Taming Reflection

Aiding Static Analysis in the Presence of Reflection and Custom Class Loaders

Eric Bodden, Andreas Sewe, Jan Sinschek,
Hela Oueslati and Mira Mezini

CASED

TECHNISCHE UNIVERSITÄT DARMSTADT

"Don't write to closed connections!"

Reflective method calls

```
public class Main {

 public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
 }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

```
Evil.doEvil(c);
```

Reflective method calls

```
Evil.doEvil(c);
```

```
public class Main {

 public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
 }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

hard to analyze statically!

Reflective method calls

```
public class Main {

 public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( args[0] );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
 }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

```
Evil.doEvil(c);
```

impossible

~~hard~~ to analyze statically!

Reflective method calls

```
Evil.doEvil(c);
```

```
public class Main {

 public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName(args[0]);
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
 }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

impossible

~~hard~~ to analyze statically!

Present a pragmatic, partial solution

Important reflective calls

- `Class.forName(String className)`
- `Class.newInstance()`
- `Method.invoke(Object tgt, Object[] args)`
- `Constructor.newInstance(Object[] args)`
- ...

Uses of Reflection

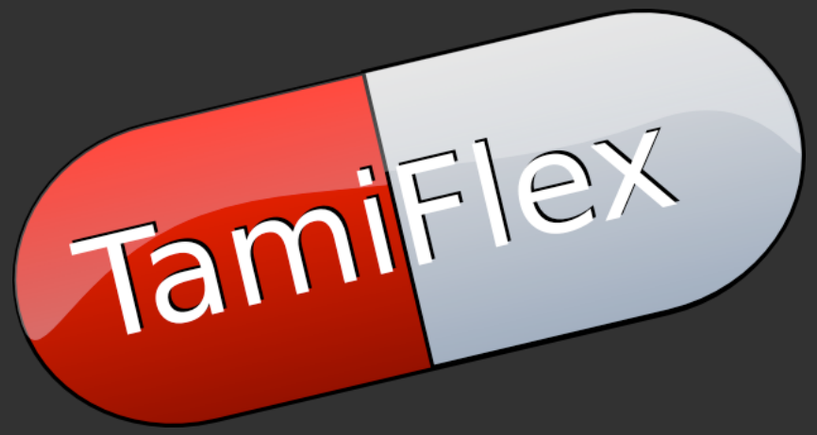
- Extensibility
- Separate Compilation
- Runtime class generation
- "Hacking": call private methods

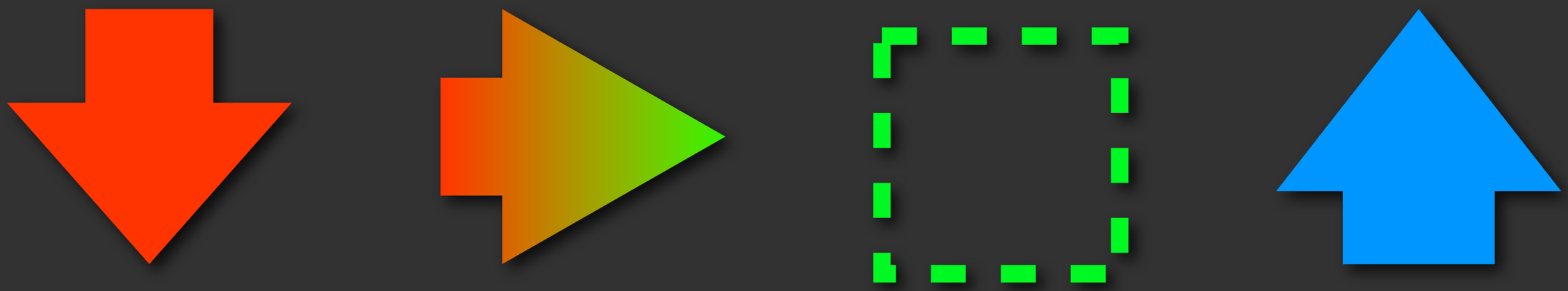
Reflection is...

used a lot

unsound to ignore

often impossible to resolve





Play-Out

Boost

Analyse & Transform

Play-In

Step 1: Play-Out

Play-Out

```
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

Play-Out

```java
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connec
     c.close();
 }

}
```

```
$ javac Main.java
$ java -javaagent:poa-trunk.jar=out Main

============================================
TamiFlex Play-Out Agent Version trunk
Found 2 new log entries.
Log file written to: out/refl.log
$ cat out/refl.log
Class.forName;Evil;Main.main;6
Method.invoke;<Evil: void doEvil(Connection)>;Main.main;8;
```

Play-Out

```java
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connec
     c.close();
 }

}
```

```
$ javac Main.java
$ java -javaagent:poa-trunk.jar=out Main

============================================
TamiFlex Play-Out Agent Version trunk
Found 2 new log entries.
Log file written to: out/refl.log
$ cat out/refl.log
Class.forName;Evil;Main.main;6
Method.invoke;<Evil: void doEvil(Connection)>;Main.main;8;
```

kind of call

Play-Out

```java
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connec
     c.close();
 }

}
```

```
$ javac Main.java
$ java -javaagent:poa-trunk.jar=out Main

============================================
TamiFlex Play-Out Agent Version trunk
Found 2 new log entries.
Log file written to: out/refl.log
$ cat out/refl.log
Class.forName;Evil;Main.main;6
Method.invoke;<Evil: void doEvil(Connection)>;Main.main;8;
```

fully resolved target of call

Play-Out

```java
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connec
     c.close();
 }

}
```

```
$ javac Main.java
$ java -javaagent:poa-trunk.jar=out Main

============================================
TamiFlex Play-Out Agent Version trunk
Found 2 new log entries.
Log file written to: out/refl.log
$ cat out/refl.log
Class.forName;Evil;Main.main;6
Method.invoke;<Evil: void doEvil(Connection)>;Main.main;8;
```

location of call

Play-Out

```
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class c = Class.forName( "Evil" );
     Method m = c.getMethod("doEvil", Connection.class);
     m.invoke(null,c);

     c.write("ohoh...");
  }

public class Evil {

 public static void doEvil(Connection c) {
     c.close();
 }

}
```

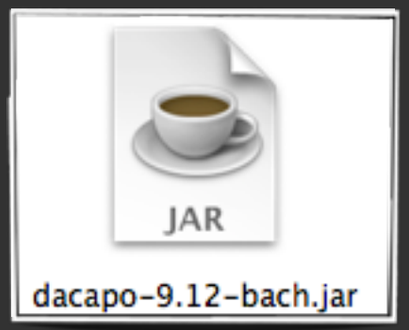
```
$ find out -name \*.class
out/com/apple/java/BackwardsCompatibility.class
out/java/io/BufferedInputStream.class
out/java/io/BufferedOutputStream.class
out/java/io/BufferedReader.class
out/java/io/BufferedWriter.class
out/java/io/ByteArrayInputStream.class
out/java/io/ByteArrayOutputStream.class
out/java/io/Closeable.class
out/java/io/DataInput.class
...
```

Why dump classes?

Why dump classes?

Static analysis needs to know the code

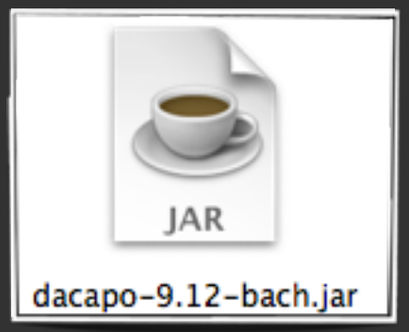
Why dump classes?



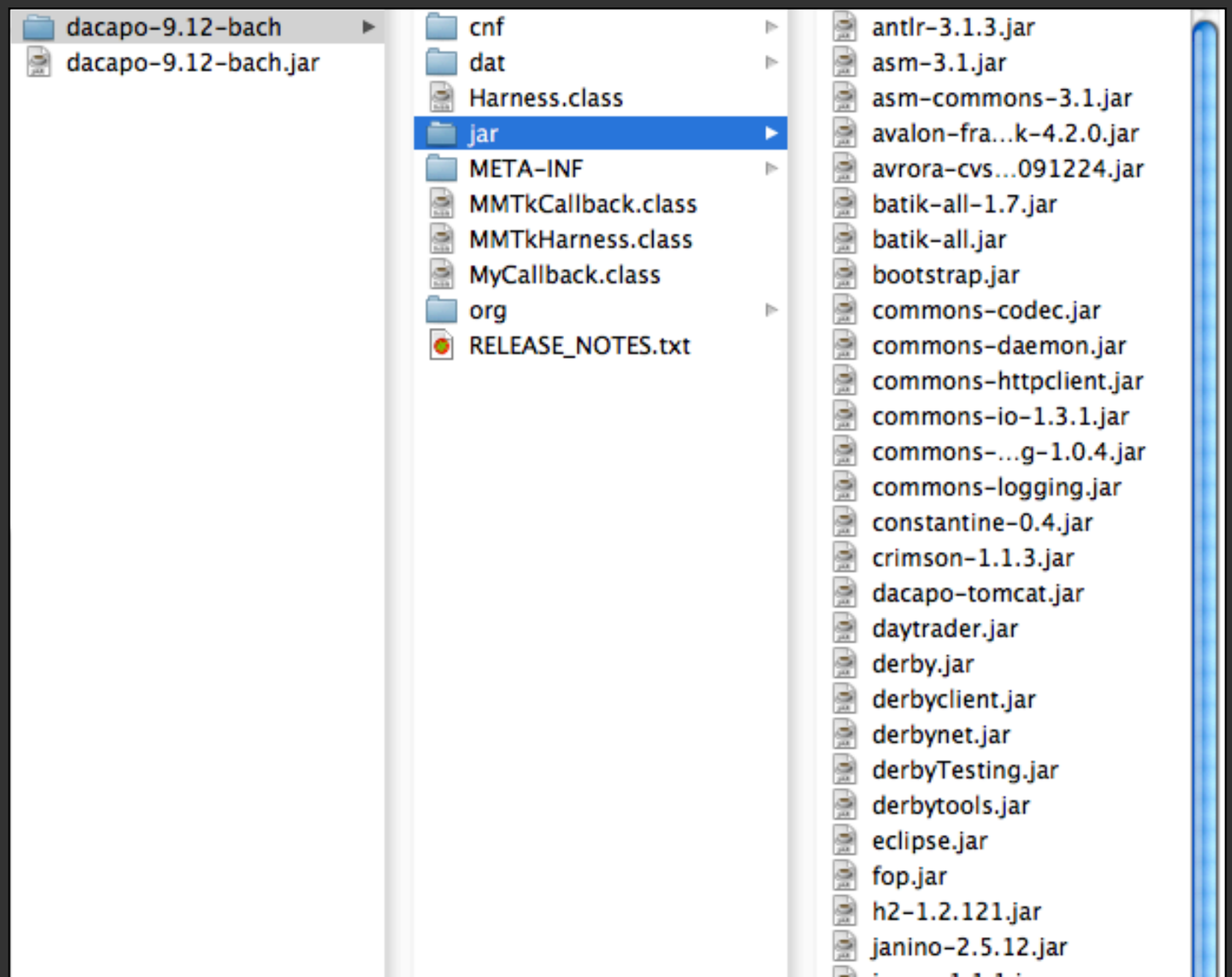

```
$ java -jar dacapo-9.12-bach.jar fop
===== DaCapo 9.12 fop starting =====
===== DaCapo 9.12 fop PASSED in 2503 msec =====
```

Static analysis needs to know the code

Why dump classes?

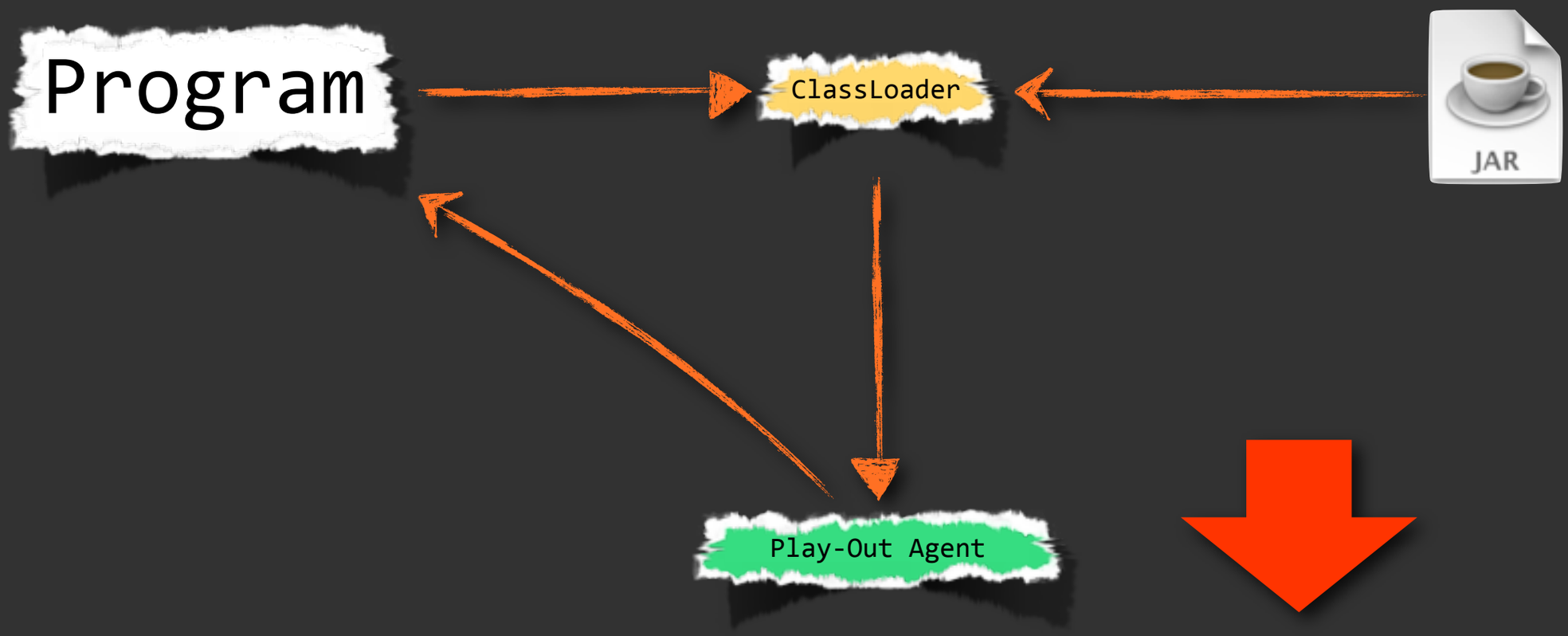


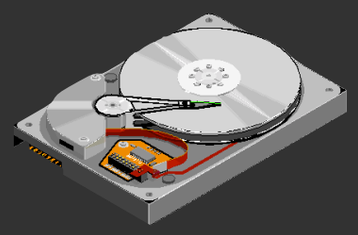

