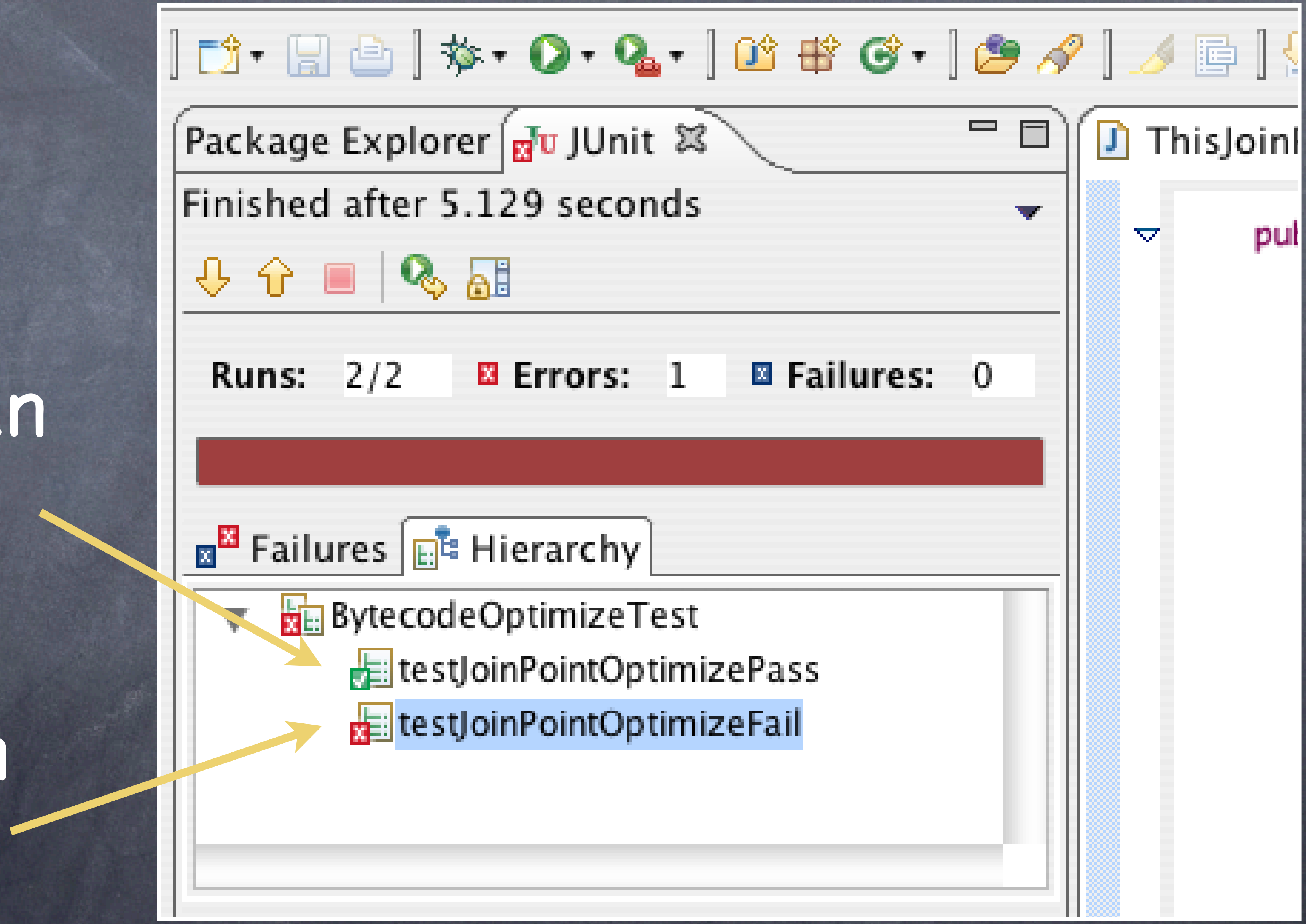
Evaluating a Lightweight Defect Localization Tool

Valentin Dallmeier Christian Lindig Andreas Zeller
Saarland University - Germany

JUnit Tests in Eclipse

Passing Run
(1+)

Failing Run
(1)



