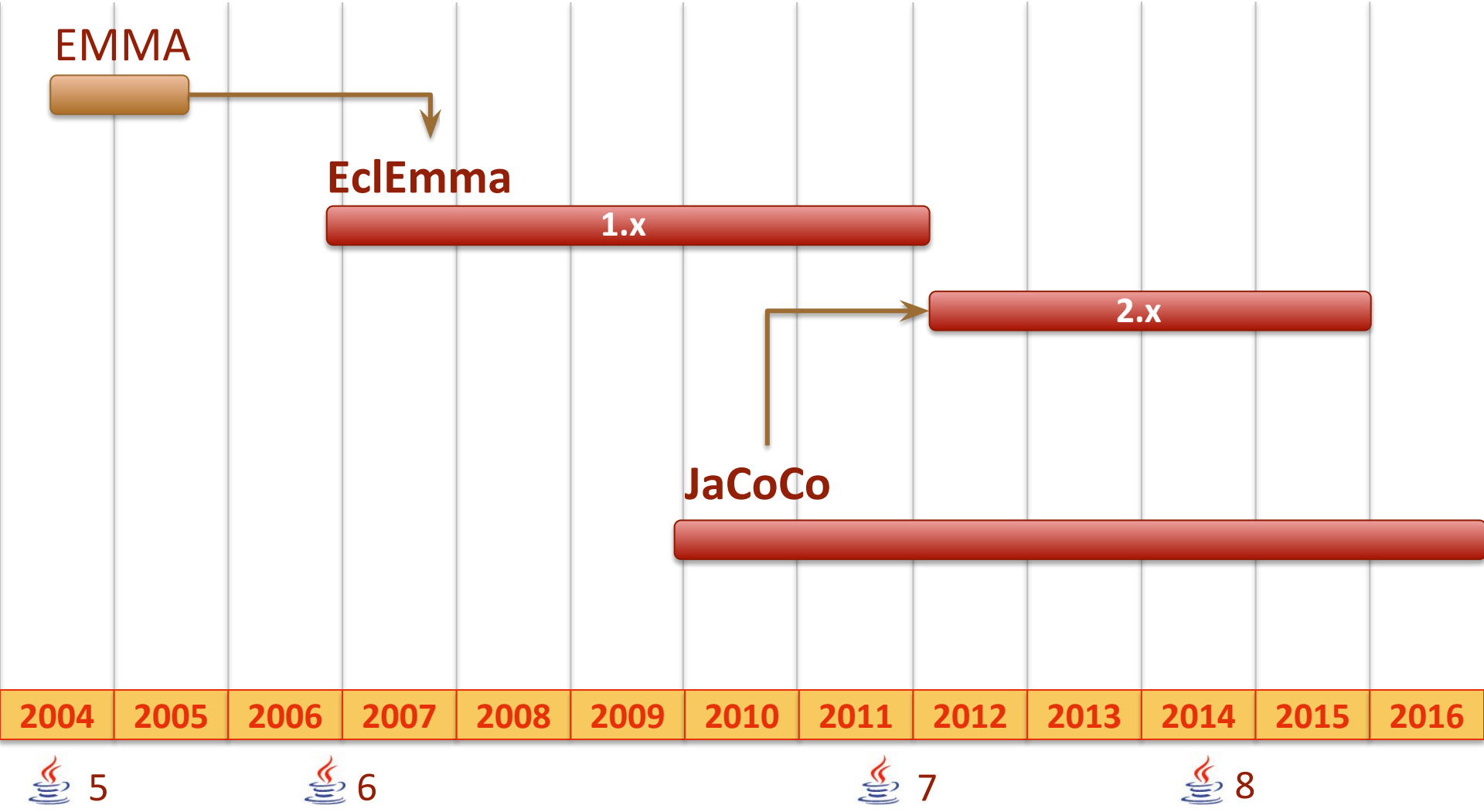
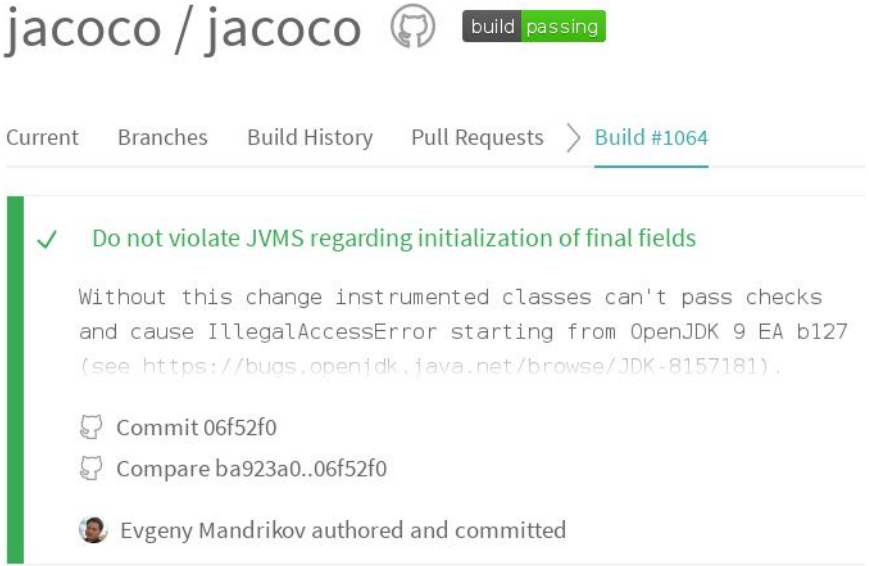
Code Coverage in Practice

