

The Future of Code Coverage for Eclipse

Marc R. Hoffmann
Eclipse Summit 2010
2010-11-03, Ludwigsburg





Outline

- Code Coverage
- EclEmma
- EMMA

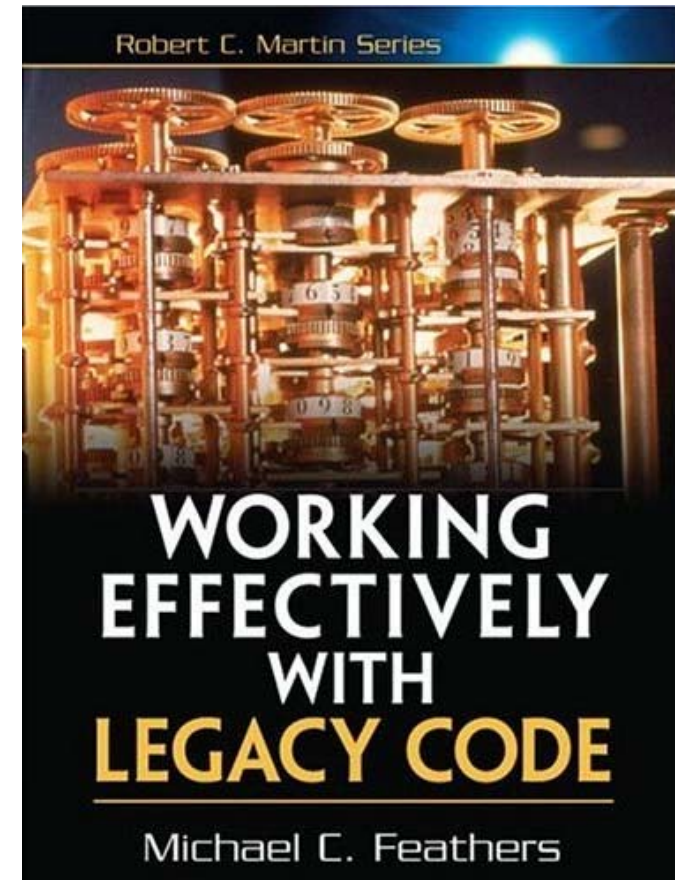
■ JaCoCo



Code Coverage

**„Legacy Code is simply
code without tests.“**

Michael Feathers:
Working Effectively with Legacy Code





The Future of Code Coverage for Eclipse

Example: JUnit with Code Coverage

The screenshot displays the Eclipse IDE interface. On the left, the 'FormulasTest.java' editor shows a JUnit test class with a single test method `testClip1()` that calls `Formulas.clip(1, 9, 0)` and asserts the result is 1. Below the editor, the JUnit test runner window is open, showing 'Finished after 0,016 seconds' and 'Runs: 7/7', 'Errors: 0', 'Failures: 0'. A tree view lists the test methods: `testClip1`, `testClip2`, `testClip3`, `testClip4`, `testMax1`, `testMax2`, and `testMax3`, all with a duration of 0,000 s. On the right, the 'Formulas.java' editor shows two methods: `clip` and `max`. The `clip` method has two conditional branches highlighted in green, indicating they were executed during the test. The `max` method has a conditional branch highlighted in red, indicating it was not executed.

```
@Test
public void testClip1() {
    int x = Formulas.clip(1, 9, 0);
    assertEquals(1, x);
}

@Test
public void testClip2() {
    // ...
}

@Test
public void testClip3() {
    // ...
}

@Test
public void testClip4() {
    // ...
}

@Test
public void testMax1() {
    // ...
}

@Test
public void testMax2() {
    // ...
}

@Test
public void testMax3() {
    // ...
}
```

```
public static int clip(int lower, int upper, int x) {
    if (x < lower) {
        x = lower;
    }
    if (x > upper) {
        x = upper;
    }
    return x;
}

public static int max(int a, int b, int c) {
    if (a < b) {
        if (b < c) {
            return c;
        } else {
            return b;
        }
    } else {
        // ...
    }
}
```



The Future of Code Coverage for Eclipse

EclEmma – Code Coverage for Eclipse

Java - CursorableLinkedList.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

JUnit Finished after 34,898 seconds

Runs: 13009/13009 Errors: 0 Failures: 0

Failures Hierarchy

- JUnit framework.TestSuite
 - TestBagUtils
 - org.apache.commons.collections.TestCloseable
 - org.apache.commons.collections.TestCollectionUtils
 - TestBufferUtils
 - TestEnumerationUtils
 - org.apache.commons.collections.TestFactories
 - TestListUtils
 - TestMapUtils
 - org.apache.commons.collections.TestPreconditions
 - TestSetUtils
 - org.apache.commons.collections.TestTraverse
 - TestArrayStack
 - TestBeanMap
 - org.apache.commons.collections.TestBinaryHeap
 - TestBoundedFifoBuffer
 - TestBoundedFifoBuffer2
 - TestCursorableLinkedList
 - TestDoubleOrderedMap
 - org.apache.commons.collections.TestExtensible
 - TestFastArrayList
 - TestFastArrayList1
 - TestFastHashMap
 - TestFastHashMap1
 - TestFastTreeMap
 - TestFastTreeMap1

```
public boolean addAll(int index, Collection c) {  
    if (c.isEmpty()) {  
        return false;  
    } else if (size == index || size == 0) {  
        return addAll(c);  
    } else {  
        Listable succ = getListableAt(index);  
        Listable pred = (null == succ) ? null : succ.prev();  
        Iterator it = c.iterator();  
        while (it.hasNext()) {  
            pred = insertListable(pred, succ, it.next());  
        }  
        return true;  
    }  
}
```

