Practical Applications of Static Bytecode Analysis and Transformation for the Java™ Platform

Eugene Kuleshov (Terracotta)
Miško Hevery (Google)

BOF-5839

Java

Sun microsystems

Java™ bytecode analysis and transformation isn't difficult and it is very cool.

Understand how code analysis tools for Java™ platform do their job and how these ideas can be used in other applications.

GOAL

Agenda

- Java Virtual Machine, Bytecode and ASM framework
- Test Coverage with Cobertura
- Google Singleton Detector
- Finding code that is hard to test: Testability Explorer

Java Virtual Machine

- A proven platform for running reliable and high-performance applications
- Built for the statically-typed Java language
- Classes can be generated and analyzed at runtime or at the compile time
- Other languages can be compiled to Java bytecode, including the dynamically typed languages
  - Groovy, JRuby, JavaFX, Jhyton...

The Class File format

- Constant Pool
  - field and method names, type descriptors, String literals and other constants
- Attributes
  - Fields
  - Methods
  - Code
    - ordered list of (stack based) instructions
  - Debug information (line numbers, local variable names)
  - Exceptions
  - User defined attributes

Class File access and modification problems

- Lots of serialization and deserialization details
- Constant pool management
  - managing constant pool indexes/references
  - missing or unused constants
- Jump offsets
  - calculation of instruction offsets
  - inserting or removing instructions from the method
- Computation of stack size and StackMapTable
  - requires a control flow analysis

ASM Bytecode Framework

http://asm.objectweb.org/

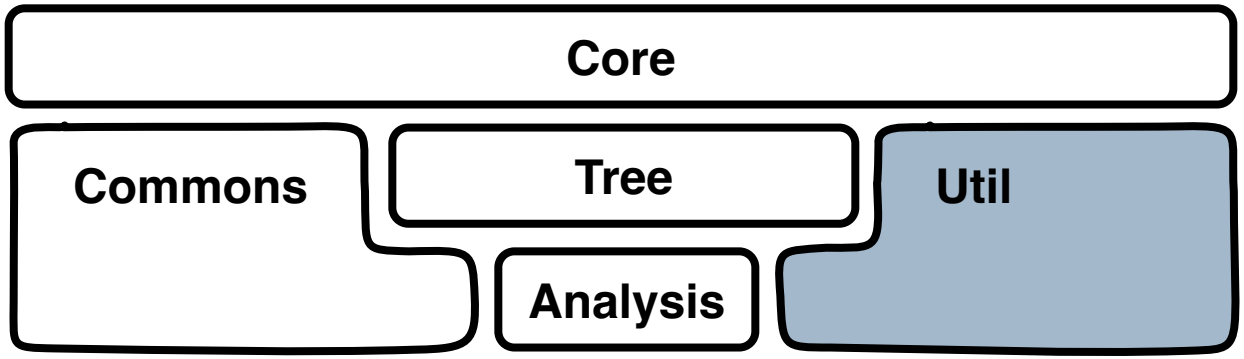
- Initial goal: dynamic class generation and modification
  - very small and very fast tool
  - primarily adapted for simple transformations
  - complete control over the produced classes is not needed
- Approach:
  - use the Visitor pattern without using in-memory object model
  - hide the (de)serialization and constant pool management details
  - represent jump offsets by Label marker objects
  - checker and ASMifier tools help to generate the method code
  - automatic computation of the max stack size and StackMapTable

ASM Bytecode Framework
Users

- Languages and AOP tools
  - AspectWerkz, AspectJ, dynaop, CGLIB, Jamaica, BeanShell, Groovy, JRuby, Open Quark, NetLogo, Retrotranslator
- Java ME
  - EclipseME, Sun Java ME emulation for Java SE, MicroEmulator
- Frameworks
  - Terracotta, RIFE, R-OSGi, Fractal, Proactive, Substance L&F
- Persistence
  - Oracle BerkleyDB and TopLink, Speedo, EasyBeans, JDBCPersistence, Ebean, JPOX
- Monitoring
  - BEA WebLogic, BTrace, JiP
- Testing and code analysis
  - Agitar, Cobertura, IBM AUS, JCarder, SemmleCode, Structure101, SonarJ

