Java^{TM} PathFinder

Neha Rungta

NASA Ames Research Center

Software Crisis

- Software crisis declared in 1968
- Programs around 100K lines of code
- What has changed?
  - Programs bigger (5M-40M)
  - Processors faster and memory larger
  - Programs in more places (Ubiquitous?)
- Software engineering relatively the same

REPORT ON A CONFERENCE SPONSORED BY THE
NATO SCIENCE COMMITTEE
ROME, ITALY, 27th to 31st OCTOBER 1969

SOFTWARE ENGINEERING TECHNIQUES

CHAIRMAN : PROFESSOR P. ERCOLI
CO-CHAIRMAN : PROFESSOR DR. F.L. BAUER

EDITORS : J.N. BUXTON and B. RANDELL

APRIL 1970

If 1968 was a crisis then, what is today?

Software Engineering

- Little engineering in software engineering
- Very little modeling and analysis
- Reuse and copy is common
- Trial and error testing
- Struggle to produce reliable software

CMP

Enron's Advanced Technology And What UBS Warburg Plans To Do With It P. 28
Long-Tenured IT Execs Explain The Advantages Of Staying In One Place P. 36

INFORMATIONWEEK

BUSINESS INNOVATION POWERED BY TECHNOLOGY JAN. 21, 2002

THE SORRY STATE OF SOFTWARE QUALITY

The number of bugs, holes, and patches more than doubled last year, keeping IT departments busy and putting business data at risk. Here's how Microsoft and other software developers got into this mess—and how they hope to get out. P. 22

PLUS

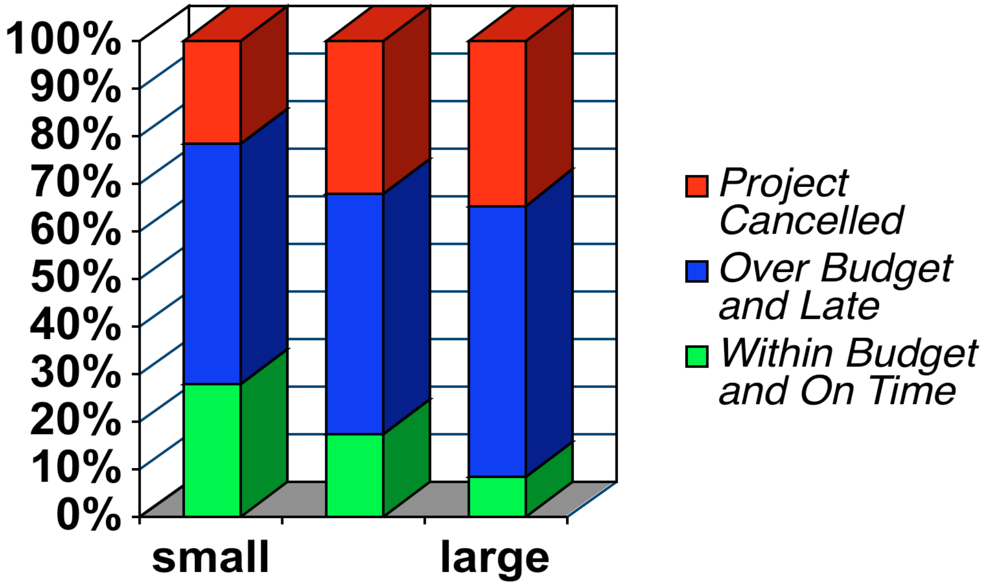
MetLife's revamped customer-service strategy depends on data integration P. 45

Bethlehem Steel re-architects IT management for closer ties with finance P. 55

informationweek.com

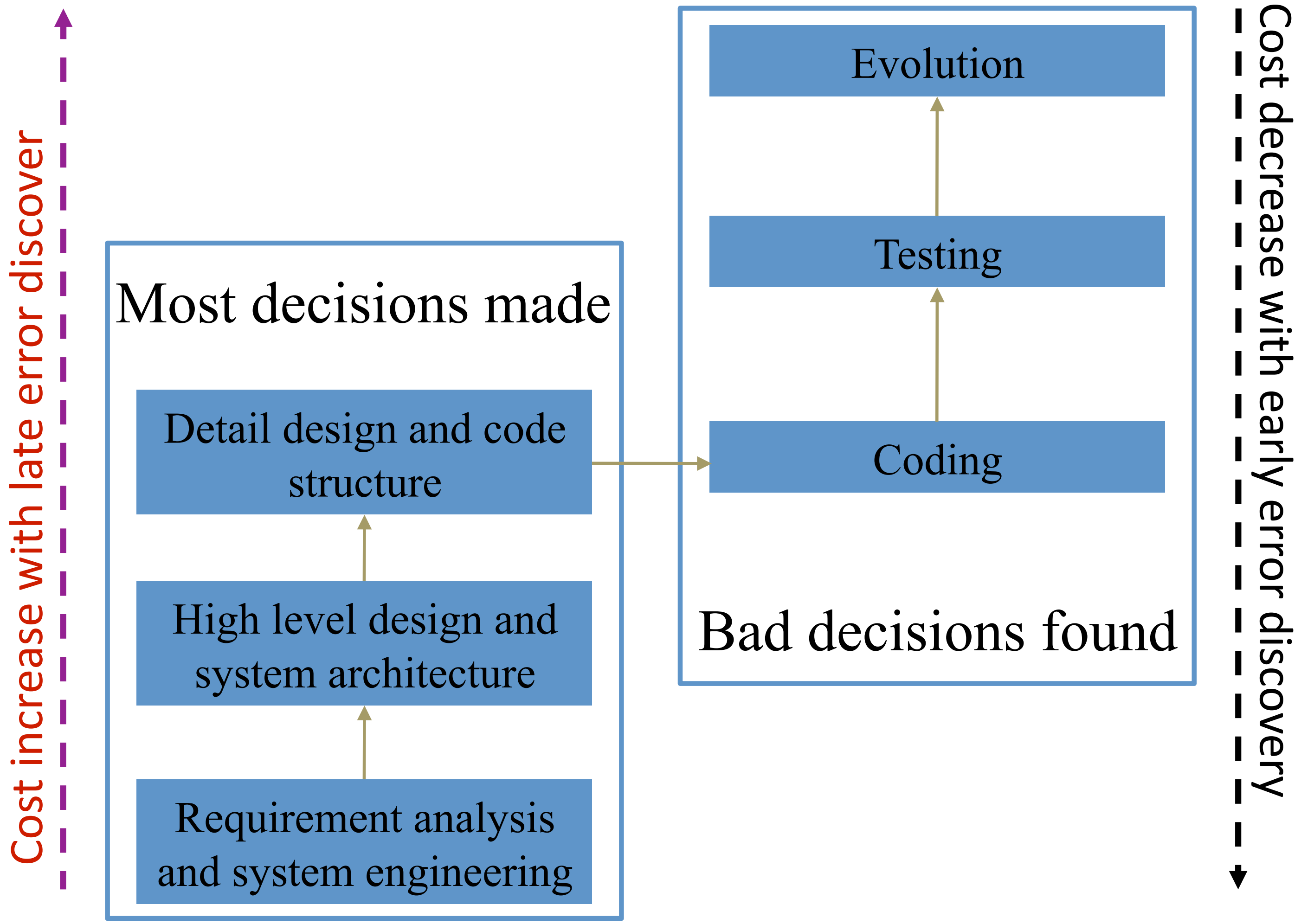
CAN $5.95 • US $4.95

Reality Check



Data: Standish Group, 1995 survey of 365 companies and 8,380 applications. NIST Report 02-3: The economic impacts of inadequate infrastructure for software testing. (May 2002).

Software Engineering



Cost increase with late error discover

Most decisions made

Detail design and code structure

High level design and system architecture

Requirement analysis and system engineering

Evolution

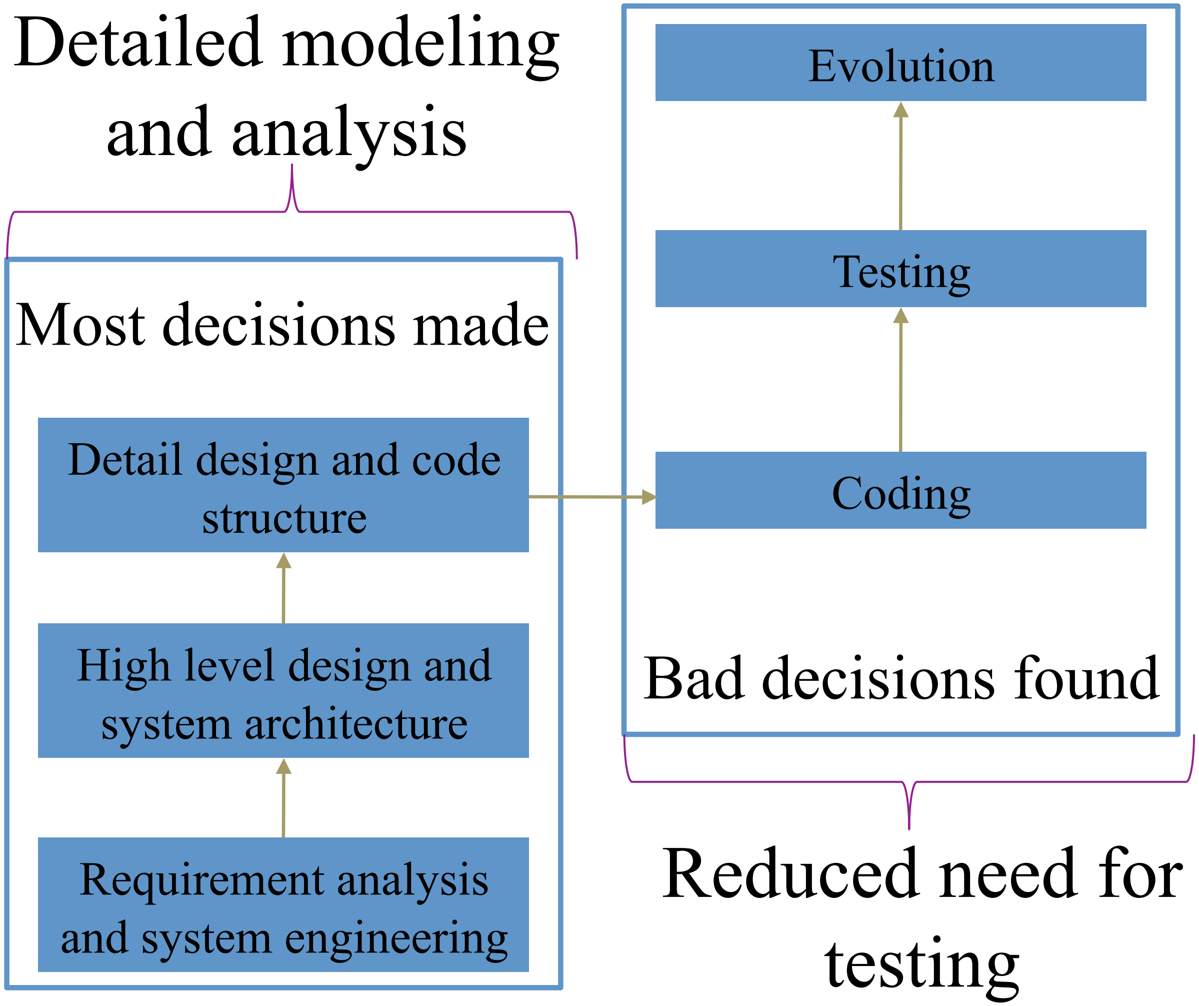
Testing

Coding

Bad decisions found

Cost decrease with early error discovery

Software Model Checking



Detailed modeling and analysis

Most decisions made

Detail design and code structure

High level design and system architecture

Requirement analysis and system engineering

Evolution

Testing

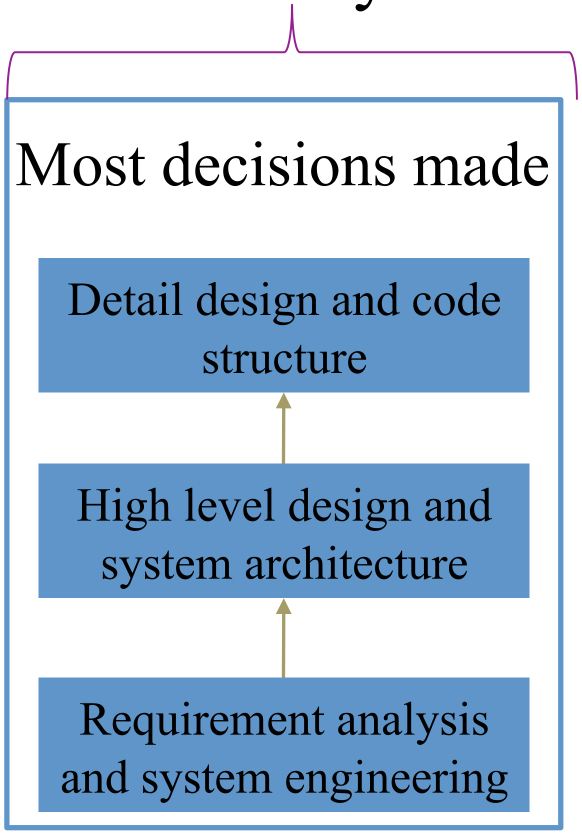
Coding

Bad decisions found

Reduced need for testing

Software Model Checking

Detailed modeling and analysis

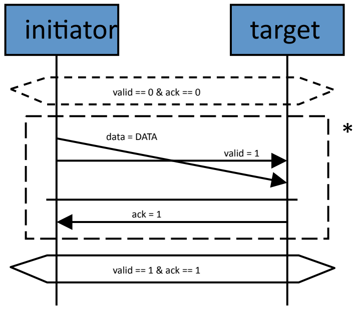


Most decisions made

Detail design and code structure

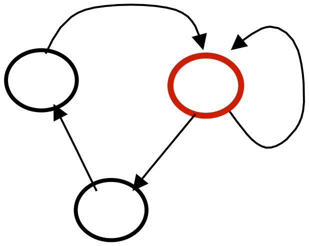
High level design and system architecture