Finished after 5.129 seconds

Runs: 2/2 Errors: 1 Failures: 0

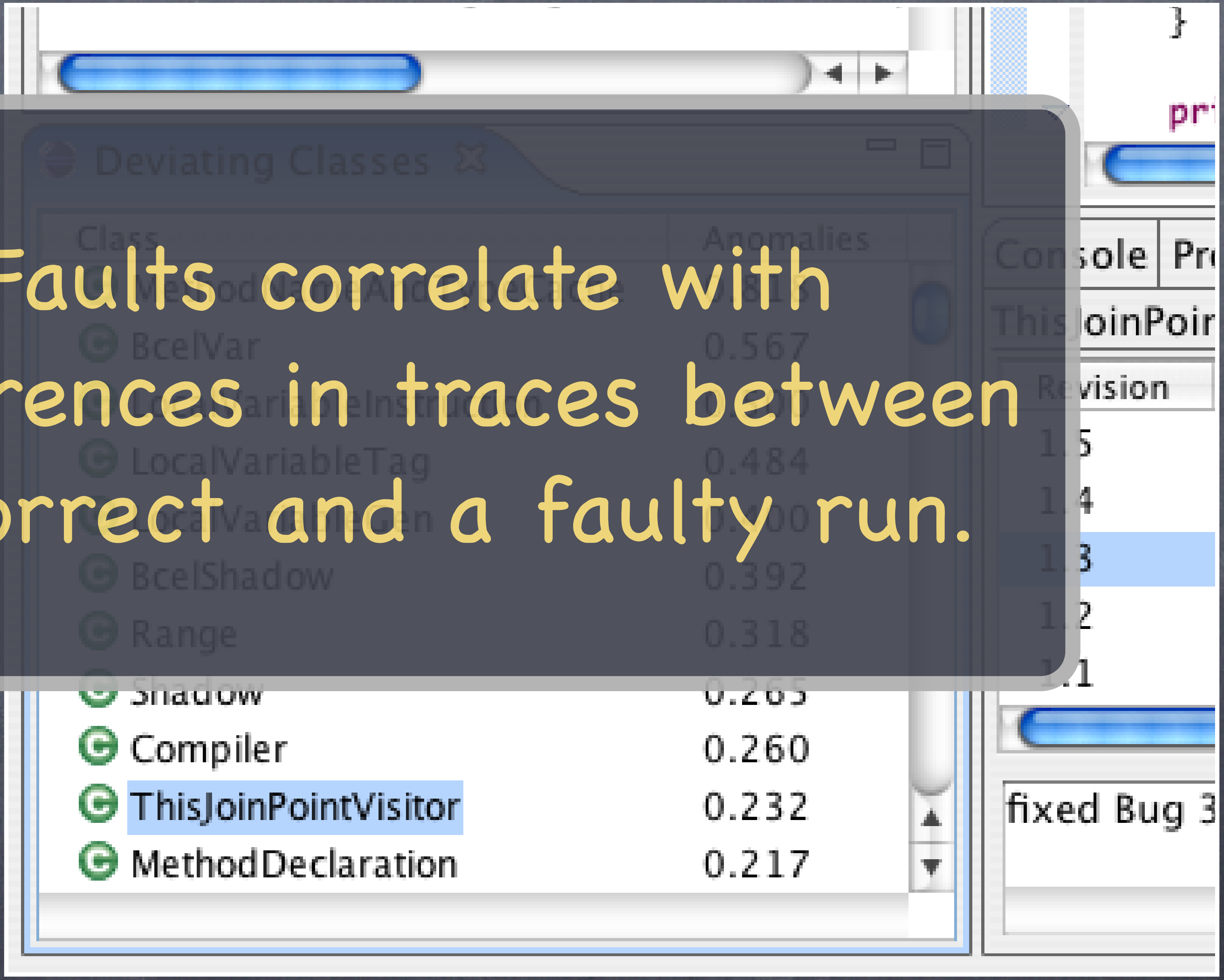
- BytecodeOptimizeTest
  - testJoinPointOptimizePass
  - testJoinPointOptimizeFail

Ample Plugin

AspectJ Bug #30168

Suspect Classes

Bug fixed here



Deviating Classes

Class	Anomalies
MethodNameAndTypeCache	[illegible]
BcelVar	0.567
LocalVariableInstruction	[illegible]
LocalVariableTag	0.484
LocalVariableGen	[illegible]
BcelShadow	0.392
Range	0.318
Shadow	0.265
Compiler	0.260
ThisJoinPointVisitor	0.232
MethodDeclaration	0.217