Problems Javadoc Declaration Console Coverage

TestAllPackages (31.10.2006 15:04:14)

Element	Coverage	Covered Lines	Total Lines
java - commons-collections	79,5 %	10927	13738
org.apache.commons.collections	74,1 %	3842	5183
ArrayStack.java	86,5 %	32	37
BagUtils.java	86,7 %	13	15
BeanMap.java	72,4 %	155	214
BinaryHeap.java	87,6 %	127	145
BoundedFifoBuffer.java	93,2 %	82	88
BufferOverflowException.java	55,6 %	5	9
BufferUnderflowException.java	88,9 %	8	9
BufferUtils.java	30,8 %	4	13
ClosureUtils.java	93,9 %	31	33
CollectionUtils.java	92,4 %	293	317
ComparatorUtils.java	8,6 %	3	35
CursorableLinkedList.java	85,4 %	444	520

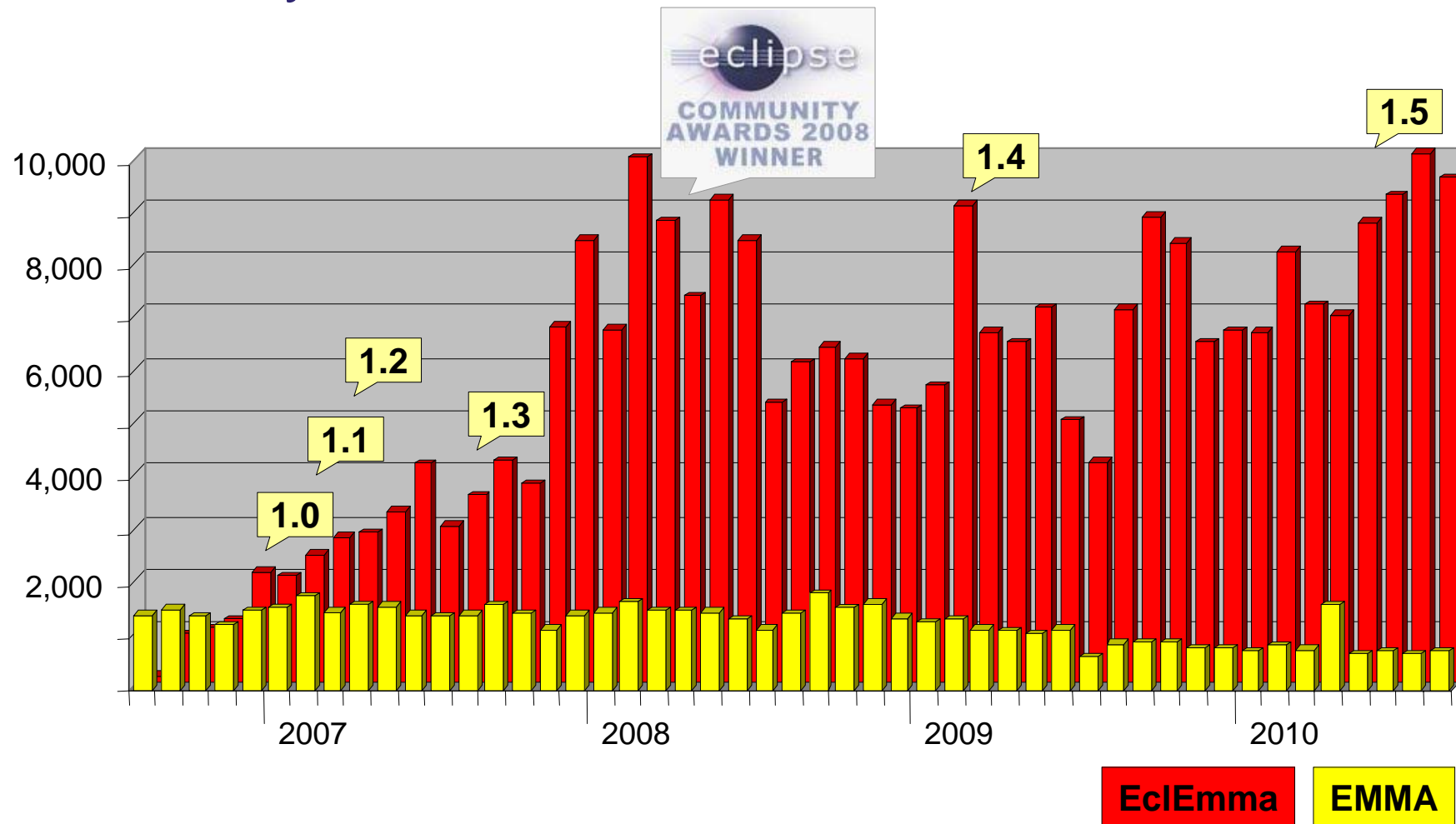
Failure Trace

Writable Smart Insert 149 : 28



The Future of Code Coverage for Eclipse

Monthly EclEmma Downloads













The Future of Code Coverage for Eclipse

Current Status of EMMA



- Great Tool! 😊
- Last Release 😞
 - ➔ 2.0.5312, 2005-06-13
- Project Activity 😞
 - ➔ Latest Commit 2006-02-23

Element	Instruction Coverage		Missed Lines
 com.vladium.emma		0%	5.412 / 5.412
 com.vladium.util		0%	2.462 / 2.462
 com.vladium.jdc		0%	1.531 / 1.531
 com.vladium.logging		0%	162 / 162
Total		0%	9.567 / 9.567



Action Required!