Evgeny Mandrikov, SonarSource, @_godin_
Marc Hoffmann, mtrail GmbH, @marcandsweep

EclipseCon 2016, Ludwigsburg


EMMA
EclEmma
1.x
2.x
JaCoCo
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
5
6
7
8

Tested on 5 major JDK versions



We do not only identify issues in JaCoCo...

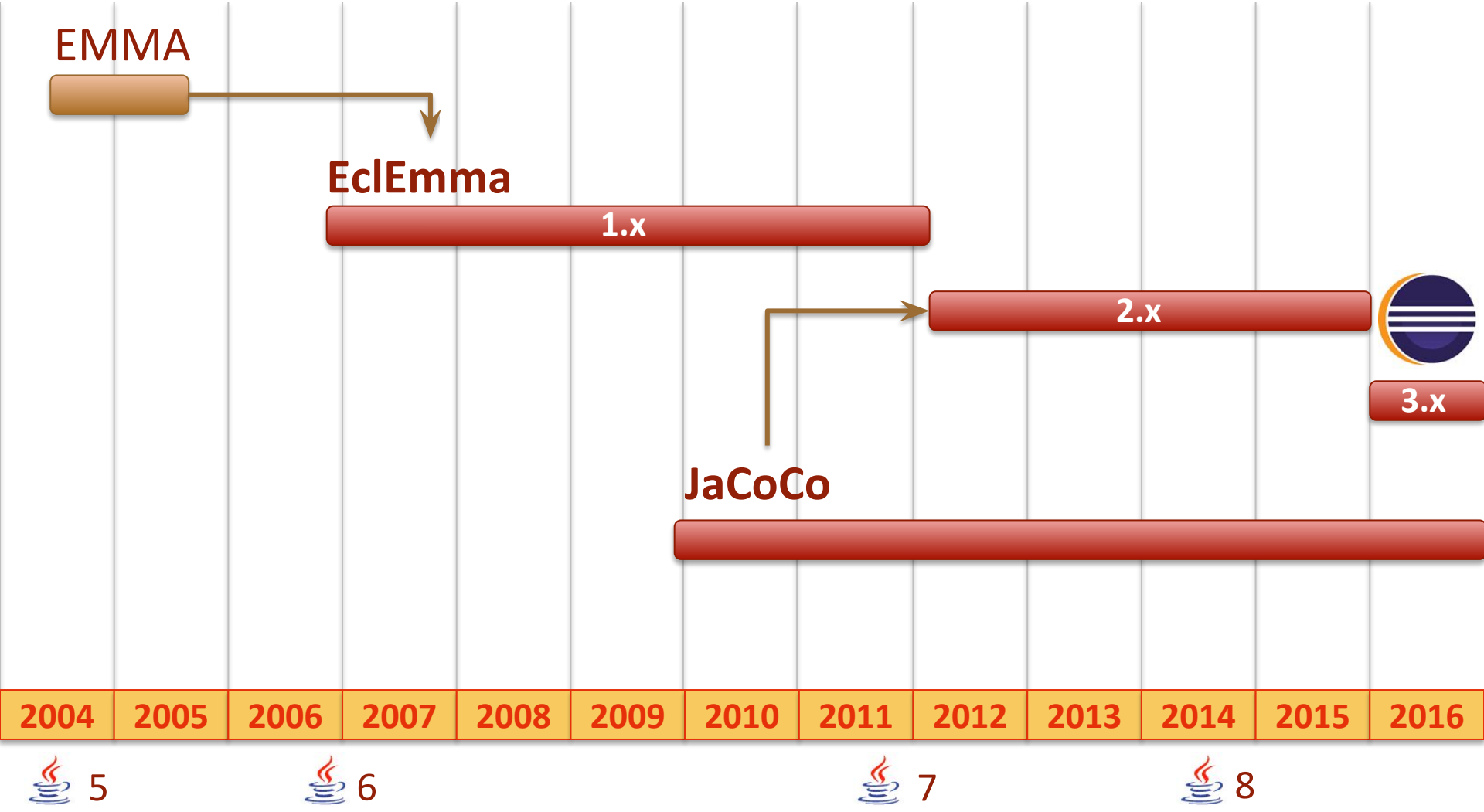
T	Key	Summary	Assignee	Reporter	P	Status
	JDK-8167607	Cyclic interface initialization causes JVM crash	Coleen Phillimore	Webbug Group	3	RESOLVED
	JDK-8164302	No initialization for super interface with default method				
	JDK-8163969	Cyclic interface initialization causes JVM crash				
	JDK-8154017	Shutdown hooks are racing against shutdown sequence System.exit()-calling thread is interrupted				
	JDK-8134862	JVM crash at PhaseIdealLoop::idom_no_update				
	JDK-8131041	Garbage in output of DecimalFormat				
	JDK-8080555	Different bytecode between JDK8u45 and JDK8u60-ea-				
	JDK-8073658	Invalid annotations in bridge methods	Adayapalam	Group		
	JDK-8071444	Annotation of argument of enum constructor causes failure of compilation	*Unassigned*	Webbug Group	3	CLOSED