ASM Bytecode Framework

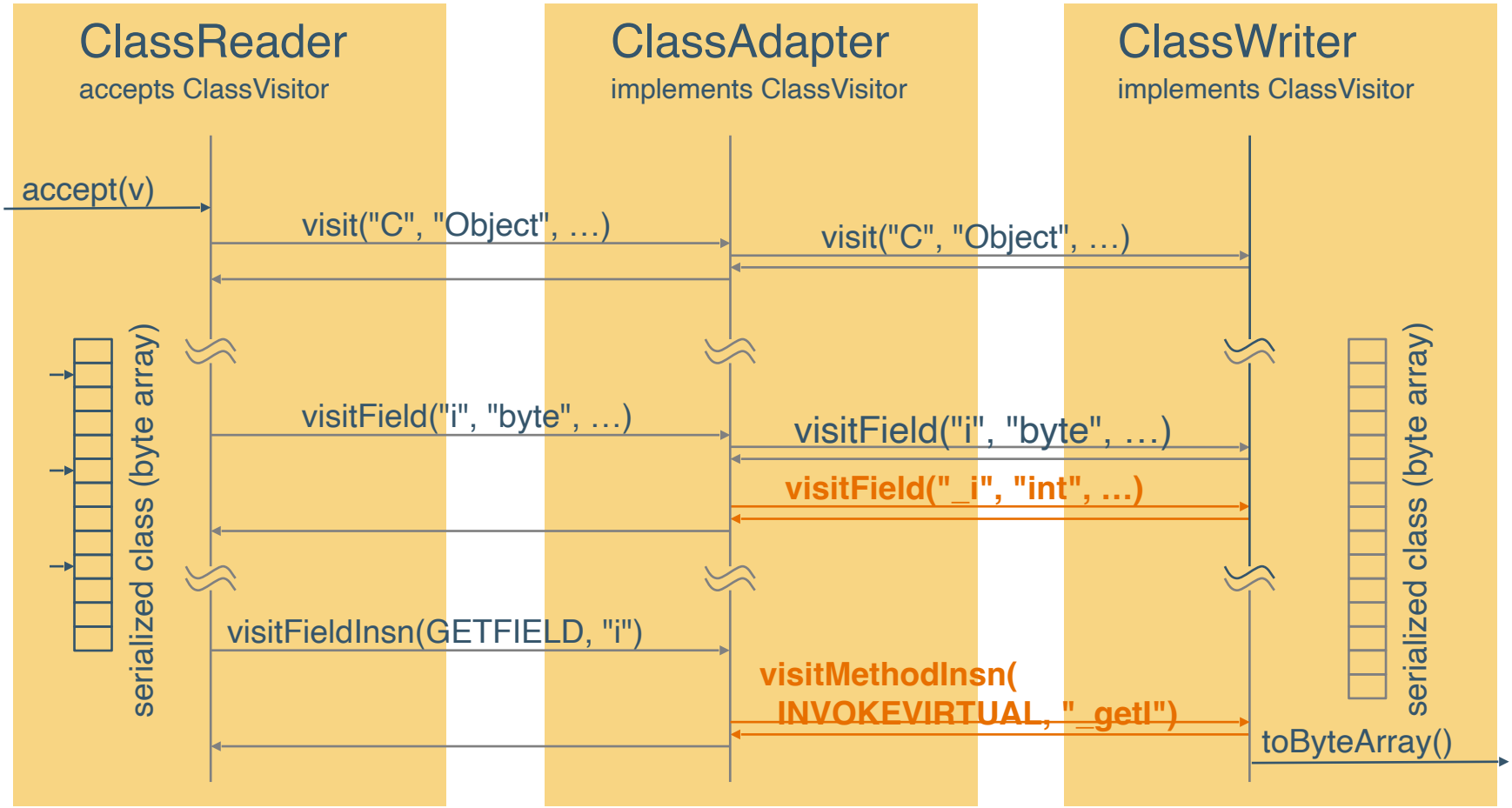
Framework Organization



- Core (42Kb for ASM 3.1)
  - generate classes
  - basic transformations
- Tree and Analysis
  - in-memory representation
  - analysis algorithms
- Commons
  - renaming, advice adapter,
  - inline JSR/RET subroutines,
  - sort local variables, calculate serialVersionUID, etc.
- Util
  - checker,
  - decompiler
  - ASMifier

ASM Bytecode Framework

Main idea



ASM Bytecode Framework
Example

```
ClassReader cr = new ClassReader(bytecode);
ClassWriter cw = new ClassWiter(cr,
    ClassWriter.COMPUTE_MAXS);

ClassVisitor cv = new CheckClassAdapter(cw);
cr.accept(new MyClassAdapter(cv), 0);
final byte[] b = cw.toByteArray();

CheckClassAdapter.verify(new ClassReader(newBytecode),
    true, new PrintWriter(System.err)));

// load the new class
final byte[] bytes = cw.toByteArray();
Class newClass = new ClassLoader(parent) {
  Class c = defineClass(name, bytes, 0, bytes.length);
}.c;
```

ASM Bytecode Framework

Tree package

```
ClassNode classNode = new ClassNode();

ClassReader cr = new ClassReader(bytecode);
cr.accept(classNode, 0);

// analyze or transform classNode structures
Iterator it = classNode.methods;
while(it.hasNext()) {
  MethodNode methodNode = (MethodNode) it.next();
  InsnList instructions = methodNode.instructions;
  ...
}

ClassWriter cw =
   new ClassWriter(ClassWriter.COMPUTE_MAXS);
cn.accept(new MyClassAdapter(cw, classNode));
```

ASM Bytecode Framework

Type info for variable and stack slots of method instruction

```
Analyzer a = new Analyzer(new BasicInterpreter());
a.analyze(className, methodNode);

Frame[] frames = a.getFrames();
for (int i = 0; i < frames.length; i++) {
  Frame f = frames[i];
  for (int j = 0; j < f.getLocals(); ++j) {
    BasicValue local = (BasicValue) f.getLocal(j);
    // ... local.getType()
  }
  for (int j = 0; j < f.getStackSize(); ++j) {
    BasicValue stack = (BasicValue) f.getStack(j);
    // ... stack.getType()
  }
}
```

