# Software Model Checking

Cyrille Artho

Research Center for Information Security (RCIS),

National Institute of Advanced Industrial Science and Technology (AIST),

Tsukuba, Japan

`c.artho@aist.go.jp`

October 26, 2011

What is Java PathFinder?

◆ Java PathFinder = Java Virtual Machine
with backtracking capability.

◆ Typical use: as a **software model checker.**

◆ Full state space is searched:

– Thread interleavings.
– Non-deterministic decision
(such as I/O failure or not).

◆ Core development by NASA Ames,
but many collaborators and outside users.

Non−deterministic
decision

success
case

failure
case
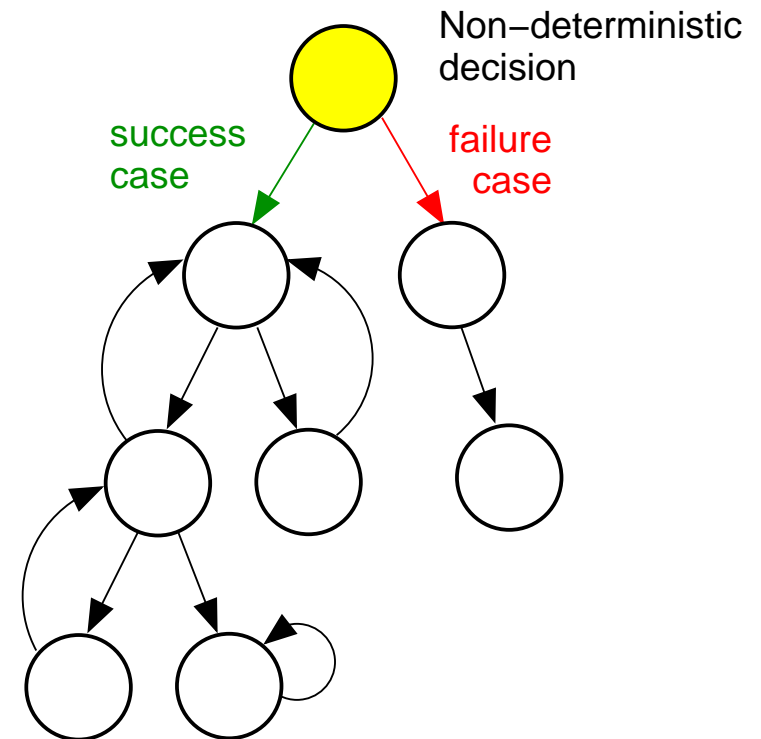
Two versions of Java PathFinder (JPF)

JPF 4

◆ Old „stable" binary release.

◆ No longer under development.

◆ No more changes.

◆ Fewer optimizations.

◆ Error trace has many transitions.

◆ Easier to use on small examples.

◆ Configuration via command line.

JPF 6

◆ Source repository (devel. version).

◆ Under active development.

◆ May change at any time.

◆ Faster on large programs.

◆ Error trace tends to be shorter.

◆ Better for real projects.

◆ Configuration via config. files.

