
F1-3.2
B6-3.0_trunk
rssowl-1.2
G1-20040331
sablecc-3.1
sandmark-3.4
scala-1.4.0.3
B6-2.5.x
sequoiaerp-0.8.2-RC1
B7-5.1.x
B8-3.3.x
B9-6.2
B10-2.0.x
tomcat-5.0.28
B11-3.1.x
H1-2.3.1_02

The joy of debugging

```
public class D {
  private String s;
  public D(String s) {
    this.s = s;
  }
  public void doD() {
    System.out.println(s.trim());
  }
}
```

- When executing an application containing this class, a `NullPointerException` occurs when `trim` is called.
- Questions we must answer in trying to identify the fault that led to the observed failure:
  - How is it that `s` is null?
  - Where did the `null` come from?

Finding null

```
public class A {
    public static void main(String[] args) {
        C c = B.create();
        D d = new D(c.get());
        d.doD();
        E.doE(c);
    }
}

public class C {
    private String str;
    public String get() {
        return str;
    }
    public void set(String str) {
        this.str = str;
    }
}
public class E {
    public static void doE(C c) {
        F f = new F();
        f.doF(c.get());
    }
}
```

```
public class B {
    String name;
    public static C create() {
        C c = new C();
        c.set(name);
    }
}

public class D {
    private String s;
    public D(String s) {
        this.s = s;
    }
    public void doD() {
        System.out.println(s.trim() );
    }
}

public class F {
    public void doF(String string) {
        D d = new D(string);
        do.doit();
    }
}
```

Use-Def Chains

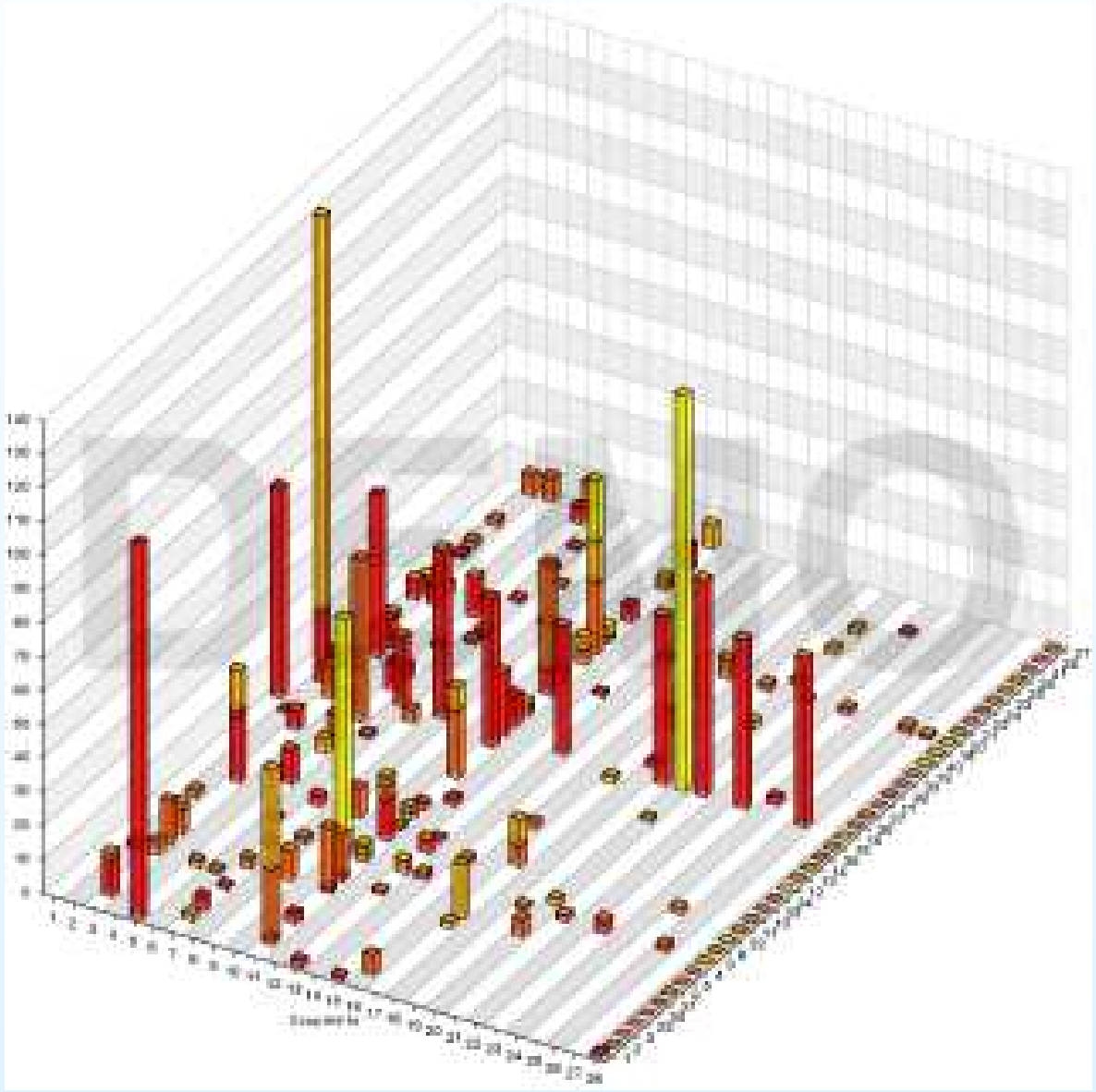
```
public class A {
    public static void main(String[] args) {
        C c = B.create();
        D d = new D(c.get());
        d.doD();
        E.doE(c);
    }
}

public class B {
    String name;
    public static C create() {
        C c = new C();
        c.set(name);
    }
}

public class C {
    private String str;
    public String get() {
        return str;
    }
    public void set(String str) {
        this.str = str;
    }
}

public class D {
    private String s;
    public D(String s) {
        this.s = s;
    }
    public void doD() {
        System.out.println(s.trim() );
    }
}

public class E {
    public static void doE(C c) {
        F f = new F();
        f.doF(c.get());
    }
}

public class F {
    public void doF(String string) {
        D d = new D(string);
        do.doit();
    }
}
```

