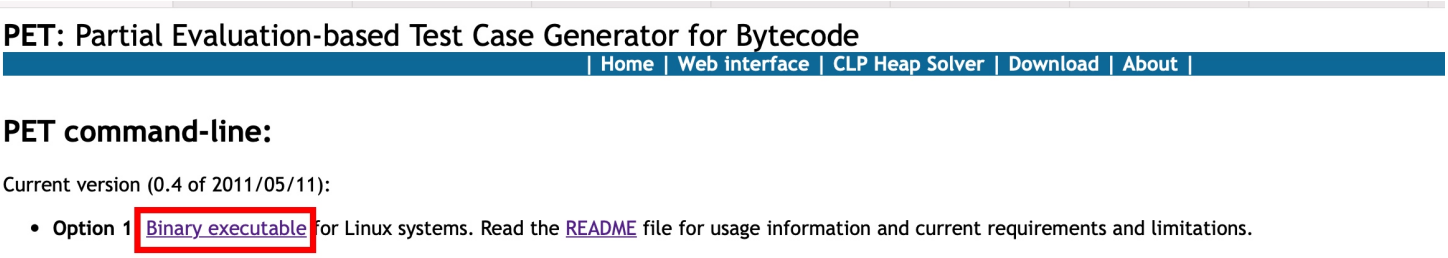
- 阅读了论文《Evaluating Code-based Test Generator Tools》（65页）和《jpet: An automatic test-case generator for java》
- 在阅读SETTE源码的时候发现sette-tool/test-generator-tools/jpet/get-tool.sh 发现

```
#!/bin/bash
# This script downloads jPET.

CWD="$(
  cd "$(dirname "$(readlink "$0" || printf %s "$0")")"
  pwd -P
)"

rm -f "$CWD/pet"
wget "http://costa.ls.fi.upm.es/pet/pet" -O "$CWD/pet"
chmod +x "$CWD/pet"

echo "0.4" > "$CWD/VERSION"
```

SETTE框架都是通过脚本的方式获取相应的test-generator-tools

- 但是非常令人恼火的是..Jpet自身给的**相应论文和YouTube教程**中都是以**jar包和eclipse插件的形式**运行的。
- 但是经过我的实验发现：