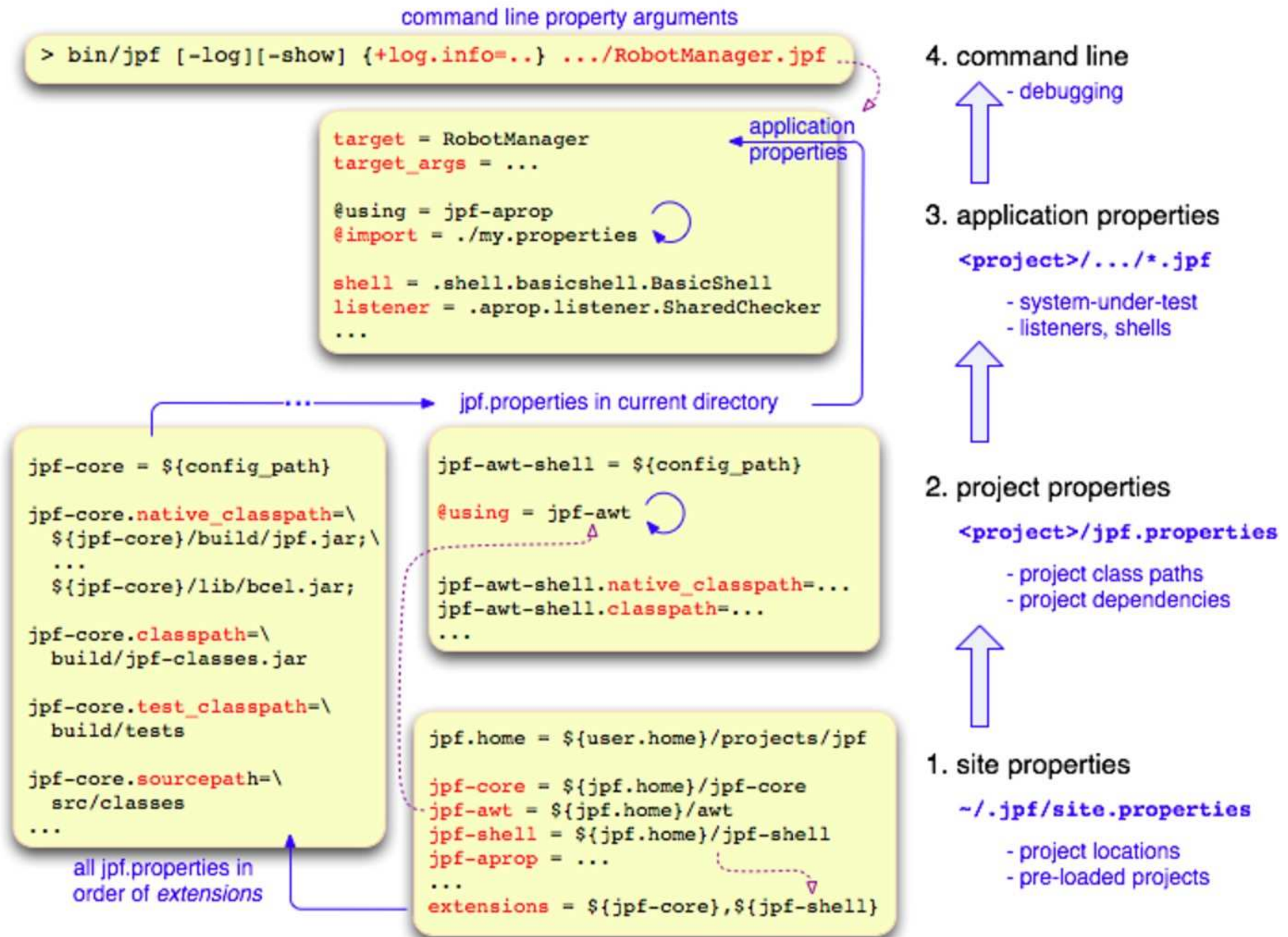**The upcoming exercises require JPF 6.**

Exercises: System requirements

◆ Version 6 is a snapshot from the new JPF web page:
   http://babelfish.arc.nasa.gov/trac/jpf
   http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/start

◆ You first need to install **Mercurial** (**hg**) to download the source code:
   http://mercurial.selenic.com/wiki/Download

◆ After installing Mercurial, clone the JPF source code repository:
   http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/repositories

◆ Run the following commands:
   mkdir ~/jpf

   cd ~/jpf

   hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-core

   ./bin/ant

◆ Configuration files have been prepared to be used with JPF 6.

Configuration under JPF 6 (slide from JPF manual)

command line property arguments

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

application properties

```
target = RobotManager
target_args = ...

@using = jpf-aprop
@import = ./my.properties

shell = .shell.basicshell.BasicShell
listener = .aprop.listener.SharedChecker
...
```

jpf.properties in current directory

```
jpf-core = ${config_path}

jpf-core.native_classpath=\
  ${jpf-core}/build/jpf.jar;\
  ...
  ${jpf-core}/lib/bcel.jar;

jpf-core.classpath=\
  build/jpf-classes.jar

jpf-core.test_classpath=\
  build/tests

jpf-core.sourcepath=\
  src/classes
...
```

all jpf.properties in order of extensions

```
jpf-awt-shell = ${config_path}

@using = jpf-awt

jpf-awt-shell.native_classpath=...
jpf-awt-shell.classpath=...
...
```

```
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}
```

4. command line
- debugging

3. application properties

`<project>/.../*.jpf`
- system-under-test
- listeners, shells

2. project properties

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

1. site properties

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

© Cyrille Artho

4

Using JPF

◆ Details and configuration files will be introduced in upcoming lectures.

◆ First: **Make sure JPF is working properly!**

◆ Try
`~/jpf/jpf-core/bin/jpf src/examples/HelloWorld.jpf`

◆ This example should finish with a message containing „no errors detected".

◆ If there are problems, please check you Java installation (should be 6 or higher), and contact me if needed: `c.artho@aist.go.jp`.