Use-Def Indirect Coupling

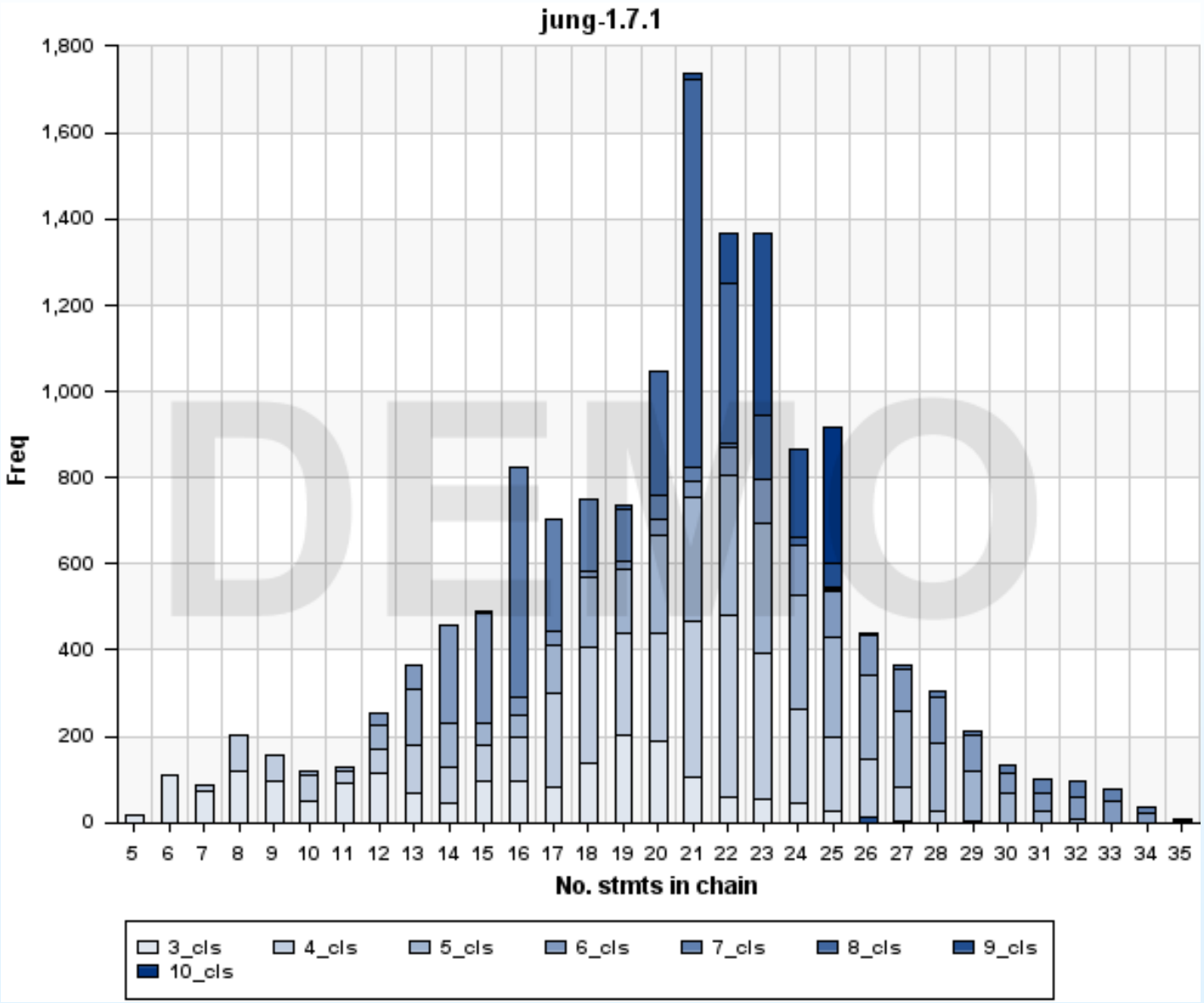
- A class A is use-def indirectly coupled to a class B if a value **defined** in B is **used** in A
- The path between use and def is called a use-def chain
- Intuition is that this kind of coupling can be a source of difficulty — when a failure occurs in D it's not obvious that one should consider B