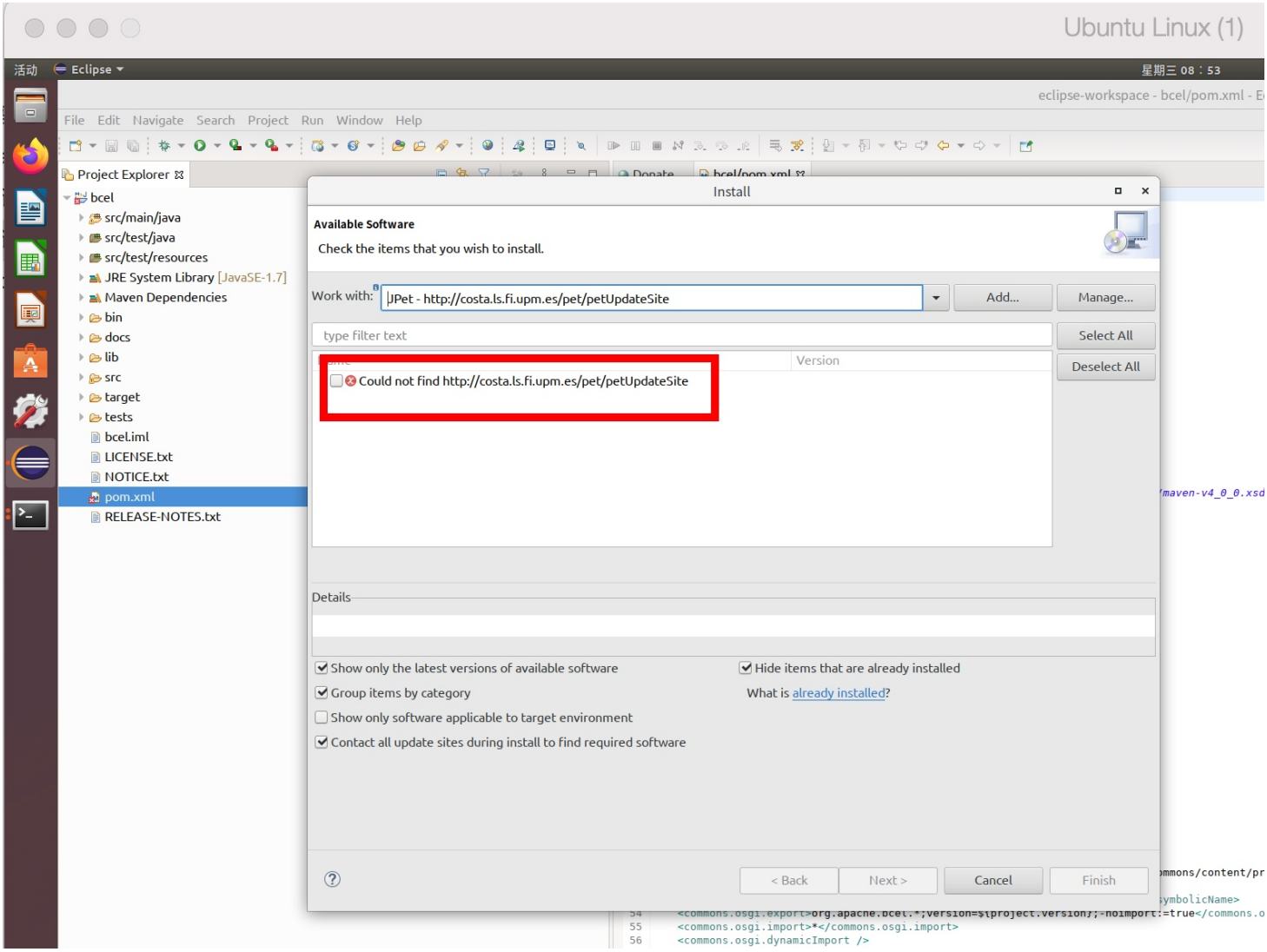


点击尝试下载jar包，结果直接来了个**404**

Not Found

The requested URL was not found on this server.

然后准备尝试用eclipse插件的形式看是否能有希望，结果：



完蛋。就是说连**jpet发布的eclipes插件都404。。**

最后仔细阅读Jpet的官网发现他做成了Web interface的形式（**就是不提供jar和eclipse，然而论文里面也没提到？**）

- web interface形式提供的功能**看起来非常强大**，可以自行上传jar包和class文件。比如（bcel.jar）
- 上传完jar后jpet会直接解析jar中的所有类并以图形界面的方式展示（很强）

PET: Partial Evaluation-based Test Case Generator for Bytecode

| Home | Web interface | CLP Heap Solver | Download | About |

Jar file uploaded : bcel-6.0

Select the class name you want to test

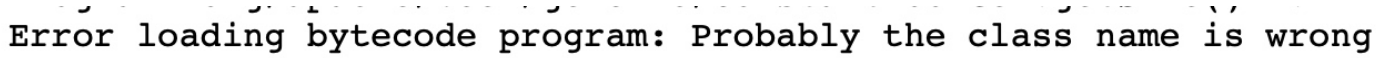
- org/apache/bcel/Constants.class
- org/apache/bcel/util/ConstantHTML.class
- org/apache/bcel/util/ClassSet.class
- org/apache/bcel/util/ClassPath$Zip.class
- org/apache/bcel/util/ByteSequence.class
- org/apache/bcel/util/ClassPathDir1.class
- org/apache/bcel/util/InstructionFinder.class
- org/apache/bcel/util/ClassPathRepository.class
- org/apache/bcel/util/BCELifier$FLAGS.class
- org/apache/bcel/util/SyntheticRepository.class
- org/apache/bcel/util/ClassQueue.class
- org/apache/bcel/util/MemorySensitiveClassPathRepository.class
- org/apache/bcel/util/ClassPath$ClassFile.class
- org/apache/bcel/util/ClassLoaderRepository.class
- org/apache/bcel/util/ClassPath$1.class
- org/apache/bcel/util/ClassVector.class
- org/apache/bcel/util/AttributeHTML.class
- org/apache/bcel/util/ClassPathZip1.class
- org/apache/bcel/util/ClassPath$Dir.class
- org/apache/bcel/util/BCELifier.class
- org/apache/bcel/util/MethodHTML.class
- org/apache/bcel/util/ClassLoader.class
- org/apache/bcel/util/Repository.class
- org/apache/bcel/util/BCELComparator.class
- org/apache/bcel/util/JavaWrapper.class
- org/apache/bcel/util/ClassPath.class
- org/apache/bcel/util/ByteSequence$ByteArrayStream.class
- org/apache/bcel/util/BCELFactory.class
- org/apache/bcel/util/InstructionFinder$CodeConstraint.class
- org/apache/bcel/util/Class2HTML.class
- org/apache/bcel/util/CodeHTML.class
- org/apache/bcel/util/ClassStack.class