```
$ java -jar dacapo-9.12-bach.jar fop
===== DaCapo 9.12 fop starting =====
===== DaCapo 9.12 fop PASSED in 2503 msec =====
```



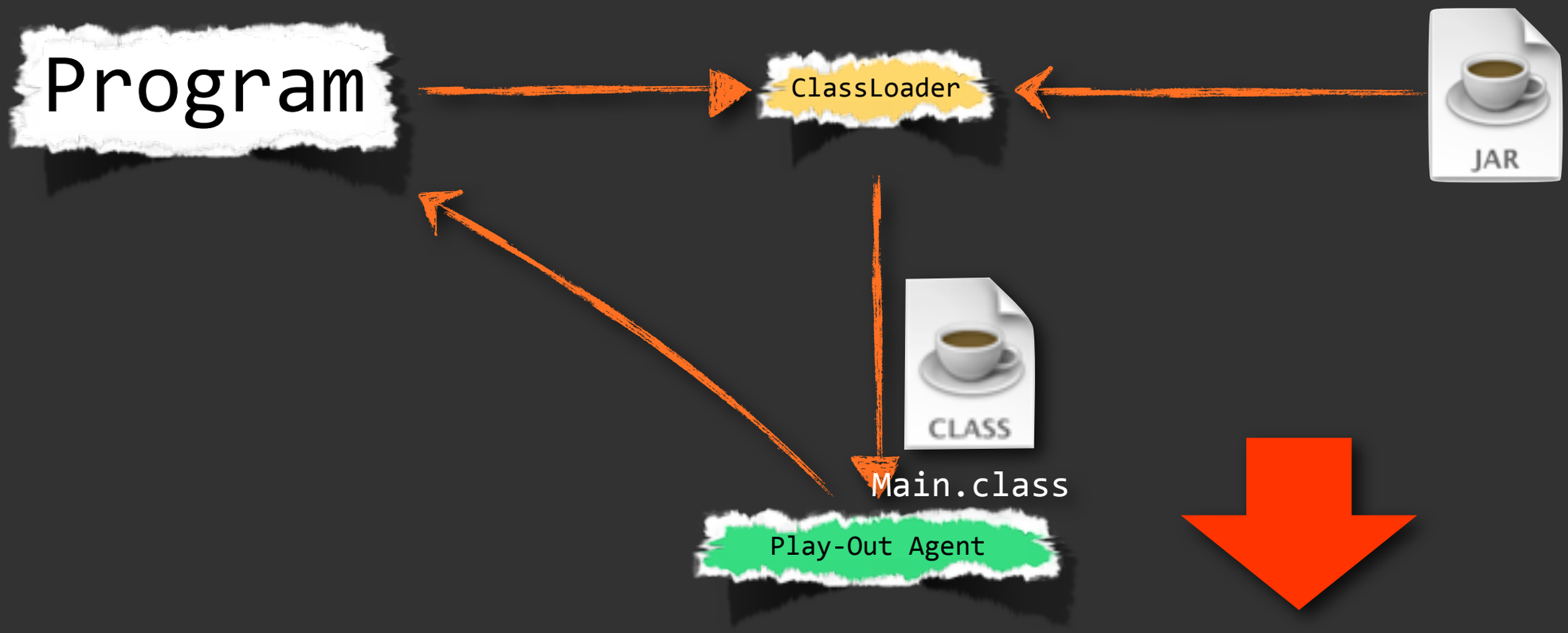
Static analysis needs to know the code

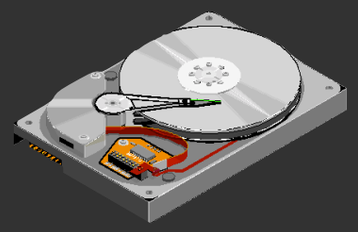
Play-out Agent





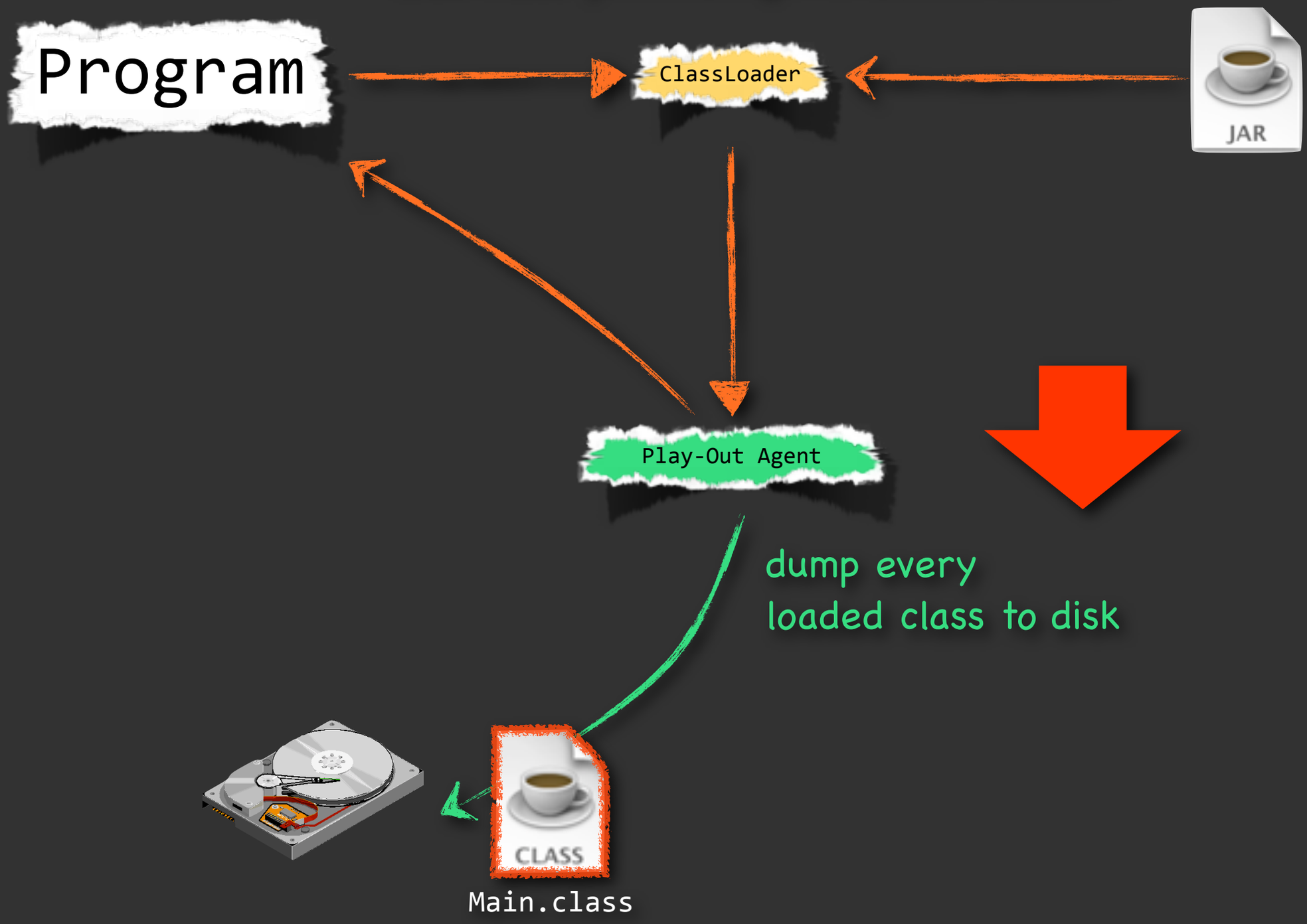
Play-out Agent





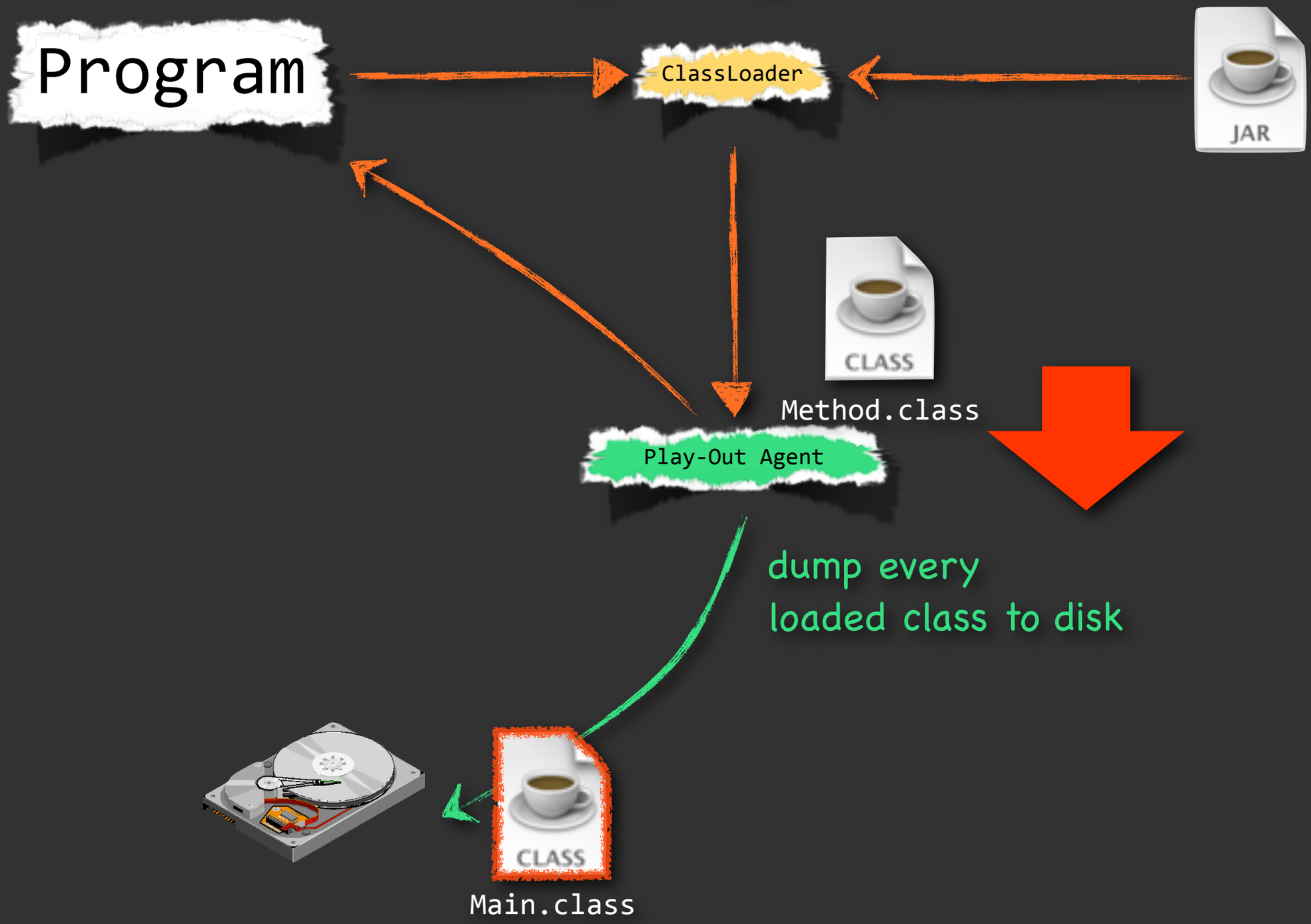
Play-out Agent

load class java.lang.reflect.Method



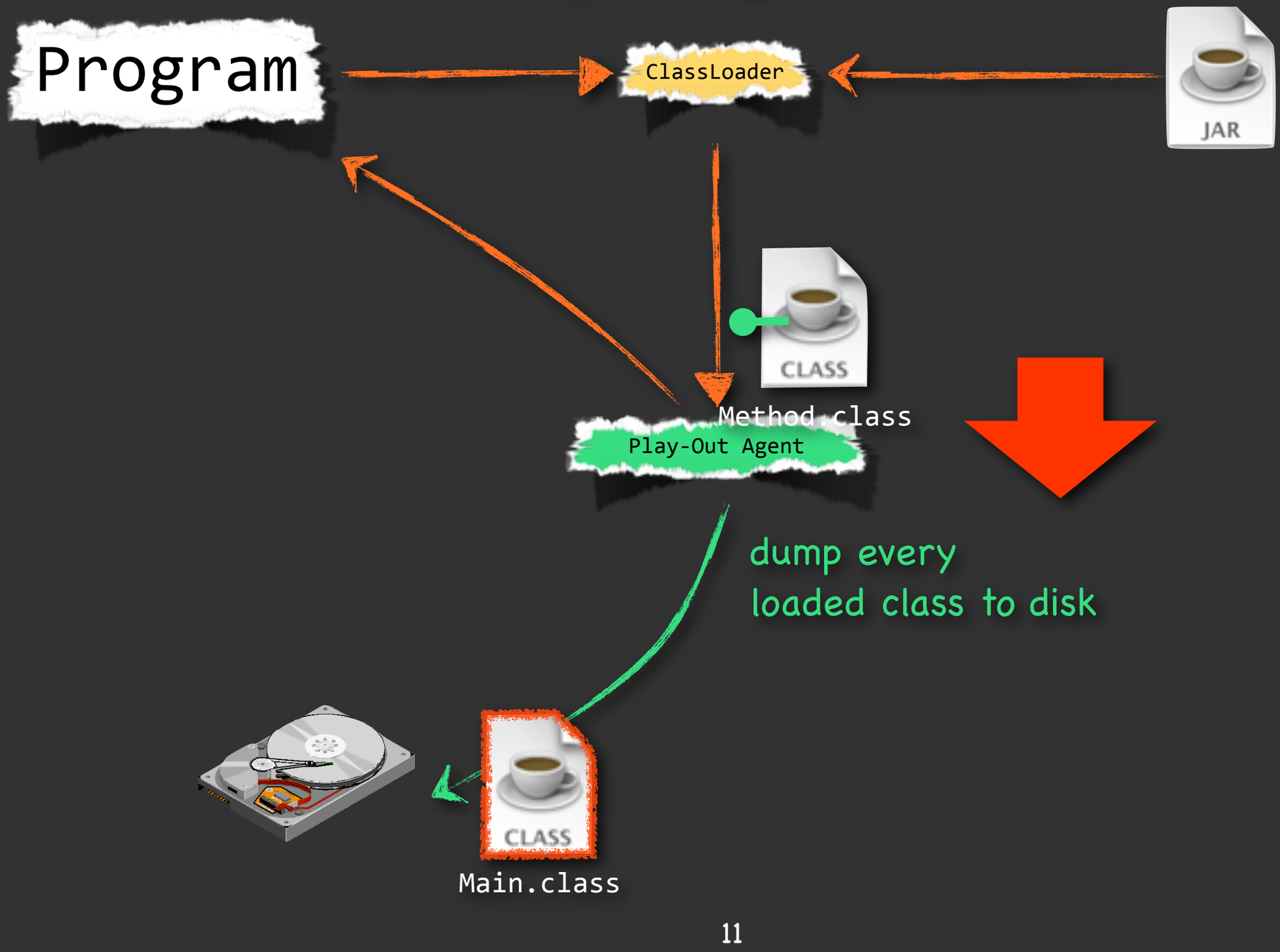
Play-out Agent

load class java.lang.reflect.Method



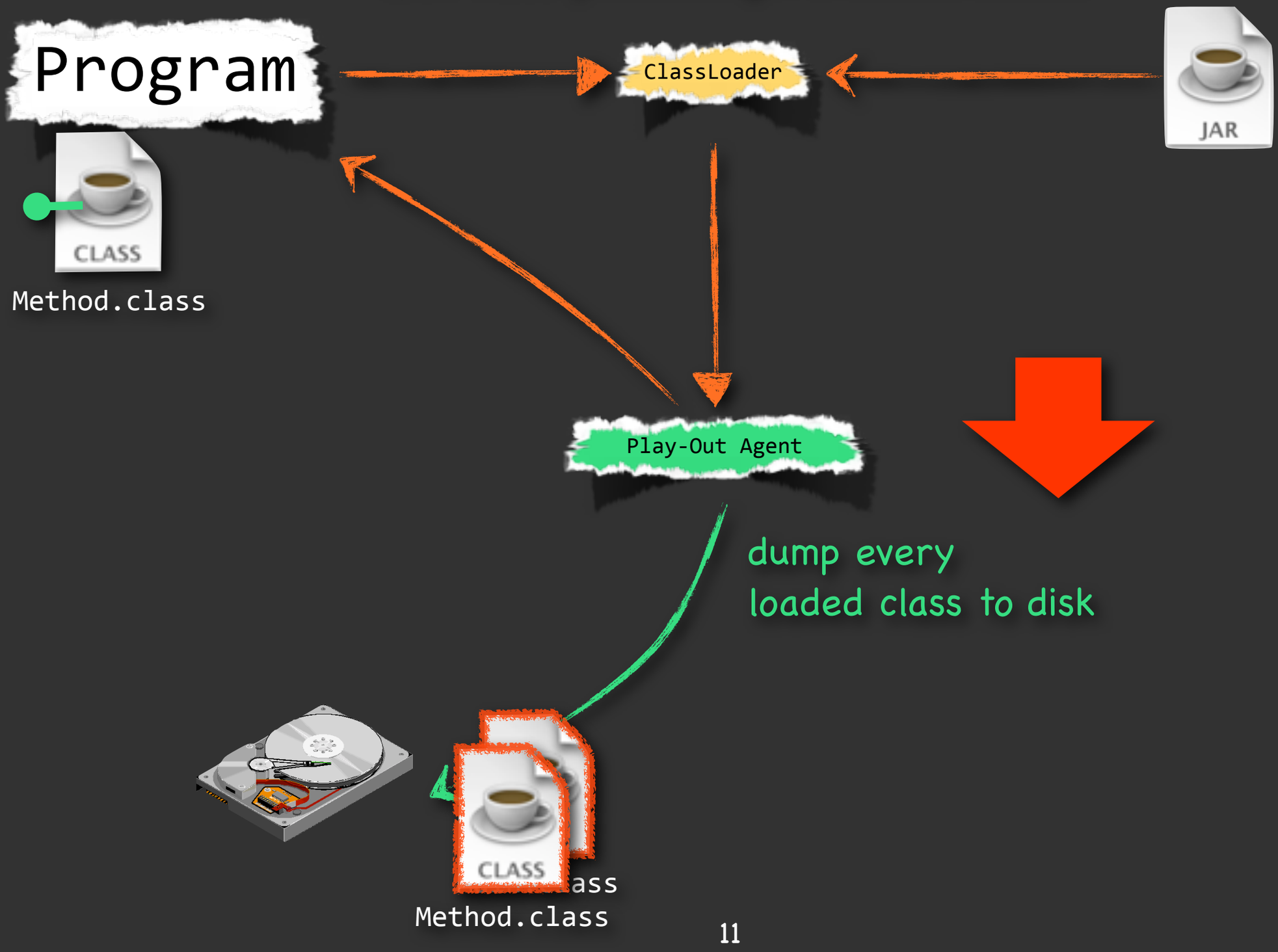
Play-out Agent

load class java.lang.reflect.Method



Play-out Agent

load class java.lang.reflect.Method



Play-out Agent

load class java.lang.reflect.Method

Program

ClassLoader

JAR

CLASS

Method.class

call to
Method.invoke

Play-Out Agent

dump every
loaded class to disk

CLASS

ass

Method.class

Play-out Agent

load class java.lang.reflect.Method

Program

ClassLoader

JAR

CLASS

Method.class

call to
Method.invoke

Play-Out Agent

dump every
loaded class to disk

LOG

CLASS

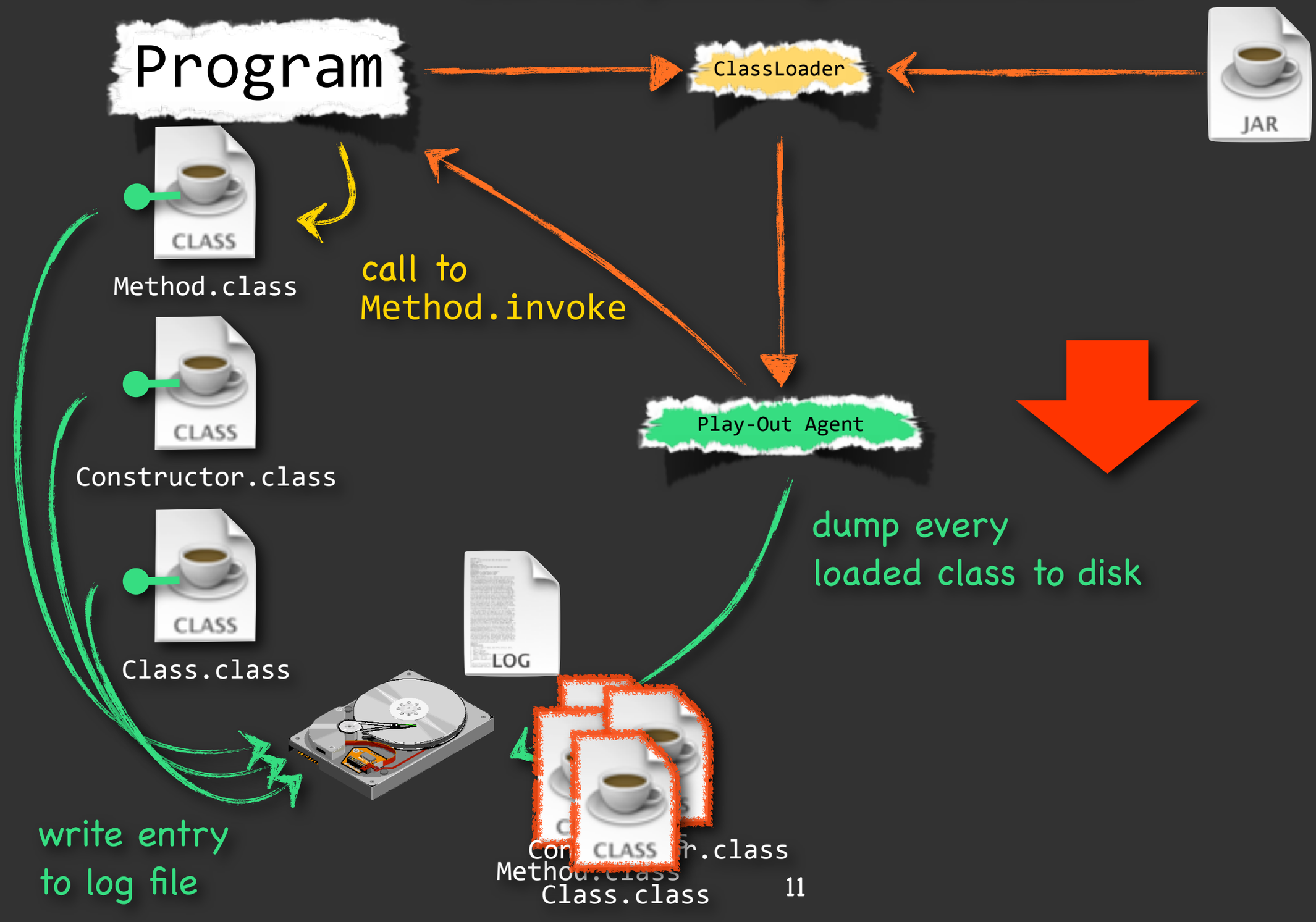
ass

Method.class

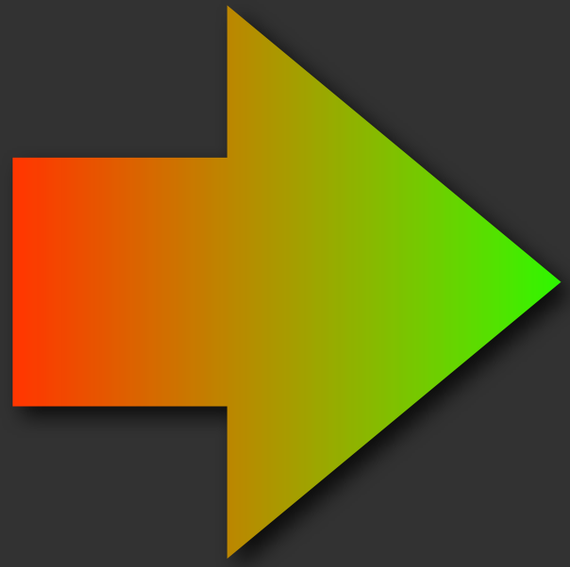
write entry
to log file

Play-out Agent

load class java.lang.reflect.Method

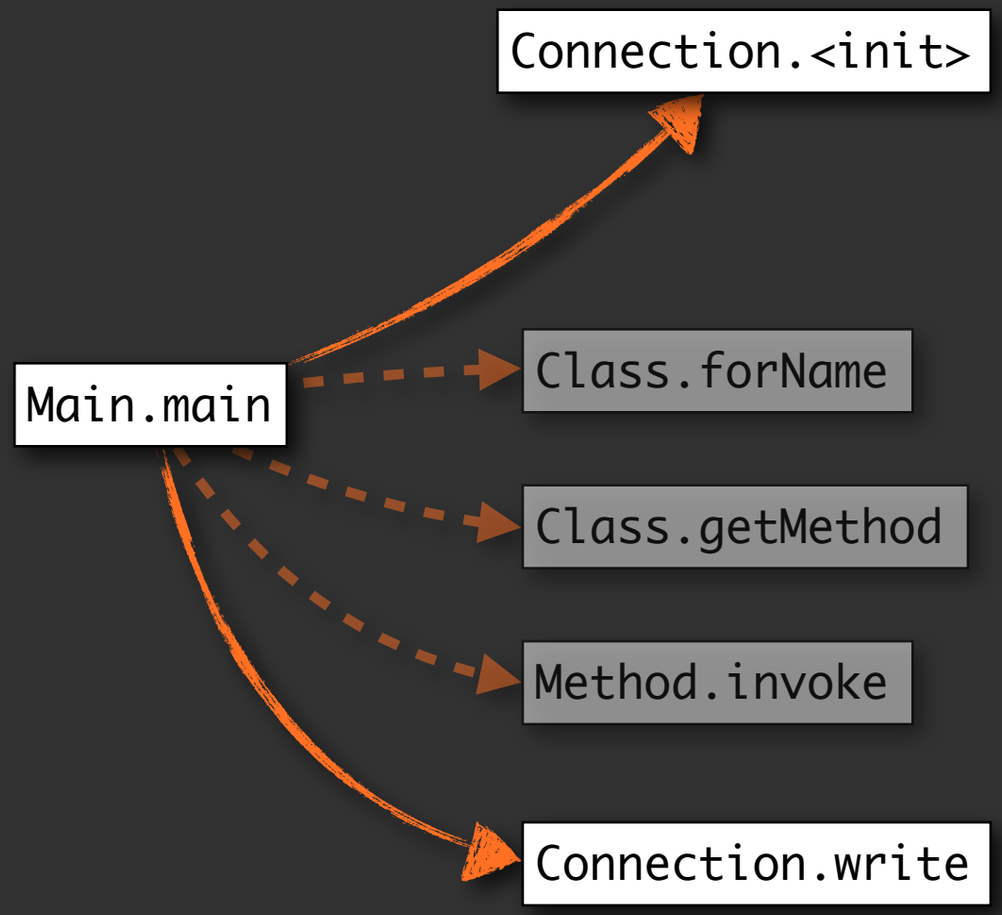


Step 2: Boosting



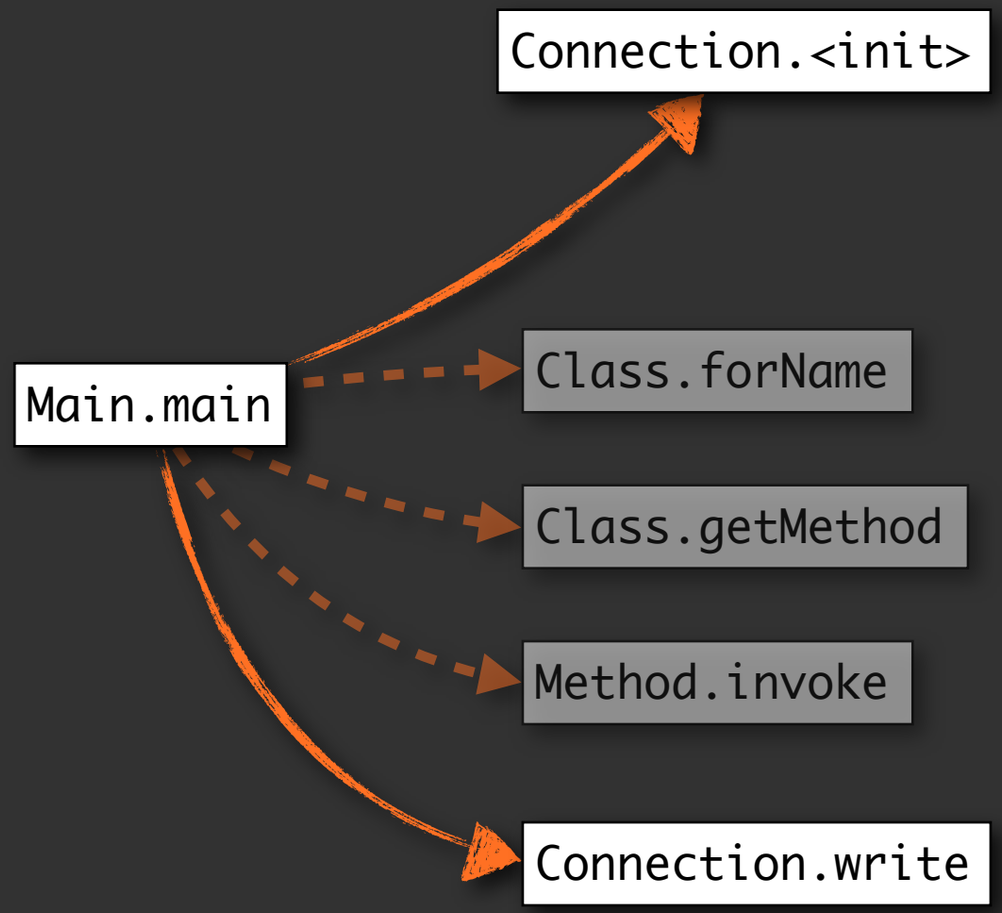
Booster transforms program into statically analyzable version

