Environment Modeling with JavaPathfinder (JPF) Part 2 - JPF Introduction

Oksana Tkachuk
SGT / NASA Ames Research Center
<oksana.tkachuk@nasa.gov>

Peter C. Mehlitz
SGT / NASA Ames Research Center
<peter.c.mehlitz@nasa.gov>
Roadmap

- Motivating Example
- What is JPF?
- Major Extension Mechanisms
- Anatomy of a JPF Installation
- Documentation
this is where we want to go
“not your usual model checking”
small system under test
(7 classes, ~500 loc)
graphical user interface app
⇒ uses huge framework (Swing, AWT)
400+ classes
user input and concurrency ⇒ hard to test
needs tool that supports
environment modeling and
faithful execution

jpf src/examples/RobotManager-thread.jpf
...
================================ error #1
NoUncaughtExceptionsProperty
java.lang.NullPointerException: Calling 'processSequence(String)' on null object
at RobotManager.sendSequence(RobotManager.java:265)
at RobotManagerView.sendSequence(RobotManager.java:565)
...
What Is JPF - Not a Black Box Tool

✦ a model checker? a virtual machine? ..

✦ and the answer is: both, and more - it depends on you

✦ not a monolithic, black box tool

No “one size fits all” - Extensibility is Paramount

✦ the quest of today: learn what is in the toolbox to find out how you can adapt JPF to your needs
What is JPF - Core and Extensions

- partitioned into core and extensions
- core consists of Search and Virtual Machine
- several extension mechanism

Java bytecode program

System under Test

JPF configuration

Virtual Machine

Search Strategy

VM driver

result + trace

publisher

verification artifacts

reports, test data
What is JPF - VM running in VM

- main stumbling block is recursive nature of JPF
- JPF’s Virtual Machine (VM) runs as a normal Java application on top of a host Java VM

![Diagram showing JPF installation process]

system under test

JPF installation

native Java installation

platform OS

application

library

classes

JPF (VM)

standard library classes

host JVM

native libraries

verification target

*.class

jpf.jar

rt.jar

classpath

native_classpath

standard Java installation

native Java installation

JPF modules

Monday, May 28, 12
JPF Extensions - Main Mechanisms

✦ ChoiceGenerators - Transitions and States
✦ VM and Search Listeners - the JPF “plugins”
✦ Native Peers - Model Java Interface (MJI)
✦ Object Attributes - data flow
✦ Instruction Factories - execution semantics
✦ ... and many more (Serializers, Publishers, PublisherExtensions, ...)

used in lab

FYI
JPF Extensions - ChoiceGenerators

- nature of state space branches *not* hardwired into JPF
- user can introduce new choice types by providing ChoiceGenerators

**iterator objects**

for statespace branches

```
ChoiceGenerator<T>
choices : set
nextChoice
hasMoreChoices()
advance()
getNextChoice() : T
```

JPF

---

**JPF core**

- IntChoiceGenerator ...
- ThreadChoiceGenerator

**JPF extension**

- UIActionGenerator ...

Monday, May 28, 12
JPF Extensions - Listeners

✦ the primary JPF “plugins”
✦ used to monitor execution & search (e.g. to implement properties)
✦ runtime configurable
JPF Extensions - Native Peers

- preferred way to model libraries that use platform specific code (I/O, GUI frameworks etc.)
- JPF’s Model Java Interface (MJI) resembles Java Native Interface (JNI)
- lowers execution from JPF VM into host VM

![Diagram showing the interaction between model class, peer class, native classpath, JPF extension, and JPF core. The diagram illustrates the process of calling a method from a model class to a peer class, and the execution flow between JPF and the host JVM.]
POJO’s that can be attached to many types of JPF objects

travel with data - good to implement data flow properties
JPF Extensions - Instruction Factories

- execution semantics encapsulated in JPF Instruction objects
- can be replaced, e.g. to add additional operations

```
*\.class

MethodInfo
- factory
- Instruction[] code
- init (JavaClass)

<<InstructionFactory>>
- Instruction ifeq(..)
- ...

Instruction
- Instruction execute()

code[i] = factory.ifeq(..);

DefaultInstructionFactory

SymbolicInstructionFactory

IFEQ
- execute()

concrete value
instruction set

symbolic value
instruction set

JPF core

JPF extension
```
all JPF projects share uniform directory layout
use ANT based build system
use same configuration scheme
support binary distributions
3rd party tools & libraries can be included (self-contained)
projects (should) have examples and regression test suites
projects (should) have IDE configuration (NB,Eclipse)
JPF Anatomy - Configuration

Configuration Levels

1. site properties
   ~/.jpf/site.properties
   - project locations
   - pre-loaded projects

2. project properties
   <project>/jpf.properties
   - project class paths
   - project dependencies

3. application properties
   <project>/*.jpf
   - system-under-test
   - listeners, shells

4. command line
   - debugging
     (+key=value pairs)

command line property arguments

> bin/jpf [-log][-show] {+log.info=..} .../RobotManager.jpf

target = RobotManager
target_args = ...
@using = jpf-aprop
@import = ./my.properties
shell = .shell.basicsshell.BasicShell
listener = .aprop.listener.SharedChecker...

all jpf.properties in order of extensions

jpf-core = ${config_path}
jpf-core.native_classpath=\
   ${jpf-core}/build/jpf.jar;\n   ...
   ${jpf-core}/lib/bcel.jar;
jpf-core.classpath=\
   build/jpf-classes.jar
jpf-core.test_classpath=\
   build/tests
jpf-core.sourceforge=\
   src/classes...

jpf-awt-shell = ${config_path}
@using = jpf-awt
jpf-awt-shell.native_classpath=...
jpf-awt-shell.classpath=...
...

jpf.properties in current directory

site properties

jpf.home = ${user.home}/projects/jpf
jpf-core = ${jpf.home}/jpf-core
jpf-awt = ${jpf.home}/awt
jpf-shell = ${jpf.home}/jpf-shell
jpf-aprop = ...
...
extensions = ${jpf-core},${jpf-shell}
http://babelfish.arc.nasa.gov/trac/jpf

- public read access
- edit for account holders (also non-NASA)

bug tracking

- Trac ticket system

project blog

- announcements
- important changes

hierarchical navigation menu

- intro
- installation
- user docu
- developer docu
- extension projects
JPF Documentation & Help - Mailing List

✧ http://groups.google.com/group/java-pathfinder
✧ anyone can join & read, member only post
✧ moderate volume (~50 messages/month)
✧ subscribe on http://groups.google.com/group/java-pathfinder/subscribe
Conclusions

✨ JPF *not* a black box tool
✨ highly extensible
✨ can - and has to - be customized for domain specific verification
✨ takes some time to master
Conclusions

✦ JPF not a black box tool
✦ highly extensible
✦ can - and has to - be customized for domain specific verification
✦ takes some time to master

Thank You!
See you at the lab