ASM Bytecode Framework

Collect class dependencies

```
final Set dependencies = new HashSet();

ClassReader cr = new ClassReader(is);
cr.accept(new RemappingClassAdapter(
  new EmptyVisitor(),
    new Remapper() {
      public String map(String typeName) {
        dependencies.add(typeName);
        return typeName;
      }
    }), ClassReader.SKIP_FRAMES);
```

Agenda

- Java Virtual Machine, Bytecode and ASM framework
- Test Coverage with Cobertura
- Google Singleton Detector
- Finding code that is hard to test: Testability Explorer

Cobertura

http://cobertura.sourceforge.net/

- An open source tool to identify test coverage
- Features
  - Ant, Maven and command line interface
  - Reports in HTML or XML
  - Percentage of lines and branches covered for each class, package and for the overall project
  - McCabe cyclomatic code complexity

Cobertura

Using ASM framework to collect test coverage

- Instruments compiled classes
  - Works with Java or non-Java languages compiled to the bytecode
- Adds callbacks to the recording engine
  - For each method line
  - For each branching instruction
- Save information about added callbacks
- Recording engine is called during test execution
  - Collects number of invocations for each callback to mark code branches as tested
  - Compares what callbacks haven't been invoked to mark code branches as untested

Cobertura Demo

Test coverage for Java and Groovy code

Cobertura

Coverage Report - org.javatx.notifier.GroovyNotifier

Classes in this File	Line Coverage	Branch Coverage	Complexity
GroovyNotifier	100% 2/2	50% 1/2	0

```
1
2     package org.javatx.notifier
3
4     public class GroovyNotifier {
5
6       List notifications = []
7
8       public void notify(String msg) {
9   1     notifications.add(msg)
10      }
11
12      public List getNotifications() {
13  1     return notifications
14      }
15
16    }
17
```

Report generated by Cobertura 1.9 on 25/04/08 7:51 PM.

Agenda

- Java Virtual Machine, Bytecode and ASM framework
- Test Coverage with Cobertura
- Google Singleton Detector
- Finding code that is hard to test: Testability Explorer

Singletons

- Who thinks Singletons are good?
  - show of hands
- Who thinks Singletons are bad?
  - show of hands

Singleton Detector

http://code.google.com/p/google-singleton-detector

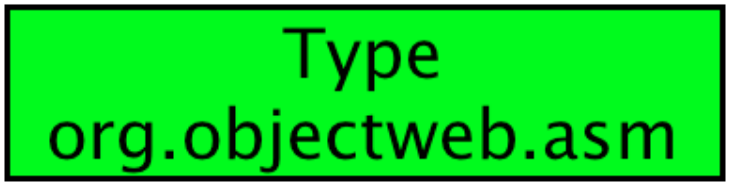
- Singletons: Private constructor & getInstance()
- Hingeltons: Helper class enforcing a Singleton
- Mingleton: Class with static method that returns some state and do not have parameters
- Fingleton: Public static field Singleton

Singleton Detector

Using ASM framework to analyze classes

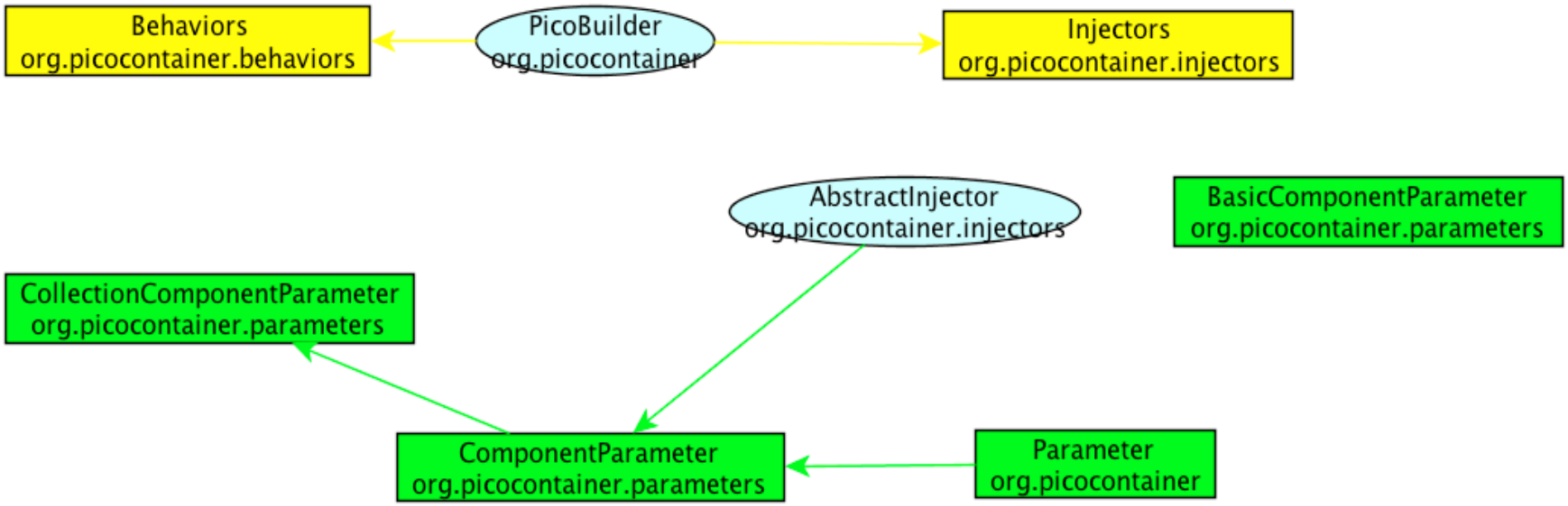
- ClassVisitor: Iterate over all classes from jar
- MethodVisitor: Look for static no arg methods which return a same type as enclosing class
- FieldVisitor: Look for public fields of same type as enclosing class