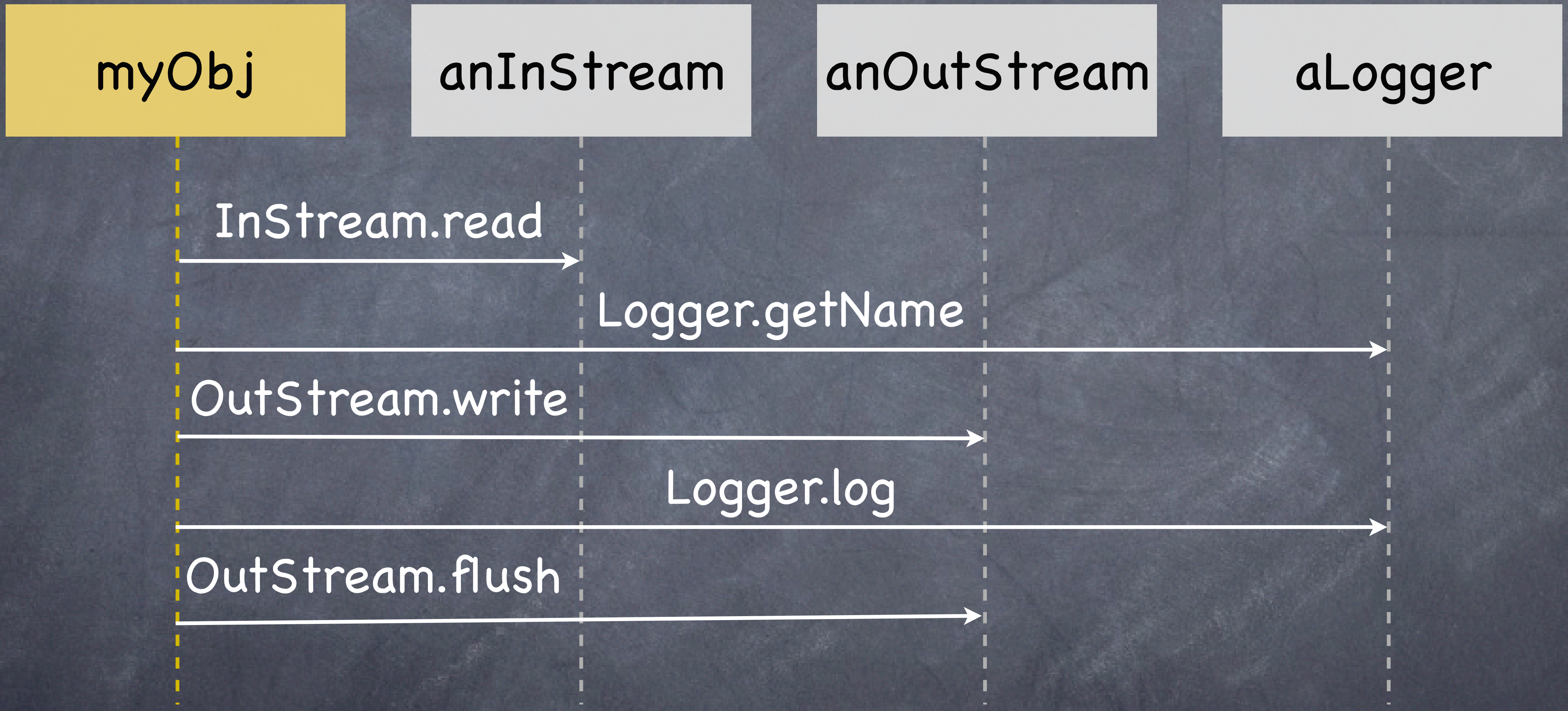
Revision
1.5
1.4
1.3
1.2
1.1

fixed Bug 3

Faults correlate with differences in traces between a correct and a faulty run.

