unibz

SOFTWARE AND SYSTEMS ENGINEERING research group

Model-based testing under uncertainty

L2: Hands-on session & extension design/implementation

Matteo Camilli
matteo.camilli@unibz.it
https://matteocamilli.github.io

Formal Methods at Work @
Gran Sasso Science Institute (GSSI)
A.Y. 2019/20

GSSI

Outline

- How do I get setup?
  - By using Gradle from command line
  - By using your IDE of choice
- Icebreaker exercise
  - Try out pre-defined examples
- Advanced exercise
  - The adaptive testing strategy
  - Hints

How do I get setup?

- Requirements
- Using Gradle from command line
- Using your IDE of choice

Requirements

- MBT-module requirements
  - JDK 1.8 — https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - R environment — https://www.r-project.org/
    - R packages
      - rJava — https://cran.r-project.org/web/packages/rJava/index.html
      - MCMCpack — https://cran.r-project.org/web/packages/MCMCpack/index.html
      - HDInterval — https://cran.r-project.org/web/packages/HDInterval/index.html
  - Hopefully.. a unix-like operating system ;-)

- MBT-module
  - Open source Git repository — https://github.com/SELab-unimi/mbt-module
  - Git clone (SSH protocol) — git clone git@github.com:SELab-unimi/mbt-module.git

Environment settings

- Set the JVM 1.8 as default
- cd into the met-module root directory
- Set the following System-dependent variables
  - JRI path as JVM option java.library.path inside the build.gradle file

```
47
48  applicationDefaultJvmArgs = ['-Dorg.slf4j.simpleLogger.defaultLogLevel=info',
49                               '-Djava.library.path=/usr/local/Cellar/r/3.6.2/lib/R/library/rJava/jri']
50
```

  - To find your own JRI path execute the following commands
    - # R console

```
> .libPaths()
[1] "/usr/local/Cellar/r/3.6.2/lib/R/library"
```

    - # bash

```
> cd /usr/local/Cellar/r/3.6.2/lib/R/library/rJava/jri
> pwd
/usr/local/Cellar/r/3.6.2/lib/R/library/rJava/jri
```

Environment settings (2)

- Set the following System-dependent variables
  - R_HOME environment variable (e.g., `export R_HOME=/my/r_home/path`
    - Tip: put this in your .bashrc or .bash_profile
    - To find your own R_HOME path execute the following command
      - # R console

```
> R.home()
[1] "/usr/local/Cellar/r/3.6.2/lib/R"
```

Command line build + execution

- From the mot-module directory
  - # bash

```
> ./gradlew clean build
…
BUILD SUCCESSFUL in 16s

> ./gradlew run -PappArgs="['-i', 'src/main/resources/tas.jmdp']"

Value Iter. Solver  (Avg)
12 states found.
********* Best Policy *********
…
Log trace
…
#test limit reached.
********* Monitor report *********
…
```