ASM



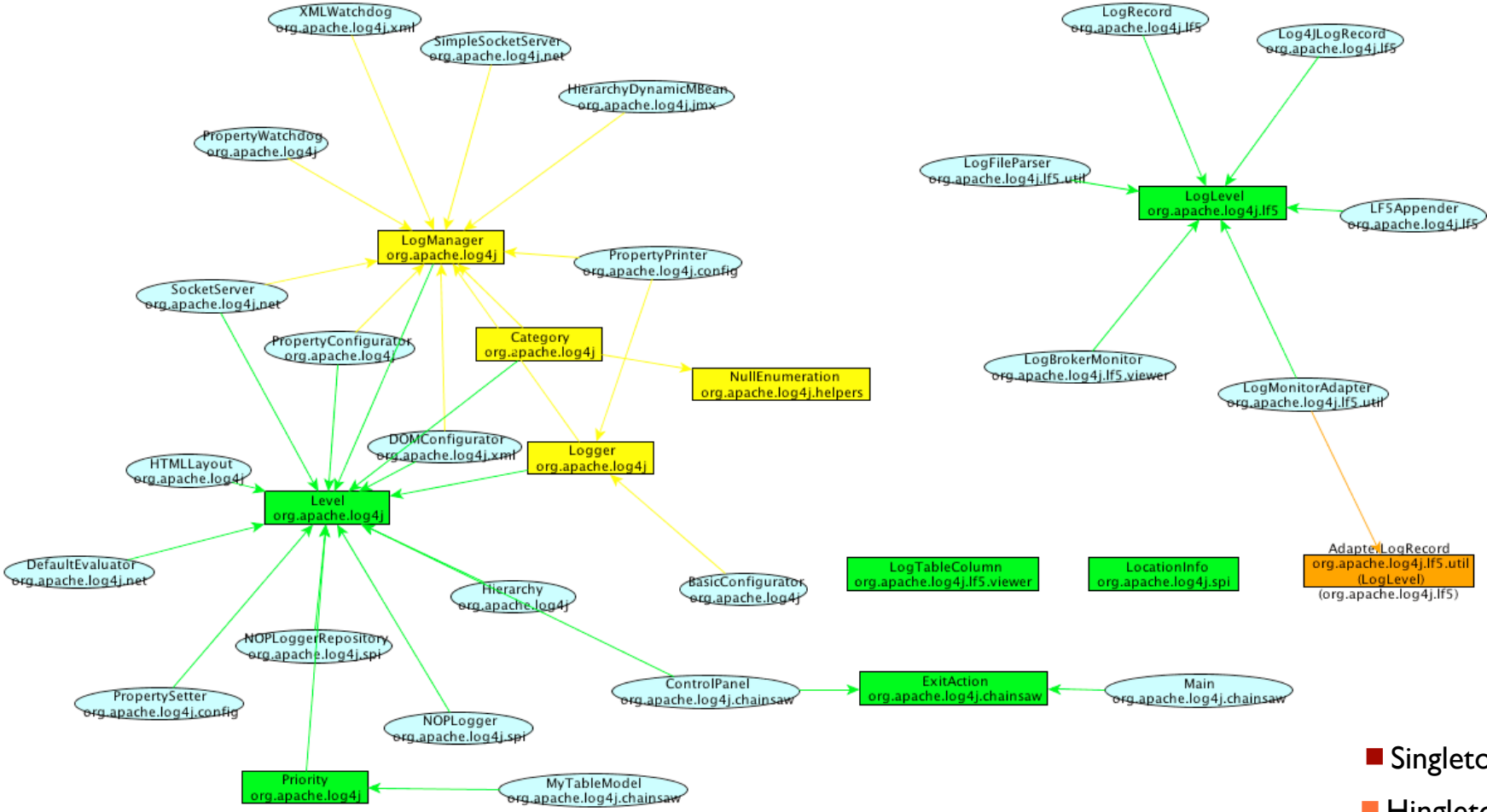
- Singleton
- Hingleton
- Mingleton
- Fingleton