Details

Type:	Bug
Priority:	3 P3
Affects Version/s:	8u45, 9
Component/s:	hotspot
Labels:	jacoco-found webbug
Subcomponent:	compiler
CPU:	x86

JaCoCo Validation Test Suite

```
// 5. Always executed while block
while (t()) { // $line-whiletrue$
    if (t()) {
        break;
    }
}

// 6. Executed while block
int i = 0;
while (i++ < 3) { // $line-whiletruefalse$
    nop(); // $line-executedwhile$
}

// 7. Executed do while block
do {
    nop(); // $line-executeddowhile$
} while (f());

// 8. Missed for block
for (nop(); f(); nop()) { // $line-missedforincrementer$
    nop(); // $line-missedfor$
}

// 9. Executed for block
for (int j = 0; j < 1; j++) { // $line-executedforincrementer$
    nop(); // $line-executedfor$
}

// 10. Missed for each block
for (Object o : Collections.emptyList()) { // $line-missedforeachincreme
    nop(o); // $line-missedforeach$
}
```


EMMA
EclEmma
1.x
2.x
3.x
JaCoCo
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
5
6
7
8

EclEmma @ Eclipse

- Proposal accepted in July 2016 (https://projects.eclipse.org/projects/technology.eclemma)
- Initial contribution and IP clearance (https://github.com/eclipse/eclemma)
- Project infrastructure
  - builds
  - website in progress
  - update path in progress
  - etc.

Target: Become part of Oxygen!

(so you don’t have to install it any more…)

What can code coverage be used for?

- White-box Testing (Unit Tests)
- Integration Testing
- Usage Analysis
- Differential Code Coverage

- JaCoCo works for any code running on the JVM
- It is not intended to be used for profiling!

Integration with Various Tools

- IDE
  - Eclipse EclEmma™
  - IntelliJ IDEA
  - NetBeans
- Build
  - Maven
  - Gradle
  - Ant

- CI
  - Jenkins
  - TeamCity
  - Visual Studio Team Services
  - SonarQube
- you name it - Java API

Demo

- Maven aggregate report
- Unit Testing within Eclipse IDE
- Differential Code Coverage within Eclipse IDE