- Making EMMA alive?
- Using a different Library?
- Starting a new Project?





Requirements for a Code Coverage Library

- Be a Library!
 - Open for Different Usage Scenarios
 - Designed for Integration
- Regression Tests
- Framework Independent
- Scalable for Large Projects
- Fast Enough for Agile Teams



The „JaCoCo“ Project

- **Java Code Coverage**
- Started Mid of 2009
- Beta Releases Since End of 2009
- EPL
- Hosted within Eclemma (SourceForge)
 - www.eclemma.org/jacoco
- Active Team
 - Marc R. Hoffmann (GER)
 - Brock Janiczak (AUS)



The Future of Code Coverage for Eclipse

JaCoCo Demo

Demo 1

```
T:\demo\eclipse>eclipse -vmargs -javaagent:jacocoagent.jar
T:\demo\eclipse>cd ..\report
T:\demo\report>ant
Buildfile: T:\demo\report\build.xml
report:
BUILD SUCCESSFUL
Total time: 28 seconds
T:\demo\report>
```

org.eclipse.rcp_3.6.0 > org.eclipse.osgi > org.eclipse.osgi.util

org.eclipse.osgi.util

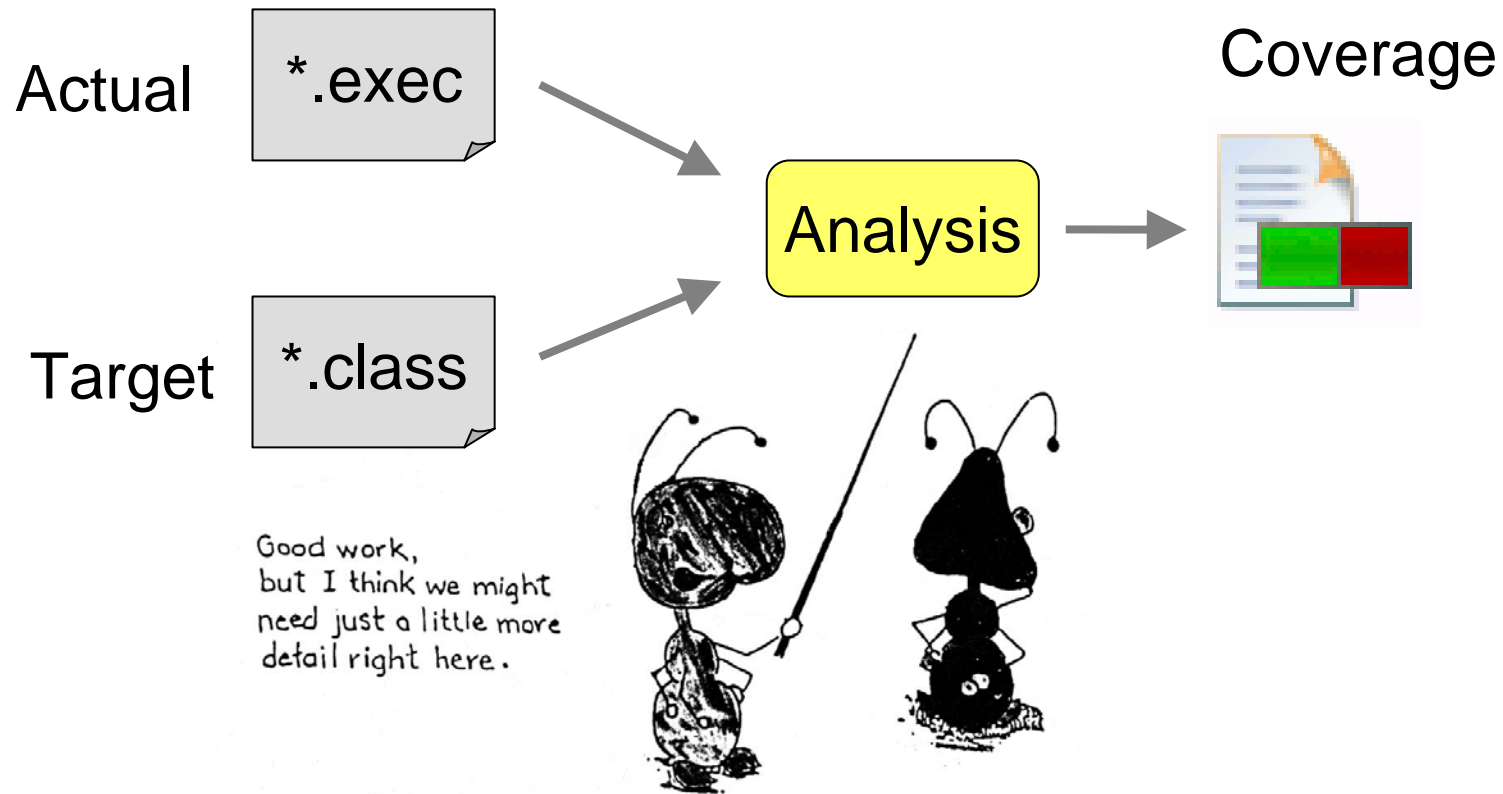
Element	Instruction Coverage	Missed Classes	Missed Methods	Missed Blocks	Missed Lines
ManifestElement	<div><div></div></div> 48%	0 / 1	9 / 21	73 / 153	83 / 189
TextProcessor	<div><div></div></div> 16%	0 / 1	3 / 8	58 / 71	49 / 61
NLS	<div><div></div></div> 63%	0 / 1	2 / 11	47 / 125	55 / 145
NLS.MessagesProperties	<div><div></div></div> 60%	0 / 1	0 / 2	6 / 15	9 / 23
NLS.new PrivilegedAction() {...}	<div><div></div></div> 0%	1 / 1	2 / 2	2 / 2	4 / 4
Total	48%	1 / 5	16 / 44	186 / 366	199 / 421