picocontainer-2.0

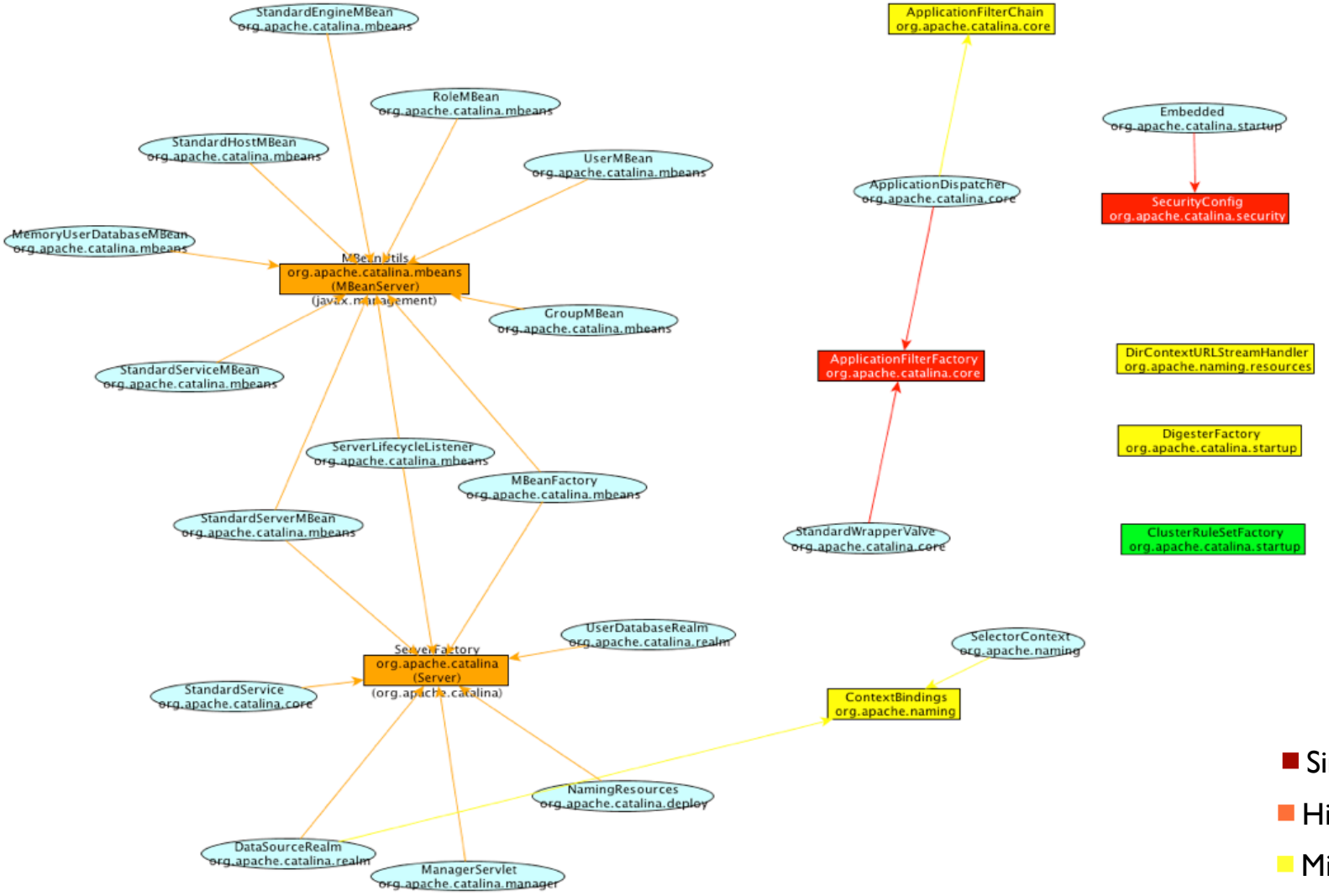


- Singleton
- Hingleton
- Mingleton
- Fingleton

log4j-1.2.15


XMLWatchdog
org.apache.log4j.xml
SimpleSocketServer
org.apache.log4j.net
HierarchyDynamicMBean
org.apache.log4j.jmx
PropertyWatchdog
org.apache.log4j
LogManager
org.apache.log4j
PropertyPrinter
org.apache.log4j.config
SocketServer
org.apache.log4j.net
PropertyConfigurator
org.apache.log4j
Category
org.apache.log4j
NullEnumeration
org.apache.log4j.helpers
DOMConfigurator
org.apache.log4j.xml
Logger
org.apache.log4j
HTMLLayout
org.apache.log4j
Level
org.apache.log4j
DefaultEvaluator
org.apache.log4j.net
Hierarchy
org.apache.log4j
BasicConfigurator
org.apache.log4j
NOPLoggerRepository
org.apache.log4j.spi
PropertySetter
org.apache.log4j.config
NOPLogger
org.apache.log4j.spi
Priority
org.apache.log4j
MyTableModel
org.apache.log4j.chainsaw
ControlPanel
org.apache.log4j.chainsaw
ExitAction
org.apache.log4j.chainsaw
Main
org.apache.log4j.chainsaw
LogRecord
org.apache.log4j.lf5
Log4JLogRecord
org.apache.log4j.lf5
LogFileParser
org.apache.log4j.lf5.util
LogLevel
org.apache.log4j.lf5
LF5Appender
org.apache.log4j.lf5
LogBrokerMonitor
org.apache.log4j.lf5.viewer
LogMonitorAdapter
org.apache.log4j.lf5.util
LogTableColumn
org.apache.log4j.lf5.viewer
LocationInfo
org.apache.log4j.spi
AdapterLogRecord
org.apache.log4j.lf5.util
(LogLevel)
(org.apache.log4j.lf5)
Singleton
Hingleton
Mingleton
Fingleton

tomcat-catalina-6.0.16



Agenda

- Java Virtual Machine, Bytecode and ASM framework
- Test Coverage with Cobertura
- Google Singleton Detector
- Finding code that is hard to test: Testability Explorer