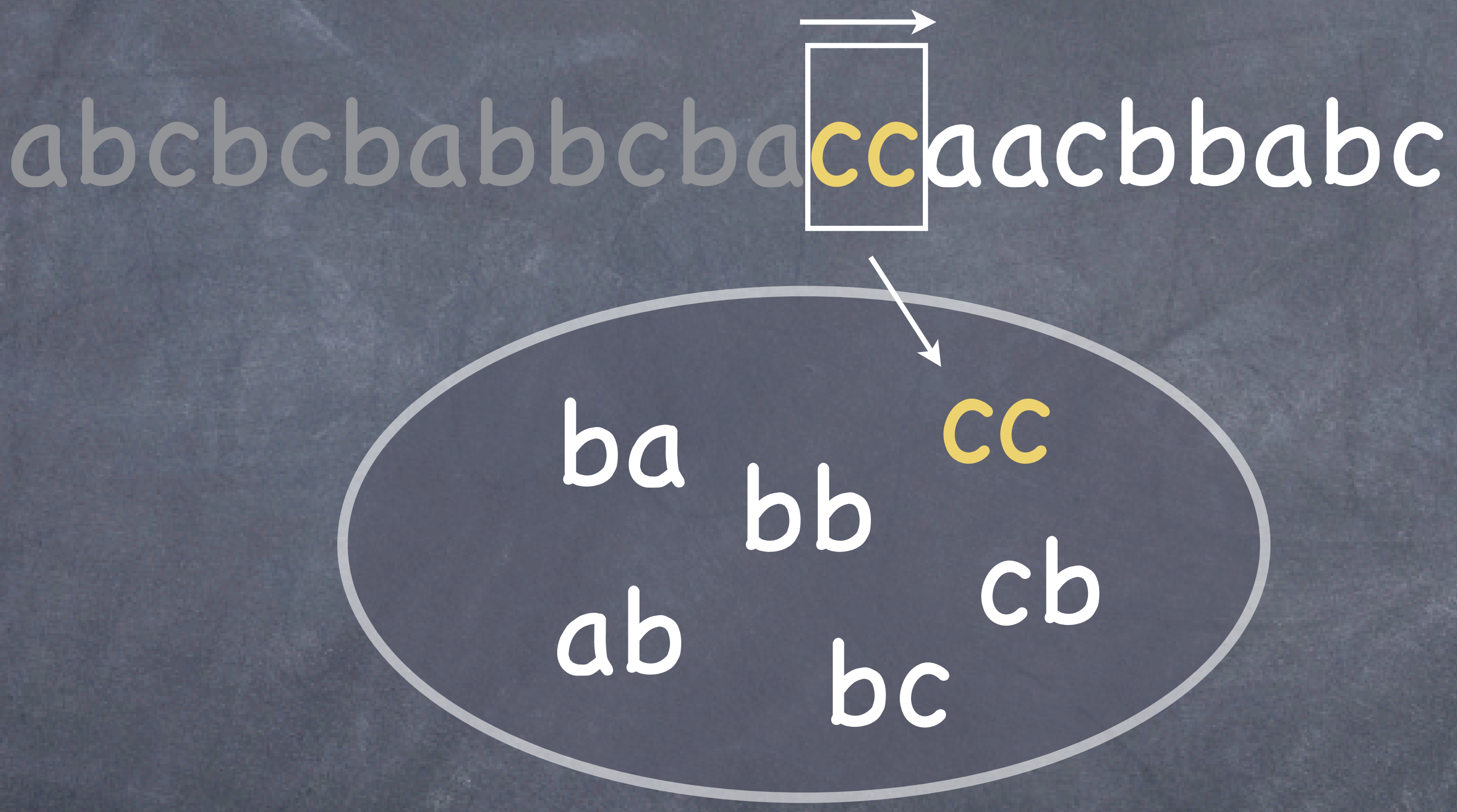
2,929 classes

Tracing Objects



InStream.read Logger.getName OutStream.write Logger.log OutStream.flush

Call-Sequence Sets



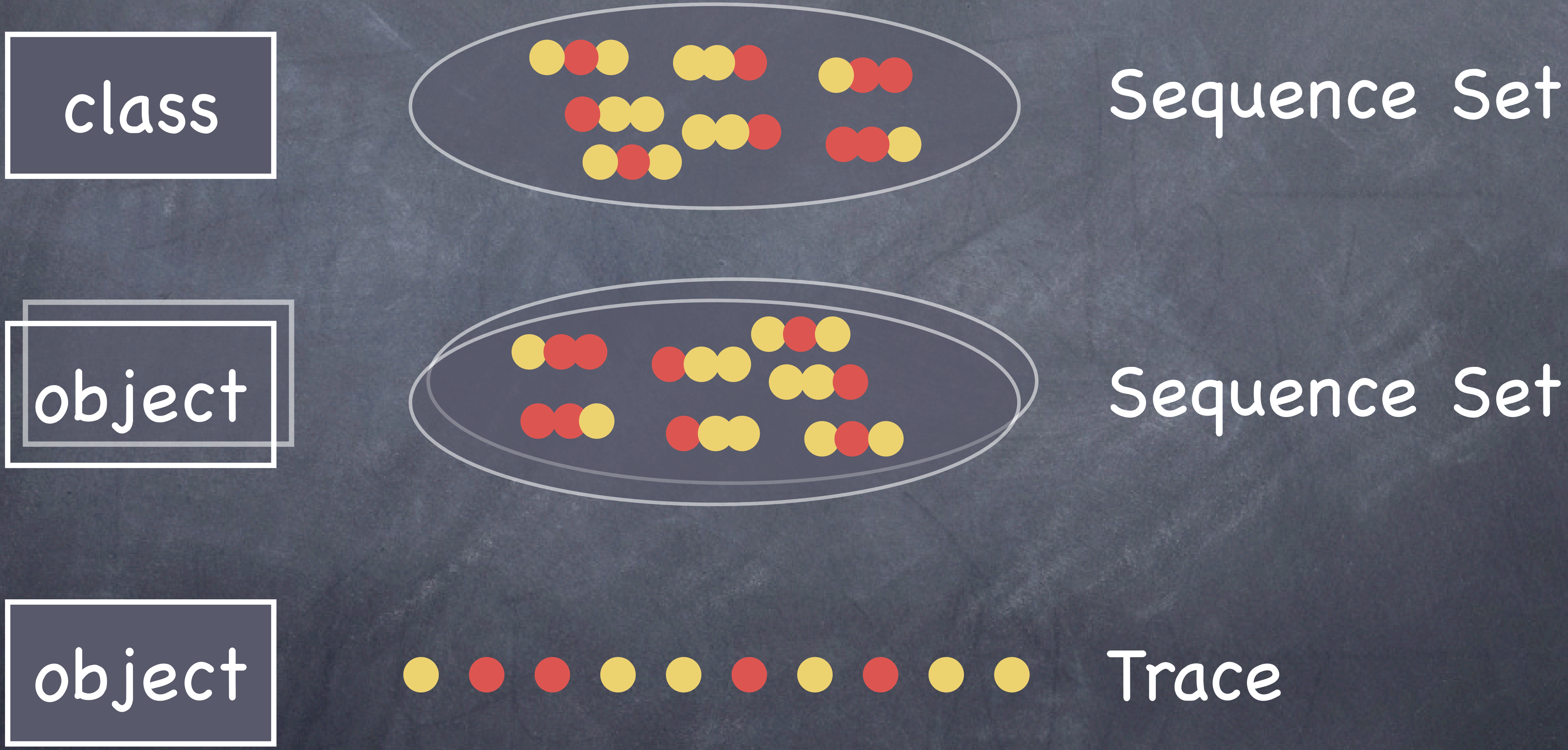
abcbcbabbbcbaccaacbbabc