Requirement analysis and system engineering



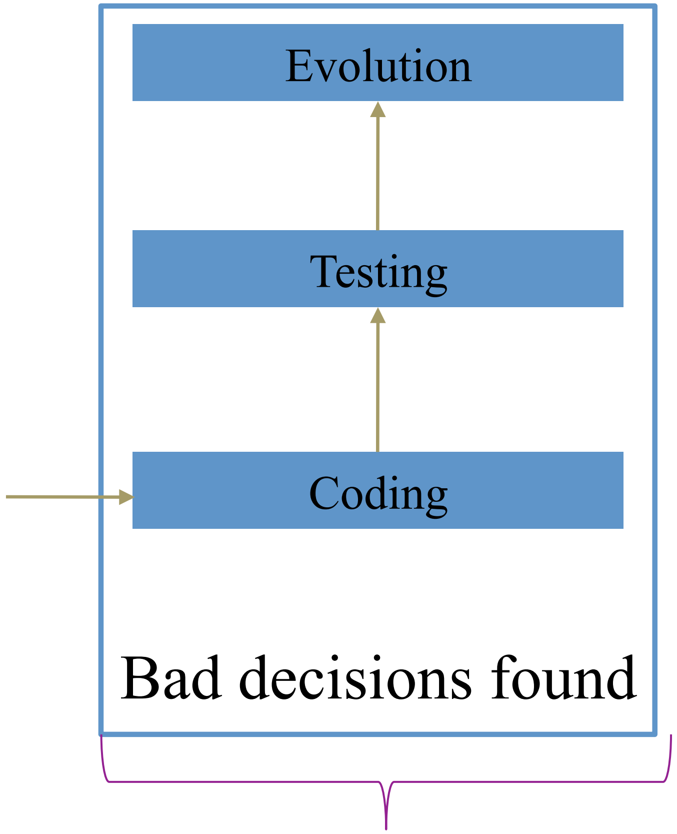
Formalize requirements in mathematically precise language

Build logical models of all designs and analyze with requirements



Do not move up until designs provably implement requirements and meet specifications

Software Model Checking



Evolution

Testing

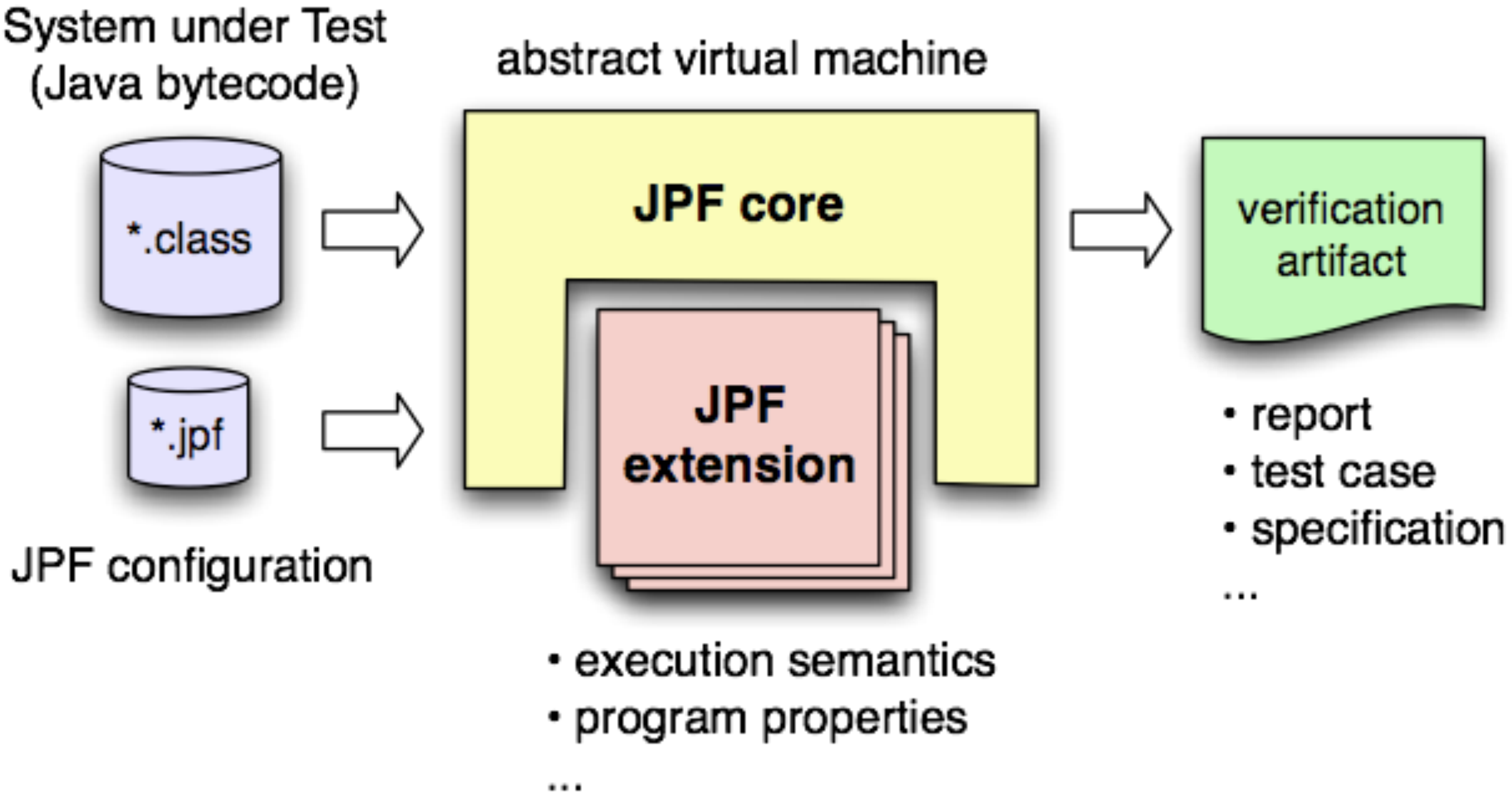
Coding

Bad decisions found

What can software model checking find?

- Errors in deep execution traces
- Deadlock, live-lock, and starvation
- Race conditions
- Priority inversion and locking problems
- Resource allocation errors
- Bounds checking
- Incompleteness and redundancy
- Logic problems
- What ever you ask!
- BTW, don’t ask don’t tell policy

JPF



System under Test (Java bytecode)

*.class

*.jpf

JPF configuration

abstract virtual machine

JPF core

JPF extension

- execution semantics
- program properties

...

verification artifact

- report
- test case
- specification

...

History

- not a new project: around for 10 years and continuously developed:
  - 1999 - project started as front end for Spin model checker
  - 2000 - reimplementation as concrete virtual machine for software model checking (concurrency defects)
  - 2003 - introduction of extension interfaces
  - 2005 - open sourced on Sourceforge
  - 2008 - participation in Google Summer of Code
  - 2009 - moved to own server, hosting extension projects and Wiki

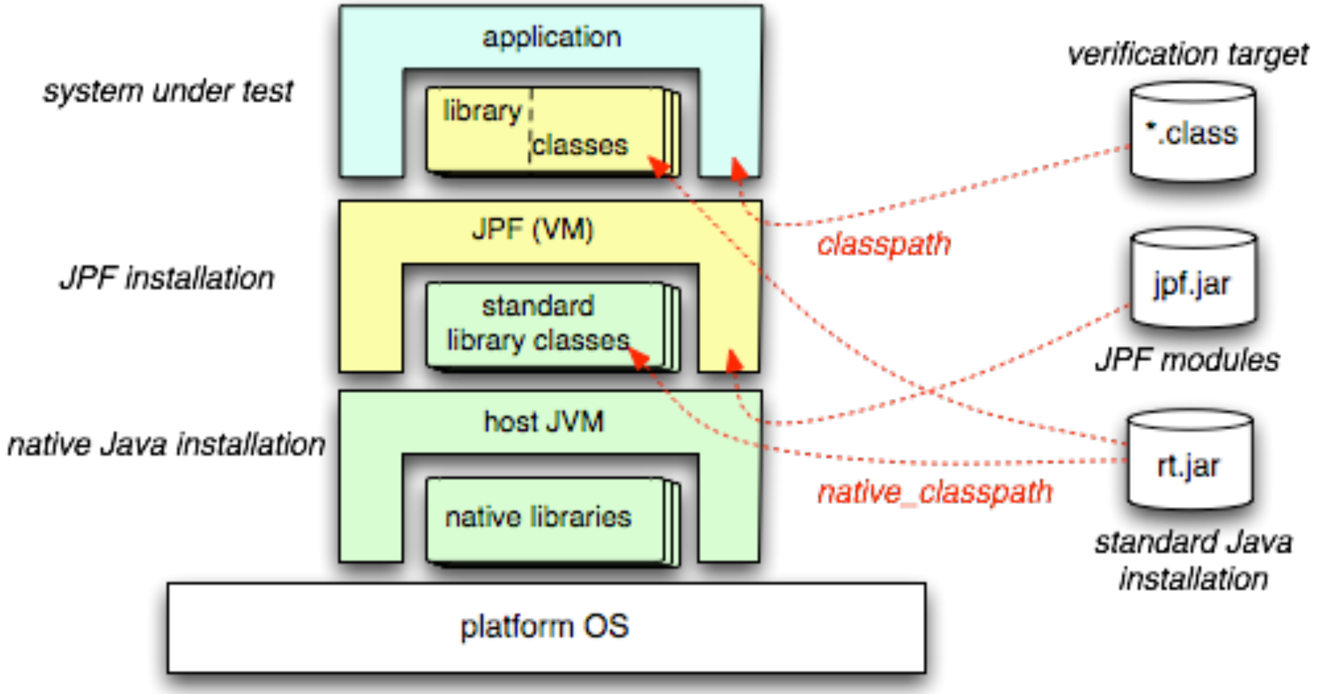
JPF’s Home

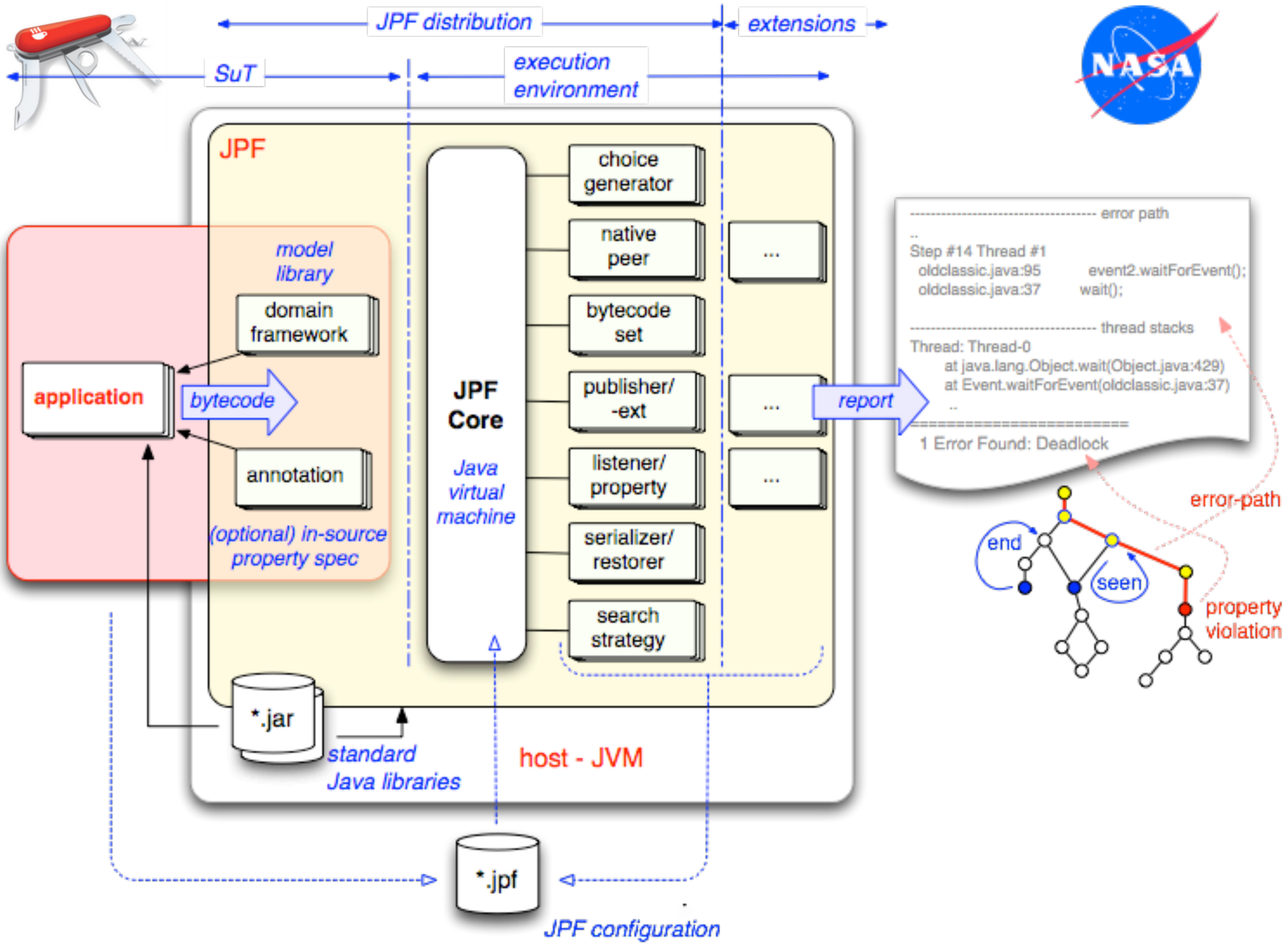
http://babelfish.arc.nasa.gov/trac/jpf

JPF’s User Forum

http://groups.google.com/group/java-pathfinder

Overall Architecture

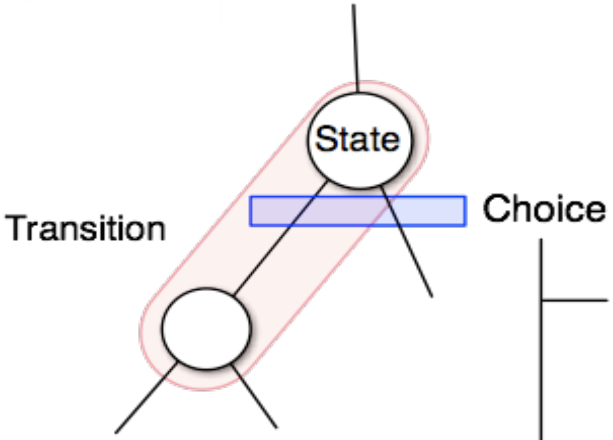




```
------------------------------------ error path
..
Step #14 Thread #1
  oldclassic.java:95          event2.waitForEvent();
  oldclassic.java:37         wait();

------------------------------------ thread stacks
Thread: Thread-0
        at java.lang.Object.wait(Object.java:429)
        at Event.waitForEvent(oldclassic.java:37)
         ..
========================
  1 Error Found: Deadlock
```