Testability Explorer

http://code.google.com/p/testability-explorer
http://testabilityexplorer.org

- Goodness
  - Dependency Injection
  - Interface / Implementation separation
- Badness
  - Global State
- Recursively

Testability Explorer

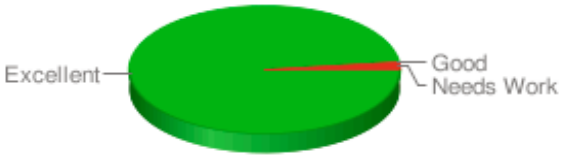
Using ASM framework to analyze classes

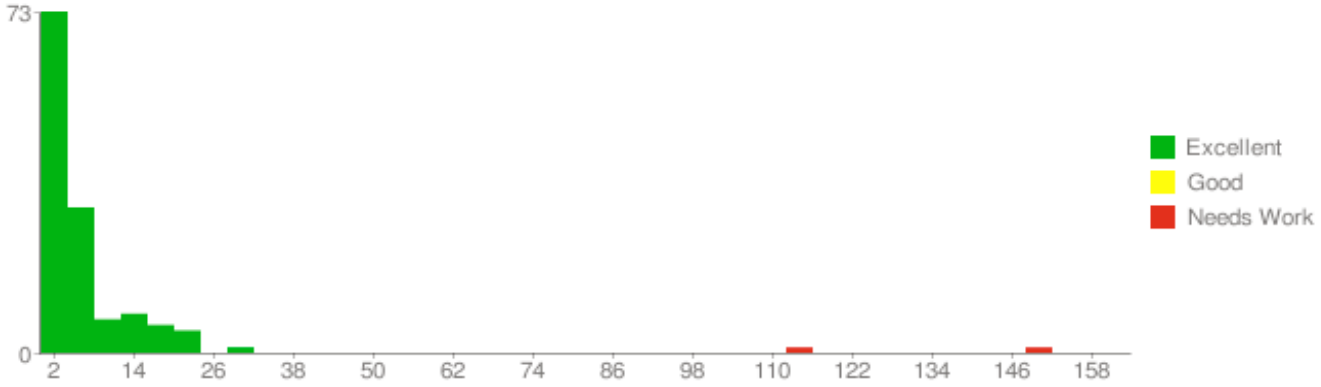
- ClassVisitor: Iterate over all classes from jar
- MethodVisitor: mark each parameter as injectable if it can be accessed from the caller.
- FieldVisitor: look for global state
- Compute the Testability Number

testability-explorer-1.1.0

Class breakdown

Analyzed classes :	131	
Excellent classes :	129	98.5%
Good classes :	0	0.0%
Needs work classes :	2	1.5%





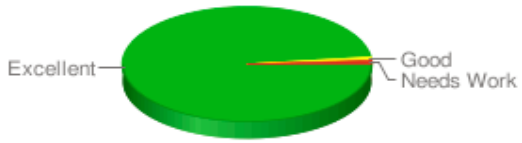
Highest Cost

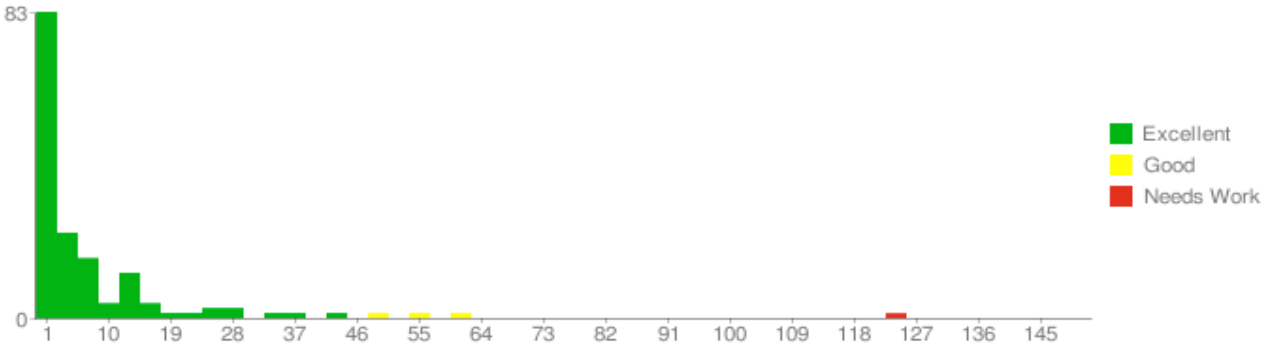
```
com.google.test.metric.Testability 149
com.google.test.metric.asm.MethodVisitorBuilder 113
com.google.test.metric.method.BlockDecomposer 29
com.google.test.metric.asm.MethodVisitorBuilder$2 23
com.google.test.metric.Type 22
```

picocontainer-2.0

Class breakdown

Analyzed classes :	153	
Excellent classes :	150	98.0%
Good classes :	2	1.3%
Needs work classes :	1	0.7%