Hong Yul Yang's PhD research

Indirect Coupling between classes



Lengths of chains



How to measure code?

- Need to parse code — create syntax model (aka abstract syntax trees)
- Need to resolve names — create semantic model
- Need sophisticated analysis for some metrics
- Need code!

Parsing Issues

- Different issues for different languages
- Even “simple” languages such as Java are not easy to parse
- Analysing code $\neq$ compiling code

Language Issues

```
#include <iostream>
#include <string>
#include "Address.cpp"

using namespace std;

int main() {
        Address add;
        cout << add.print() << endl;
}
```

But Java is a simple language!

- lots of parsers around, none of them easy to use
- lots of grammars around, all of questionable quality
- and Java isn't simple . . .
- (but it is simpler than C++)
- (and there's lots of code around)
- (and we can analyse bytecode!)

Source vs Bytecode

```
class A {
    E field;
    public void method(B aB, C aC) {
      D aD = aB;
      field.doit(aC.getIt(aD), Math.PI);
    }
}
```

Source vs Bytecode

```
class A {
    E field;
    public void method(B aB, C aC) {
      D aD = aB;
      field.doit(aC.getIt(aD), Math.PI);
    }
}
```

Source vs Bytecode

```
class A {
    E field;
    public void method(B aB, C aC) {
      D aD = aB;
      field.doit(aC.getIt(aD), Math.PI);
    }
}
```

Source vs Bytecode

```
enum MyEnum { A, B, C; }
```

```
class MyEnum extends java.lang.Enum {

}
```

Analysing code $\neq$ compiling code

- Need to build model for whole “program”
  - E.g., Eclipse is about 12,000 classes, 1.5 million LOC — one model?
- Some metrics require analysis not required in compilation
  - E.g., use-def chains

Software Corpora

- Corpus — A collection of writings or recorded remarks used for linguistic analysis
- If all analysis is done on the same corpus, then results are more likely to be comparable (benchmarks)
- If the corpus is representative of a population, then findings due to analysis might apply to the whole population
- Should work for software too!
  - $>$ 100 open-source Java applications
  - $>$ 250 versions (e.g. 13 versions of eclipse)
  - source code of most
  - byte code of most
  - needs to be managed