ba bb cc cb ab bc

Call-Sequence Set - sequences of length k

Benefits: simple, compact, set semantics

Aggregating Traces



class

Sequence Set

object

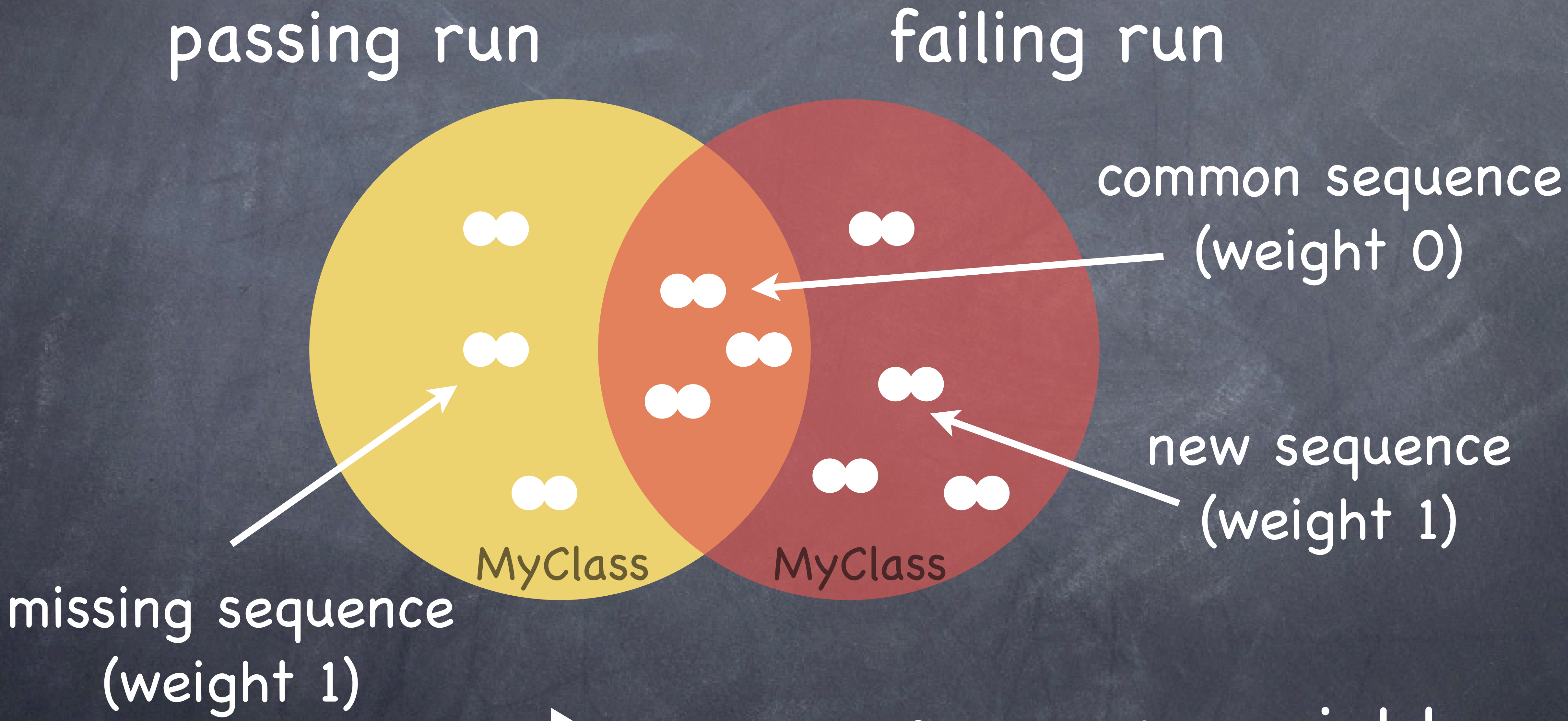
Sequence Set

object

Trace

Comparing Program Runs