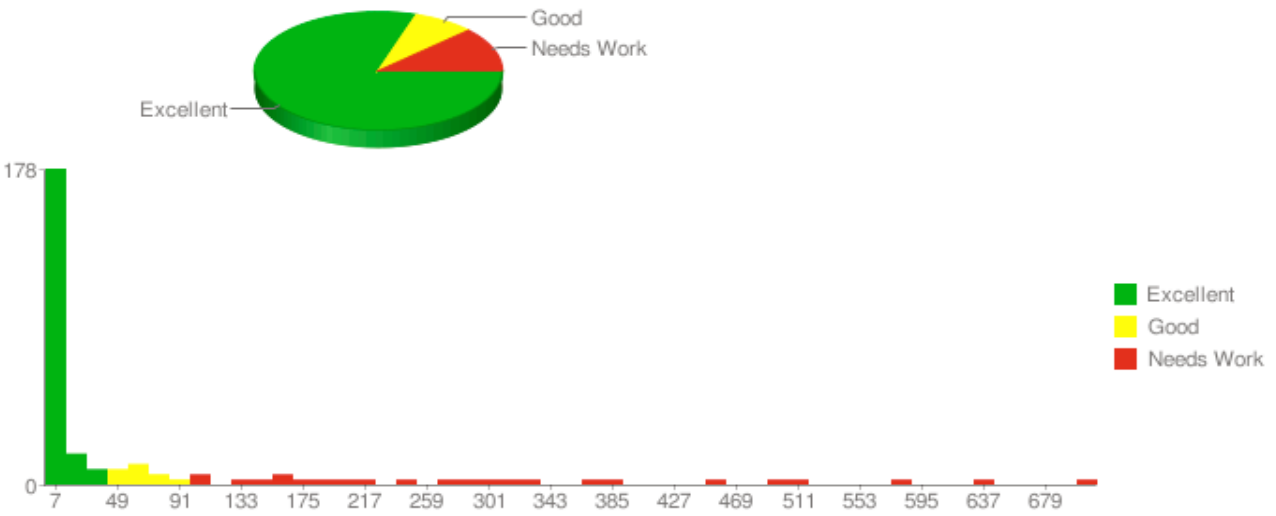
Highest Cost

```
org.picocontainer.paranamer.BytecodeReadingParanamer 124
org.picocontainer.PicoBuilder 62
org.picocontainer.paranamer.BytecodeReadingParanamer$ClassReader 56
org.picocontainer.behaviors.PropertyApplicator 48
org.picocontainer.paranamer.BytecodeReadingParanamer$TypeCollector 42
org.picocontainer.parameters.BasicComponentParameter 36
org.picocontainer.DefaultPicoContainer 34
org.picocontainer.injectors.ConstructorInjector 29
org.picocontainer.paranamer.DefaultParanamer 28
```

log4j-1.2.15

Class breakdown

```
  Analyzed classes :   256
 Excellent classes :   206   80.5%
      Good classes :    20    7.8%
Needs work classes :    30   11.7%
```



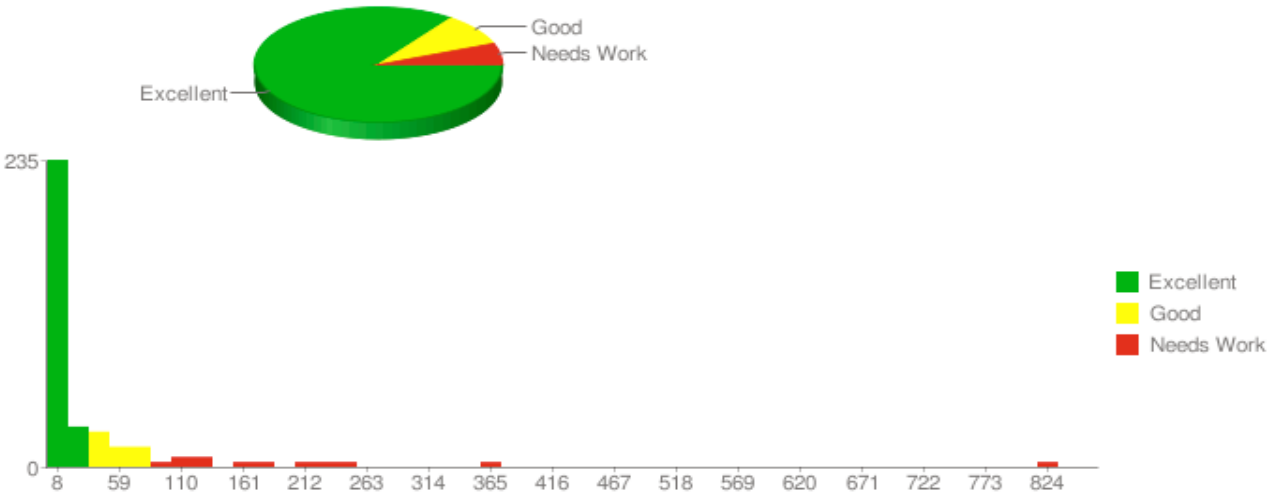
Highest Cost

```
org.apache.log4j.net.SocketServer 708
org.apache.log4j.chainsaw.Main 642
org.apache.log4j.net.SimpleSocketServer 574
org.apache.log4j.lf5.DefaultLF5Configurator 510
org.apache.log4j.PropertyConfigurator 497
org.apache.log4j.PropertyWatchdog 497
org.apache.log4j.config.PropertySetter 453
```

