Unittests

IDEAS is developed as a test driven Modelica library. The idea is to make small tests for a specific class in order to check if this class is has no programming errors and produces the expected output. Every model (Modelica class) that we test is seen as a unit, and the testing is called unittesting.

Table of Contents

How?

For every model (.mo file) that is included in the tests, we write a script (.mos file) that can be run in Dymola (only simulation environment supported at the moment) and generates a plot. These scripts are saved in IDEAS and are part of the version management. The models that are to be tested are top-level models. This means they contain all the inputs they need and can be run directly from the IDEAS library without changing any setting. In IDEAS we specify these as examples. A simulation exemplifies how the model works.

The unittest is done using python code from the BuildingsPy module. The BuildingsPy package needs to be downloaded to your computer. The unittests are carried out by runninga small python script. It first calls a tester instance which can be run(). This tester instance then collects all possible .mos files in the IDEAS/Resources/Scripts/dymola/… directory. An instance of Dymola is opened and the files are run.

- The whole IDEAS library is copied to a temp directory. This means the unittesting process can run in the background.
- For every example model with corresponding .mos file, a selection of simulation outputs (as specified in the script by a plot command) is compared to a reference (expected output) which is also part of IDEAS
- Dymola log (command window in dymola-*simulation*) and unittest log (python output of the tester instance) are saved to the working directory (IDEAS root dir) for user feedback.

What to do?

- Get the last version of IDEAS from the mech git repository.
- Download BuildingsPy from internet.
  - The easiest option is to use pip according to these instructions: http://simulationresearch.lbl.gov/modelica/buildingspy/install.html
  - Alternatively, clone the BuildingsPy repository on github and run “python setup.py install” in the main buildingspy folder (in a command window, not a python console).
- Download and install tidylib:
  - easiest: pip install tidylib (in a command window, not a python console)
  -

- Make minor change to the file \BuildingsPy\buildingspy\development\unittest.py:
  - The input files in IDEAS are programmed to be one directory above the working directory. This means that besides the library (IDEAS) the input files need to be copied to the temp directory as well.
  - In line 1118 in method setTemporaryDirectories()add the following code in the for loop in front of the return statement:

    “

    try:

    # For IDEAS the inputfiles should be in the directory above the workdir (this is an implementation choice for easy simulation sweeps)

    shutil.copytree(".." + os.sep + "Inputs", os.path.join(dirNam, "Inputs"), symlinks=False, ignore=shutil.ignore_patterns('.svn', '.mat'))

    except:

    pass

    return”

The result should look like this:

```
def __setTemporaryDirectories(self, nPro):
    import tempfile
    import shutil
    import os

    self.__temDir = []

    # Make temporary directory, copy library into the directory and
    # write run scripts to directory
    for iPro in range(self.__nPro):
        #print "Calling parallel loop for iPro=", iPro, " self.__nPro=", self.__nPro
        dirNam = tempfile.mkdtemp(prefix='tmp-' + self.__libraryName + '-'+ str(iPro) +  "-")
        self.__temDir.append( dirNam )
        shutil.copytree(".." + os.sep + self.__libraryName,
                        os.path.join(dirNam, self.__libraryName),
                        symlinks=False,
                        ignore=shutil.ignore_patterns('.svn', '.mat'))
        try:
            # For IDEAS the inputfiles should be in the directory above the workdir (this is an ir
            shutil.copytree(".." + os.sep + "Inputs", os.path.join(dirNam, "Inputs"), symlinks=Fa
        except:
            pass

    return
```

-
- Set **environmental variables**:
  (A system restart might be necessary for environmental variable changes to take place.)
    - Buildingspy root directory needs to be in the "PYTHONPATH" environmental variable (e.g. on my comp: *add* ; D:\work\Python\BuildingsPy; *to the already existing path*) (if your computer does not yet have an environment variable called PYTHONPATH, simply create it)
    - Dymola installation folder (where you find the dymola.exe) needs to be in the "PATH" environmental variable (e.g. on my comp: *add* ; C:\Program Files (x86)\Dymola 2013 FD01\bin64; *to the already existing path*)
- RUN THE TESTER: In IDEAS root directory you find a python file: **RunUnitTests.py**. Run this file in python (interpreter of your choice)
    - Open python interpreter
    - Set working directory to root directory of library (IDEAS) (use "cd" command)
    - "run –i runUnitTests.py"
      ("-i" is a flag to indicate some output printing by python – can be omitted)
    - Alternative:
        - Open cmd line
        - Set working directory to root directory of IDEAS.
        - "python RunUnitTests.py"
    - http://stackoverflow.com/questions/1522564/how-do-i-run-a-python-program
    - Although Dymola is used, it does not pop open when it is called.
      If you want to open Dymola, to see the tests being run, remove *, "/nowindow"* in line 28 of unittest.py before running the script

How to use Git version management to get files from an online repository

BASICS

- Download your preferred git software: e.g. tortoiseGit
- Restart your computer after installation
- Go to the online repository (github, BitBucket or TME (https://git.mech.kuleuven.be/tme/ideas-public.git)) and copy the url. (also called clone)
- Go to the directory you want to use for this library (in your windows directories)
    - In the dropdown menu from windows select clone directory (this is for toirtoiseGit)
    - Check if the paths are ok
    - (deselect checkbox load putty key)
    - Hit enter