Exploring Choices

- model checker needs choices to explore state space
- there are many potential types of choices (scheduling, data, ..)
- choice types should not be hardwired in model checker



Scheduling Choice

```
synchronized (..) {..}
wait (..)
x = mySharedObject
..
```

Data Choice

```
boolean b = Verify.getBoolean();
double d = Verify.getDouble("MyHeuristic");
..
```

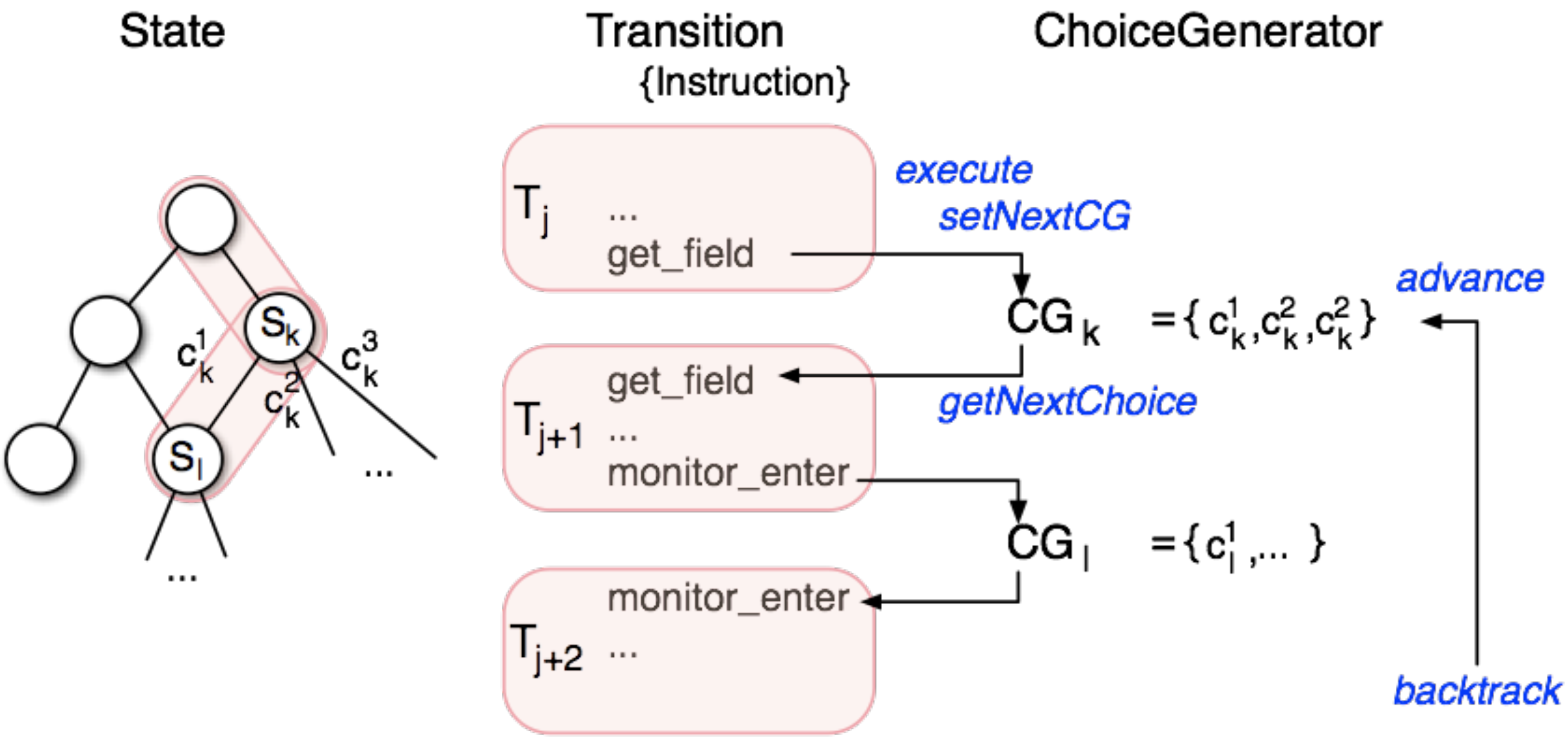
Control Choice

```
if (<cond>) ..
INVOKECG.setInvocations(..)
..
```

Choice Generators

- transitions begin with a choice and extend until the next ChoiceGenerator (CG) is set (by instruction, native peer or listener)
- advance positions the CG on the next unprocessed choice (if any)
- backtrack goes up to the next CG with unprocessed choices

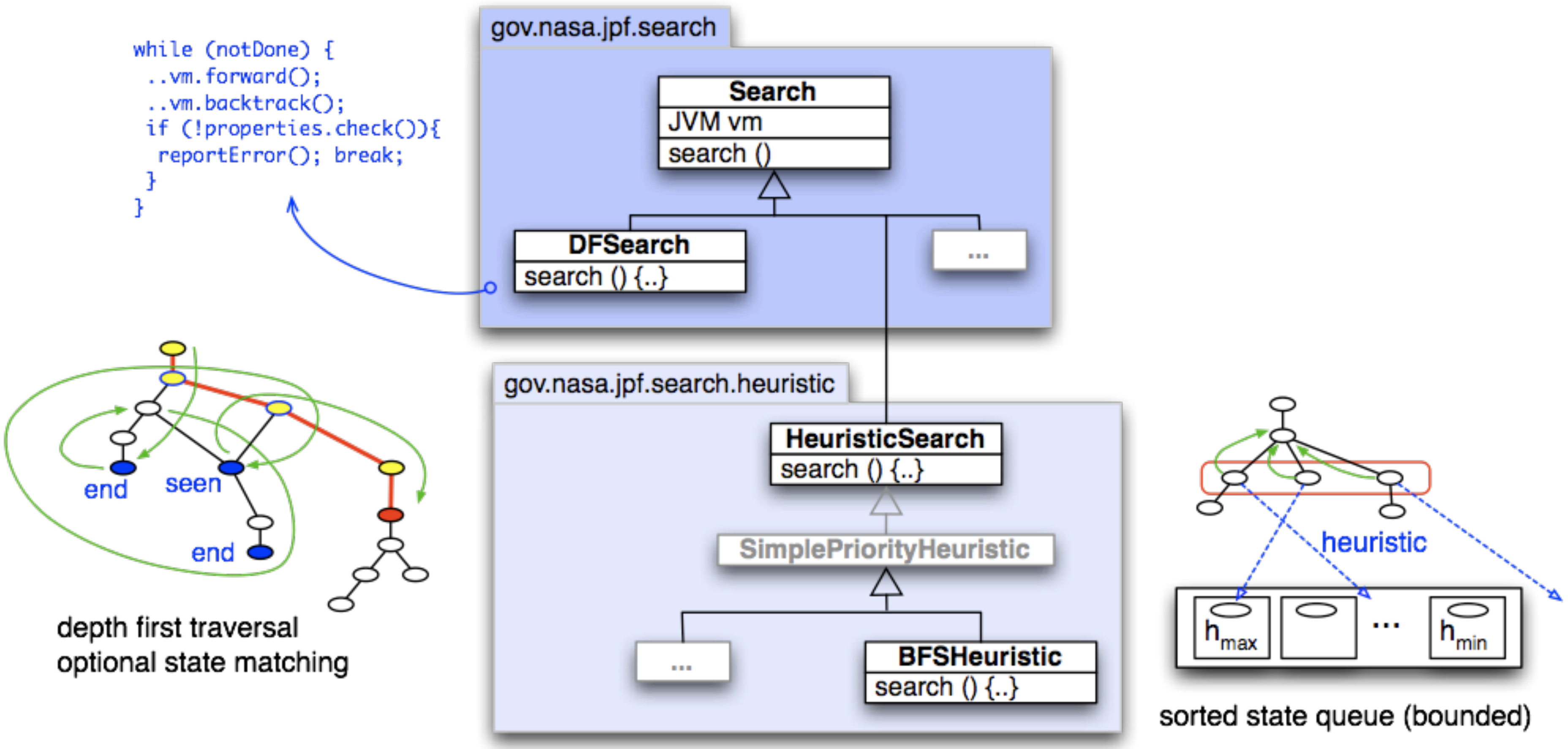
Choice Generators



Search Strategies

- state explosion mitigation: search the interesting state space part first ("get to the bug early, before running out of memory")
- Search instances encapsulate (configurable) search policies

Search Strategies

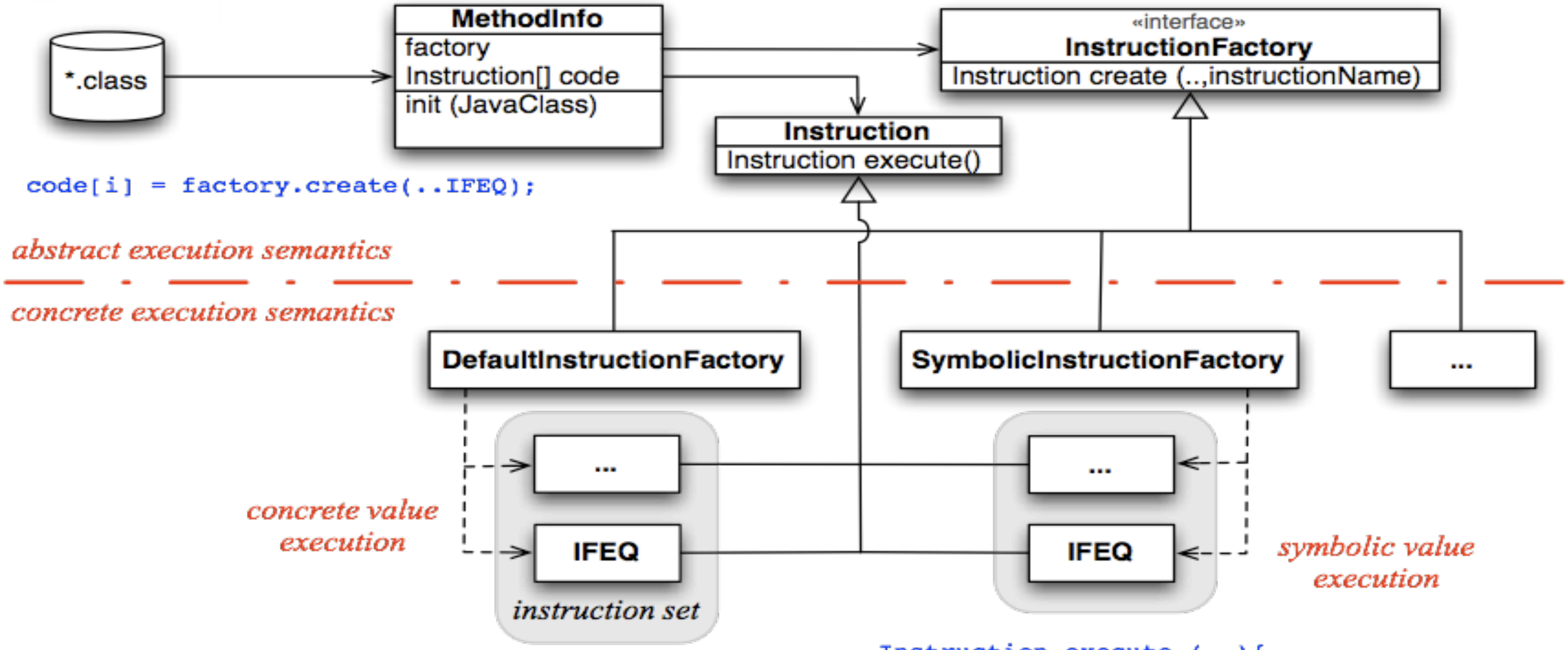


```
while (notDone) {
 ..vm.forward();
 ..vm.backtrack();
 if (!properties.check()){
  reportError(); break;
 }
}
```

depth first traversal
optional state matching

sorted state queue (bounded)

Bytecode Factory

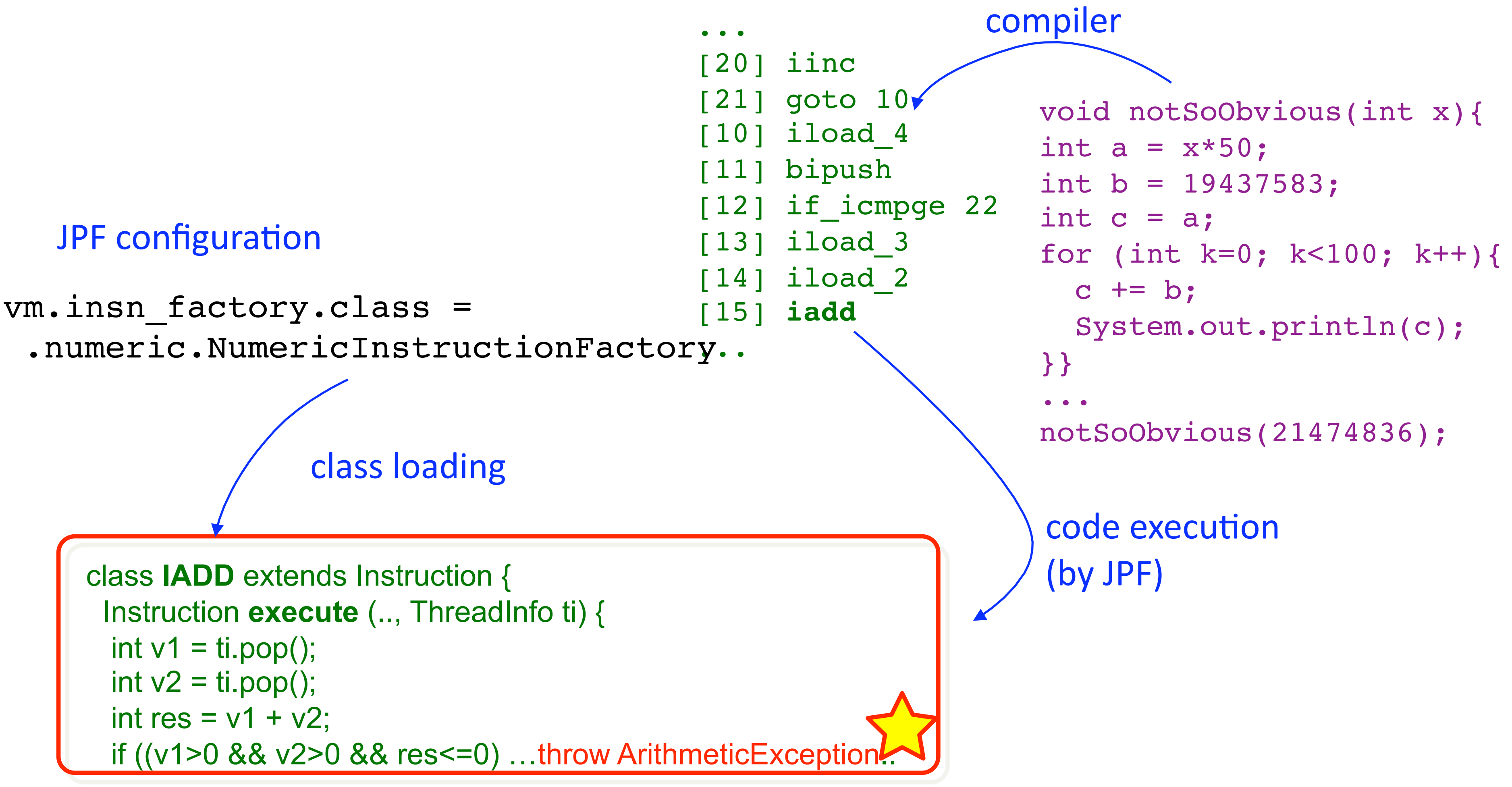


```
code[i] = factory.create(..IFEQ);
```

```
Instruction execute (..){
  cond = popCondition();
  if (cond)
    return jumpTarget;
  else
    return getNextInsn();
}
```

```
Instruction execute (..){
  if (!firstStepInsn()){
    setNextCG(new PCChoiceGenerator());
    return this;
  }
  popCondition(); // not interested
  cond = getCG().getNextChoice();
  if (cond){...
    updatePathCondition(.., EQ);
    return jumpTarget;
  } else {...
    updatePathCondition(.., NE);
    return getNextInsn();
  }
}
```

