**ModelicaPy Capabilities**

**Development:** Development is primarily used for testing Modelica models. It includes unit testing and a validator for validating html syntax in .mo files.

* Error\_dictionary: importable dictionary that is used in regressiontest for error reporting
* Merger: class that merges a Modelica library with the ‘Annex60’ library
* Refactor:
  + \* create Modelica packages and autopopulate for example the
  + `package.mo` and `package.order` files
  + \* move Modelica classes include any associated `.mos` scripts,
  + reference results and images, and
  + \* rewrite the `package.order` file.
* Unittest: essentially useless script that just refers to the regressiontesttwo testing
* Validator: Class that validates ``.mo`` files for the correct html syntax.
* Regressiontesttwo: script that can perform unit testing on Modelica models with a variety of options and functions. See detailed breakdown below.

**Fmi:** This module provides functions to analyse FMUs. Return the input and state dependencies of an FMU as a dictionary.

* Only contains an \_\_init\_\_ script

**Io:** responsible for using dymat

* Outputfile: opens and reads the log file generated by a simulation
* Postprocess: This class contains static methods that can be used to create plots. For an example of a simple plot, see also the example
* Reporter: contains reporter class that is used to report errors

**Simulator:** Contains Simulator script that can simulate Modelica models

* Simulator: class that simulates Modelica models with extensive options including:
  + Adding pre and post processing
  + Set output directory
  + addModelModifier
  + getSimulatorSettings
  + set start and stop times
  + set tolerance, solver, number of intervals
  + actual simulation functions are simulate, translate, and simulate translated

**Tests:** contains scripts that test all scripts mentioned above against a set library.

**Breakdown of Regressiontesttwo**

runSimulation Line 12-57

* Function actually responsible for opening dymola and giving it the required commands
* Has cmd equal an array to pass to Popen command
* Makes log file in the directory in cwd and opens said log file, used as output parameter in Popen
* Ends with error reporting

Class Tester:

\_init\_ Line 12-207

* Sets class variables and instance
* Parameters include…
  + Check\_html=True
  + Executable=”dymola” (can also be omc”)
  + cleanup=true

setMosLocation Line 215-221

* sets specific location of the .mos script file

\_initialize\_error\_dict Line 215-220

* sets a dictionary for errors

setLibraryRoot Line 230-246

* sets up specified home directory (location of top level .mo package)
* sets up root directory (where Resources folder is contained)

useExistingResults Line 248-272

* does not run a simulation, will instead use existing results

TestSinglePackage Line 274-289

* This function allows the user to specify a single model to test.

setNumberOfThreads Line 291-299

* set how many processor threads to use

showGUI Line 301-307

* show Dymola (or omc) when running simulations

batchMode Line 309-327

* runs without interactive prompts or plot windows

pedanticModelica Line 329-344

* runs Modelica in pendantic mode

include\_fmu\_tests Line 347-362

* will test the export of FMU’s

getModelicacommand Line 364-370

* returns name of Modelica command

isExecutable Line 373-393

* tests to see if a program is executable
* adds .exe to given name and checks if it is executable
* if not it will check the path for the executable

isValidLibrary Line 396-422

* checks that the home lib and root lib exist

getLibraryName Line 424-429

* returns the base name of libHome

checkPythonModuleavailability Line 431-4451

* checks for matplotlib, numpy, scipy and tidylib of checkHTML=true, raises error if not found

\_checkKey Line 453-474

* checks the given file for the string “key” as the first string

\_includeFile Line 477-505

* determines if file needs to be included in the list of scripts to run

setSinglePackage Line 507-539

* takes full directory path to package
* doesn’t test on only one model

writeOpenModelicaResultsDictionary Line 542-582

* write files useful to the regression testing done by the openmodelica team

get\_plot\_variables Line 585-631

* turn y={aa,bb,cc} into [aa,bb,cc]

setDataDictionary Line 634-929

* Build the data structures that are needed to parse the output files

\_checkDataDictionary Line 929-963

* Check if the data used to run the regression tests do not have duplicate .fmu files and .mat names
* Since Dymola writes all fmu and mat files to the current working directory, duplicate file names would cause a translation or simulation to overwrite the files of a previous test. This would make it impossible to check the fmu export and to compare the results to previously obtained results.