```
public class Main {

   public static void main(String[] args) throws Exception {
      Connection c = new Connection();

      Class cl = Class.forName(args[0]);

      Method m = cl.getMethod("do"+"Evil", Connection.class);

      m.invoke(null,c);

      c.write("ohoh...");
    }

public class Evil {

 public static void doEvil(Connection c) {
      c.close();
 }

}
```



Booster transforms program into statically analyzable version

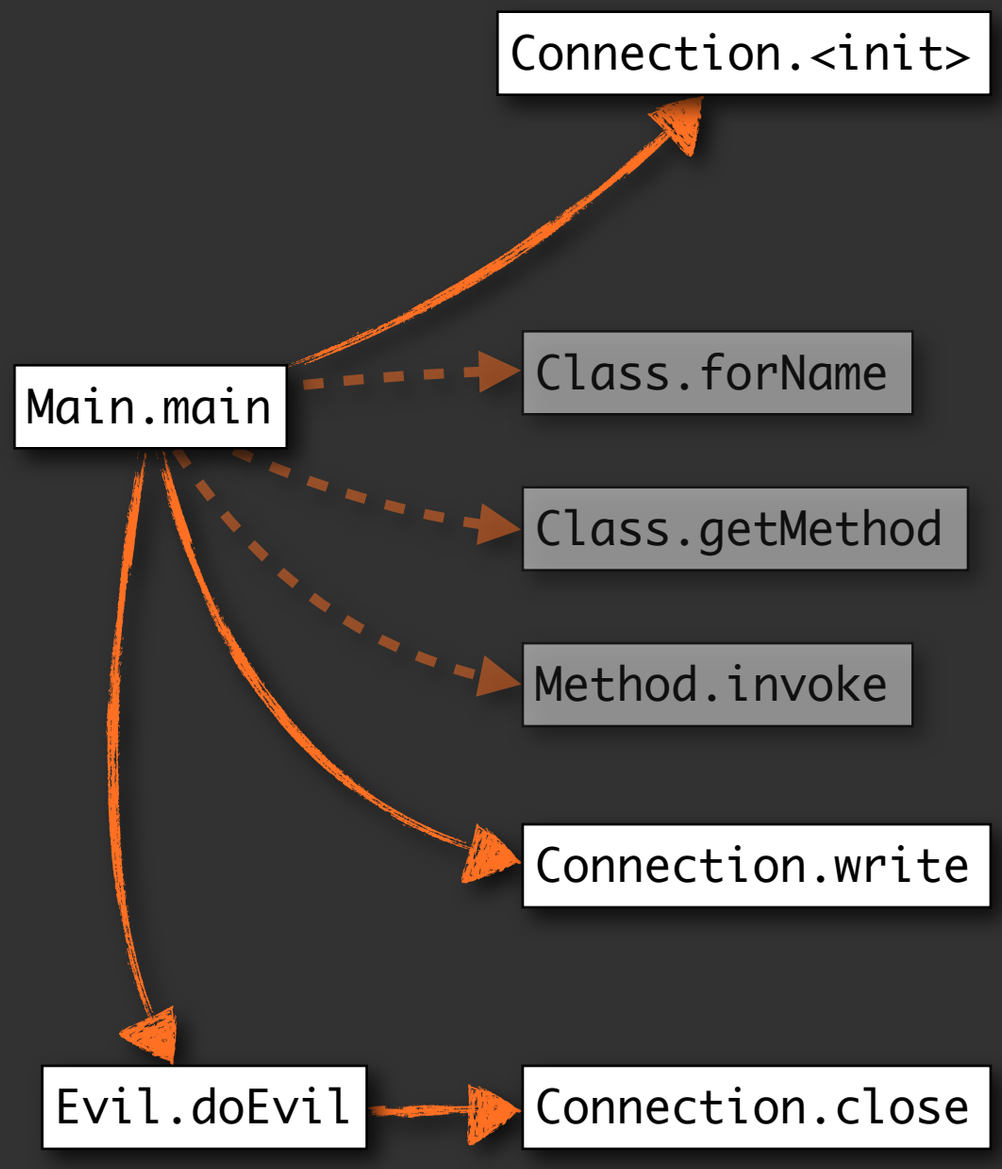
```
public class Main {

   public static void main(String[] args) throws Exception {
      Connection c = new Connection();