Example



compiler

```
void notSoObvious(int x){
int a = x*50;
int b = 19437583;
int c = a;
for (int k=0; k<100; k++){
  c += b;
  System.out.println(c);
}}
...
notSoObvious(21474836);
```

```
...
[20] iinc
[21] goto 10
[10] iload_4
[11] bipush
[12] if_icmpge 22
[13] iload_3
[14] iload_2
[15] iadd
```

JPF configuration

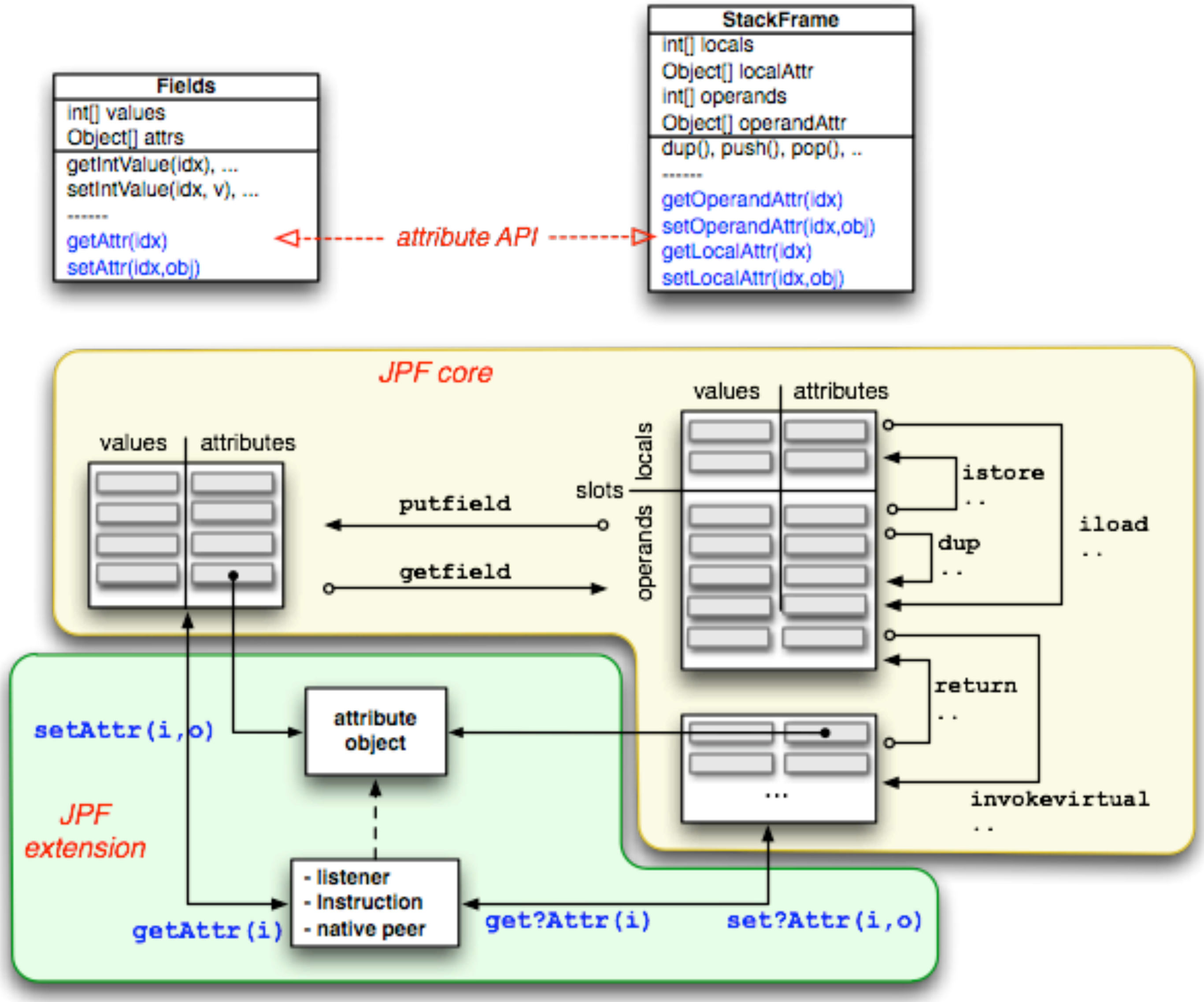
```
vm.insn_factory.class =
 .numeric.NumericInstructionFactory..
```

class loading

code execution (by JPF)

```
class IADD extends Instruction {
 Instruction execute (.., ThreadInfo ti) {
  int v1 = ti.pop();
  int v2 = ti.pop();
  int res = v1 + v2;
  if ((v1>0 && v2>0 && res<=0) …throw ArithmeticException..
```

Attributes



Fields

int[] values
Object[] attrs

getIntValue(idx), ...
setIntValue(idx, v), ...
------
getAttr(idx)
setAttr(idx,obj)

attribute API

StackFrame

int[] locals
Object[] localAttr
int[] operands
Object[] operandAttr

dup(), push(), pop(), ..
------
getOperandAttr(idx)
setOperandAttr(idx,obj)
getLocalAttr(idx)
setLocalAttr(idx,obj)

JPF core

values | attributes

putfield

getfield

slots

locals

operands

istore
..

iload
..

dup
..

return
..

invokevirtual
..

...

setAttr(i,o)

attribute object

JPF extension

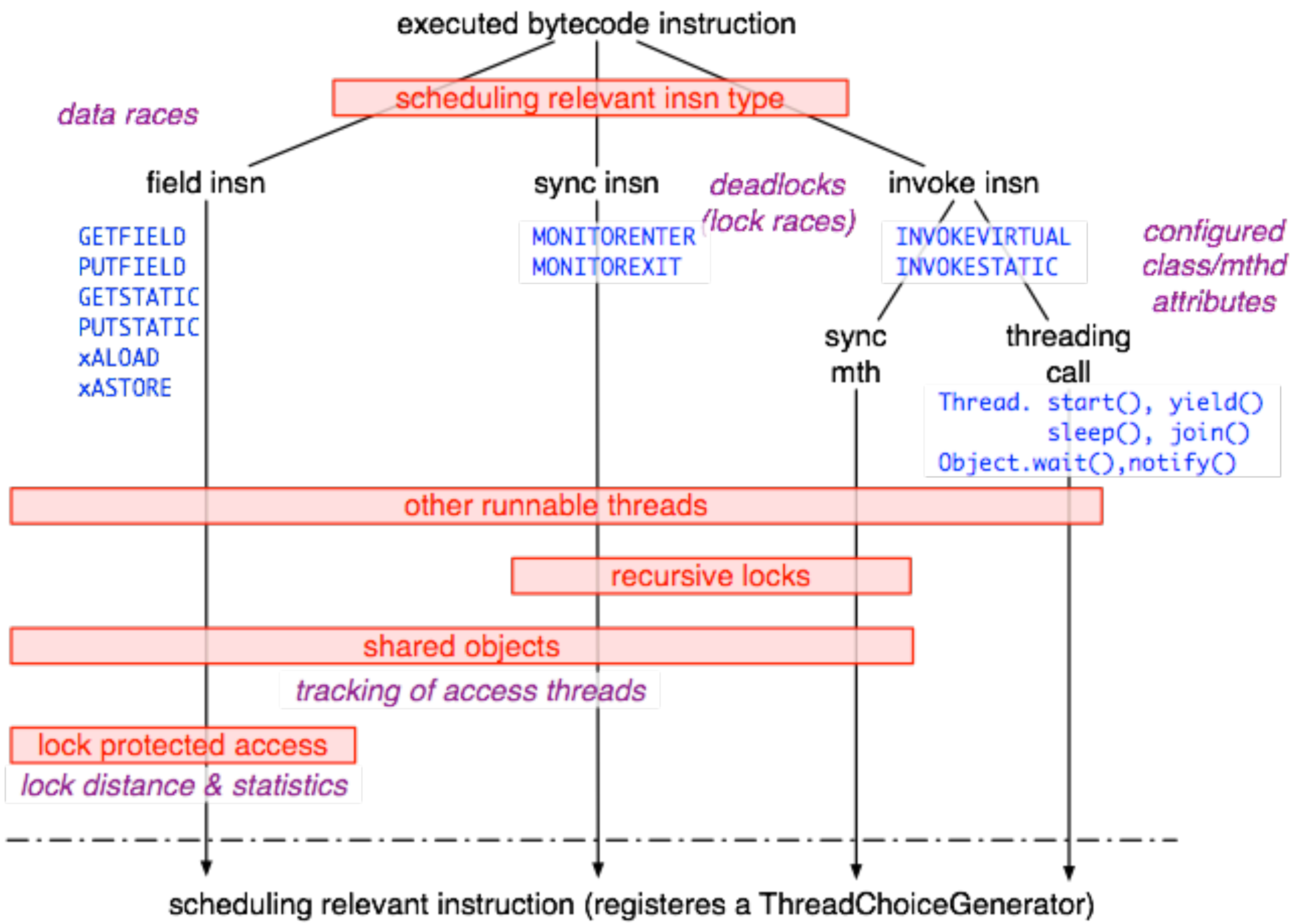
- listener
- Instruction
- native peer

getAttr(i)

get?Attr(i)

set?Attr(i,o)

Partial Order Reduction



executed bytecode instruction

scheduling relevant insn type

data races

field insn

GETFIELD
PUTFIELD
GETSTATIC
PUTSTATIC
xALOAD
xASTORE

sync insn

MONITORENTER
MONITOREXIT

deadlocks (lock races)

invoke insn

INVOKEVIRTUAL
INVOKESTATIC

configured class/mthd attributes

sync mth

threading call

```
Thread. start(), yield()
        sleep(), join()
Object.wait(),notify()
```

other runnable threads

recursive locks

shared objects

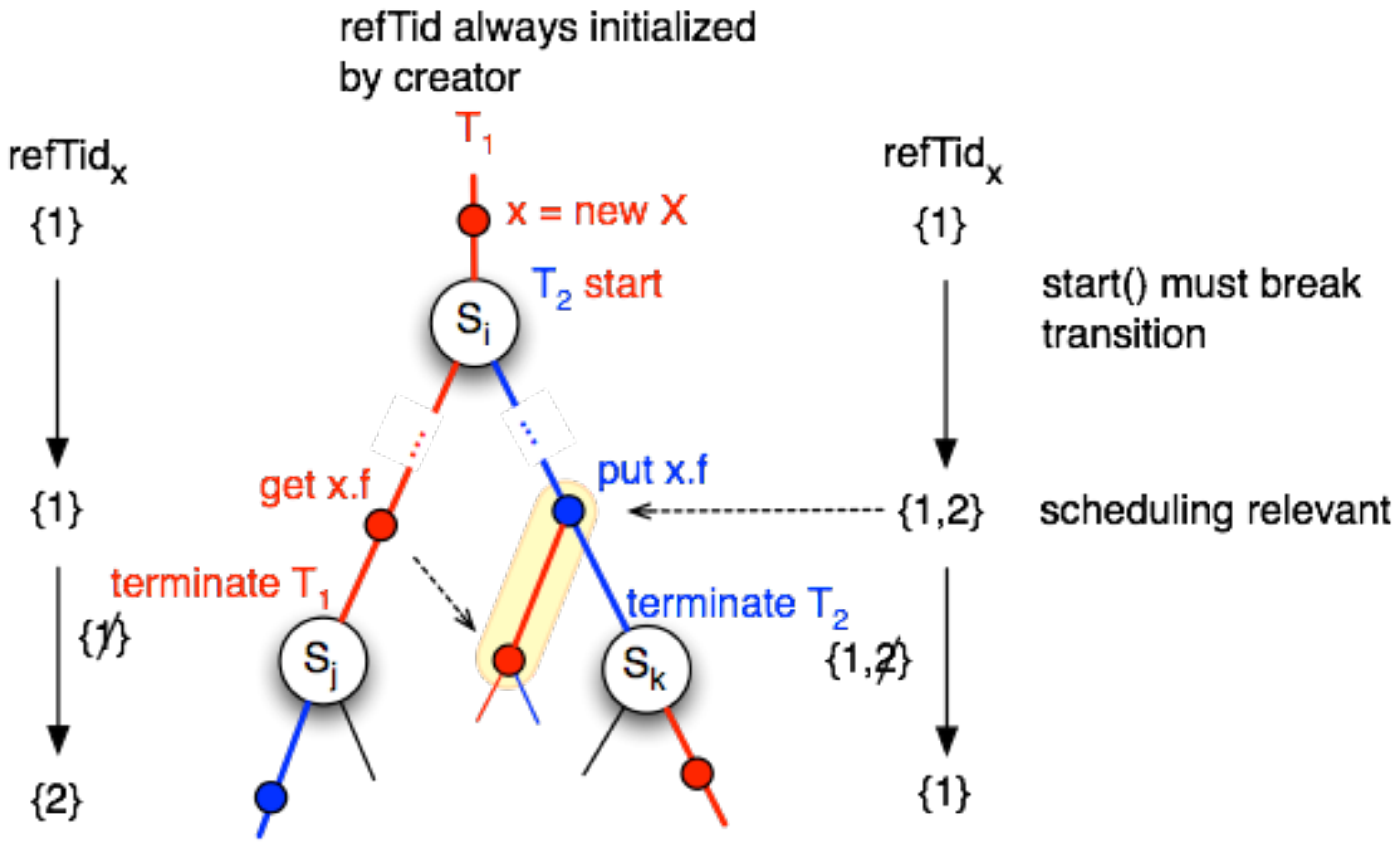
tracking of access threads

lock protected access

lock distance & statistics

scheduling relevant instruction (registeres a ThreadChoiceGenerator)