Created with JaCoCo 0.3.1.20100311234431



The Future of Code Coverage for Eclipse

Coverage Analysis





JaCoCo Ant Tasks: Coverage

```
1 <jacoco:coverage>
2   <java classname="org.jacoco.examples.HelloJaCoCo" fork="true">
3     <classpath>
4       <pathelement location="./bin"/>
5     </classpath>
6   </java>
7 </jacoco:coverage>
8
9
10 <jacoco:coverage>
11   <junit fork="true" forkmode="once">
12     <test name="org.jacoco.examples.HelloJaCoCoTest"/>
13     <classpath>
14       <pathelement location="./bin"/>
15     </classpath>
16   </junit>
17 </jacoco:coverage>
```



JaCoCo Ant Tasks: Report

```
1 <jacoco:report>
2
3   <executiondata>
4     <file file="jacoco.exec"/>
5   </executiondata>
6
7   <structure name="Example Project">
8     <classfiles>
9       <fileset dir="bin"/>
10    </classfiles>
11    <sourcefiles encoding="UTF-8">
12      <fileset dir="src"/>
13    </sourcefiles>
14  </structure>
15
16  <html destdir="report"/>
17
18 </jacoco:report>
```



JaCoCo Ant Tasks: Report

```
1 <structure name="JaCoCo">
2   <group name="org.jacoco.core">
3     <classfiles>
4       <path refid="bundle-org.jacoco.core"/>
5     </classfiles>
6     <sourcefiles>
7       <fileset dir="${workspace.dir}/org.jacoco.core/src"/>
8     </sourcefiles>
9   </group>
10  <group name="org.jacoco.report">
11    <classfiles>
12      <path refid="bundle-org.jacoco.report"/>
13    </classfiles>
14    <sourcefiles>
15      <fileset dir="${workspace.dir}/org.jacoco.report/src"/>
16    </sourcefiles>
17  </group>
18
19  ...
20
21 </structure>
```