From Metrics to Process

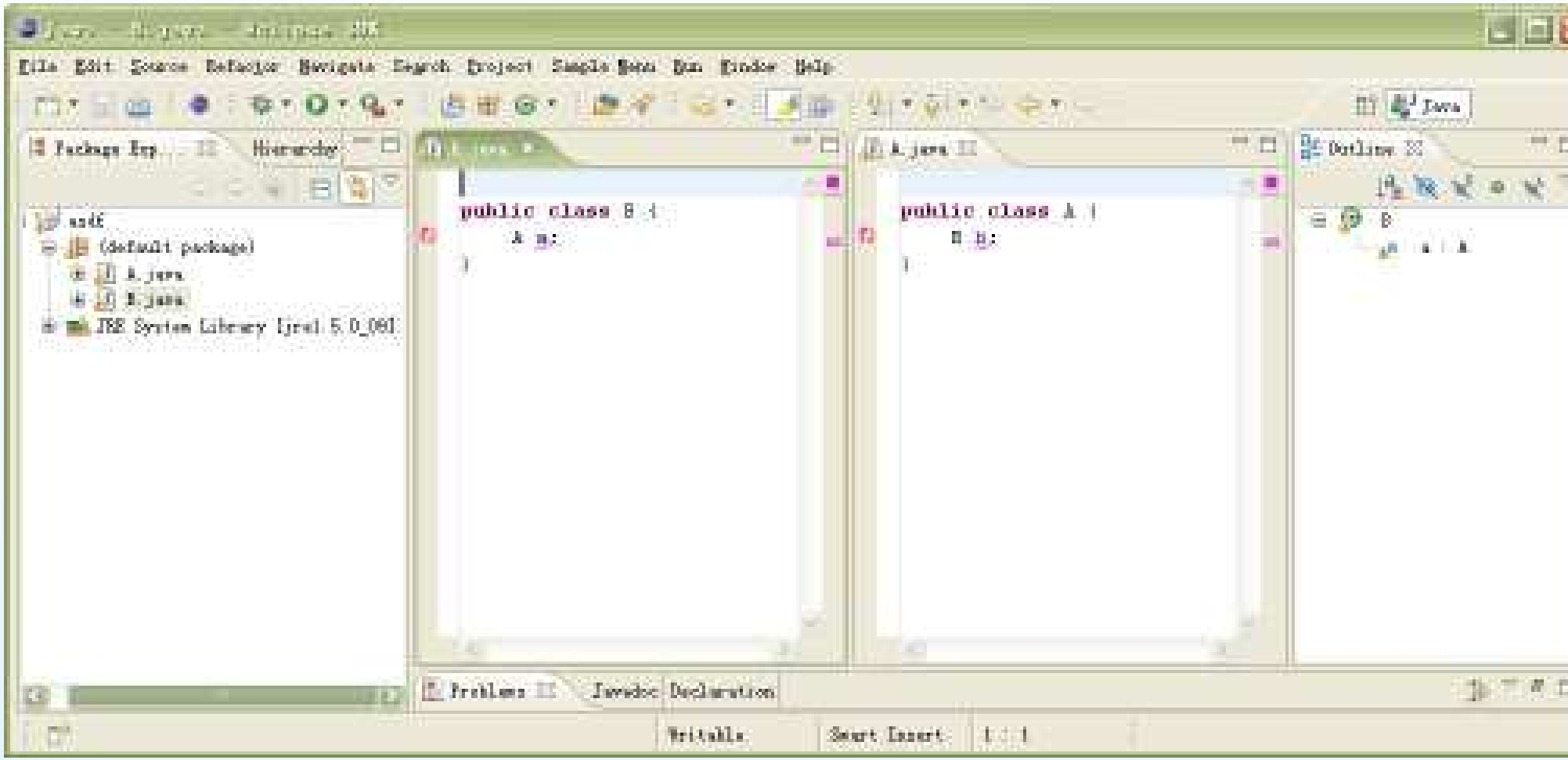
- If cycles are bad, let's not have any

$\Rightarrow$ tool