POR

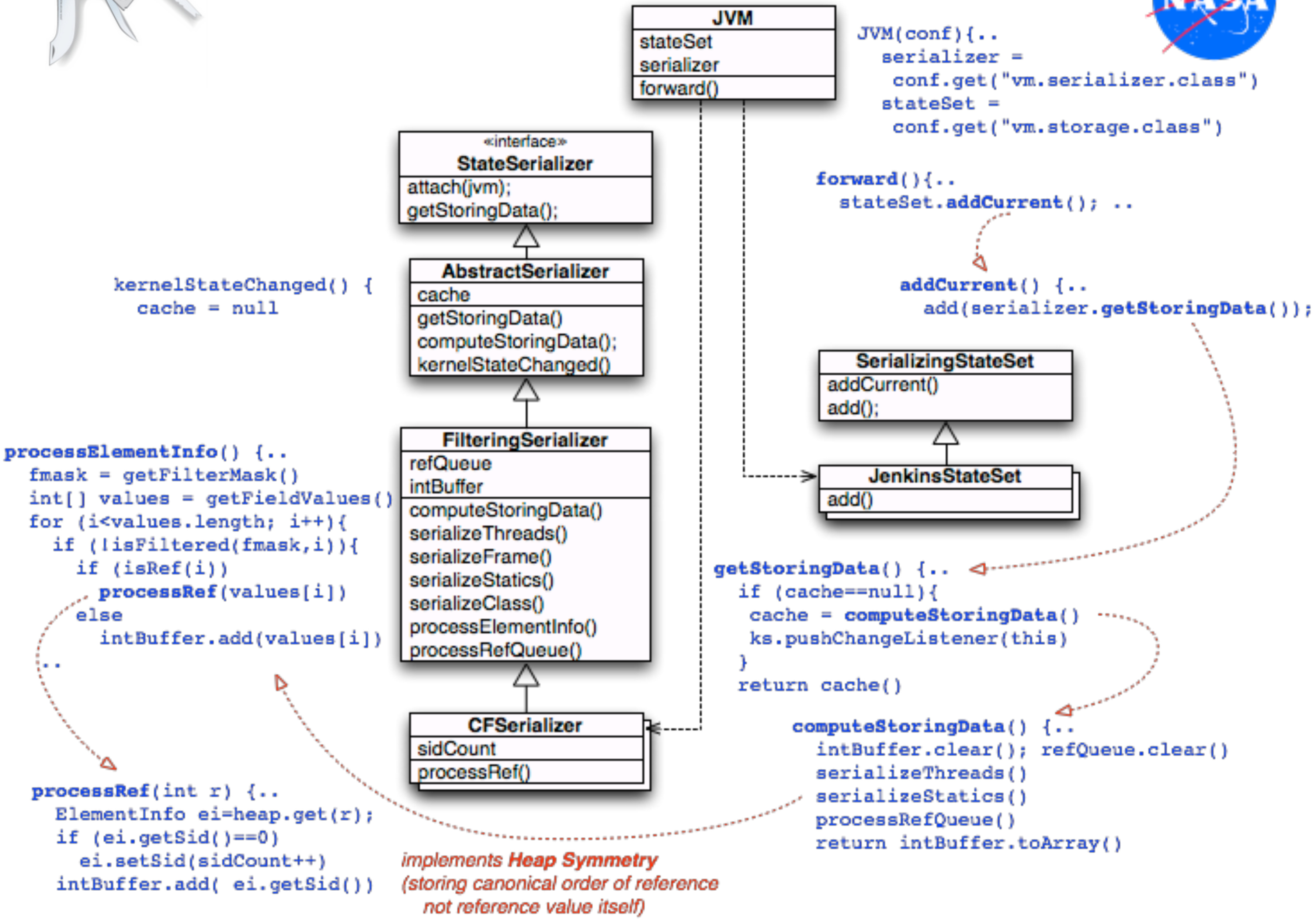


refTid always initialized by creator

start() must break transition

scheduling relevant

State Serialization



JVM
stateSet
serializer
forward()

«interface»
StateSerializer
attach(jvm);
getStoringData();

AbstractSerializer
cache
getStoringData()
computeStoringData();
kernelStateChanged()

FilteringSerializer
refQueue
intBuffer
computeStoringData()
serializeThreads()
serializeFrame()
serializeStatics()
serializeClass()
processElementInfo()
processRefQueue()

CFSerializer
sidCount
processRef()

SerializingStateSet
addCurrent()
add();

JenkinsStateSet
add()

```
JVM(conf){..
  serializer =
   conf.get("vm.serializer.class")
  stateSet =
   conf.get("vm.storage.class")
```

```
forward(){..
  stateSet.addCurrent(); ..
```

```
addCurrent() {..
  add(serializer.getStoringData());
```

```
kernelStateChanged() {
  cache = null
```

```
processElementInfo() {..
  fmask = getFilterMask()
  int[] values = getFieldValues()
  for (i<values.length; i++){
    if (!isFiltered(fmask,i)){
      if (isRef(i))
        processRef(values[i])
      else
        intBuffer.add(values[i])
..
```

```
getStoringData() {..
  if (cache==null){
   cache = computeStoringData()
   ks.pushChangeListener(this)
  }
  return cache()
```

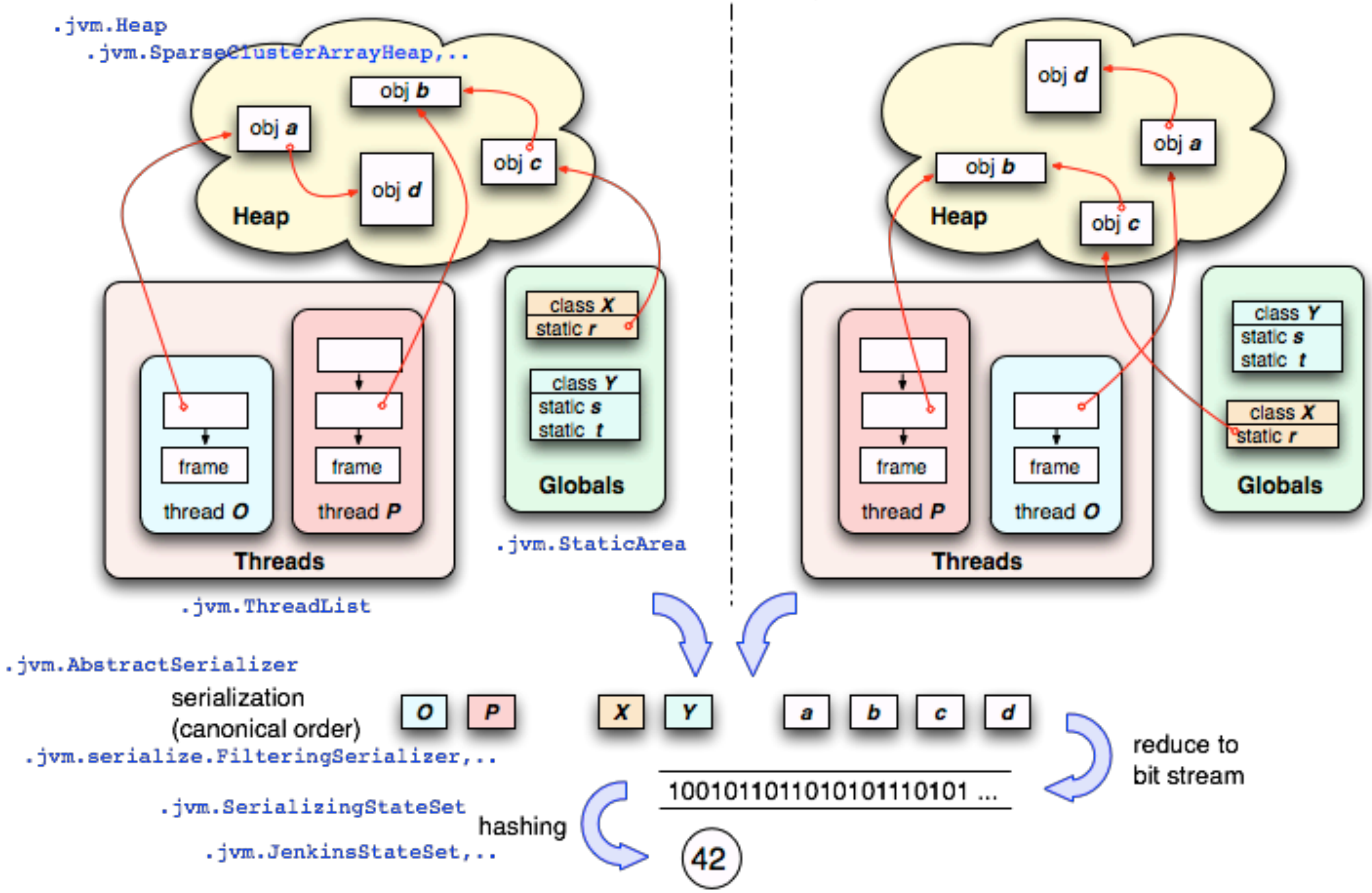
```
computeStoringData() {..
  intBuffer.clear(); refQueue.clear()
  serializeThreads()
  serializeStatics()
  processRefQueue()
  return intBuffer.toArray()
```

```
processRef(int r) {..
  ElementInfo ei=heap.get(r);
  if (ei.getSid()==0)
    ei.setSid(sidCount++)
  intBuffer.add( ei.getSid())
```

*implements **Heap Symmetry***
(storing canonical order of reference not reference value itself)

Heap Symmetry

$S_i = f(T_1, T_2, ..., T_n) \cong S_j = f(T_1, T_2, ..., T_m)$



.jvm.Heap

.jvm.SparseClusterArrayHeap,..

.jvm.StaticArea

.jvm.ThreadList

.jvm.AbstractSerializer

serialization
(canonical order)

.jvm.serialize.FilteringSerializer,..

reduce to
bit stream

.jvm.SerializingStateSet

hashing

.jvm.JenkinsStateSet,..

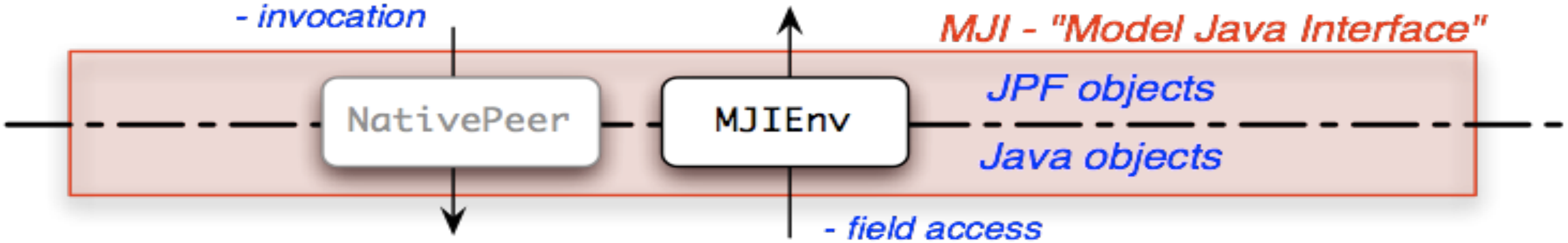
Native Methods

"Model" Class

JPF Class

```
package x.y.z;
class MyClass {
   ..
   native String foo (int i, String s);
}
```