Report Sorting

Project A 80%



Project B 35%

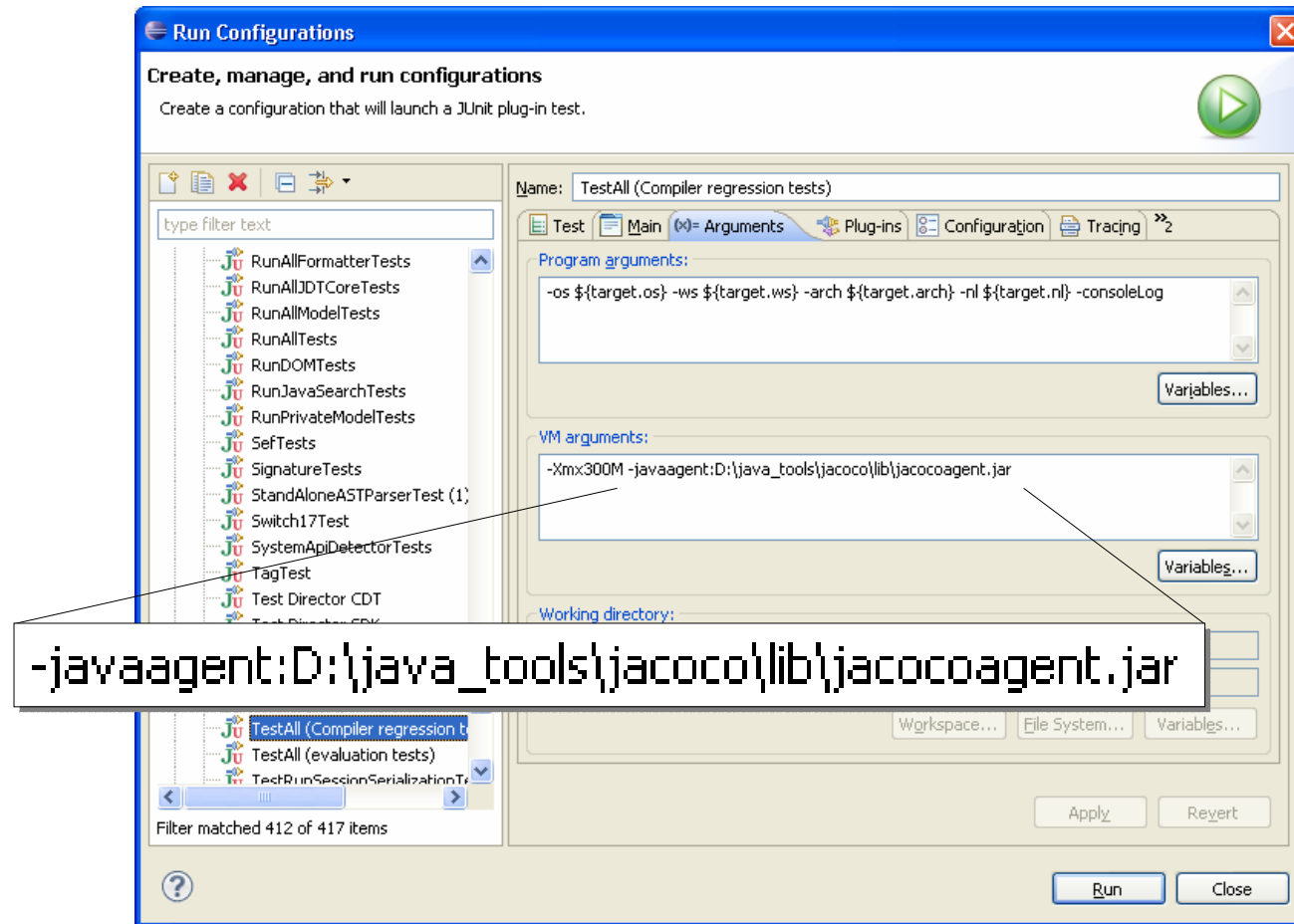


Sort Items by absolut amount of missed code.