support for detecting cycles

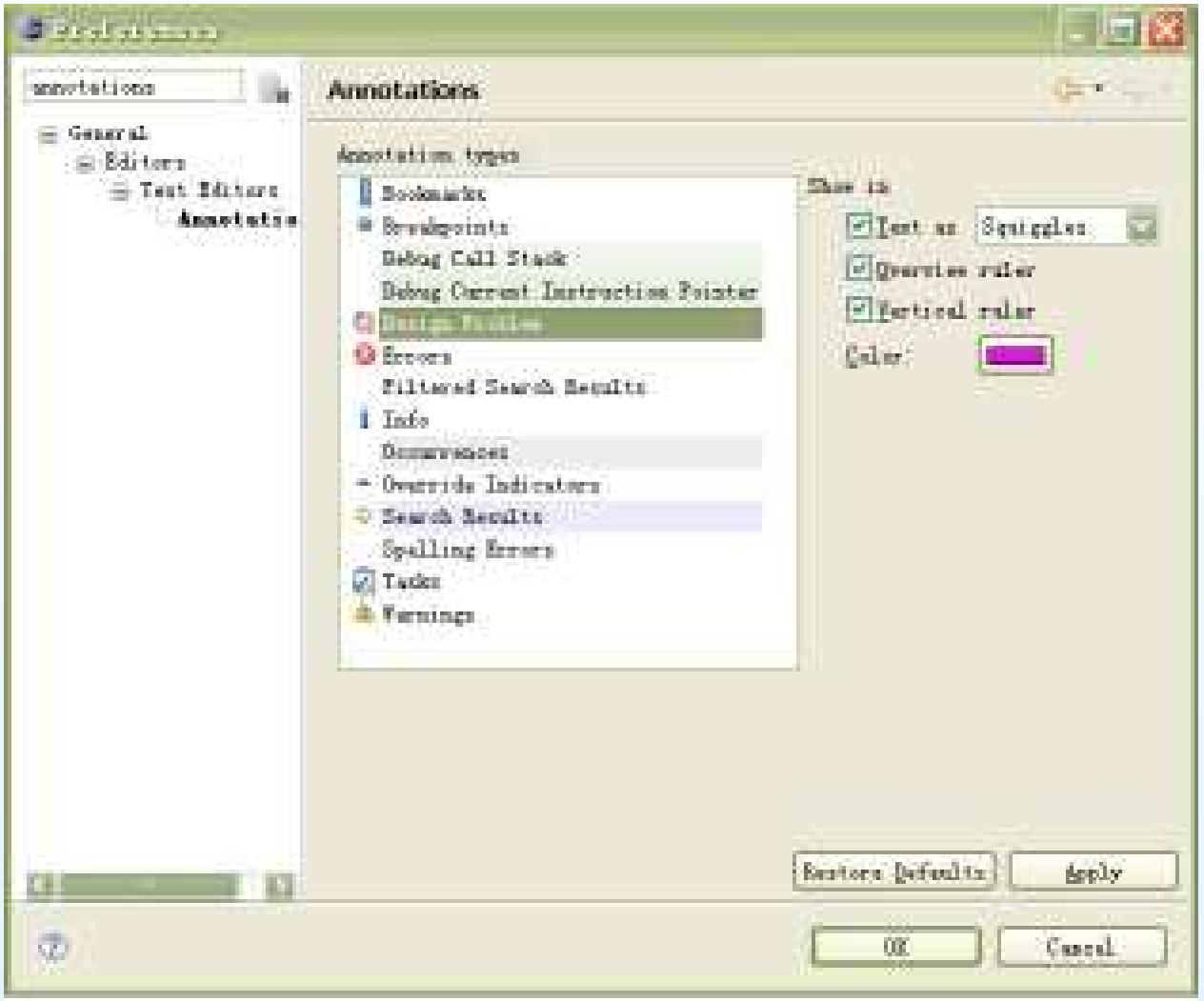
$\Rightarrow$ should be part of development environment