- *method lookup*
- *parameter conversion*
- *invocation*

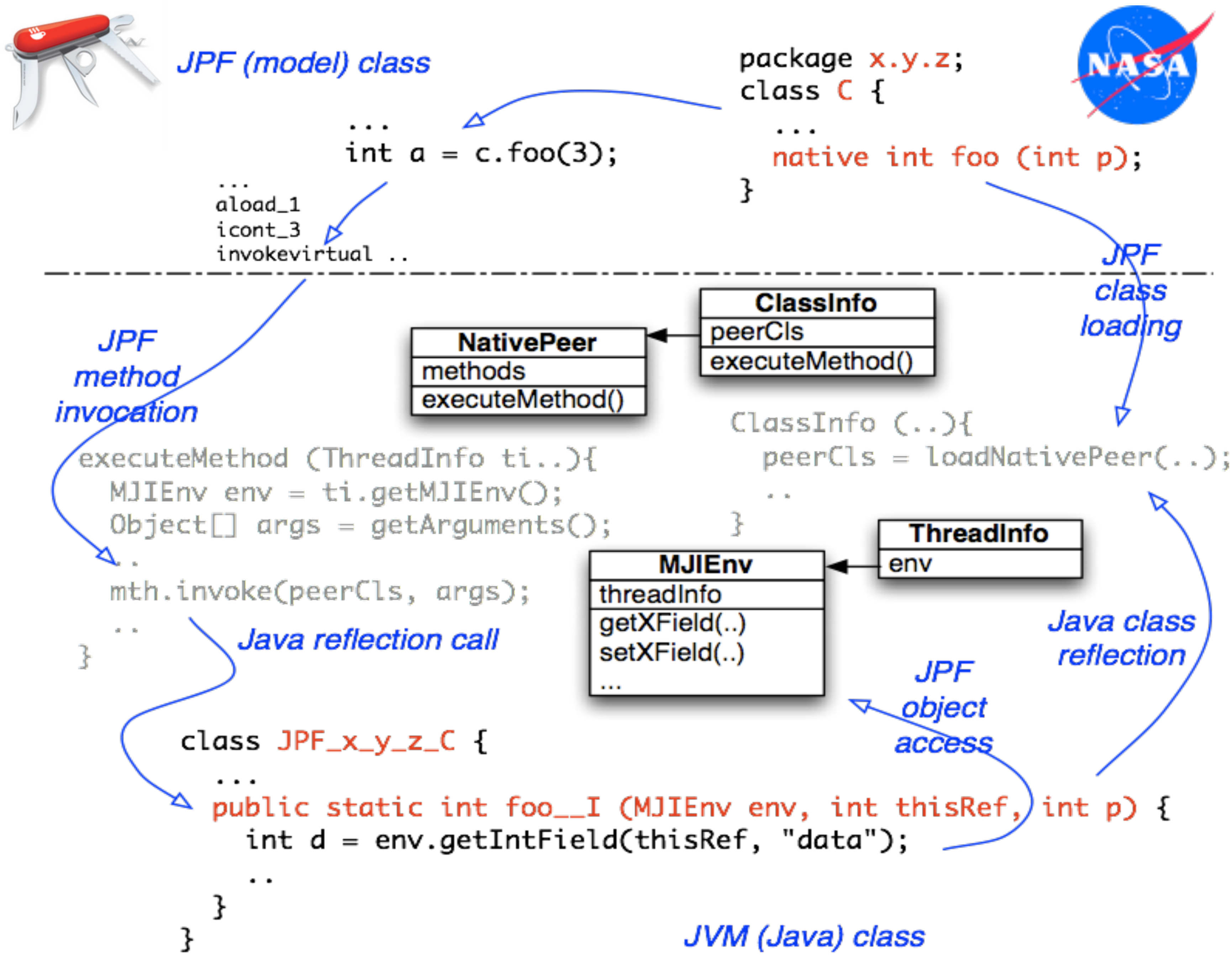


- *field access*
- *object conversion*
- *JPF intrinsics access*

```
class JPF_x_y_z_MyClass {
  public static
      int foo__ILjava_lang_String__2 (MJIEnv env, int objRef,
                                          int i, int sRef) {
    String s = env.getStringObject(sRef);
    ..
    int ref = env.newString(..);
    return ref;
  }
}
```

Java Class

"NativePeer" Class



```
package x.y.z;
class C {
  ...
  native int foo (int p);
}
```

```
...
int a = c.foo(3);
```

```
...
aload_1
icont_3
invokevirtual ..
```

ClassInfo
peerCls
executeMethod()

NativePeer
methods
executeMethod()

```
ClassInfo (..){
  peerCls = loadNativePeer(..);
  ..
}
```

```
executeMethod (ThreadInfo ti..){
  MJIEnv env = ti.getMJIEnv();
  Object[] args = getArguments();
  ..
  mth.invoke(peerCls, args);
  ..
}
```

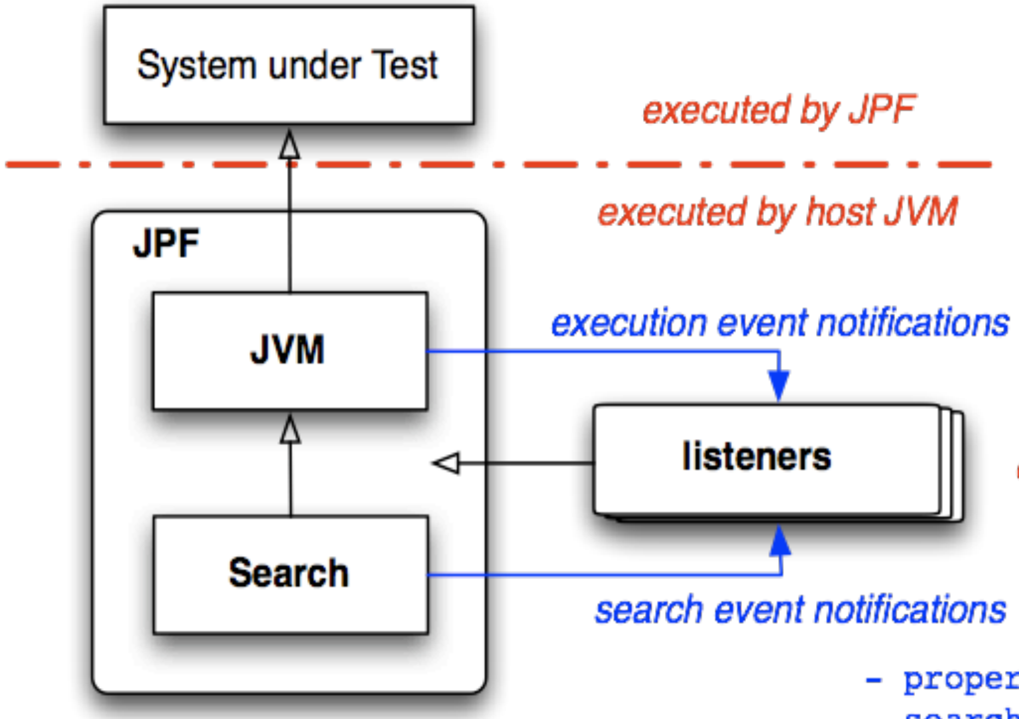
ThreadInfo
env

MJIEnv
threadInfo
getXField(..)
setXField(..)
...

```
class JPF_x_y_z_C {
  ...
  public static int foo__I (MJIEnv env, int thisRef, int p) {
    int d = env.getIntField(thisRef, "data");
    ..
  }
}
```

Listeners

```
- classLoaded()
- threadScheduled()
- threadNotified()
  ...
- executeInstruction()
- instructionExecuted()
- objectCreated()
  ...
- exceptionThrown()
  ...
- choiceGeneratorAdvanced()
  ...
```



System under Test

executed by JPF

executed by host JVM

JPF

JVM

execution event notifications

listeners

Search

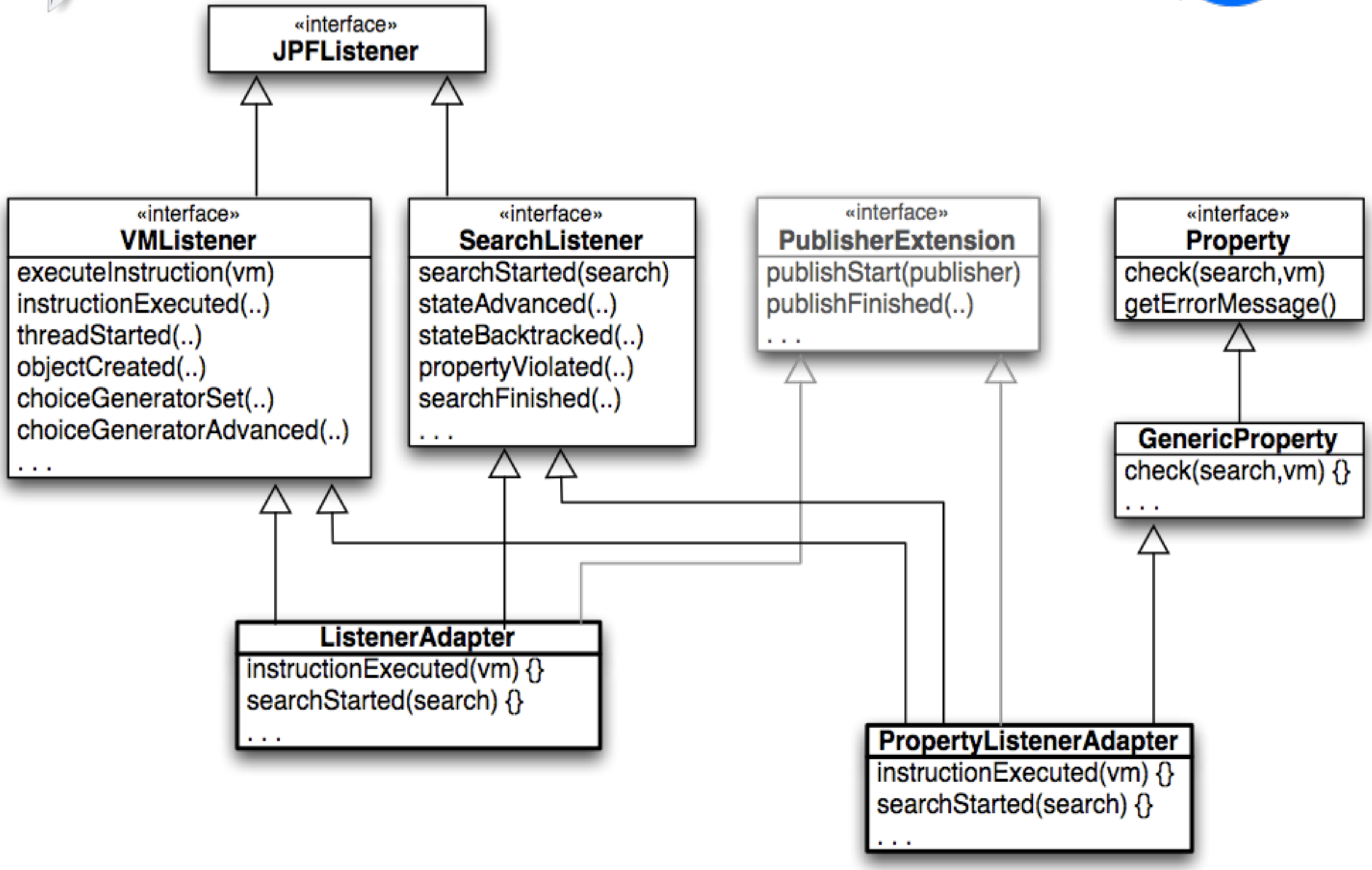
search event notifications

configured

```
- +listener=<listener-class>
- @JPFConfig(..)
- listener.autoload=<annotations>
- jpf.addListener(..)
  ...
```

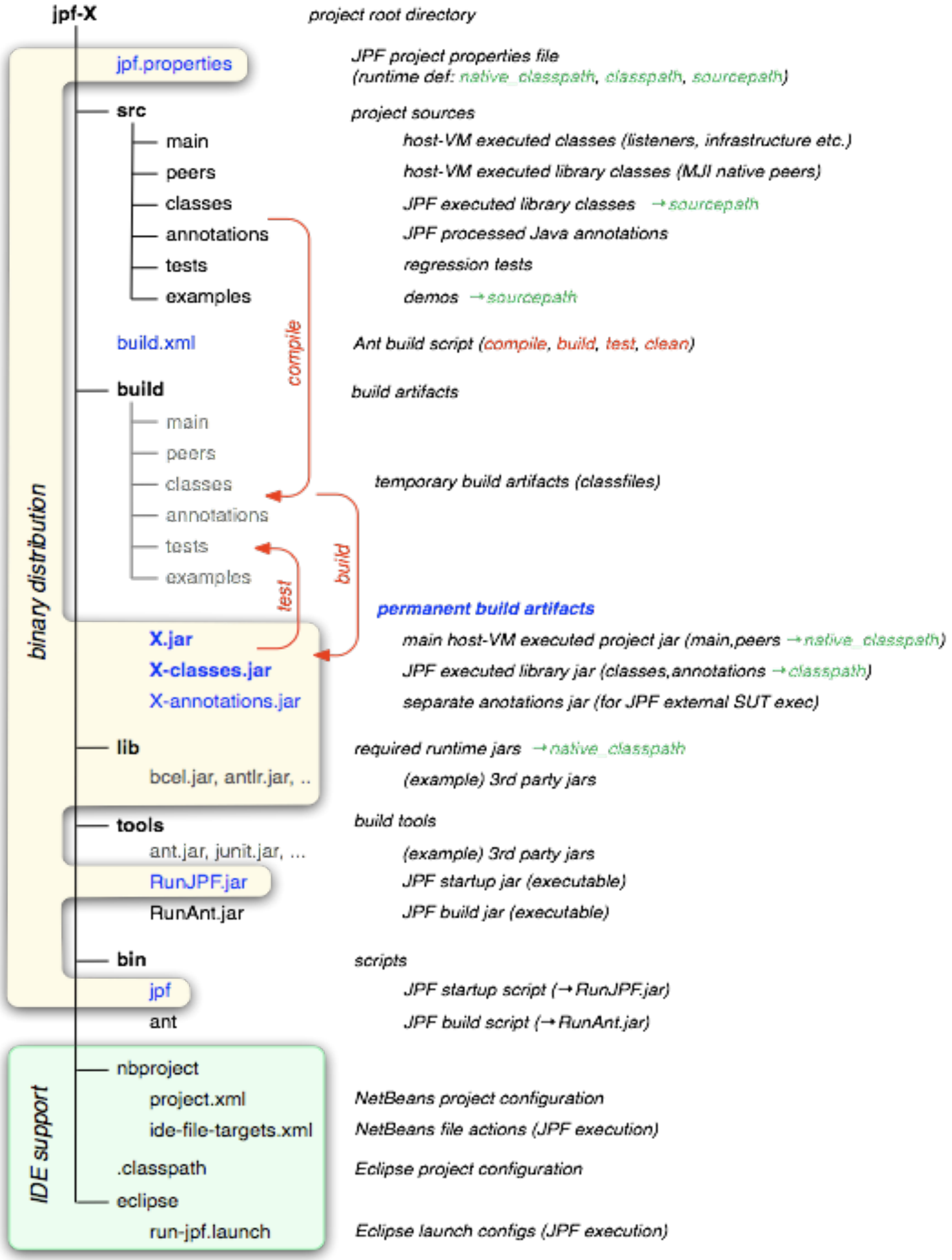
```
- propertyViolated()
- searchContraintHit
- searchFinished()
  ...
```