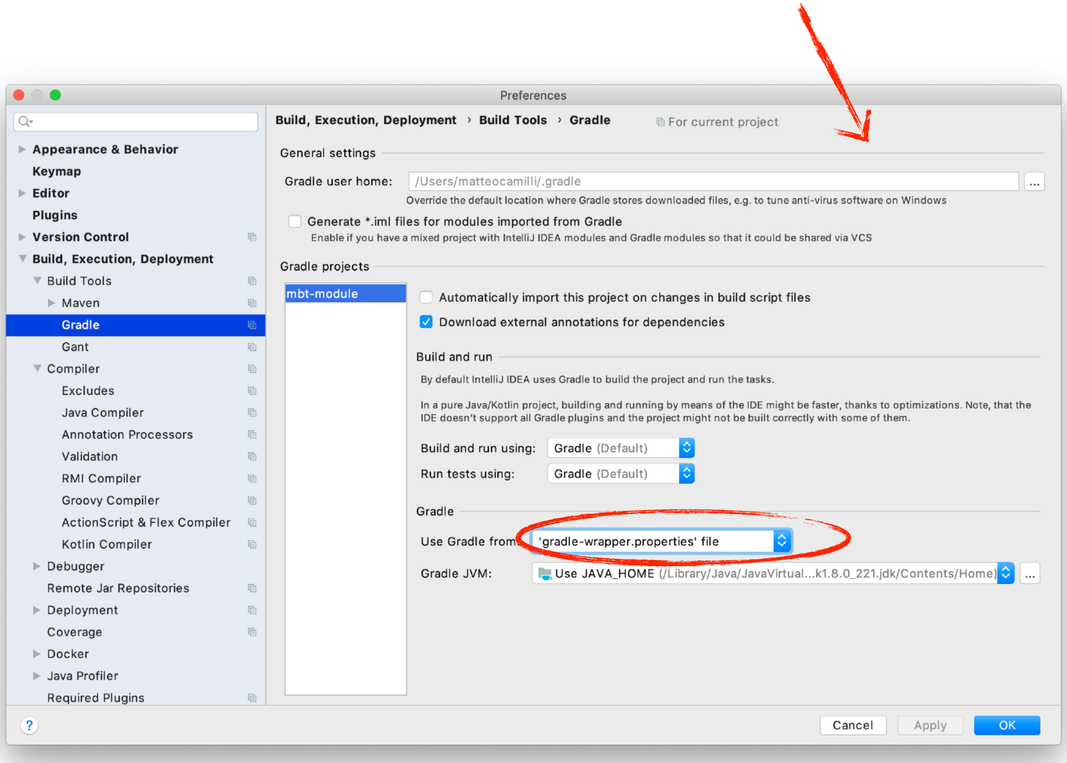
Import into the IDE

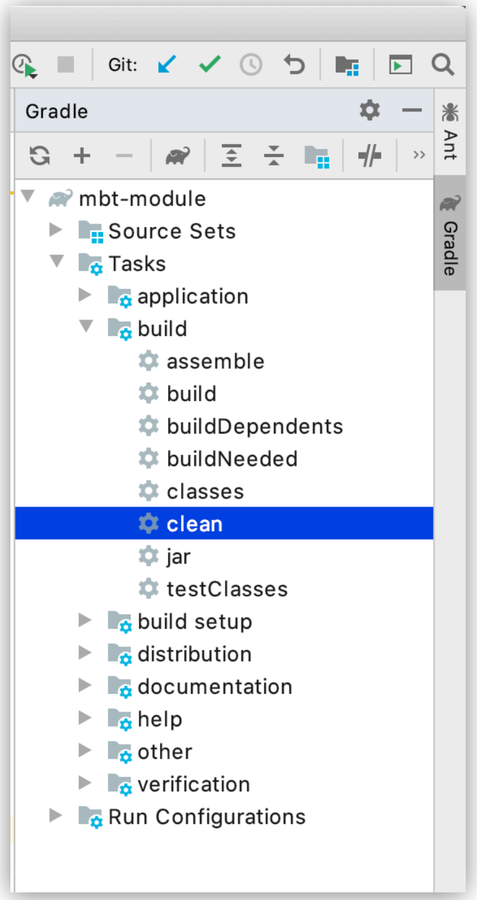
- Intellij Idea IDE
  - https://www.jetbrains.com/idea/
  - Ultimate edition needed (it includes the aspectJ plugin)
- Eclipse IDE
  - https://www.eclipse.org/downloads/
  - AspectJ Development Tools (AJDT) plugin (https://marketplace.eclipse.org/content/aspectj-development-tools)

- Once you have the IDE (with the aspectJ plugin)
  - Open/import an existing gradle project
  - Select the mbt-module directory

Build from the IDE

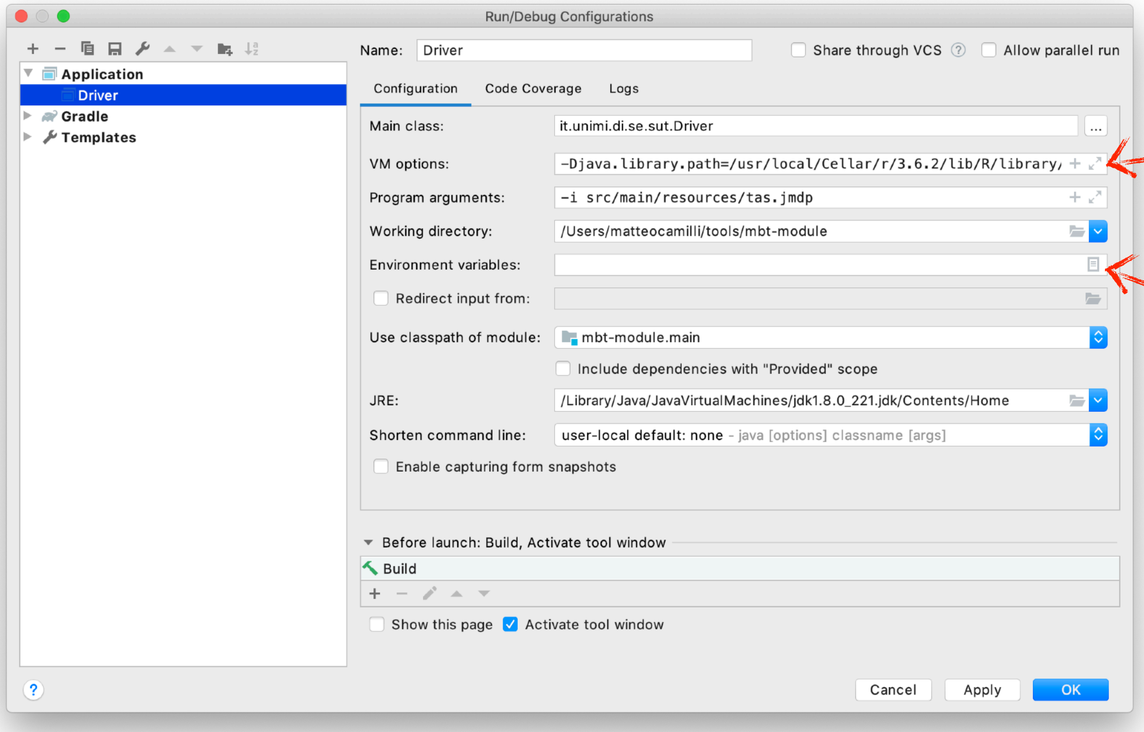
- Clean + build
  - Use the IDE to run **gradle** commands: **clean** and then **build**
  - Be sure that **gradle-wrapper** option is selected