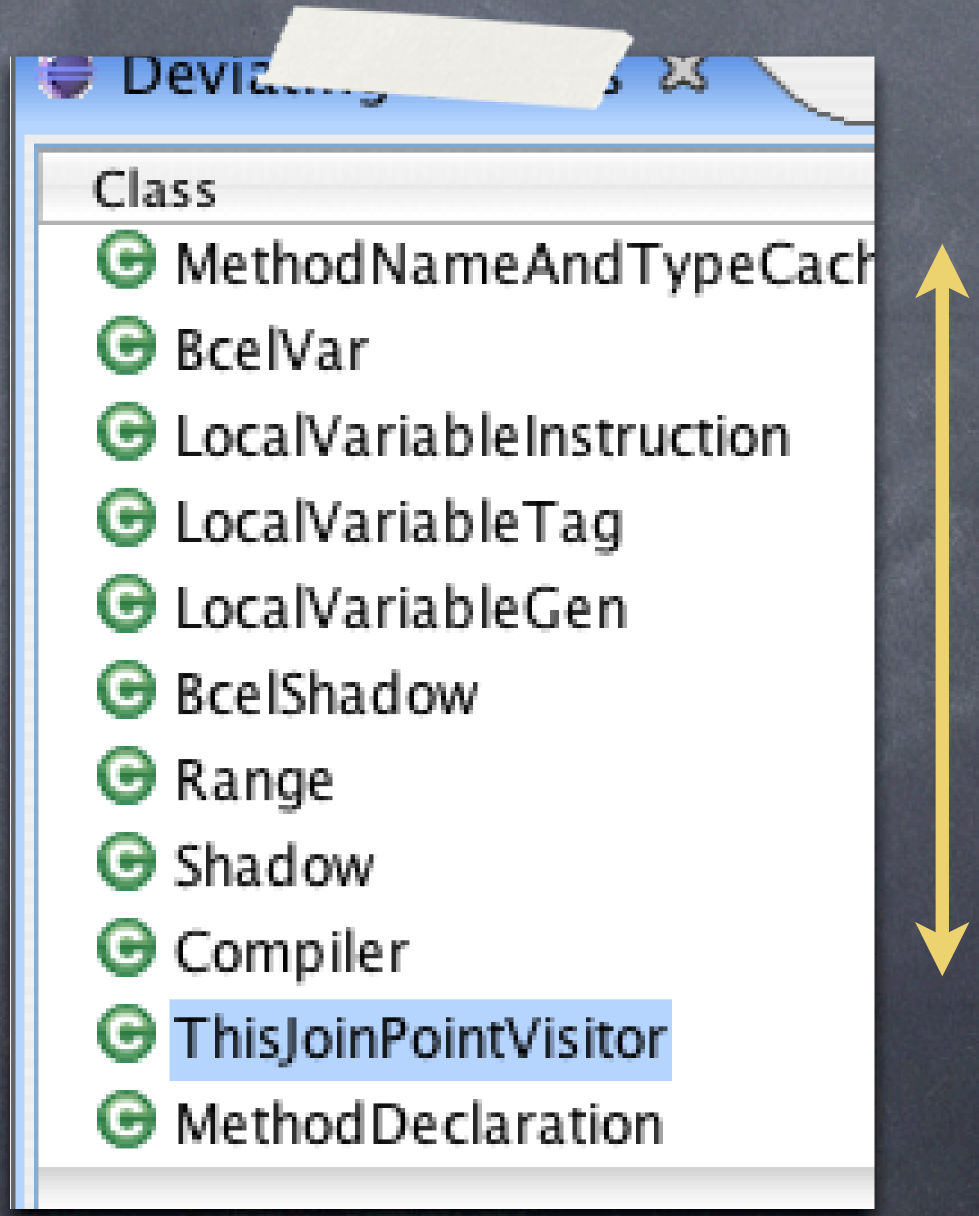
class-by-class



→ average sequence weight for ranking classes

Search Length

- search length: classes in front of faulty class in ranking
- smaller is better
- evaluated for programs with one known bug