Design Hierarchy



Checking NonNull Annotation on Return

```
public class NonnullChecker extends ListenerAdapter {
...
public void executeInstruction (JVM vm) {
   Instruction insn = vm.getLastInstruction();
   ThreadInfo ti = vm.getLastThreadInfo();

   if (insn instanceof ARETURN) { // check @NonNull method returns
     ARETURN areturn = (ARETURN)insn;
     MethodInfo mi = insn.getMethodInfo();
     if (areturn.getReturnValue(ti) == null) {
       if (mi.getAnnotation(“java.annotation.Nonnull”) != null) {
         Instruction nextPc = ti.createAndThrowException(
                                          "java.lang.AssertionError",
                                          "null return from @Nonnull method: " +
                                          mi.getCompleteName());
         ti.setNextPC(nextPC);
         return;
       }
     }
..
```



jpf-X	project root directory
jpf.properties	JPF project properties file (runtime def: native_classpath, classpath, sourcepath)
src	project sources
main	host-VM executed classes (listeners, infrastructure etc.)
peers	host-VM executed library classes (MJI native peers)
classes	JPF executed library classes →sourcepath
annotations	JPF processed Java annotations
tests	regression tests
examples	demos →sourcepath
build.xml	Ant build script (compile, build, test, clean)
build	build artifacts
main	
peers	
classes	temporary build artifacts (classfiles)
annotations	
tests	
examples	
	permanent build artifacts
X.jar	main host-VM executed project jar (main,peers →native_classpath)
X-classes.jar	JPF executed library jar (classes,annotations →classpath)
X-annotations.jar	separate anotations jar (for JPF external SUT exec)
lib	required runtime jars →native_classpath
bcel.jar, antlr.jar, ..	(example) 3rd party jars
tools	build tools
ant.jar, junit.jar, ...	(example) 3rd party jars
RunJPF.jar	JPF startup jar (executable)
RunAnt.jar	JPF build jar (executable)
bin	scripts
jpf	JPF startup script (→RunJPF.jar)
ant	JPF build script (→RunAnt.jar)
nbproject	
project.xml	NetBeans project configuration
ide-file-targets.xml	NetBeans file actions (JPF execution)
.classpath	Eclipse project configuration
eclipse	
run-jpf.launch	Eclipse launch configs (JPF execution)

JPF and JUnit

- derive your test cases from `gov.nasa.jpf.util.test.TestJPF`
- run normally under JUnit or from Ant `<junit ..>` task
- be aware of that test case is run by JVM *and* JPF

```
public class ConstTest extends TestJPF {
  static final String[] JPF_ARGS = { "+listener=.aprop.listener.ConstChecker" };

  //--- standard driver to execute single test methods
  public static void main(String[] args) {
    runTestsOfThisClass(args);
  }

  //--- the test methods
 @Test
  public void testStaticConstOk () {
    if (verifyNoPropertyViolation(JPF_ARGS)){
      ConstTest.checkThis();
  } }
 …
```

Verification goal

code checked by JPF

Obtaining JPF

- Mercurial repositories on http://babelfish.arc.nasa.gov/hg/jpf/{jpf-core,jpf-aprop,...}
- Eclipse Steps

  (1) Get Mercurial

  (1) Eclipse Update site: http://cbes.javaforge.com/update

  (2) Get jpf-core

  (1) File – Import – Mercurial - Clone repository using Mercurial - Next

  (2) Specify http://babelfish.arc.nasa.gov/hg/jpf/jpf-core

  (3) Check the box for 'Search for .project files in clone and use them to create projects'

  (4) Finish

  (3) Build

  (1) Project – Properties - Select Builders - Ant Builder - Click Edit

  (2) Click JRE tab - Separate JREs - Installed JREs

  (3) Pick a JDK 1.6xxx…JRE will not find javac

Running JPF (1)

- Create site.properties in $(user.home)/.jpf
  - One line is enough for now:
  - $(user.home)/My Documents/workspace/jpf-core
- Install Eclipse Plugin (from the website description)
  - Ensure that you are running Eclipse >= 3.5 (Galileo)
  - In Eclipse go to Help -> Install New Software
  - In the new window selected "Add"
  - The name is up to you but, set "Location" to http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/install/eclipse-plugin/update/
  - From the "Work with:" drop down menu select the update site that you just entered from the previous step
  - Check the "Eclipse-JPF" check box, select "Next" and go through the install process.

Running JPF (2)

- Right click on *.jpf file and pick “Verify”
  - Go to src/examples and right click on oldclassic.jpf
  - Should see a deadlock!

Configuring JPF

- almost nothing in JPF is hardwired ⇒ great flexibility but config can be intimidating
- all of JPFs configuration is done through Java properties (but with some extended property file format)
  - keyword expansion `jpf-root = ${user.home}/jpf`
    - previously defined properties
    - system properties
  - append `extensions+=,jpf-aprop` no space between key and ‘+’ !
  - prepend `+peer_packages=jpf-symbc/build/peers,`
  - directives
    - dependencies `@requires jpf-awt`
    - recursive loading `@include ../jpf-symbc/jpf.properties`
- hierarchical process
  - system defaults (from jpf.jar)
  - site.properties
  - project properties from all site configured projects (<project-dir>/jpf.properties)
  - current project properties (./jpf.properties)
  - selected application properties file (*.jpf)
  - command line args (e.g. `bin/jpf +listener=.listeners.ExecTracker ...`)

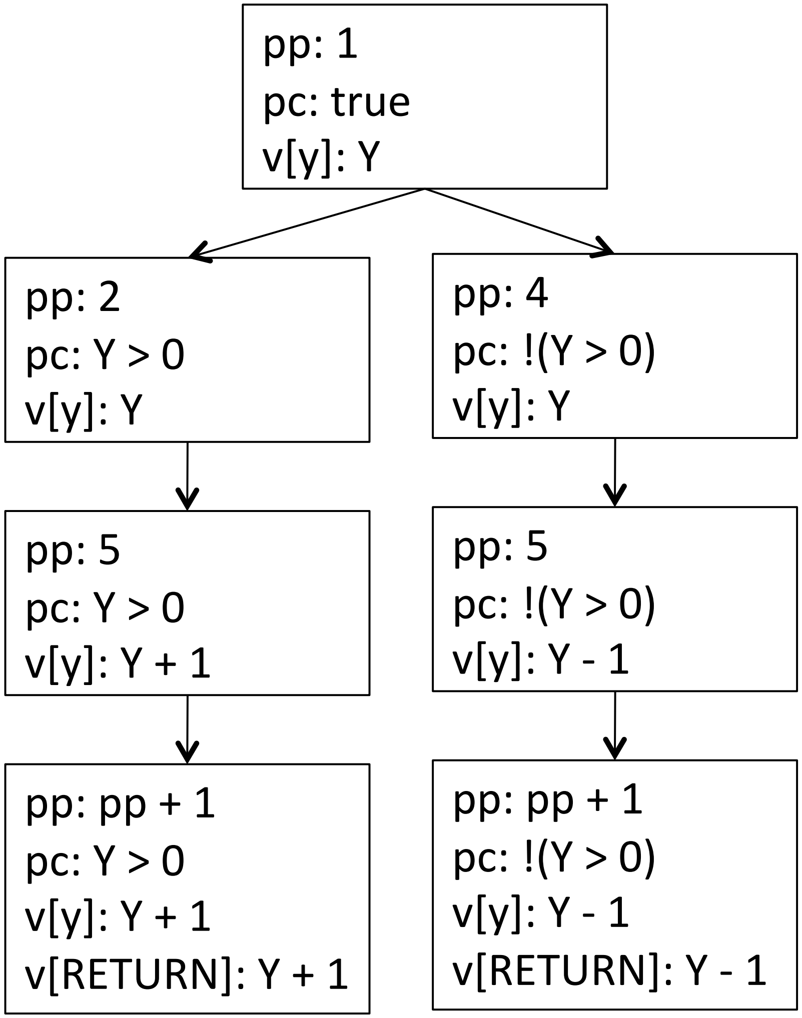
Demo

Automated Test Case generation

- Symbolic Execution

```
int m(int y){
1:  if (y>0)
2:    y++;
3:  else
4:     y--;
5:  return y;
}
```

m_{sum}=
{((Y>0), RETURN=Y+1),
!(Y>0), RETURN=Y-1)}



pp: 1
pc: true
v[y]: Y

pp: 2
pc: Y > 0
v[y]: Y

pp: 4
pc: !(Y > 0)
v[y]: Y

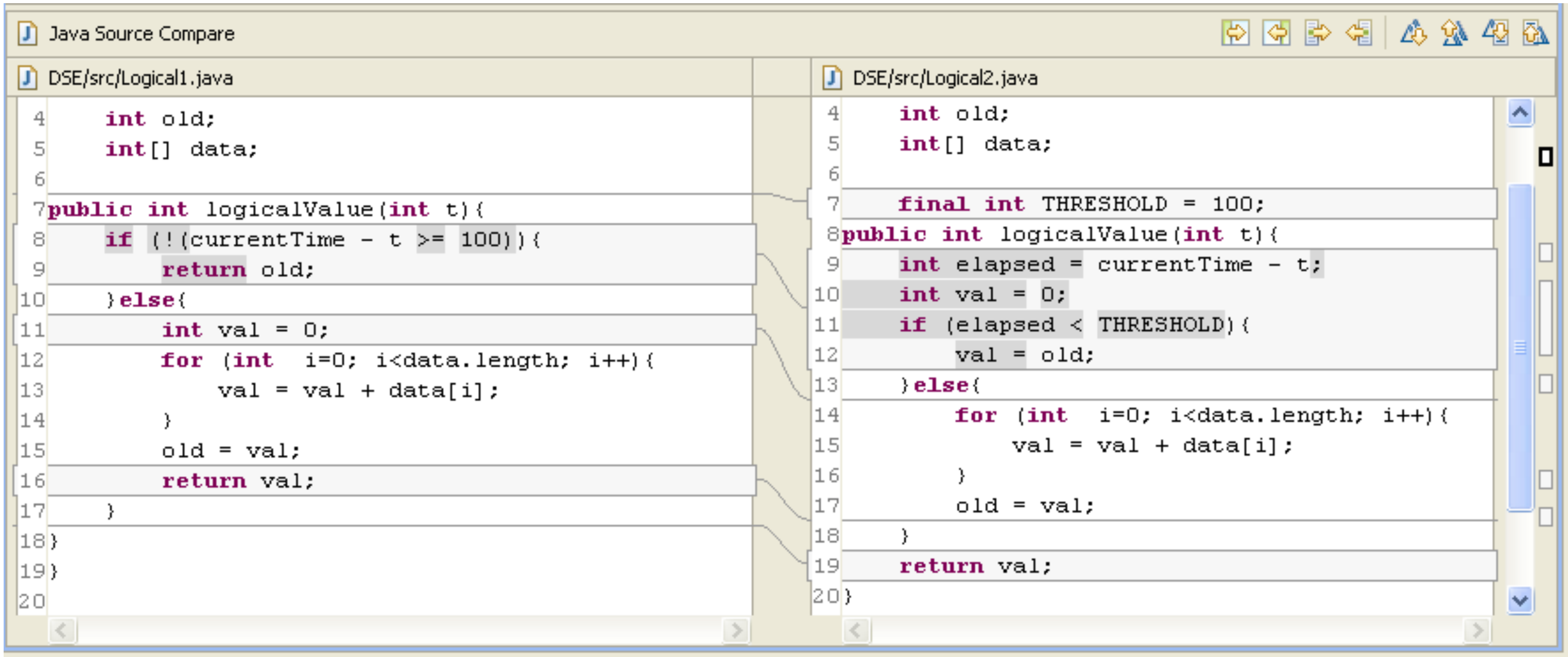
pp: 5
pc: Y > 0
v[y]: Y + 1

pp: 5
pc: !(Y > 0)
v[y]: Y - 1

pp: pp + 1
pc: Y > 0
v[y]: Y + 1
v[RETURN]: Y + 1

pp: pp + 1
pc: !(Y > 0)
v[y]: Y - 1
v[RETURN]: Y - 1

Agile Development

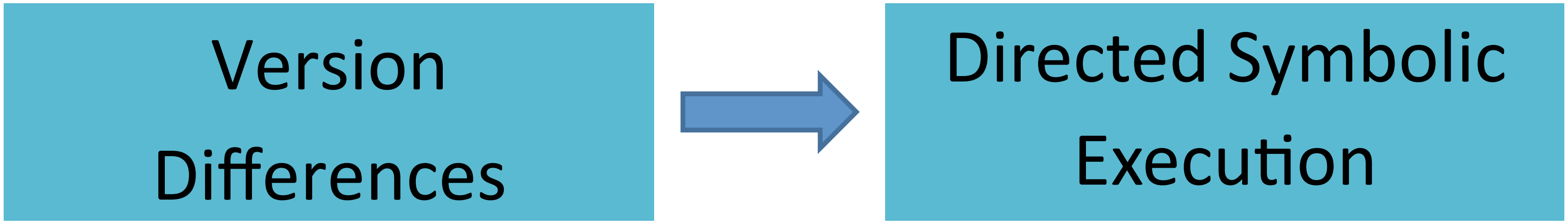


```java
    int old;
    int[] data;

public int logicalValue(int t){
    if (!(currentTime - t >= 100)){
        return old;
    }else{
        int val = 0;
        for (int  i=0; i<data.length; i++){
            val = val + data[i];
        }
        old = val;
        return val;
    }
}
}
```

```java
    int old;
    int[] data;

    final int THRESHOLD = 100;
public int logicalValue(int t){
    int elapsed = currentTime - t;
    int val = 0;
    if (elapsed < THRESHOLD){
        val = old;
    }else{
        for (int  i=0; i<data.length; i++){
            val = val + data[i];
        }
        old = val;
    }
    return val;
}
```

Evolution

- Regression analysis technique focused on version differences
- Combines syntactic and semantic analysis techniques
- Identify and characterize effects of program changes