The Future of Code Coverage for Eclipse

JaCoCo in Eclipse





























The Future of Code Coverage for Eclipse

JaCoCo on JDT Test Suite

JDT Core tests > org.eclipse.jdt.core

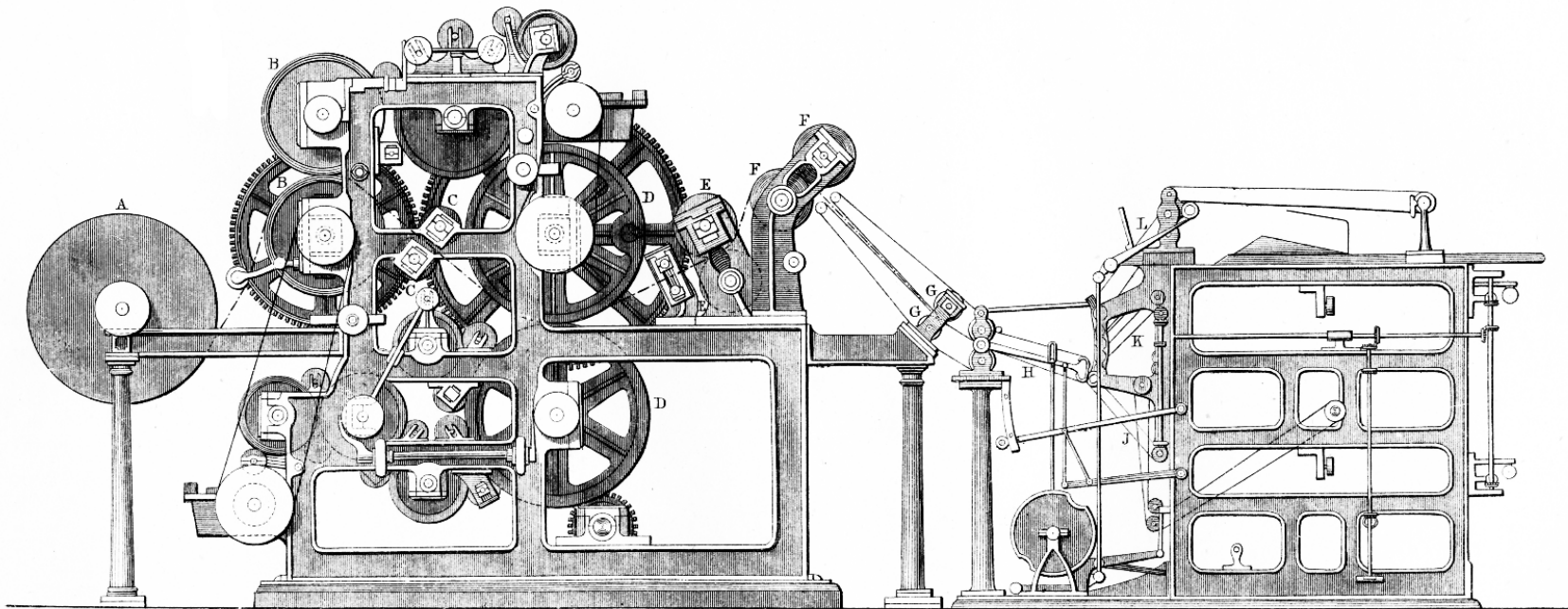
org.eclipse.jdt.core

Element	Instruction Coverage	Missed Classes	Missed Methods	Missed Blocks	Missed Lines
org.eclipse.jdt.internal.core	 81 %	15 / 237	404 / 2 656	3 436 / 16 103	3 943 / 21 297
org.eclipse.jdt.internal.core.util	 64 %	2 / 73	343 / 1 220	3 042 / 7 348	4 133 / 11 272
org.eclipse.jdt.core.dom	 83 %	0 / 146	212 / 2 898	3 396 / 14 734	1 935 / 17 196
org.eclipse.jdt.internal.compiler	 71 %	4 / 29	109 / 575	1 418 / 4 370	2 531 / 8 540
org.eclipse.jdt.internal.compiler.lookup	 86 %	1 / 60	103 / 1 053	1 767 / 11 792	1 579 / 12 967
org.eclipse.jdt.internal.compiler.parser	 86 %	0 / 25	86 / 815	1 284 / 7 949	1 610 / 12 529
org.eclipse.jdt.internal.codeassist	 82 %	1 / 33	74 / 524	1 482 / 6 464	1 927 / 11 207
org.eclipse.jdt.internal.compiler.ast	 89 %	1 / 112	120 / 1 108	1 382 / 11 418	1 604 / 14 752
org.eclipse.jdt.internal.formatter	 86 %	1 / 20	58 / 429	1 091 / 6 715	1 255 / 9 107
org.eclipse.jdt.internal.core.search.matching	 84 %	1 / 58	66 / 682	1 166 / 6 635	963 / 7 026
org.eclipse.jdt.internal.compiler.problem	 81 %	0 / 9	74 / 490	397 / 1 950	1 142 / 5 924
org.eclipse.jdt.internal.compiler.codegen	 83 %	0 / 20	55 / 519	787 / 3 355	1 079 / 6 076
org.eclipse.jdt.internal.core.builder	 72 %	3 / 28	50 / 278	717 / 2 583	771 / 3 163
org.eclipse.jdt.internal.codeassist.complete	 82 %	0 / 44	84 / 539	731 / 3 517	872 / 5 149
org.eclipse.jdt.internal.compiler.batch	 68 %	1 / 17	43 / 180	618 / 2 056	920 / 3 097
org.eclipse.jdt.internal.eval	 74 %	2 / 35	69 / 294	768 / 2 039	905 / 3 009
org.eclipse.jdt.internal.compiler.flow	 78 %	1 / 13	41 / 185	409 / 1 712	529 / 2 351
org.eclipse.jdt.internal.core.dom.rewrite	 85 %	0 / 39	60 / 518	443 / 2 703	663 / 3 960
org.eclipse.jdt.internal.core.search	 73 %	2 / 24	39 / 170	536 / 1 807	517 / 2 172
org.eclipse.jdt.internal.compiler.util	 71 %	1 / 23	70 / 220	425 / 1 408	490 / 1 822
org.eclipse.jdt.core	 76 %	2 / 26	142 / 418	569 / 1 941	703 / 2 676
org.eclipse.jdt.internal.compiler.impl	 86 %	0 / 13	47 / 168	683 / 2 702	284 / 2 097
org.eclipse.jdt.internal.compiler.classfmt	 79 %	1 / 13	35 / 171	318 / 1 205	316 / 1 533
org.eclipse.jdt.internal.core.hierarchy	 84 %	0 / 17	18 / 197	332 / 1 638	388 / 2 157
org.eclipse.jdt.internal.core.search.indexing	 83 %	0 / 22	13 / 193	320 / 1 587	243 / 1 935
org.eclipse.jdt.internal.codeassist.impl	 77 %	3 / 13	42 / 156	181 / 857	282 / 1 419