Class

MethodNameAndTypeCach
BcelVar
LocalVariableInstruction
LocalVariableTag
LocalVariableGen
BcelShadow
Range
Shadow
Compiler
ThisJoinPointVisitor
MethodDeclaration

search length: 9

Evaluation Subjects

- NanoXML - Java XML Parser (Do et al.)
  - 4 Versions, 16–23 classes, 4.3-7.6 kLOC
  - 33 known bugs, 214 test cases
  - 386 rankings, each for:
    1 bug, 1 failing run, 1+ passing runs
- AspectJ - Java Compiler (v1.1.1)
  - 979 classes, 112 kLOC
  - 5 rankings for real bugs from bug db

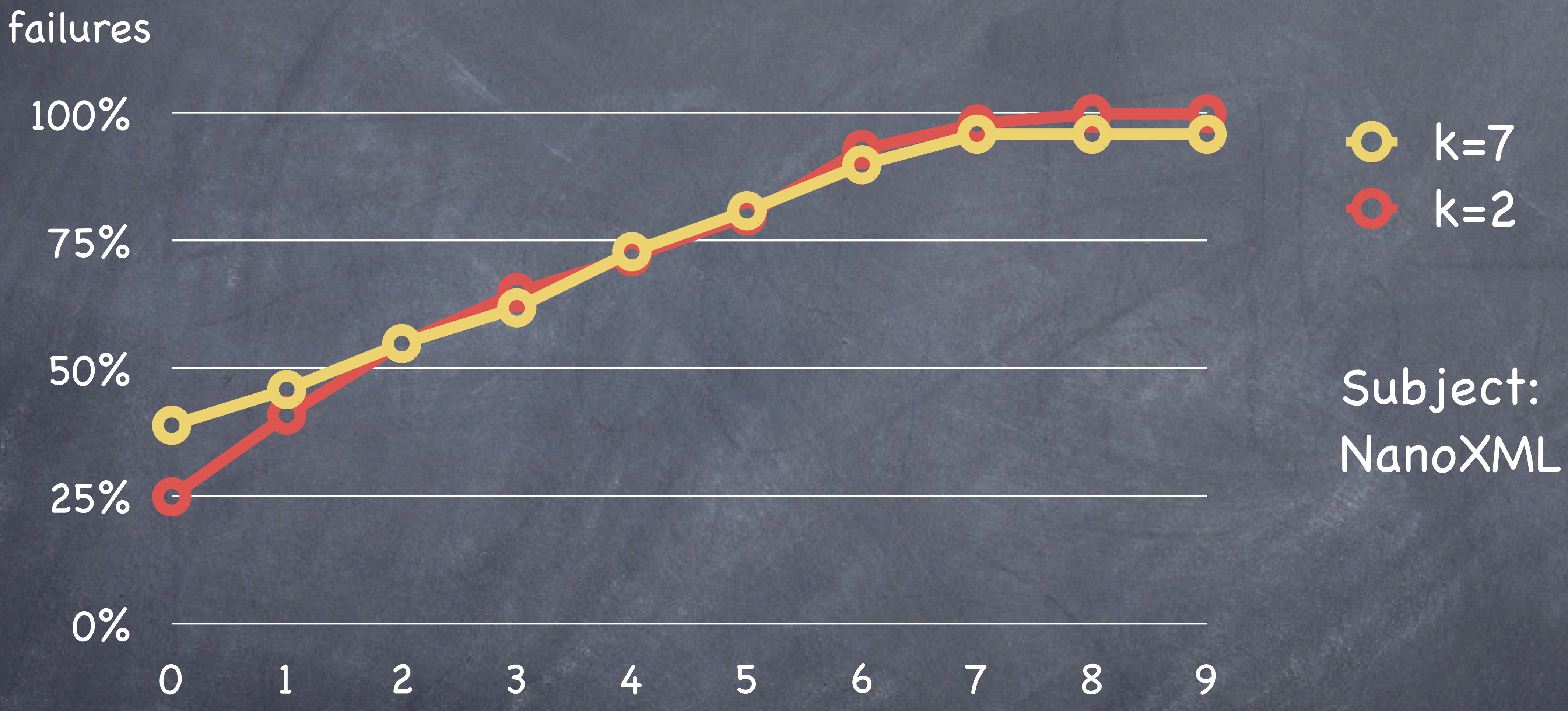
Results

Subject	Rand Guess	search length window size 1	2	4	5	8	10
NanoXML	4.78	2.53	2.31	2.17	2.04	2.12	2.14
AspectJ	209	32.4	31.8	10.2	8.6	23.8	24.0

Search beats random guessing (no surprise)