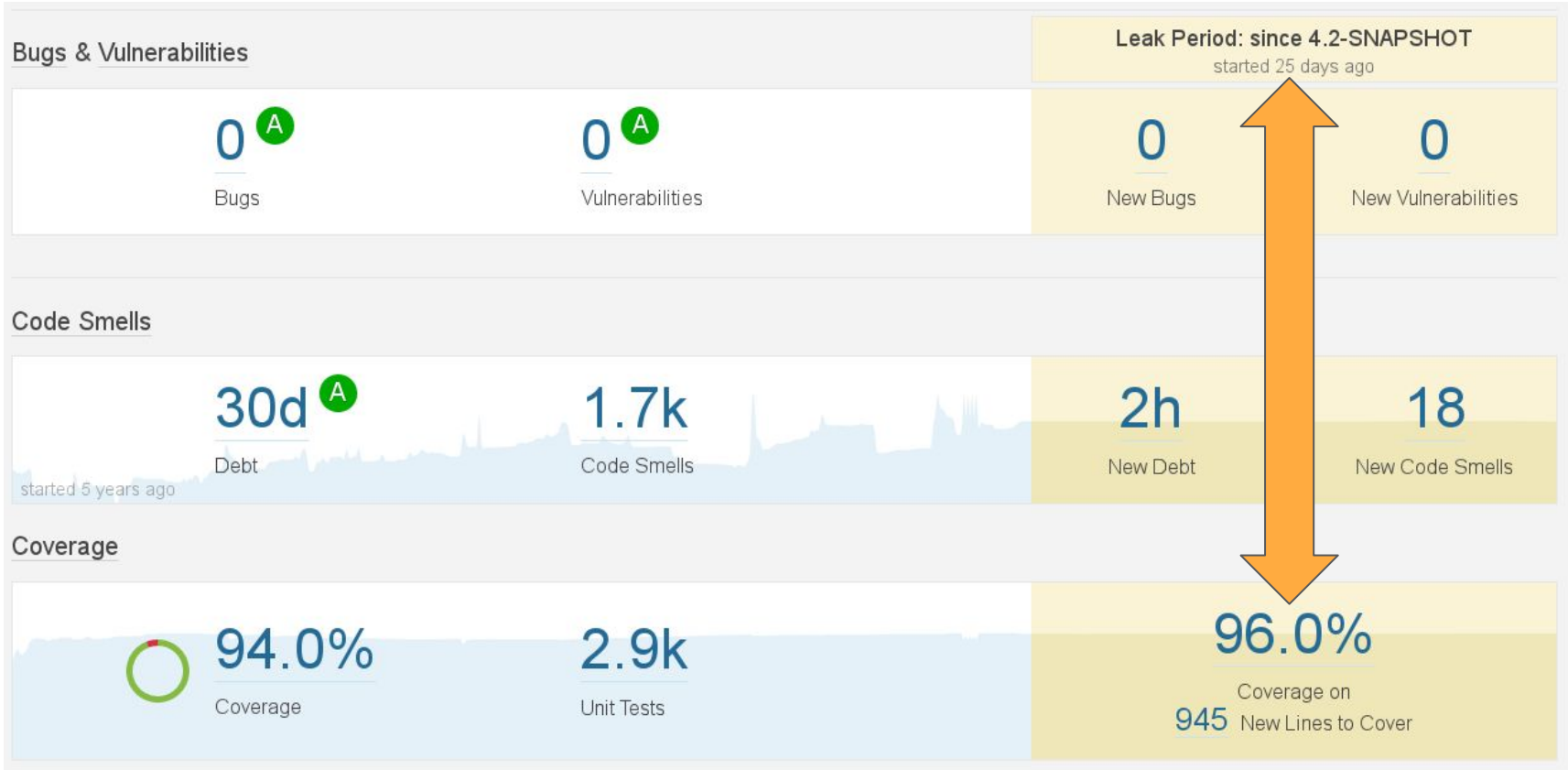
GitHub, Inc. [US] | https://github.com/jacoco/jacoco/blob/master/org.jacoco.doc/pom.xml

```
94      <dependency>
95        <groupId>${project.groupId}</groupId>
96        <artifactId>jacoco-maven-plugin</artifactId>
97        <version>${project.version}</version>
98      </dependency>
99      <dependency>
100       <groupId>${project.groupId}</groupId>
101       <artifactId>jacoco-maven-plugin.test</artifactId>
102       <version>${project.version}</version>
103       <scope>test</scope>
104     </dependency>
105   </dependencies>
106
107   <build>
108     <sourceDirectory>src</sourceDirectory>
109
110     <plugins>
111       <plugin>
112         <groupId>org.jacoco</groupId>
113         <artifactId>jacoco-maven-plugin</artifactId>
114         <version>${project.version}</version>
115         <executions>
116           <execution>
117             <id>report-aggregate</id>
118             <phase>prepare-package</phase>
119             <goals>
120               <goal>report-aggregate</goal>
121             </goals>
122             <configuration>
```