Credits: Olivier Thomann



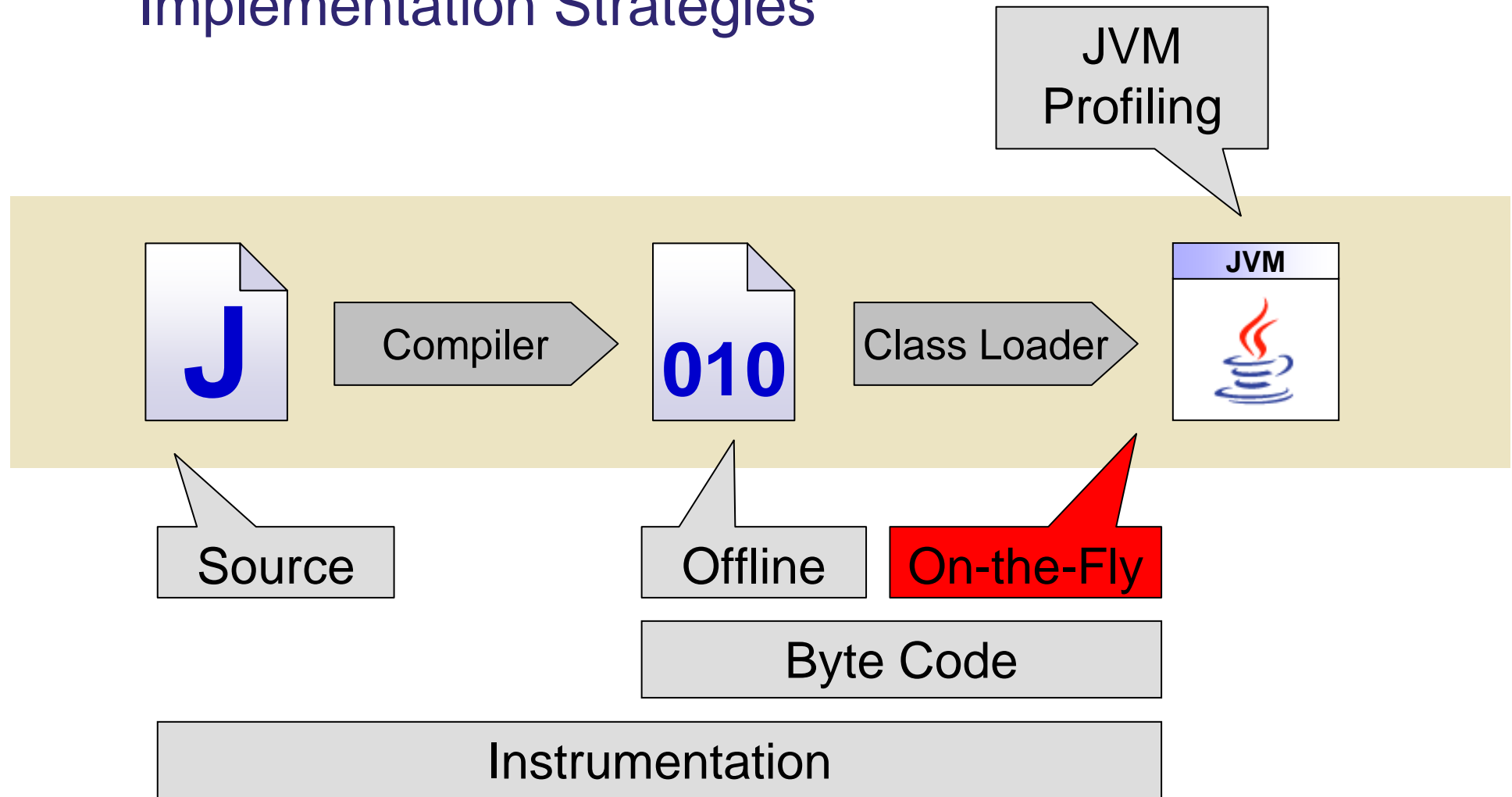
Implementation Details





The Future of Code Coverage for Eclipse

Implementation Strategies





Java Agent

- `java.lang.instrument`
 - In-Memory
 - No class file preprocessing

```
byte[] transform(ClassLoader loader,  
                String className,  
                Class<?> classBeingRedefined,  
                ProtectionDomain protectionDomain,  
                byte[] classfileBuffer)  
throws IllegalClassFormatException
```