Sequences perform better than coverage

Search Length



Subject:
NanoXML

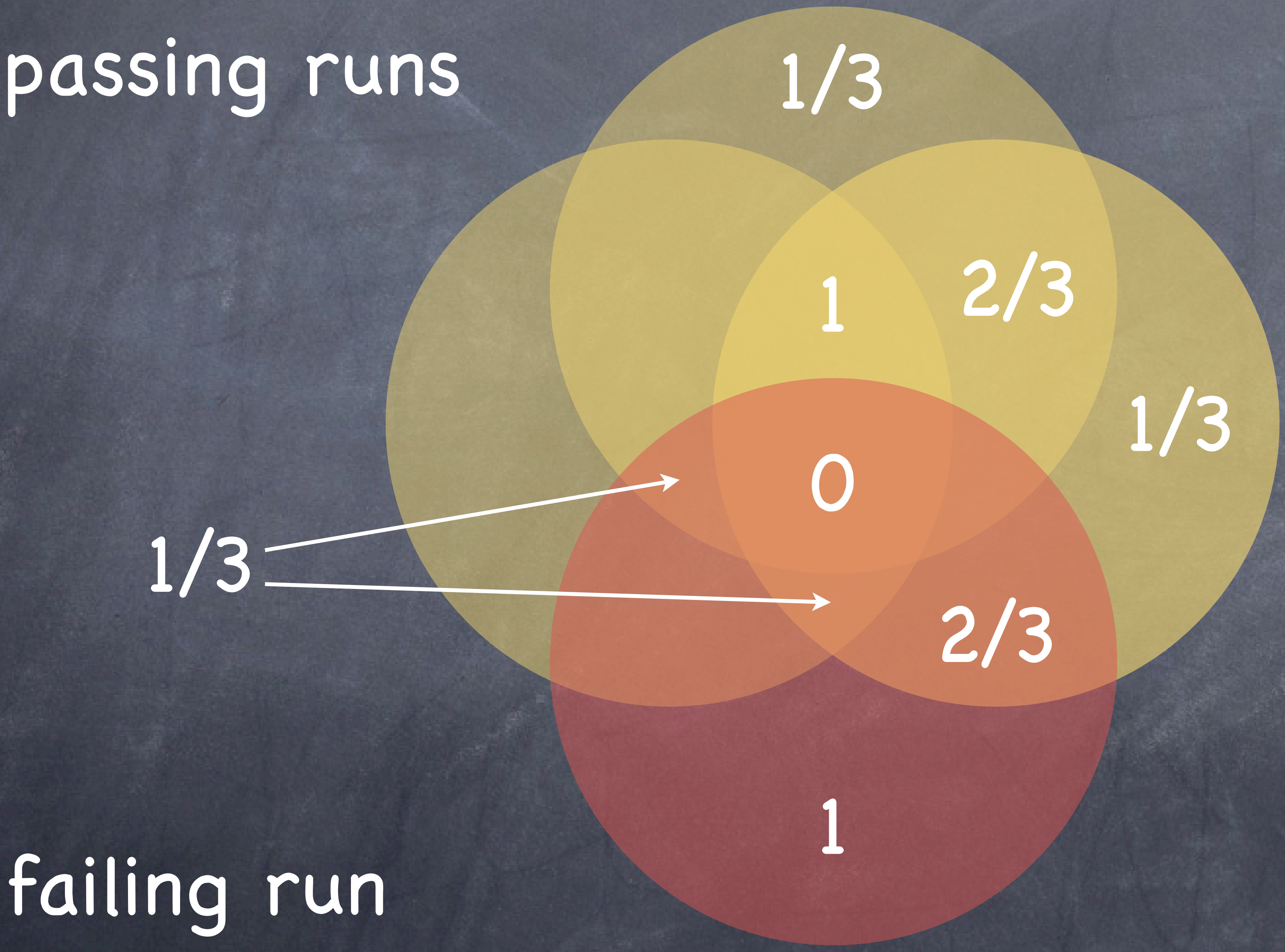
Inspecting the 3 top-ranked classes, a programmer finds over 50% of all bugs in NanoXML.

Conclusions

- Ample works (NanoXML) and scales (AspectJ)
- Sequence sets facilitate aggregation and comparison of runs
- Ample is first approach to leverage objects
- Search length is measure for performance
- Sequences outperform coverage analysis

Dallmeier, Lindig, Zeller: Lightweight Defect Localization for Java, ECOOP 2005.

1 Failing, 3 Passing Runs



Runtime Overhead

- Measured for SPEC JVM 98 Benchmarks
- Memory: factor 1.1 – 22.7 (typical: ≤ 2)
- Time: factor 1.2 – ≥ 100 (varies widely)
- comparable to coverage analysis (JCoverage)
- found low overhead for AspectJ