      Class cl = Class.forName(args[0]);

      Method m = cl.getMethod("do"+"Evil", Connection.class);

      m.invoke(null,c);

      c.write("ohoh...");
    }

public class Evil {

 public static void doEvil(Connection c) {
      c.close();
 }

}
```




```
$ cat out/refl.log
Class.forName;Evil;Main.main;6
Method.invoke;<Evil: void doEvil(Connection)>;Main.main;8;
```

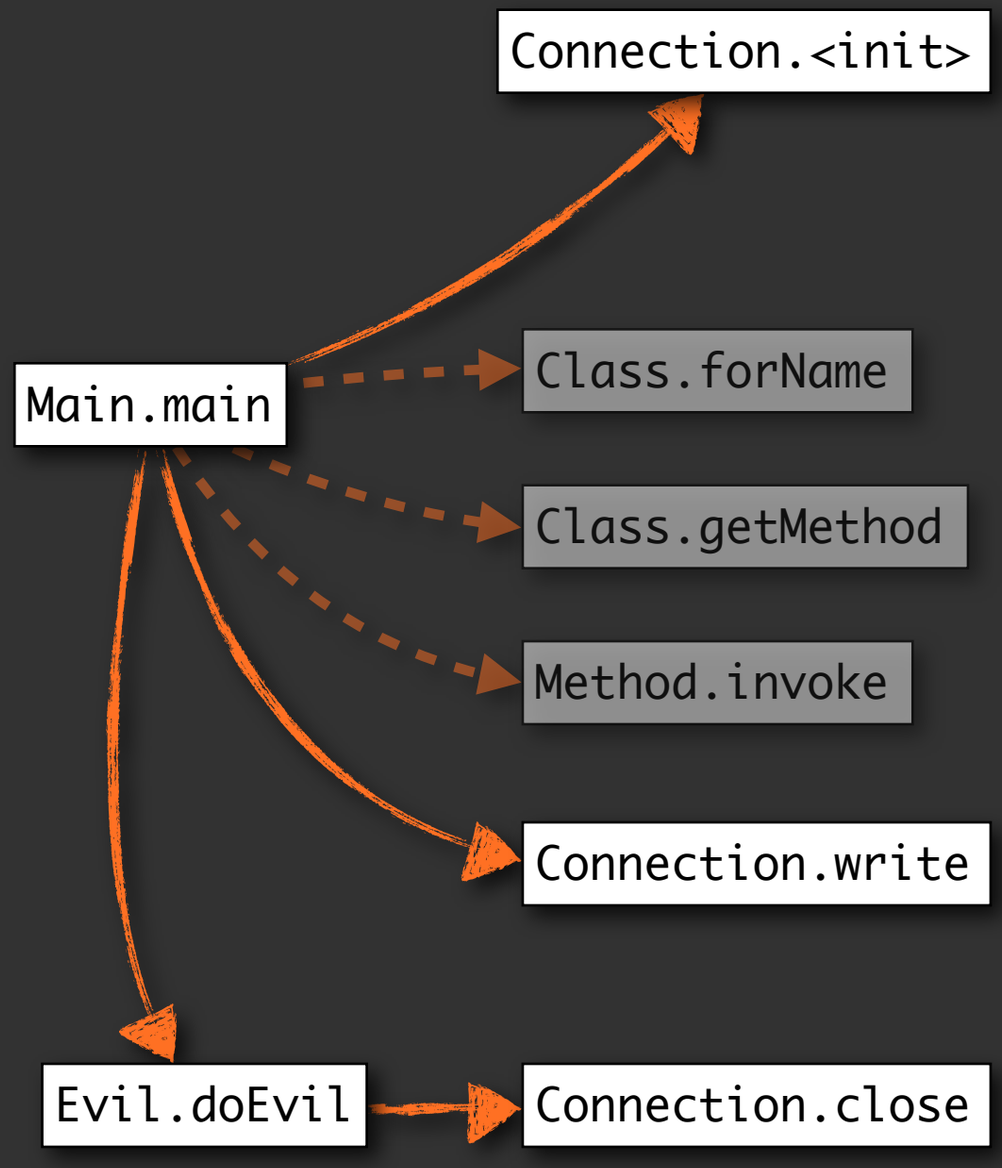
Booster transforms program into statically analyzable version

```
public class Main {

   public static void main(String[] args) throws Exception {
      Connection c = new Connection();

      Class cl;
      if(args[0].equals("Evil"))
          cl = Evil.class;
      else {
          TamiFlexRT.warnClassForName(args[0]);
          cl = Class.forName(args[0]);
      }

      Method m = cl.getMethod("do"+"Evil", Connection.class);

      if(m.getDeclaringClass().getName().equals("Evil") &&
        m.getName().equals("doEvil"))
          Evil.doEvil(c);
      else {
          TamiFlexRT.warnMethodInvoke(m);
          m.invoke(null,c);
      }

      c.write("ohoh...");
   }

   ...

}
```



Booster transforms program into statically analyzable version

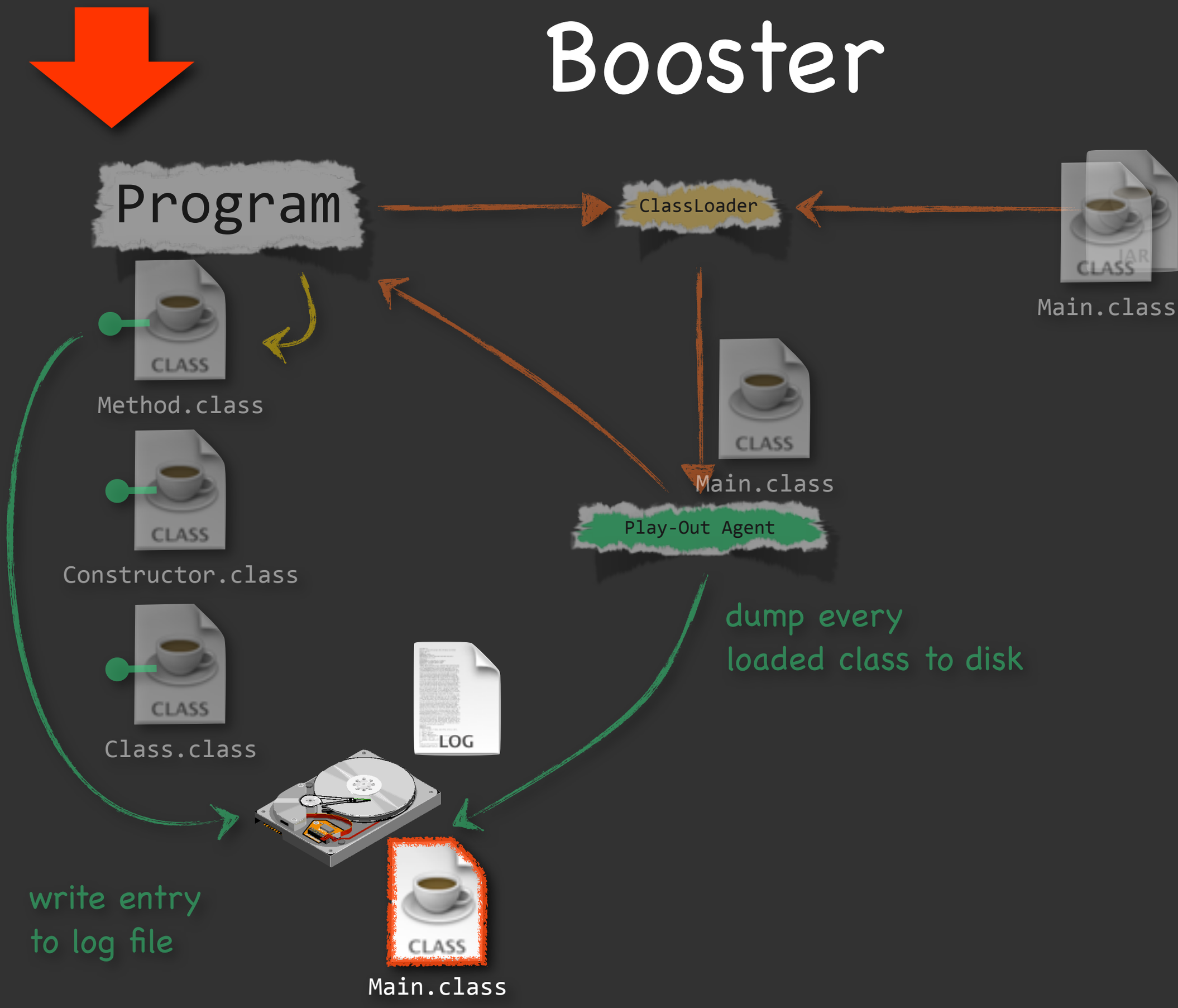
```
public class Main {

  public static void main(String[] args) throws Exception {
     Connection c = new Connection();

     Class cl;
     if(args[0].equals("Evil"))
         cl = Evil.class;
     else {
         TamiFlexRT.warnClassForName(args[0]);
         cl = Class.forName(args[0]);
     }

     Method m = cl.getMethod("do"+"Evil", Connection.class);

     if(m.getDeclaringClass().getName().equals("Evil") &&
       m.getName().equals("doEvil"))
         Evil.doEvil(c);
     else {
         TamiFlexRT.warnMethodInvoke(m);
         m.invoke(null,c);
     }

     c.write("ohoh...");
  }

  ...