可以选择对应的待测类。比如ConstantPoolGen.class
然后jpet会进一步把class的所有方法展示出来

string(78) "javap -classpath /tmp/pet/web/org/apache/bcel/generic -s ConstantPoolGen 2>&1 " string(89) "Warning: Binary file ConstantPoolGen contains
org.apache.bcel.generic.ConstantPoolGen
" string(40) "Compiled from "ConstantPoolGen.java"
" string(58) "public class org.apache.bcel.generic.ConstantPoolGen {
" array(4) { [0]=> string(49) "c class org.apache.bcel.generic.ConstantPoolGen {" [1]=> string(2) "c " ["name"]=> string(39) "org.apache.bcel.generic.ConstantPoolGen" [2]=> string(39)
"org.apache.bcel.generic.ConstantPoolGen" } <init>([Lorg/apache/bcel/classfile/Constant;)V
<init>(Lorg/apache/bcel/classfile/ConstantPool;)V
<init>()V
adjustSize()V
lookupString(Ljava/lang/String;)I
addString(Ljava/lang/String;)I
lookupClass(Ljava/lang/String;)I
addClass(Ljava/lang/String;)I
addClass(Lorg/apache/bcel/generic/ObjectType;)I
addArrayClass(Lorg/apache/bcel/generic/ArrayType;)I
lookupInteger(I)I
addInteger(I)I
lookupFloat(F)I
addFloat(F)I
lookupUtf8(Ljava/lang/String;)I
addUtf8(Ljava/lang/String;)I
lookupLong(J)I
addLong(J)I
lookupDouble(D)I
addDouble(D)I
lookupNameAndType(Ljava/lang/String;Ljava/lang/String;)I
addNameAndType(Ljava/lang/String;Ljava/lang/String;)I
lookupMethodref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
lookupMethodref(Lorg/apache/bcel/generic/MethodGen;)I
addMethodref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
addMethodref(Lorg/apache/bcel/generic/MethodGen;)I
lookupInterfaceMethodref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
lookupInterfaceMethodref(Lorg/apache/bcel/generic/MethodGen;)I
addInterfaceMethodref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
addInterfaceMethodref(Lorg/apache/bcel/generic/MethodGen;)I
lookupFieldref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
addFieldref(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I
getConstant(I)Lorg/apache/bcel/classfile/Constant;

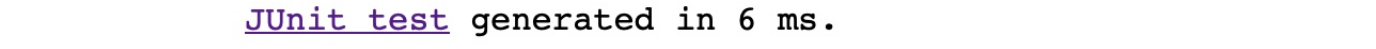
选对待测的方法，提供一系列的参数，点击生成按钮。
其中一个参数是

它可以为方法生成相应的测试套件。。
但是经过多次运行发现不是



就是**仅仅有输出信息**，但是根本无法下载任何的测试套件，**测了个寂寞**。
比如用官网给的样例进行测试：

```
Code for [ints/Factorial.superFact(I)I,ints/Factorial.hyperFact(I)I] decompiled into a CLP program in 10 ms.

Test-cases (with dpk(20)) for method(s) ['ints/Factorial.superFact(I)I','ints/Factorial.hyperFact(I)I'] generated in 3 ms.

2 test-cases generated for ints/Factorial.superFact(I)I (with dpk(20)).

3 test-cases generated for ints/Factorial.hyperFact(I)I (with dpk(20)).

TDG Statistics for method(s) ['ints/Factorial.superFact(I)I','ints/Factorial.hyperFact(I)I']:
        97 unfolding steps. 0 summaries reused or generated. 0 summaries composed.

JUnit test generated in 6 ms.
```



输出信息里面有

Not Found

The requested URL was not found on this server.

但是点进去 （测了个寂寞）
代码覆盖率也是玄学

```
Code Coverage checked in 2 ms:
         Full Code Coverage of 'ints/IntExponential.intExpRec(II)I': 0% (0/32)
         Top Code Coverage of 'ints/IntExponential.intExpRec(II)I': 0% (0/31)

         Full Code Coverage of 'ints/IntExponential.intExpBuggy(II)I': 0% (0/27)
         Top Code Coverage of 'ints/IntExponential.intExpBuggy(II)I': 0% (0/26)


JUnit test generated in 26 ms.
```

放到现在就是完全不可能运行。。

- 感觉Jpet和Jcute都是一样的结果，年久失修导致问题很多...
- **感觉SETEE跑起来也难...**