$\Rightarrow$ JooJ

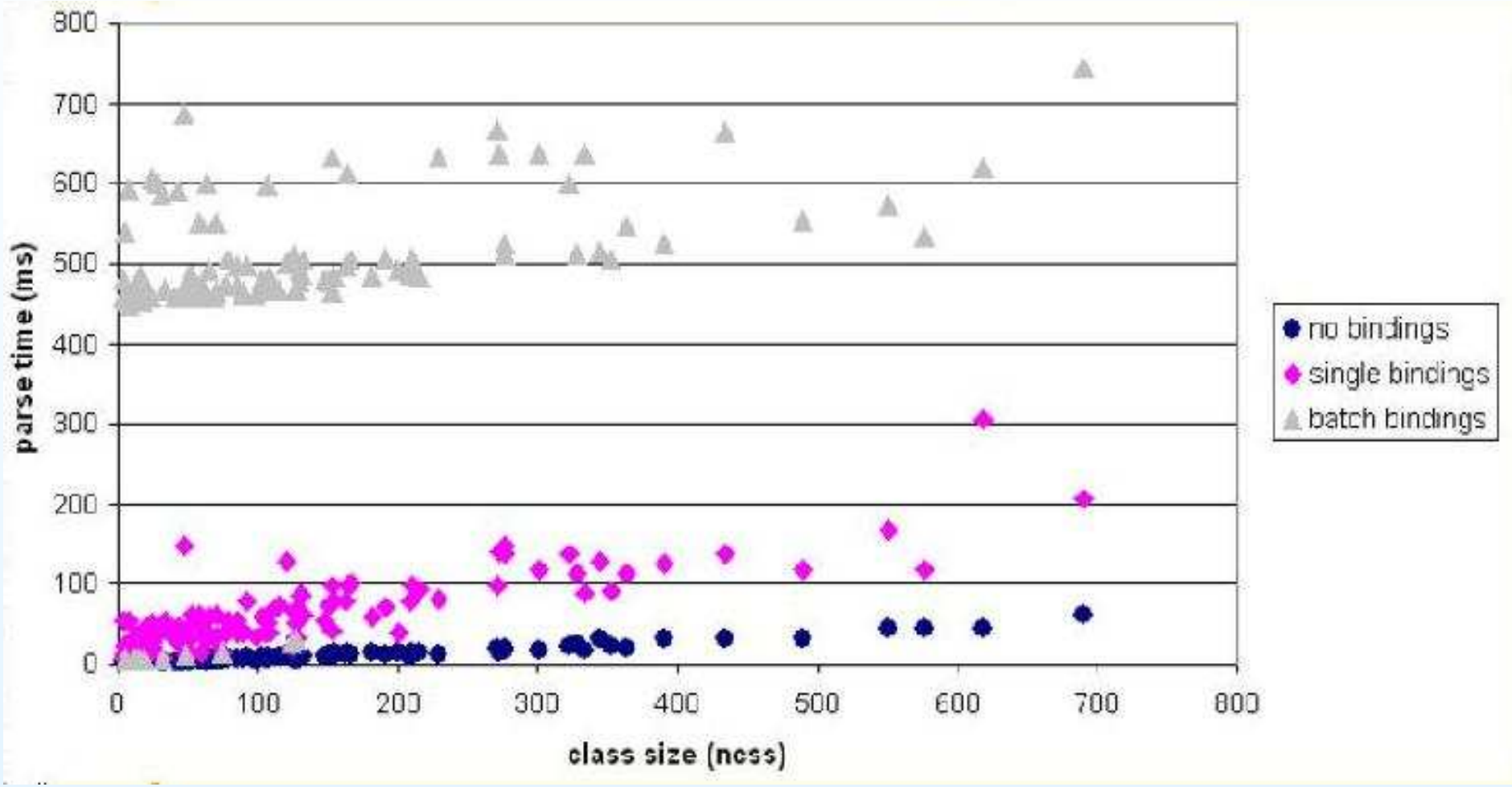
JooJ


File Edit Source Refactor Navigate Search Project Sample Menu Run Window Help
Package Exp
Hierarchy
(default package)
A.java
B.java
JRE System Library [jre1.5.0_06]
public class B {
A a;
}
A.java
public class A {
B b;
}
Outline
Problems
Javadoc
Declaration
Writable
Smart Insert

JooJ


Annotations
General
Editors
Text Editors
Annotations
Annotation types
Bookmarks
Breakpoints
Debug Call Stack
Debug Current Instruction Pointer
Errors
Filtered Search Results
Info
Override Indicators
Search Results
Spelling Errors
Tasks
Warnings
Text as
Squiggles
Vertical ruler
Color
Restore Defaults
Apply
OK
Cancel

How Fast?


800
700
600
500
400
300
200
100
0
parse time (ms)
0
100
200
300
400
500
600
700
800
class size (ncss)
no bindings
single bindings
batch bindings

Tools for Producing Quality Software

- Measurement instruments
  $\Rightarrow$ development support
- Corpus Management
- Data analysis and presentation