}
```



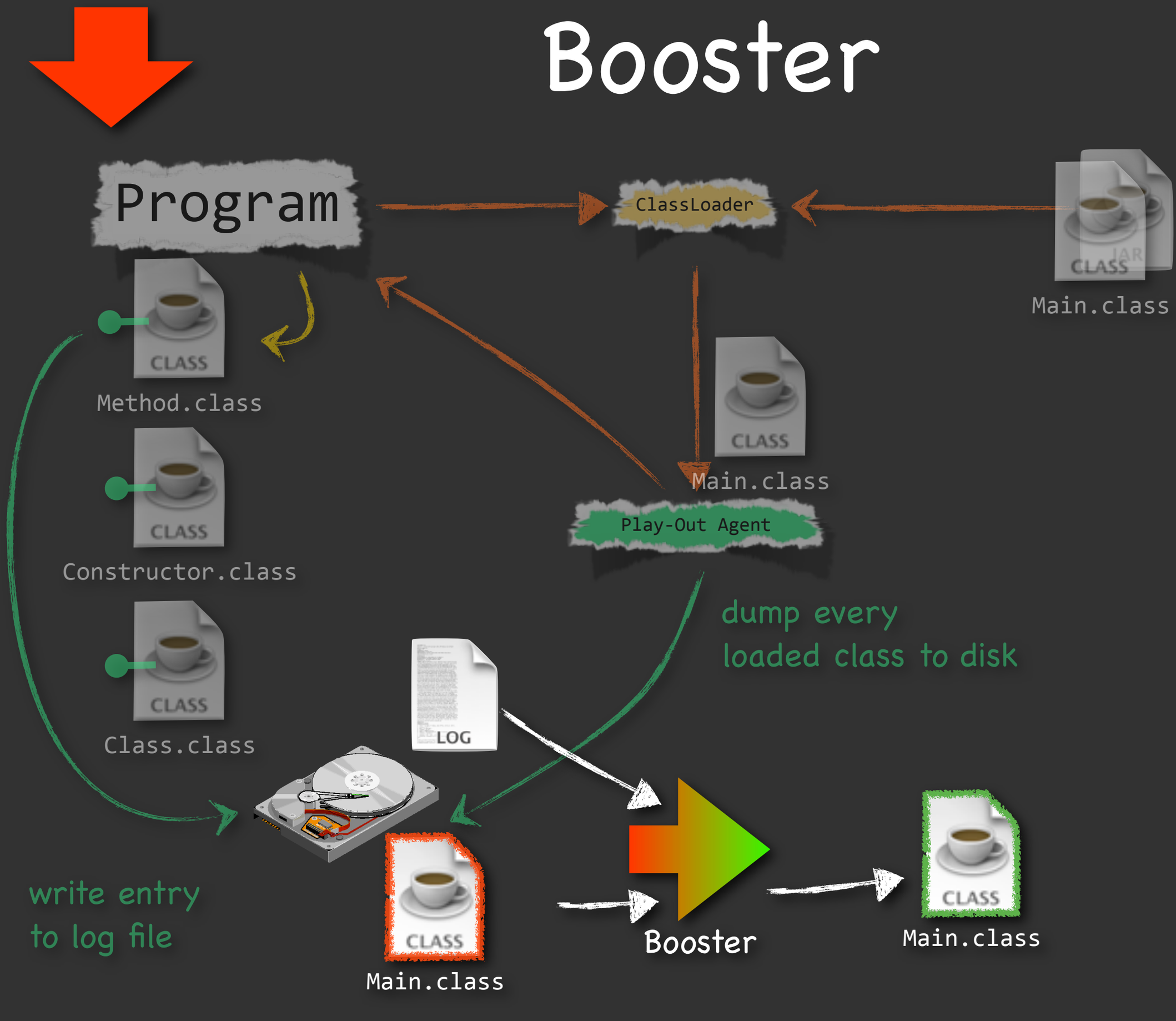
Booster

- Implemented on top of Soot
- Operates on 3-address code (Jimple) generated from bytecode
- Input: class files recorded by play-out agent
- Emits class files of "boosted" program
- Some nitty-gritty details with respect to the use of non-standard class loaders (see paper)

Booster


Program
ClassLoader
CLASS
JAR
Main.class
CLASS
Method.class
CLASS
Main.class
Play-Out Agent
CLASS
Constructor.class
CLASS
Class.class
LOG
dump every
loaded class to disk
write entry
to log file
CLASS
Main.class

Booster


Program
ClassLoader
Main.class
CLASS
JAR
Method.class
Constructor.class
Class.class
Main.class
Play-Out Agent
dump every
loaded class to disk
LOG
write entry
to log file
Main.class
Booster
Main.class

Booster

Program

ClassLoader

Main.class

Method.class

Main.class

Play-Out Agent

Main.class

Constructor.class

Class.class

dump every
loaded class to disk

LOG

write entry
to log file

Main.class

Booster

Main.class

Static Analysis

Booster

Program

ClassLoader

Main.class

Method.class

Main.class

Play-Out Agent

Constructor.class

Class.class

dump every
loaded class to disk

LOG

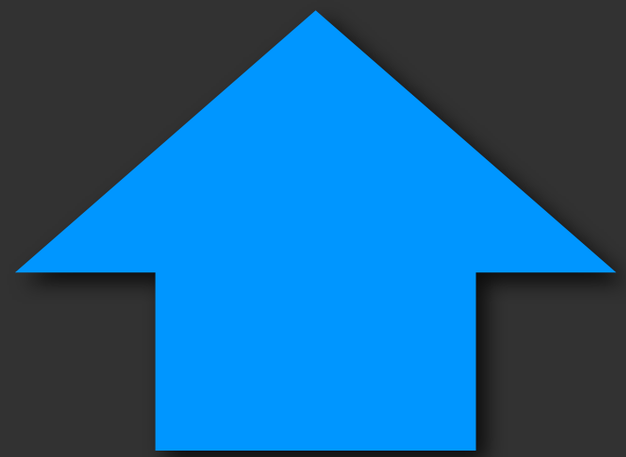
write entry
to log file

Main.class

Booster

Main.class

Static Analysis



Step 3: Play-In

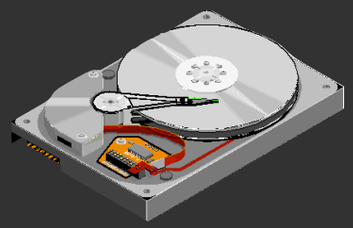
Play-in Agent

load from somewhere, or generate

load class Main

Program

Custom ClassLoader



Main.class

Play-In Agent