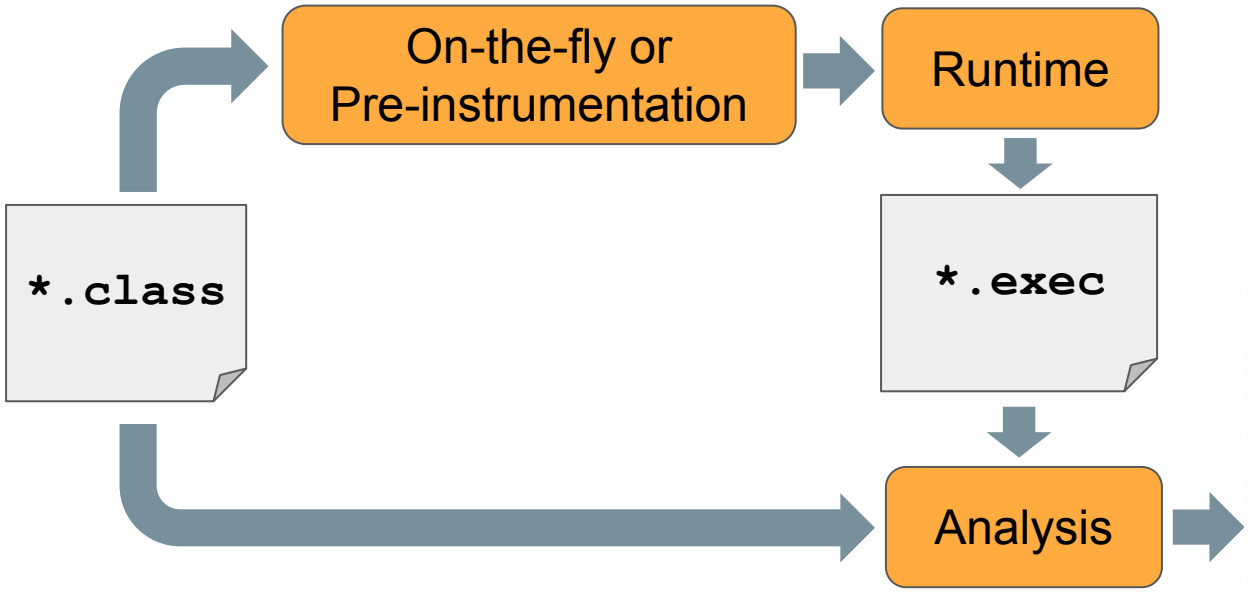
Principles and Best Practices

- Don’t care about percentage value, observe the amount of untested code
- Focus on coverage of modified and new code
- Make coverage analysis an inherent part of your build/test chain
- Make coverage reports available to everybody in the team
  - But not the management!
- Always go for functional coverage when writing tests
  - Just executing code will not improve its quality!

Continuous Inspection



JaCoCo works on Java class files only




```
120.    public static IRuntime createFor(final Instrumentation
121.            final String className, final String accessFiel
122.            throws ClassNotFoundException {
123.        final ClassFileTransformer transformer = new ClassF
124.            public byte[] transform(final ClassLoader loade
125.                    final String name, final Class<?> class
126.                    final ProtectionDomain protectionDomain
127.                    throws IllegalClassFormatException {
128.                if (name.equals(className)) {
129.                    return instrument(source, accessFieldNa
130.                }
131.                return null;
132.            }
133.        };
134.        inst.addTransformer(transformer);
135.        final Class<?> clazz = Class.forName(className.repl
136.        inst.removeTransformer(transformer);
137.        try {
138.            clazz.getField(accessFieldName);
139.        } catch (final NoSuchFieldException e) {
140.            throw new RuntimeException(format(
141.                    "Class %s could not be instrumented.",
142.        }
143.        return new ModifiedSystemClassRuntime(clazz, access
144.    }
```

Common Pitfalls and how to avoid them

- Different Class Files runtime/analysis
  - different compiler **implementations**
  - different compiler **versions**
  - different compiler **settings**
  - Pack200
- Different version of the same class in same group
- Not graceful JVM termination
- Reflection (synthetic fields and methods)
- Pre-instrumentation requires direct dependency on the JaCoCo runtime
- Interoperability with other agents
  - PowerMock might bypass agents, because reads files
  - overrides JaCoCo init method

Get Involved

- EclEmma and JaCoCo user's group:
  - jacoco@googlegroups.com
  - https://groups.google.com/forum/#!forum/jacoco
- EclEmma
  - http://www.eclemma.org/
  - https://github.com/eclipse/eclemma
- JaCoCo
  - http://www.jacoco.org/jacoco/index.html
  - https://github.com/jacoco/jacoco

eclipsecon Europe 2016

OSGi™ Community Event 2016

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

- 1 0 + 1