catalina-6.0.16

Class breakdown

```
  Analyzed classes :   338
 Excellent classes :   287   84.9%
      Good classes :    32    9.5%
Needs work classes :    19    5.6%
```



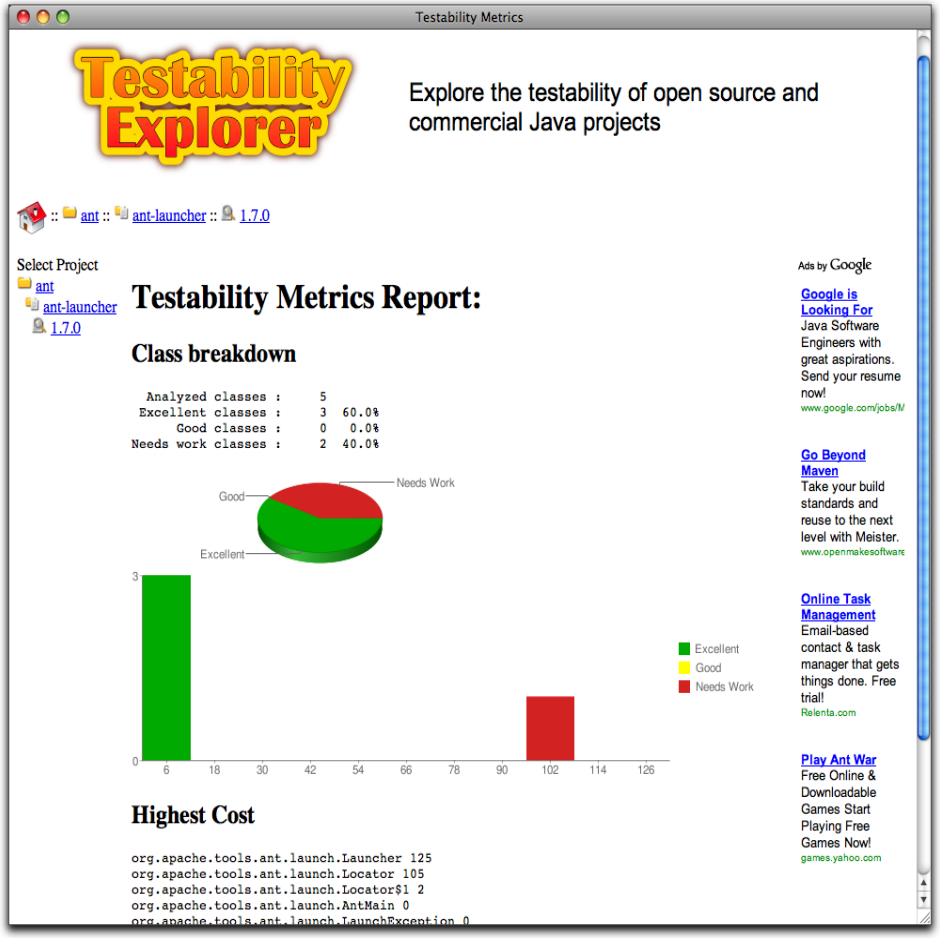
Highest Cost

```
org.apache.catalina.loader.WebappLoader 819
org.apache.catalina.core.StandardContext 362
org.apache.catalina.startup.TldConfig 240
org.apache.catalina.startup.ContextConfig 222
org.apache.catalina.startup.Catalina 217
org.apache.catalina.startup.Embedded 208
org.apache.catalina.startup.Bootstrap 206
org.apache.catalina.util.ExtensionValidator 183
```

Website Launch

http://testabilityexplorer.org/

- Catalog and publish
  - Open Source Projects
  - Releases
  - Compare
  - Commercial projects can also be catalogued if owners wish



Links

- Eugene blog
  - http://jroller.com/eu
- ASM Framework
  - http://asm.objectweb.org/
- Cobertura
  - http://cobertura.sourceforge.net/
- Singleton Detector
  - http://code.google.com/p/google-singleton-detector
- Testability Explorer
  - http://code.google.com/p/testability-explorer
  - http://testabilityexplorer.org

THANK YOU

Eugene Kuleshov (Terracotta)
Miško Hevery (Google)

BOF-5839