Play-in Agent

load from somewhere, or generate

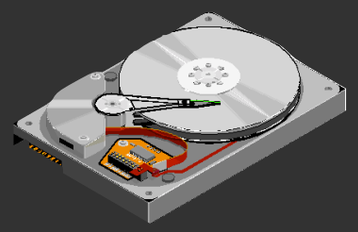
load class Main

Program

Custom ClassLoader

JAR

Play-In Agent



CLASS

Main.class

Play-in Agent

load from somewhere, or generate

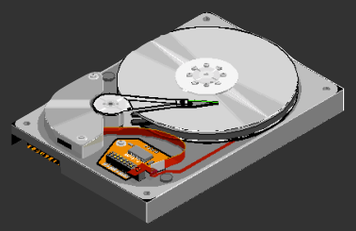
load class Main

Program

Custom ClassLoader

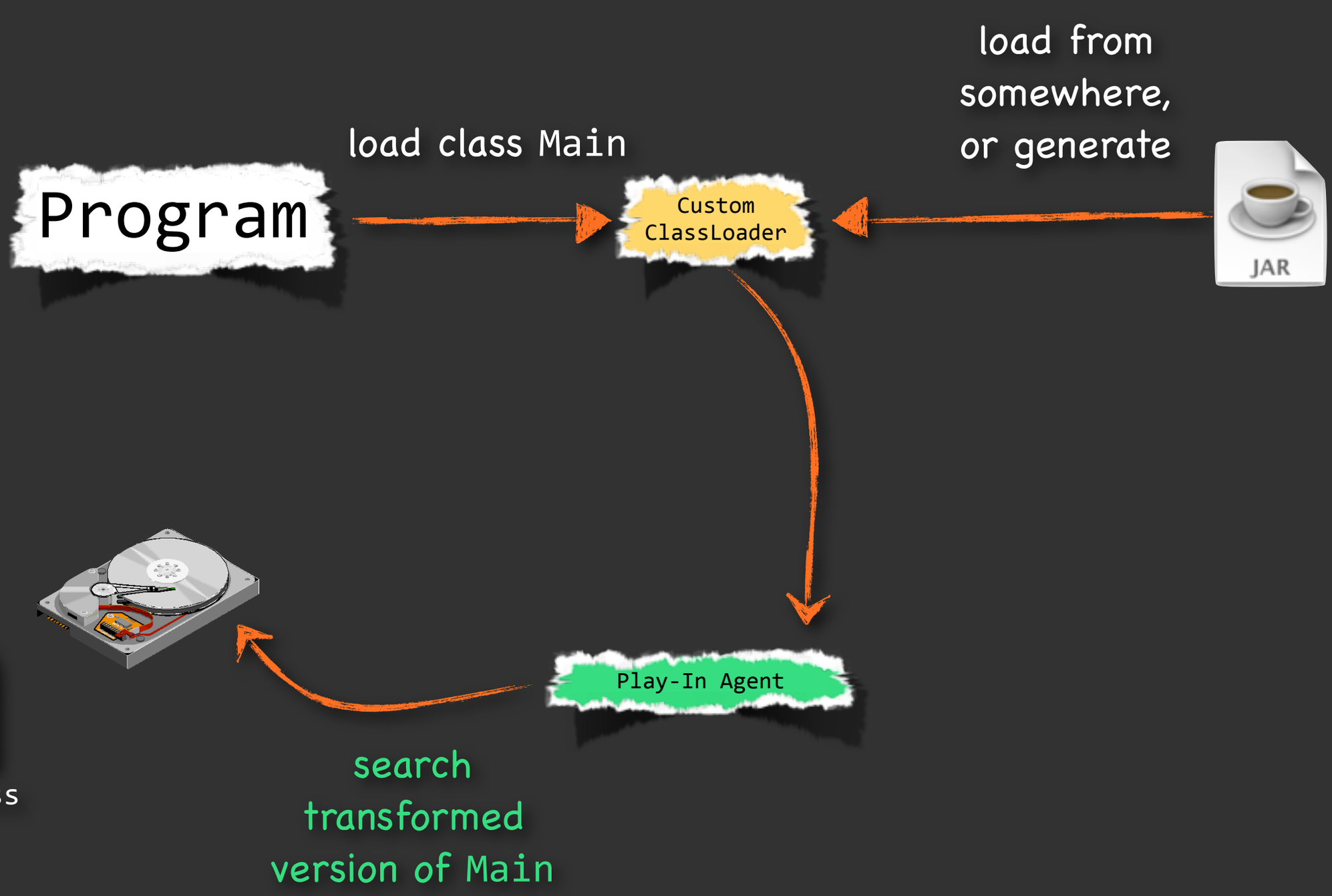
Main.class

Play-In Agent

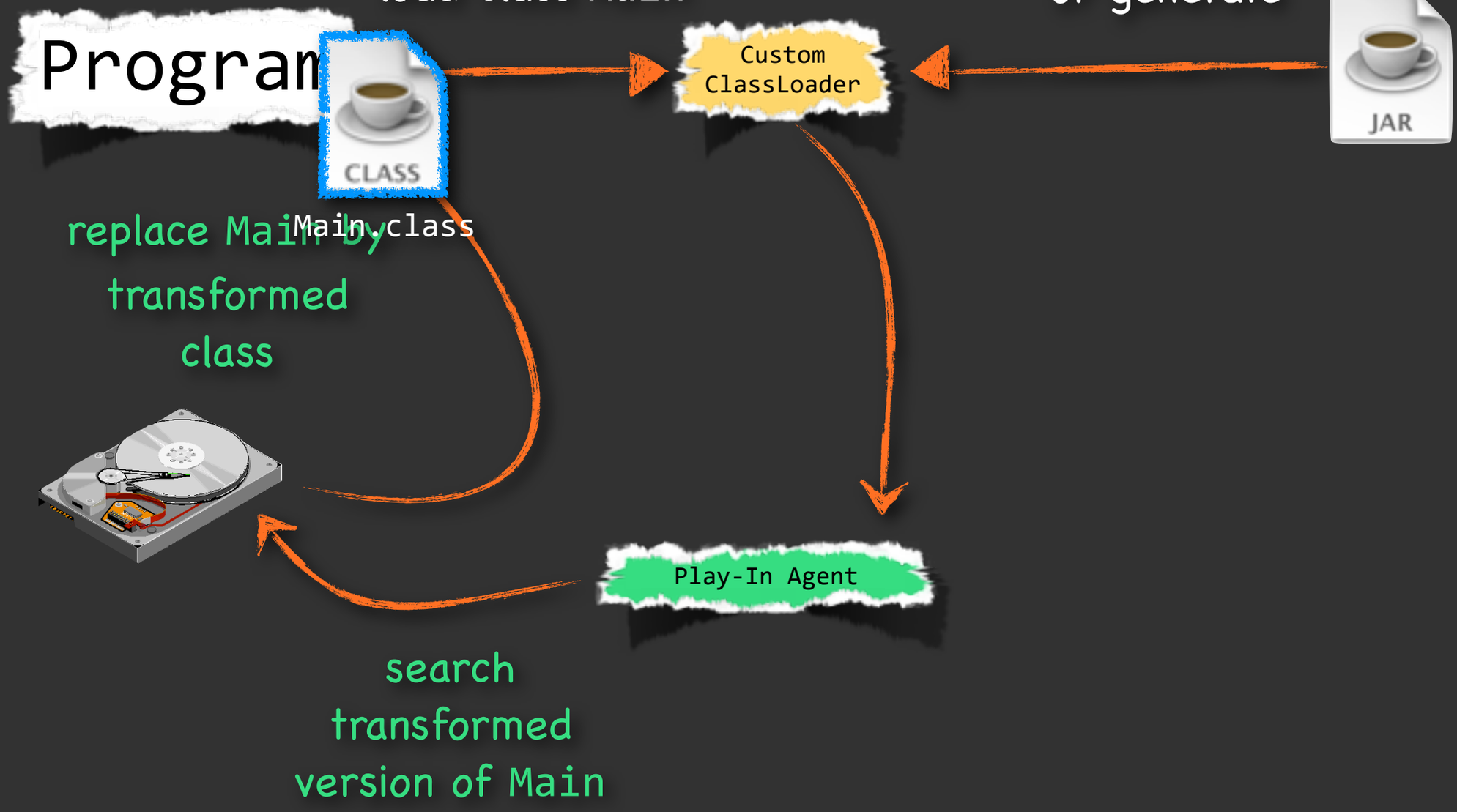


Main.class

Play-in Agent



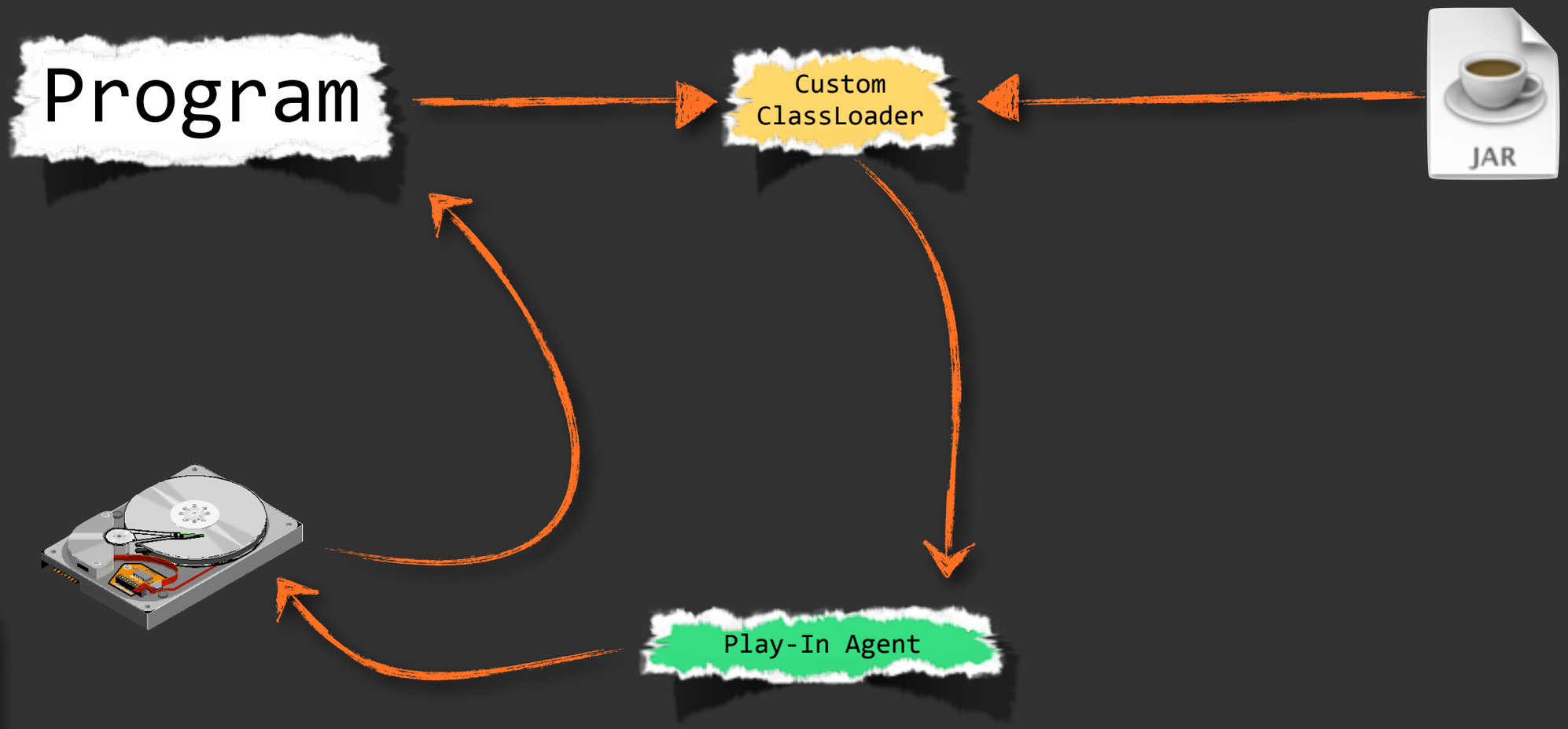
Play-in Agent



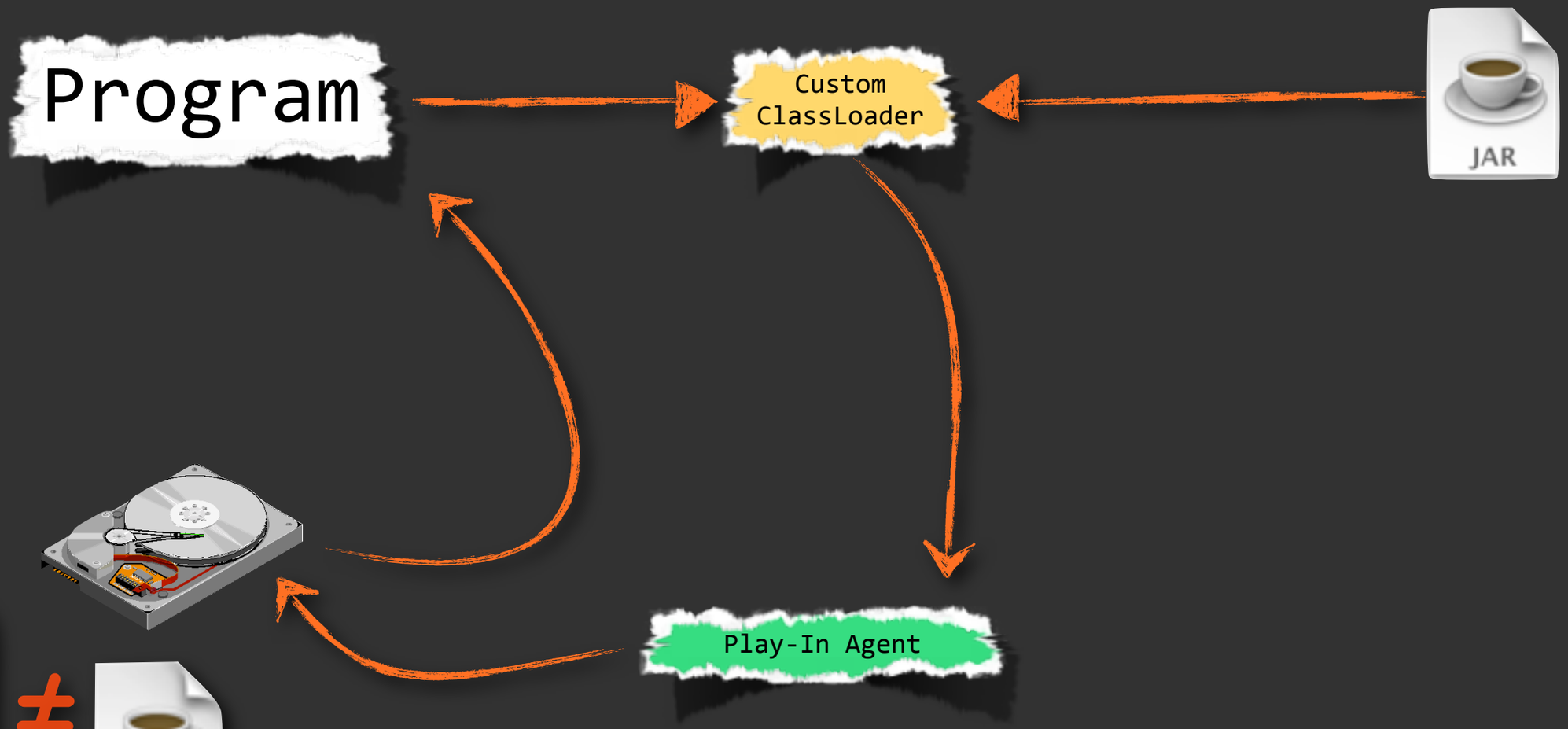
Just when
we thought
we were
done...

generated classes
may bear
randomized names!

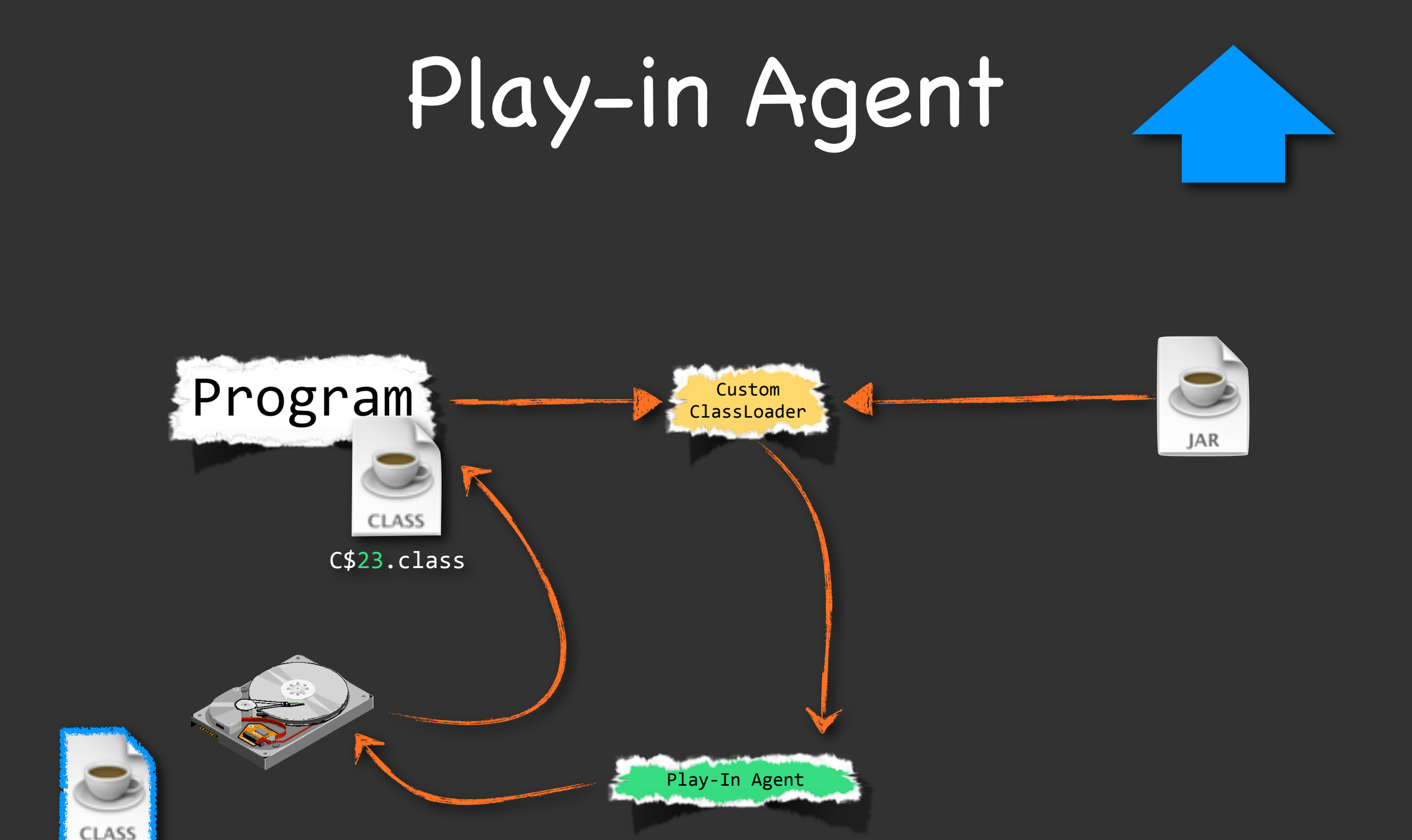
Play-in Agent



Play-in Agent

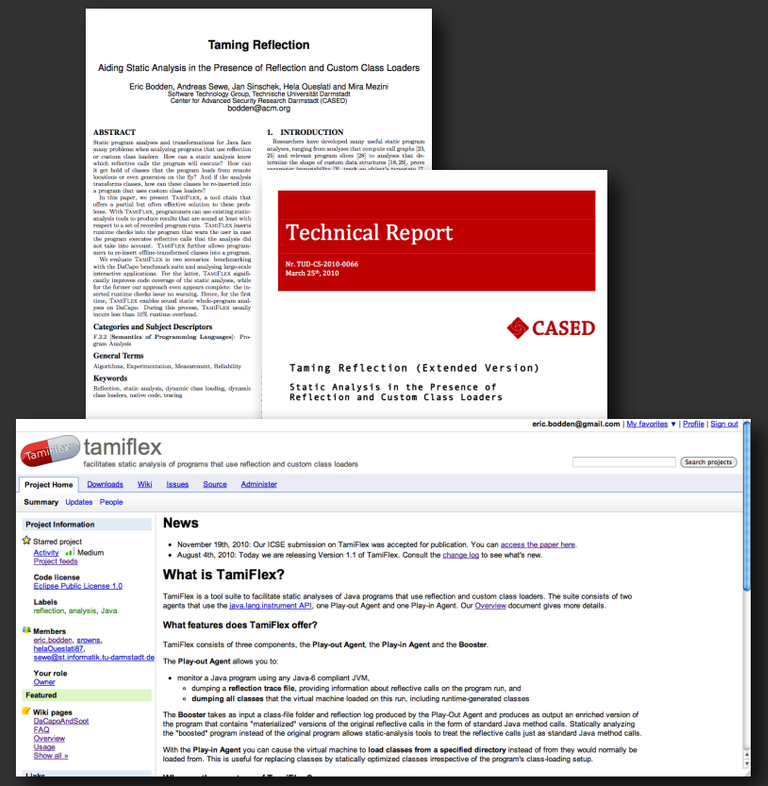


Play-in Agent



Solution

- Normalization: replace randomized name by normalized name
  - in the log, the binaries, everywhere...
- Quite involved: classes to normalize may be interdependent
- Details in the paper
- More details in our Tech Report
- Yet more details in the source



Evaluation

Evaluation

How can this be sound?

Evaluation

How can this be sound?

It's not!
But it's sound**er**!

Evaluation

How can this be sound?

It's not!
But it's sound**er**!
It's "representative"!

Correctness

static call graph ⊇ U{recorded call graph} ?

Correctness

static call graph ⊇ U{recorded call graph} ?

computed with Soot/Spark

recorded with JVMTI agent

Correctness

static call graph ⊇ U{recorded call graph} ?