\_getTimeGrid Line 965-973

* Return the time grid for the output result interpolation

\_getSimulationResults Line 975-1053

* Get the simulation results for a single unit test

\_getTranslationStatistics Line 1055-1073

* Get the translation statistics for a single unit test

areResultsEqual Line 1076-1210

* Returns True is data series are equal within a tolerance
* Set max time error line 965
* Set tolerance in line 967

\_isParameter Line 1212-1219

* Return True if dataSeries is from a parameter

format\_float Line 1221-1229

* Cuts the trailing zeros to avoid outputs like 1.00000000

\_writeRefrenceResults Line 1232-1266

* Writes the reference results

\_readRefrenceResults Line 1268-1329

* Read the reference results written by \_writeReferenceResults

\_askNoReferenceResultsFound Line 1331-1353

* Function that asks what to do if no reference results found

\_check\_statistics Line 1355-1390

* Checks the simulation or translation stats and return True if there is a new stat

\_compareResults Line 1392-1560

* Compares the new and old results, accepts a MatLab file

are\_statistics\_equal Line 1563-1585

* Compares Simulation statistics and returns True if they are

\_compare\_and\_rewrite\_fmu\_dependencies Line 1589-1656

* Compares whether the .fmu dependencies have been changed, if they…
* …are the same, nothing is changed
* …differ from the reference results, it asks whether to accept new ones

\_check\_fmu\_statistics Line 1659-1714

* Check the fmu statistics from each regression test and compare it with previous ones
* If the statistics differ, show a warning
* If there are no statistics, ask if they should be generated

\_checkReferencePoints Line 1716-1798

* Check reference points from each regression test and compare it with the previously saved reference point of the same test stored in the library home folder.
* If all points are not within a certain tolerance with the previous results, show a warning message
* If no .mat file exists then ask the user if it should be generated.

\_checkSimulationError Line 1800-1871

* Check whether the simulation had any errors and write error messages to self.\_reporter

printNumberOfClasses Line 1873-1897

* Print the number of models, blocks and function s to the standard output stream

\_getModelCheckCommand Line 1899-1929

* Return lines that conduct a model check in pedantic mode

\_removePlotCommands Line 1931-1958

* Remove all plot commands from the mos file
* This fixes an issue with Dymola 2012 that caused issues in Windows and Linux

\_write\_runscripts Line 1960-2233

* Creates the runAll.mos script, one per processor
* Commands in the script depend on the executable

deleteTemporaryDirectories Line 2235-2244

* Flag, if set to False then temp directories will not be deleted after the regression test

\_setTemporaryDirectories Line 2247-2271

* Creates the temporary directories

run Line 2274-2488

* The ‘master’ run scripts that ties everything together it…
  + Sets the number of threads
  + Checks if it should use existing results
  + Validates the home library
  + Initiates the error dictionary
  + Prints number of classes
  + Checks HTML
  + Writes runsctipts
  + Initiates simulations
  + Opens, writes, and closes log file
  + Checks reference results
  + Deletes temp directories
  + Checks for errors
  + Prints list of files that may be excluded
  + Prints the time
  + Deletes statistics file

\_get\_test\_models Line 2490-2515

* Return a list with the full path of test models that were found in “packages”

\_model\_from\_mo Line 2517-2526

* Return the model name from a .mo file

test\_JModelica Line 2529-2651

* High level test for compliance with JModelica.org.
* Will work even if there is no .mos file
* Note: sets self.\_npro (number of threads) to 1

analyseJMStats Line 2653-2706

* Analyze the statistics dictionary resulting from a \_test\_Jmodelica() call

\_writeOMRunScript Line 2708-2740

* Write an OpenModelica run script to test model compliance

Test\_OpenModelica Line 2742-2864

* High level test for OpenModelica compliance
* Also sets the number of threads to 1
* Will work even if there is no .mos file

\_analyseOMStats Line 2866-2942

* Analyze the log file of the OM compatibility test

**ModelicaPy User’s Manual**

our\_test.py overview:

* This is the top level script that interacts with the BuildingsPy framework. To call this script run our\_test.py in the command prompt (make sure Python 2.7 is in your path before Python 3). Also be sure that Dymola is in the system path so that simulations can properly run.
* Specify the library to be tested in lines 111 and 112.
  + 111- specify the path to the top level directory with the package.mo file
  + 112- specify the path to the direcroty containing the “Resources” folder
* Our\_test.py can accept either 1 input or no inputs
  + Single input- the name of the .mos script to be tested (do not include the .mos extension)
  + No input- tests all .mos scripts found in the root directory
* The os.chdir(rootDir) is required to write simulator log to rootDir. If these logs should be written somewhere else then specify.
* Options:
  + setLibraryRoot: lets you differentiate between the directory containing the package.mo file and the directory containing the “Resources” folder.
  + showGUI: set this to true to show the Dymola GUI
    - note that when using continuous integration this must be turned off, otherwise it will hang Jenkins/CMake
  + setNumberOfThreads: set the number of threads for testing, if unspecified it defaults to using all threads available/all threads needed.
  + batchMode: run simulations without interactive user input
    - note that this should be turned on when using continuous integration, waiting on input from the user will hang the script
  + deleteTemporaryDirectories: set to False for temp directories to be preserved. Since all simulations are run from temp many of the log files are also written in temp. Preserving temp directories is useful for troubleshooting
  + setMosLocation: specify the exact location of the .mos file to be tested, not useful at this point but could be further down the road
  + useExistingResults: Does not run a simulation, instead uses the previously stored results located where the user specifies
  + pendanticModelica: run in pendantic Modelica mode
  + include\_fmu\_tests: set to “False” to not test the FMU export