Implementation and Packaging

- Set of OSGi Bundles
- < 3,000 LOC
- < 400 KB





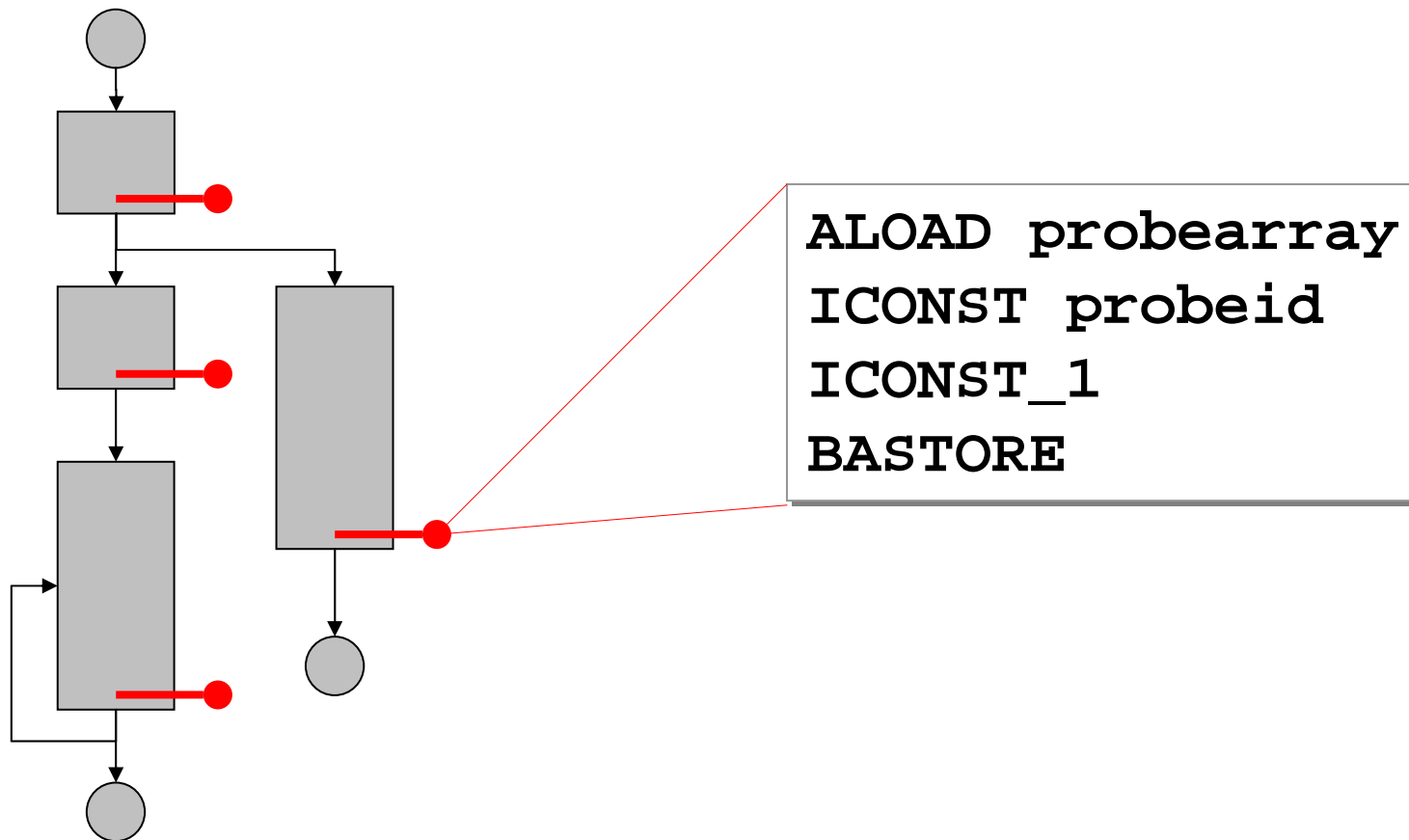
Keep the Good Ideas of EMMA



- Byte Code Instrumentation
 - JRE and Platform Independent
- Basic Block Coverage
 - Good Tradeoff between Details and Overhead
- Using `boolean[]` Arrays for Probe Storage
 - Concurrency Possible
 - Sharing the Instance



Basic Block Coverage





Class Identity

- Issues

- Multiple Versions of the Same Class in one VM
- Modified Classes over Time

- Use CRC64 Hash

- Fits into Java `long`



Avoid Coverage Runtime Dependency

- Avoid Class Loading Issues
- Use JRE APIs only!

```
Object access = ... // Retrieve instance
```

```
Object[] args = new Object[3];  
args[0] = Long.valueOf(0x89f47a04b2881d38); // class id  
args[1] = "com/example/MyClass"; // class name  
args[2] = Integer.valueOf(24); // probe count
```

```
access.equals(args);
```

```
boolean[] probes = (boolean[]) args[0];
```



How to Share an Object Instance?

- The Challenge:

**Share a given object instance
by using JDK APIs only.**

- **Current Solutions:**
 - ➔ **Object as System Property**
 - ➔ **Install Custom Handler with Java Logging**
 - ➔ **Install Custom URL Protocol Handler**
 - ➔ **Instrumented JRE Class**



Runtime Isolation

**Eating one's own dog food:
Run JaCoCo on JaCoCo?**

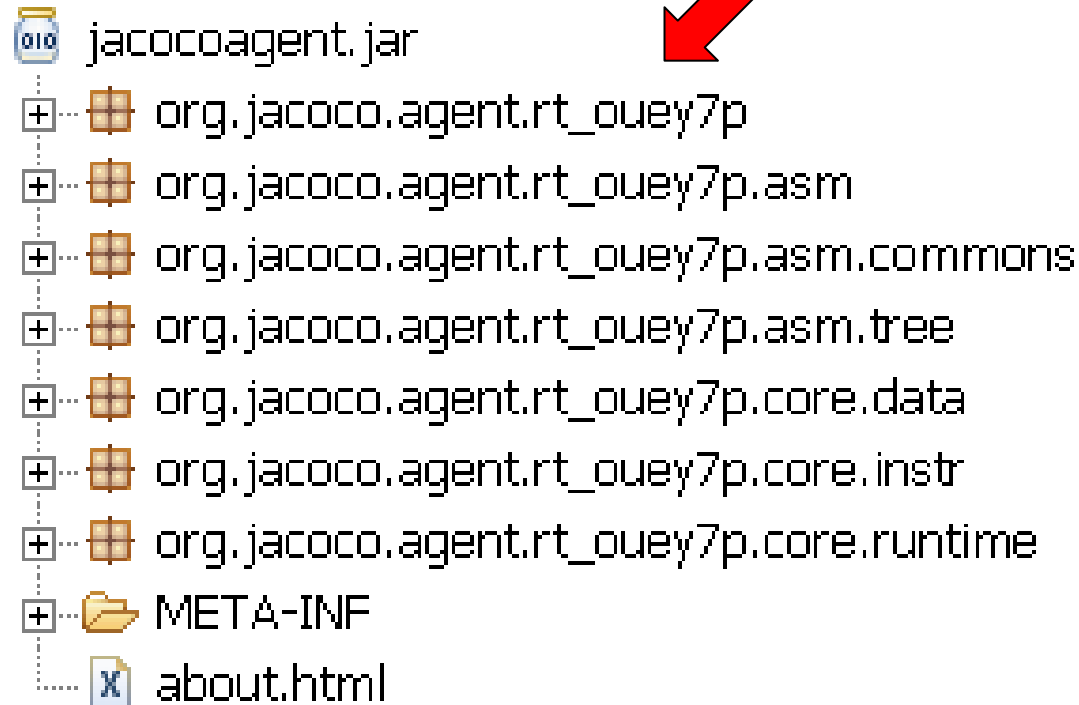
- Java Agent becomes part of the application classpath ☹
- Rename classes in jacocoagent.jar during build ☺





The Future of Code Coverage for Eclipse

Runtime Isolation





JaCoCo Implementation Maxims

- Test First
- Keep it simple and **fast**
- Release Often and Consistent
 - ➔ Every Work Item released as Trunk Build



JaCoCo Status

- Statement Coverage
- HTML, XML, CSV Reports
- Ant Tasks
- Documentation
- APIs not frozen yet!

3rd Party Integrations

- Sonar Plug-in



Future Plans

- Branch Coverage
- Filters
- Eclipse Plugin
- Maven Integration





Get Involved

- Download the Latest Build at
<http://www.eclemma.org/jacoco>
- Integrate it With Your
 - ➔ Build
 - ➔ Whatever Tool
- Get in Touch for
 - ➔ Feature Requests
 - ➔ Bug Reports
 - ➔ Contributions



Questions?

Thank You!