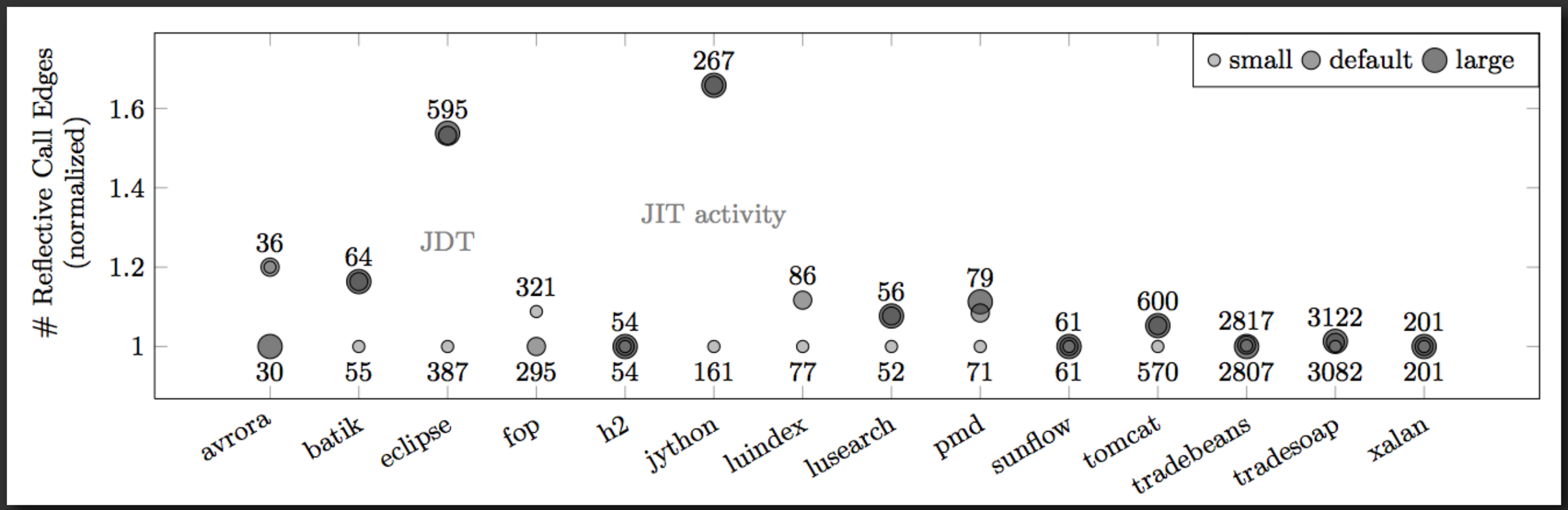
computed with Soot/Spark

recorded with JVMTI agent

Yes!

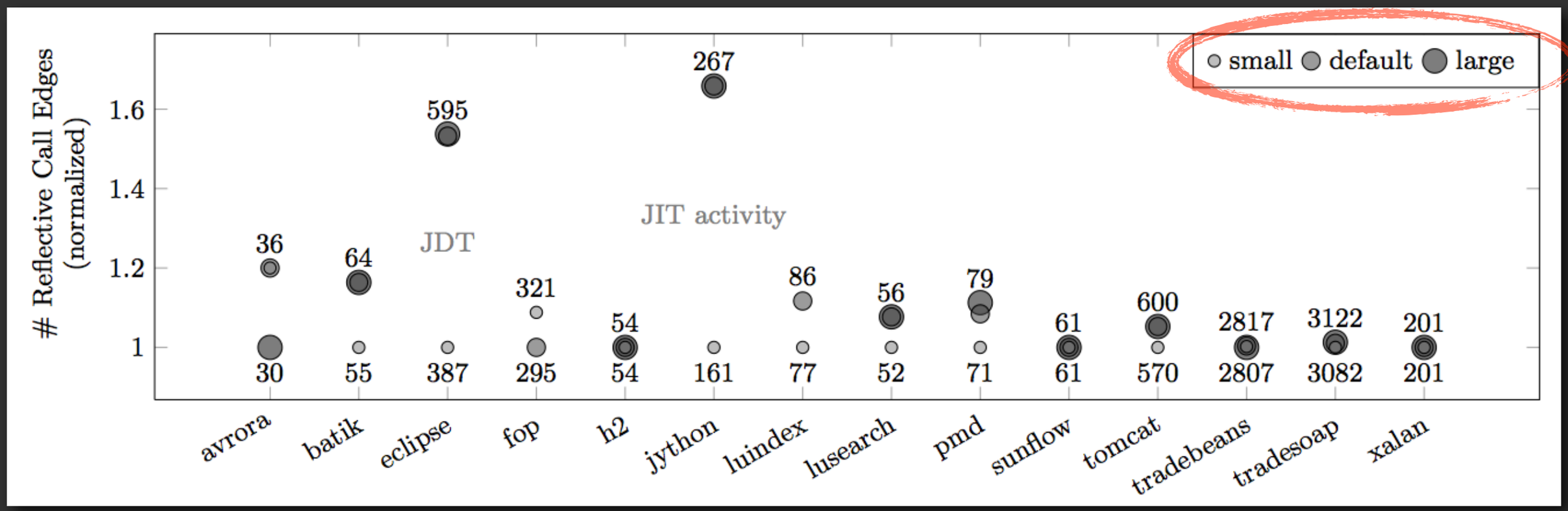
(call graphs are representative;
tested with Lhotak's Call-graph comparison tool ProBe)

Effectiveness: How much does log-file quality depend on program input?



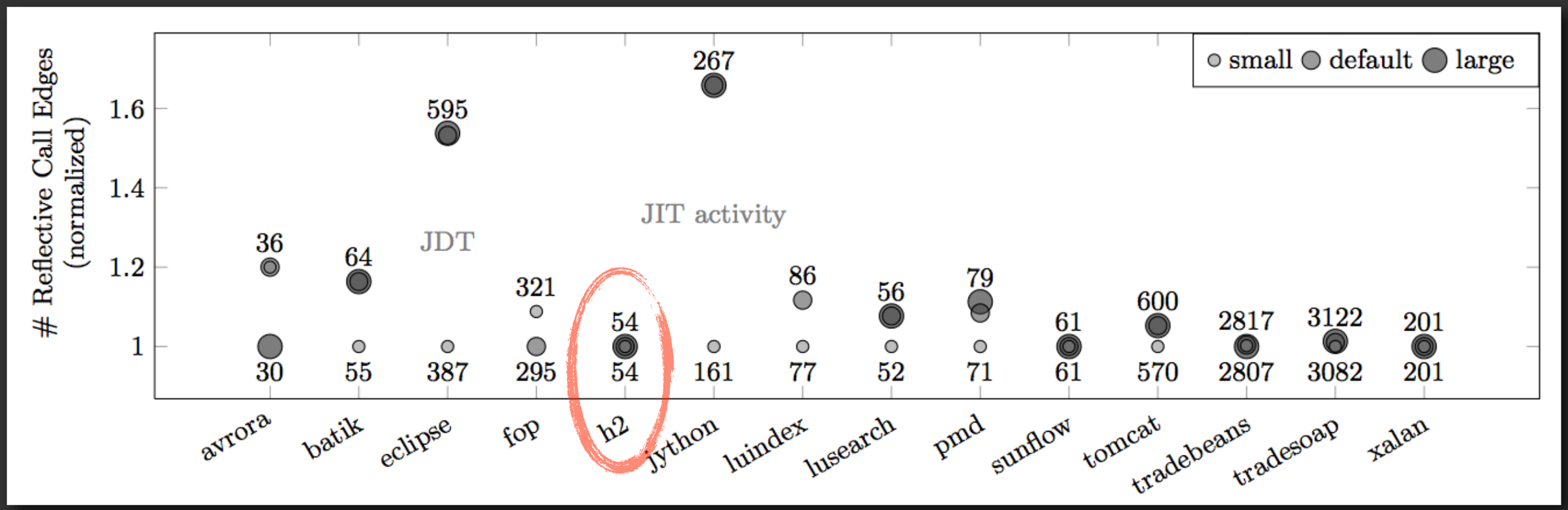
Dacapo "bach" release

Effectiveness: How much does log-file quality depend on program input?



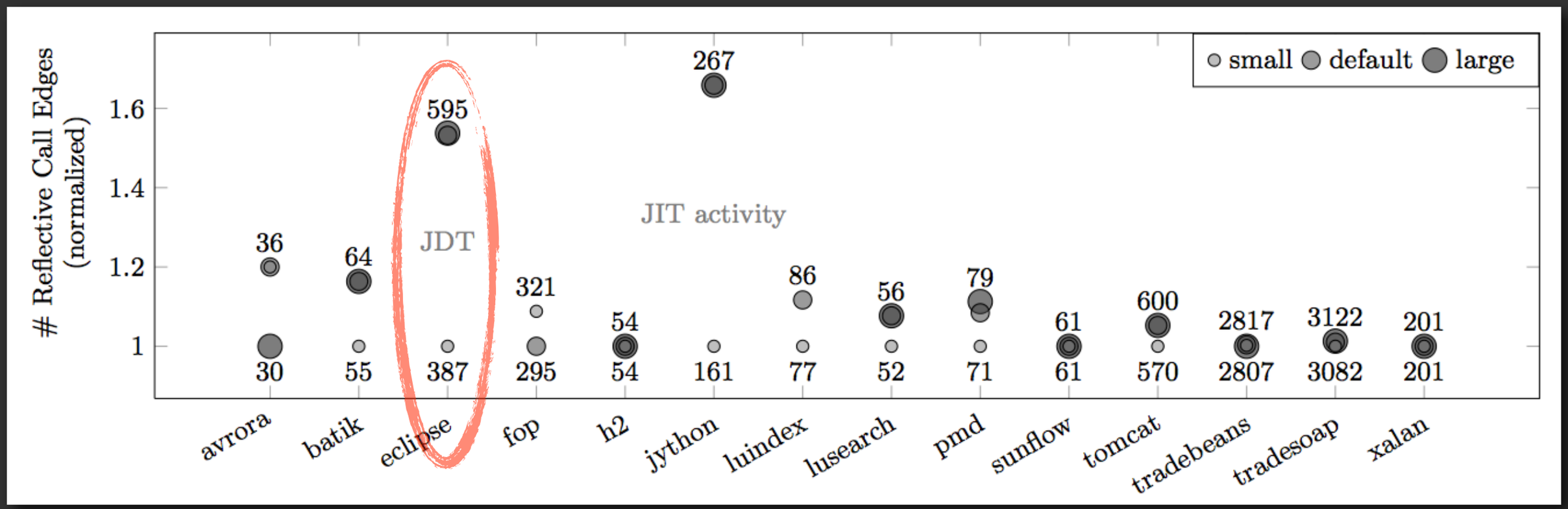
Dacapo "bach" release

Effectiveness: How much does log-file quality depend on program input?



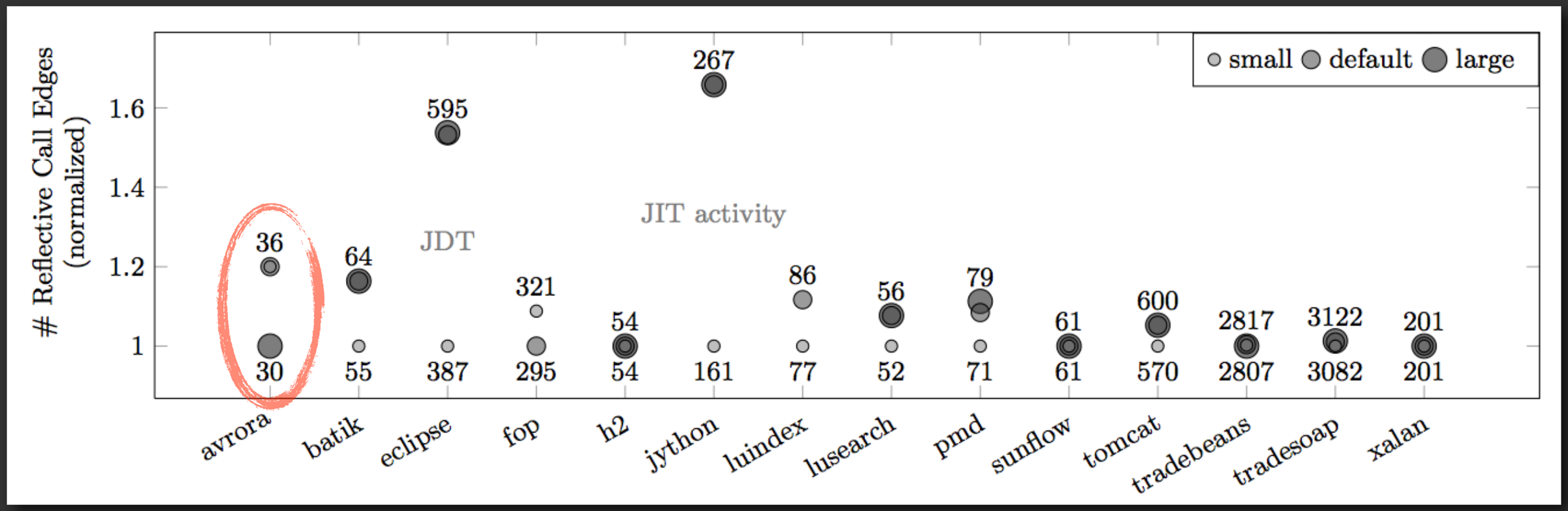
Dacapo "bach" release

Effectiveness: How much does log-file quality depend on program input?



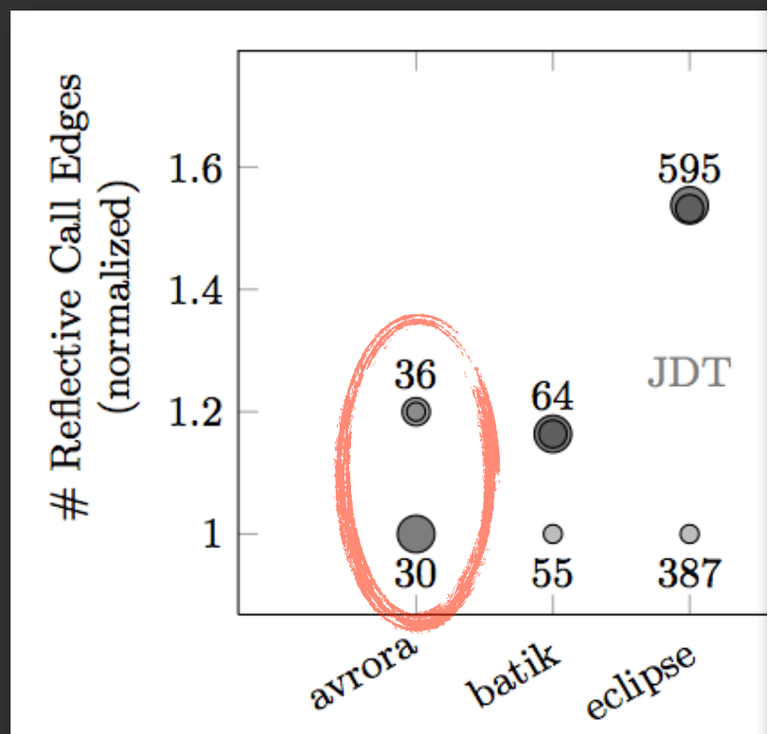
Dacapo "bach" release

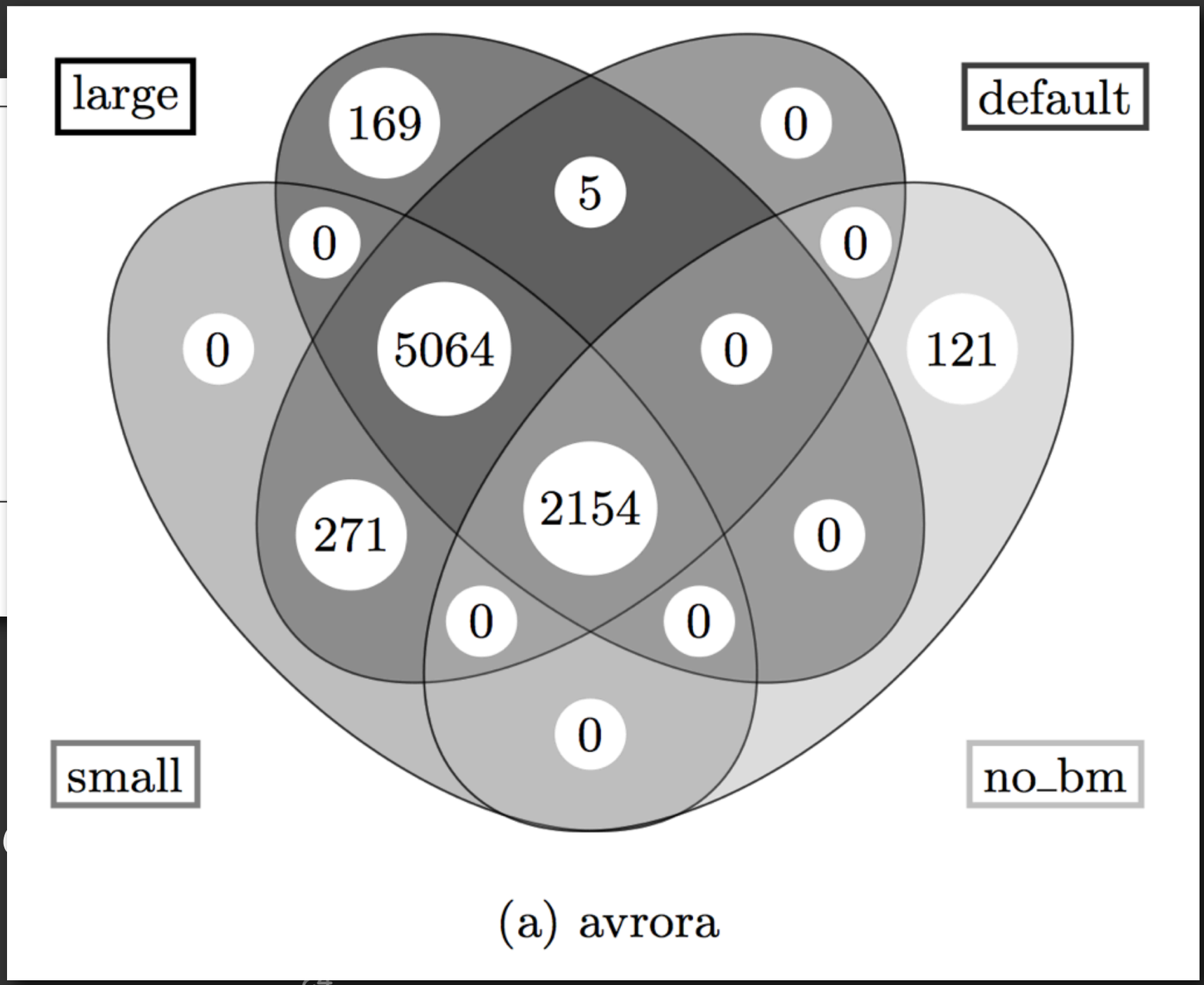
Effectiveness: How much does log-file quality depend on program input?



Dacapo "bach" release

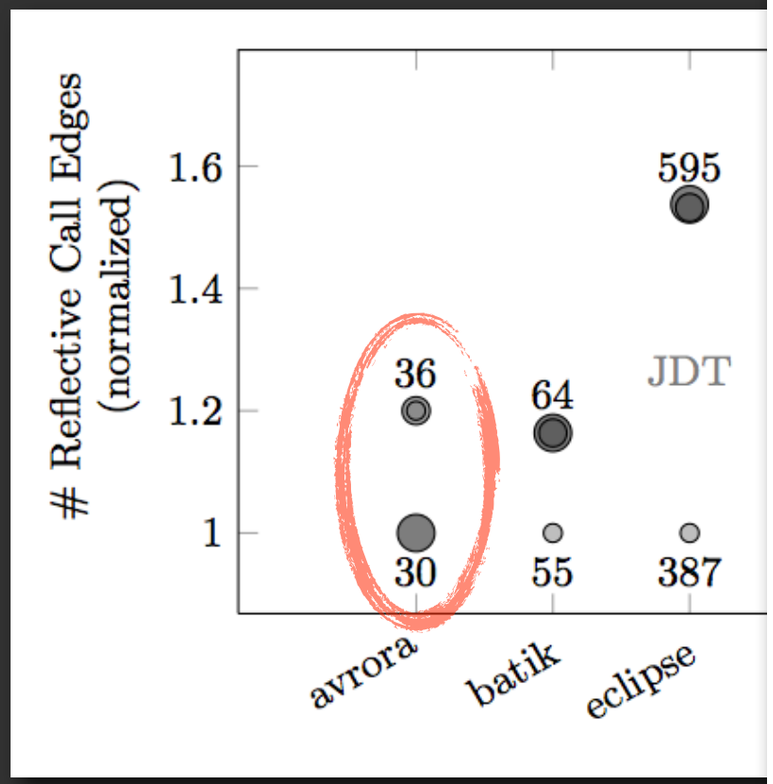
Effectiveness: How much does log-file quality depend on program input?

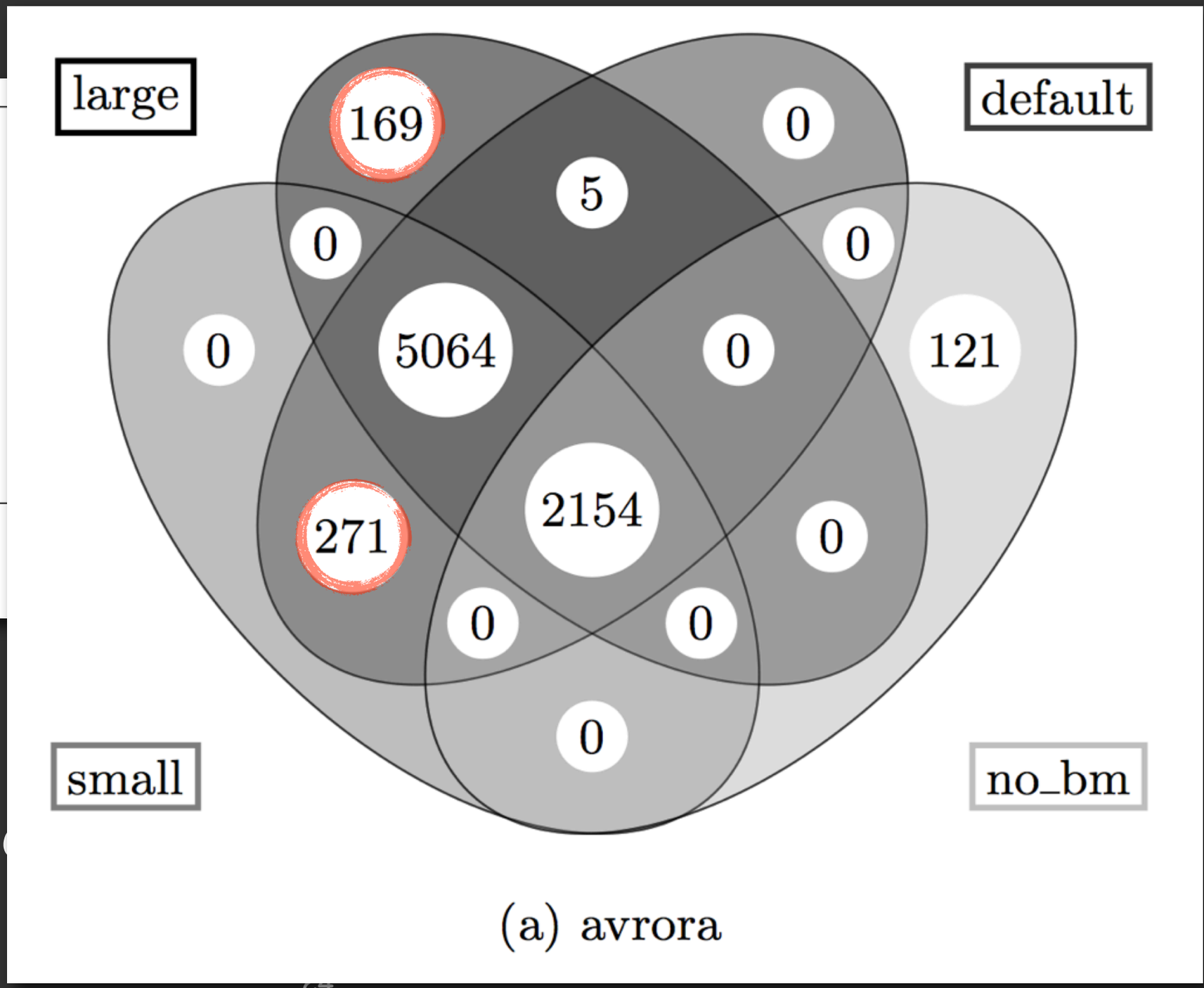




(a) avrora

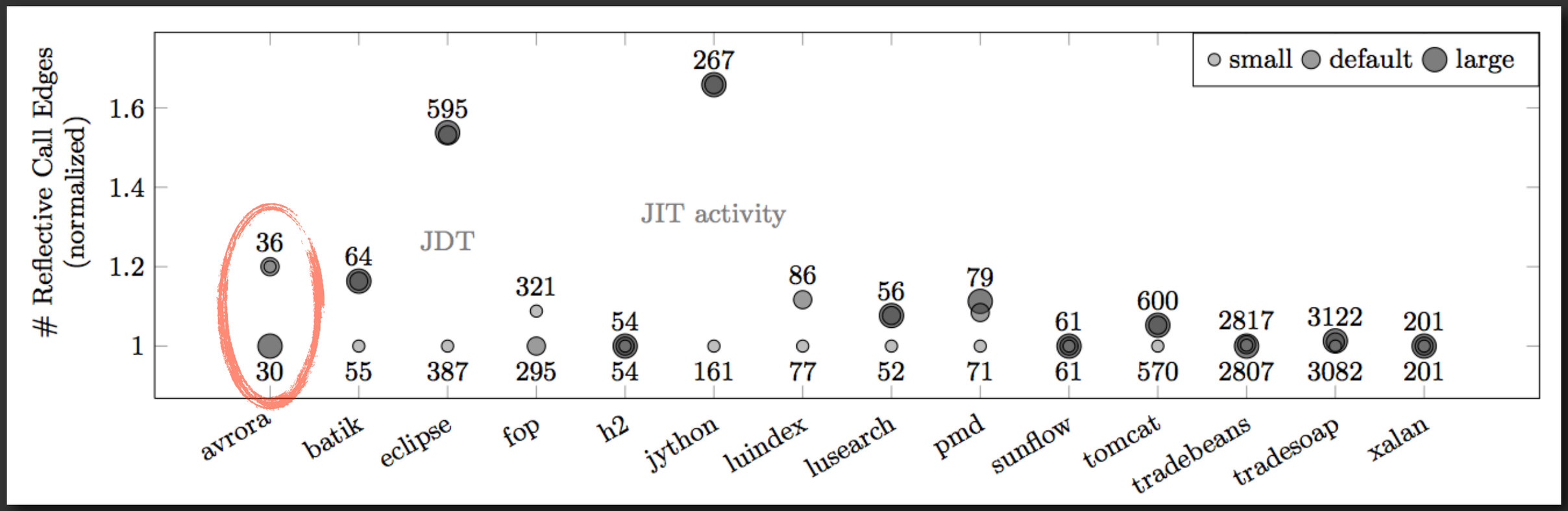
Effectiveness: How much does log-file quality depend on program input?





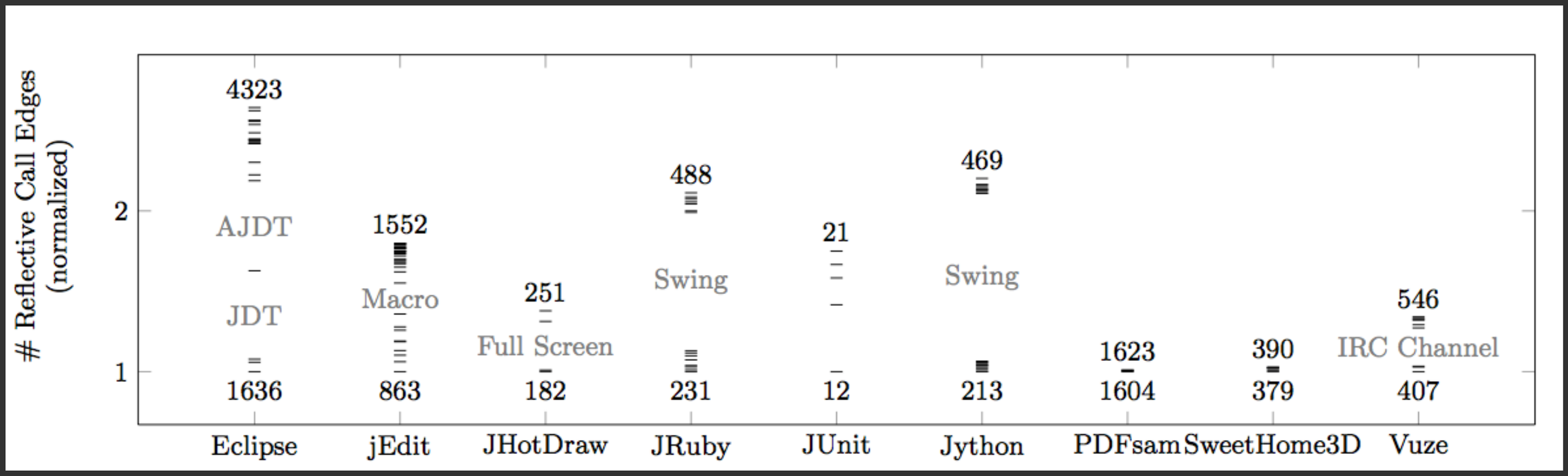
(a) avrora

Effectiveness: How much does log-file quality depend on program input?



Can now safely use static analyses on DaCapo: Warnings never triggered!

Effectiveness: How much does log-file quality depend on program input?