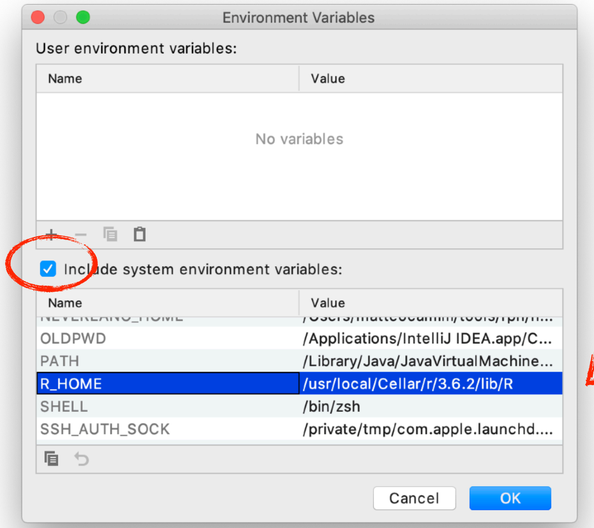
Run from the IDE

- Run configuration
  - Execute the Driver class with the following configuration



VM options (taken from build.gradle)

-Djava.library.path=/usr/local/Cellar/r/3.6.2/lib/R/library/rJava/jri
-Dorg.slf4j.simpleLogger.defaultLogLevel=warn



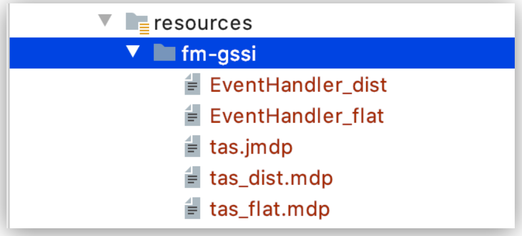
R_HOME must be defined

Icebreaker exercise

- Execute pre-defined examples

Examples

- The TAS example
  - The resources/fm-gssi directory contains
    - [SUT] definition of the (simulated) TAS — tas.jmdp
    - [flat strategy] model and test harness — tas_flat.mdp + EventHandler_flat
    - [dist strategy] model and test harness — tas_dist.mdp + EventHandler_dist



- Select and run a strategy (e.g., the dist strategy)
  - Copy both tas.jmdp and tas_dist.mdp in the resource folder (i.e., the parent of fm-gssi)
  - Replace the content of it.di.unimi.di.se.monitor.EventHandler.aj with EventHandler_dist
  - Execute gradle clean + build and then run
    - This executes the dist strategy on the TAS example with an effort limit of #tests = 2k

Exercise

- Execute both the flat and the dist strategy multiple times (e.g., 10 times)
- Which one is likely to be superior in terms of overall delivered confidence (i.e., small HPD width) on the two uncertain regions defined upon the TAS? Why?

- Hints
  - The answer can be found by inspecting the definition of the Prior knowledge (tas_dist.mdp) and thinking about how the two strategies work
  - Reminder
    - Flat —> uniform sampling of the actions
    - Dist —> sampling of the actions by taking into account the HPD width of the regions

Advanced exercise

- Design/implement a new testing strategy

The adaptive strategy

- Problem
  - All the testing strategies considered up to this point are built considering a pre-computed and fixed set of best policies maximizing the probability of reaching each region (in isolation)
  - The incremental Posterior knowledge could be leveraged to update to update the best policies. This approach has potential impact in the cost-effectiveness in case of wrong initial assumptions

- Idea — design/implement an adaptive strategy by following this guideline
  - Periodically (e.g., every sample-size tests) update a reward structure depending on the HPD width
    - The reward associated to arcs of each region should be directly proportional to the HPD width of that region (i.e., the higher the confidence (small HPD width), the lower the reward)
  - Compute the best policy given the updated reward structure
  - Select actions by following the new best policy
  - Repeat until termination

The adaptive strategy (2)

- Hints
  - The testing strategies follow the strategy design pattern and extend the DecisionMaker class
  - The DecisionMaker receives information on HPD width of regions by means of the callback updateDistance invoked by the Monitor (see the DistanceDecisionMaker)

```
50      @Override
51      public void updateDistance(int stateIndex, double distance) {
52          hpdDistance.put(new IntegerState(stateIndex), distance);
53      }
```

DistanceDecisionMaker.java

  - The DecisionMaker interface should be extended with a new method updatePolicy called by the Monitor every sample-size observations

```
326                 if(tests % EventHandler.SAMPLE_SIZE >= EventHandler.SAMPLE_SIZE-1) {
327                     //log.info(coverageInfo.toString());
328                     if(EventHandler.TERMINATION_CONDITION == Termination.COVERAGE && coverageInfo.getCo
329                         log.info("[Monitor] convergence reached.");
330                         addEvent(Event.stopEvent());
331                     }
```

Monitor.java

else updatePolicy call

License of these slides