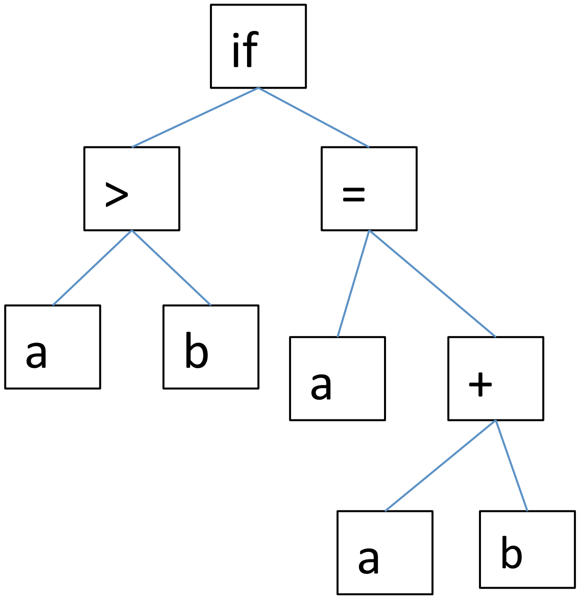
Version Differences

Directed Symbolic Execution

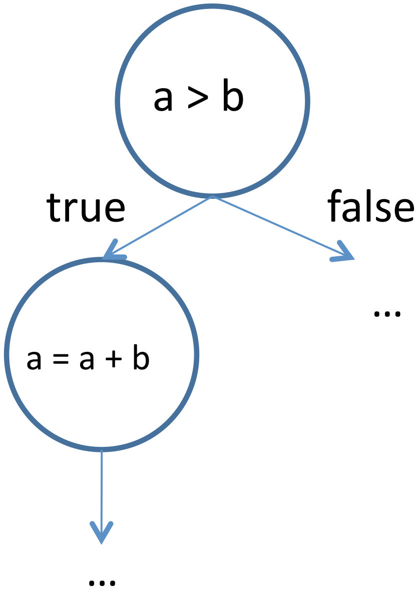
Background

- Abstract Syntax Tree

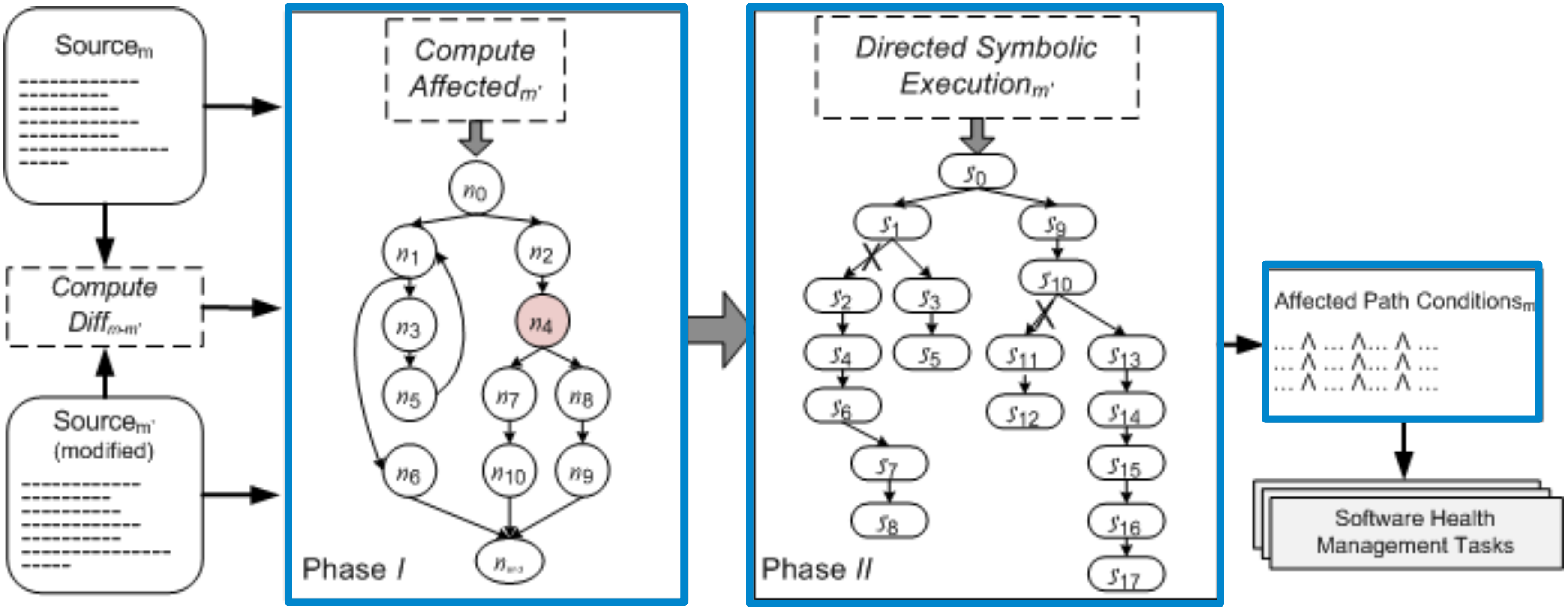
```
if (a > b)
  a = a + b;
```



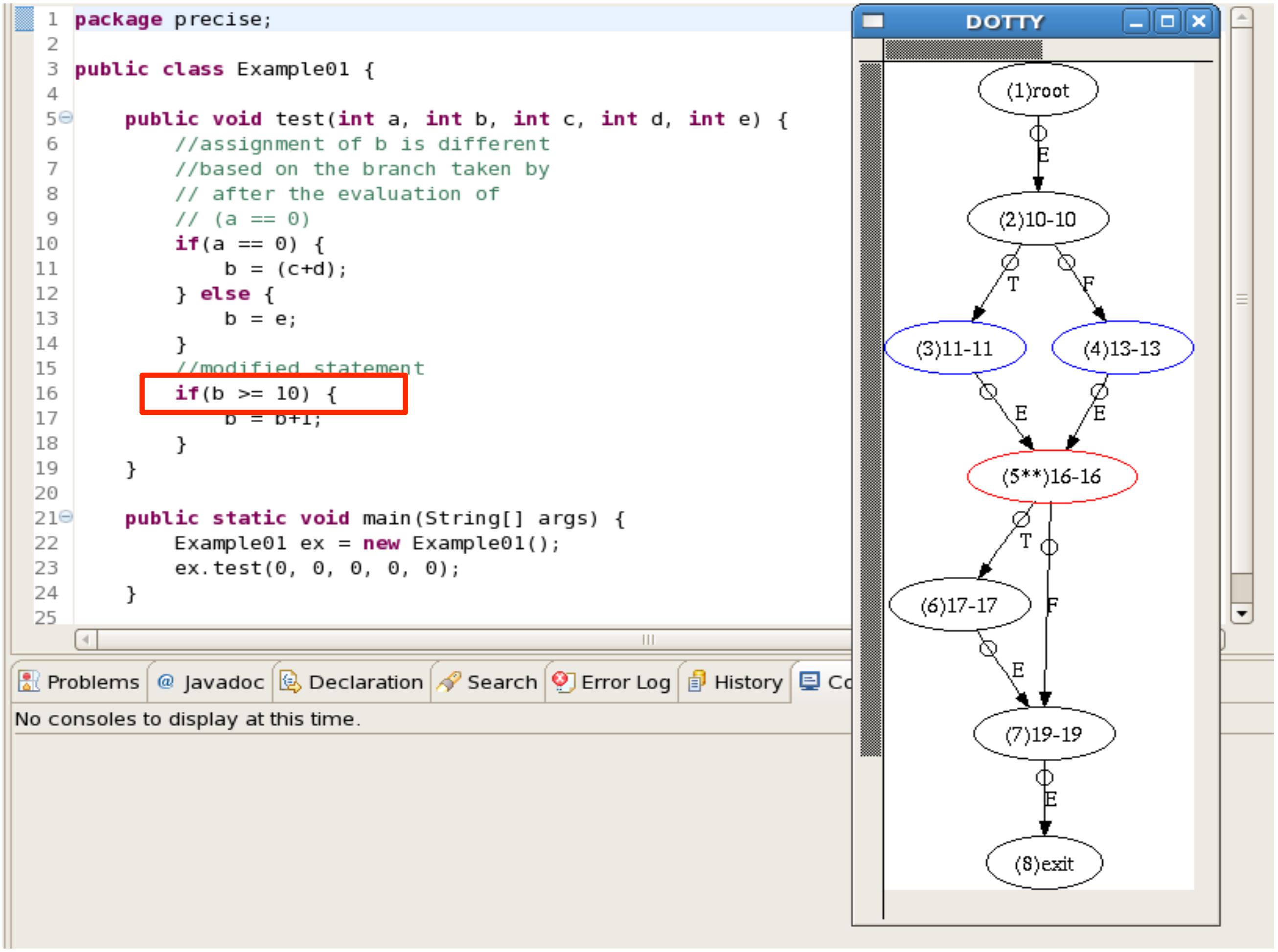
□ Control Flow Graph



Incremental Execution



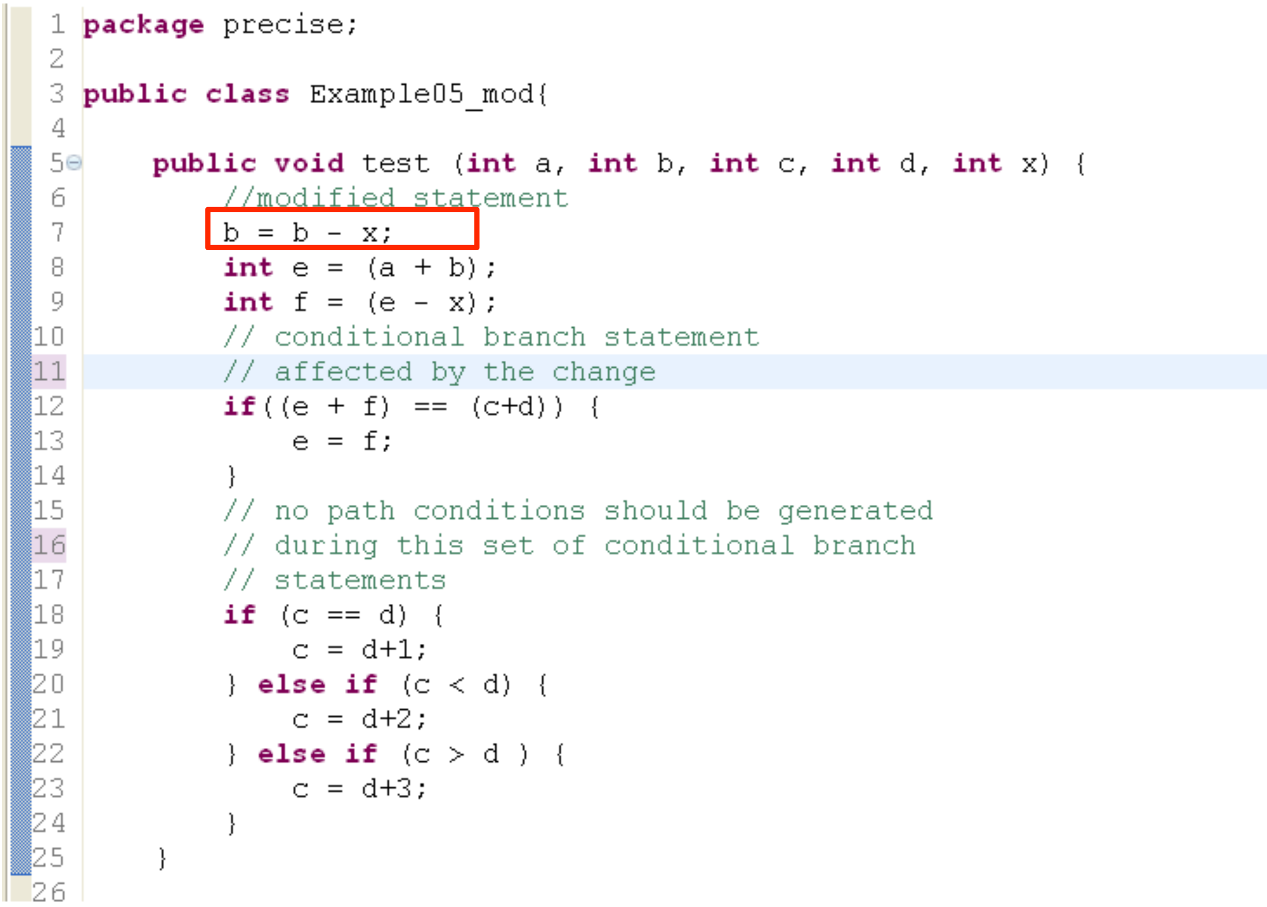
Incremental Analysis



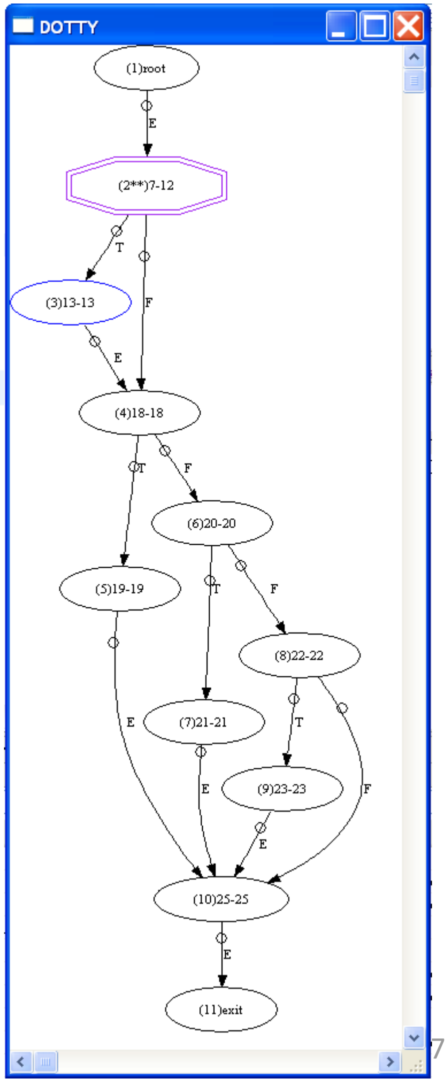
```
package precise;

public class Example01 {

    public void test(int a, int b, int c, int d, int e) {
        //assignment of b is different
        //based on the branch taken by
        // after the evaluation of
        // (a == 0)
        if(a == 0) {
            b = (c+d);
        } else {
            b = e;
        }
        //modified statement
        if(b >= 10) {
            b = b+1;
        }
    }

    public static void main(String[] args) {
        Example01 ex = new Example01();
        ex.test(0, 0, 0, 0, 0);
    }

```

No consoles to display at this time.

Incremental Analysis



```
 1 package precise;
 2 
 3 public class Example05_mod{
 4 
 5     public void test (int a, int b, int c, int d, int x) {
 6         //modified statement
 7         b = b - x;
 8         int e = (a + b);
 9         int f = (e - x);
10         // conditional branch statement
11         // affected by the change
12         if((e + f) == (c+d)) {
13             e = f;
14         }
15         // no path conditions should be generated
16         // during this set of conditional branch
17         // statements
18         if (c == d) {
19             c = d+1;
20         } else if (c < d) {
21             c = d+2;
22         } else if (c > d ) {
23             c = d+3;
24         }
25     }
26 
```



2 *affected* path conditions

Extensions!