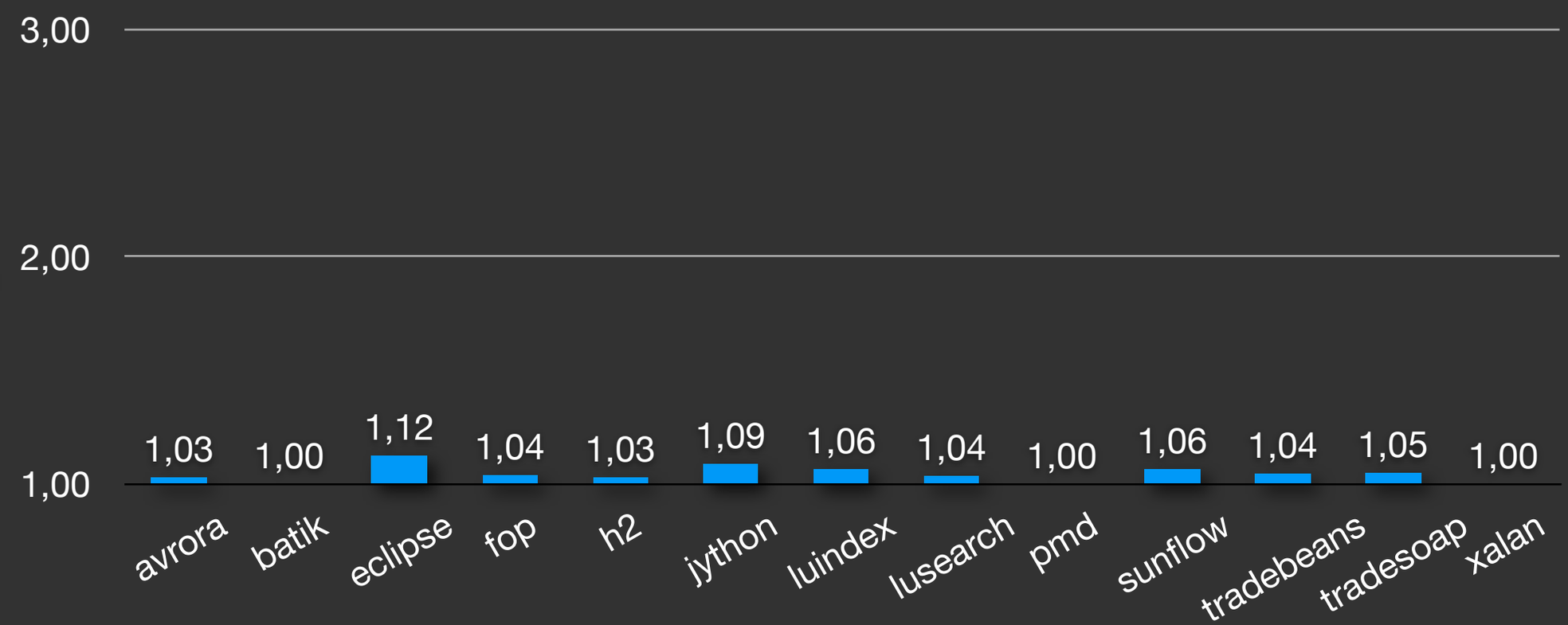
Other prominent programs/frameworks

Effectiveness: How much does log-file quality depend on program input?

Approach requires good "feature coverage

Play-Out

Play-In



Related Work

- Partial evaluation
  Braux & Noyé, PEPM 1999

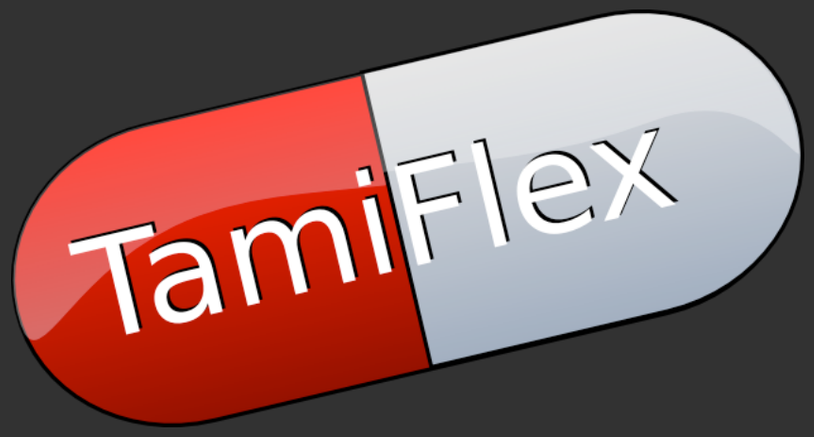
- String analysis
  Christensen et al., SAS 2003
  Livshits et al., APLAS 2005

- Application extraction
  Tip et. al, TOPLAS 2002

- "Finding the code"
  Bessey et. al (Coverity), CACM 2010

Related Work

- Online Call-Graph Construction
  Hirzel et al., TOPLAS 2007
- Blended Analysis
  Dufour et al., ISSTA 2007
- Runtime type inference in ~~P~~Ruby
  Furr et al., OOPSLA 2009



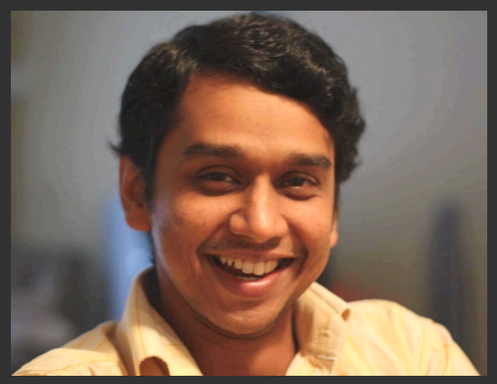
http://tamiflex.googlecode.com/

First external users

Cheng Zhang, SJTU

Play-out, Soot, Play-In

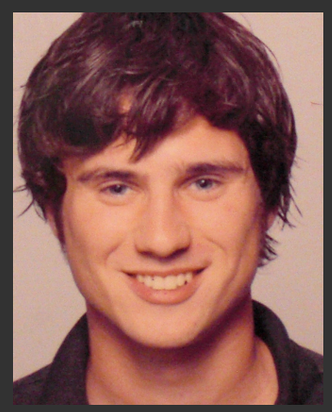
Call frequency analysis



Saswat Anand, Georgia Tech

Play-out, Soot

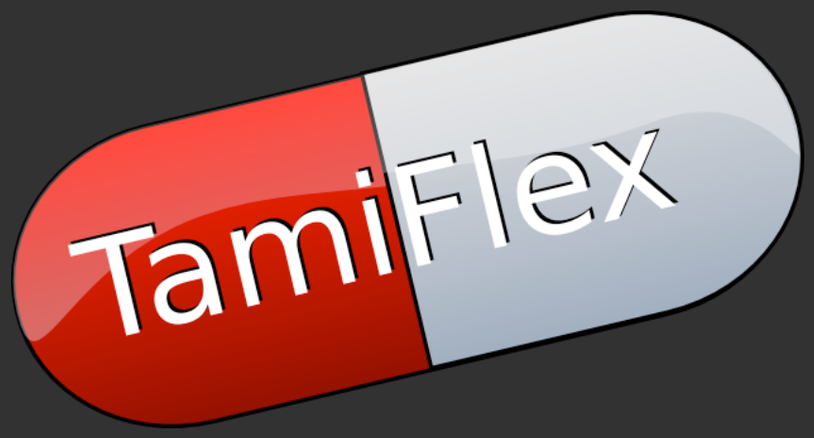
Points-to, Symb. Exec.



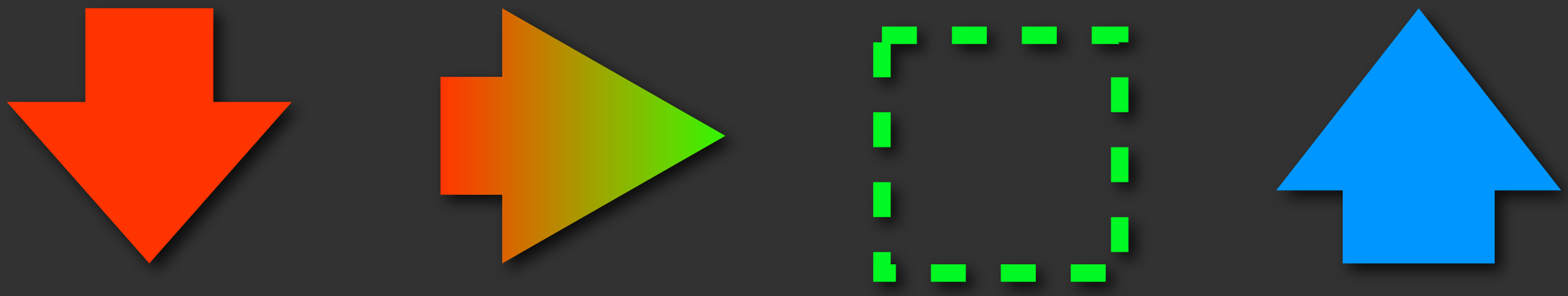
Jochen Huck, KIT

Play-out, Soot

Side-Effects, Dominance



http://tamiflex.googlecode.com/



Play-Out Boost Analyse & Transform Play-In

Correct Effective Efficient

TamiFlex

http://tamiflex.googlecode.com/

MODULARITY - aosd 2012

submit to

2nd of 3 submission deadlines: July 14th

July 17th-21st, 2011, Toronto

Record number of submitted & accepted papers

attend