Training Session 4: TRANSFORM Advanced II

M. Scott Greenwood, PhD

Oak Ridge, TN

September 11-12, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

TRANSFORM Advanced II (60 Minutes)

- Additional background to supplement Session 3

Grab Bag:

- Some random, important information

Resources Folder

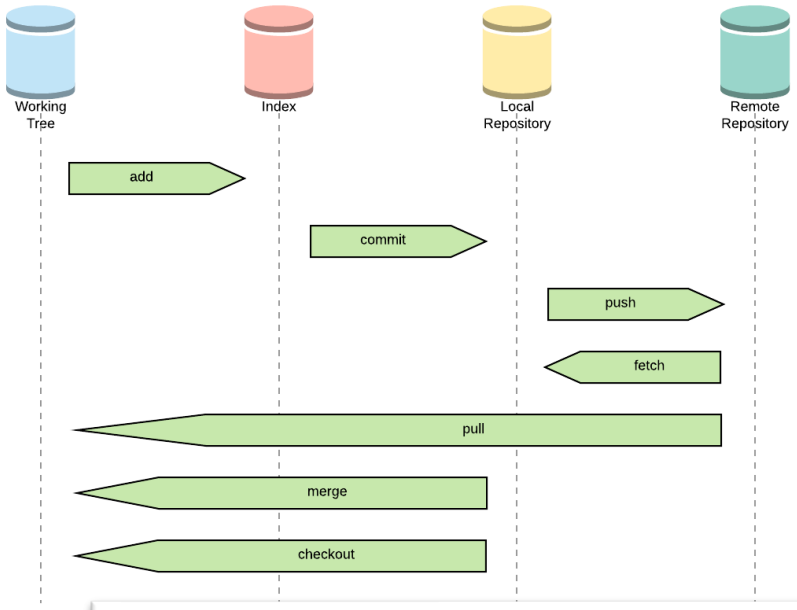
- Location to store source code, scripts, images, etc.
- Will get deleted if not careful if library is saved as a single file

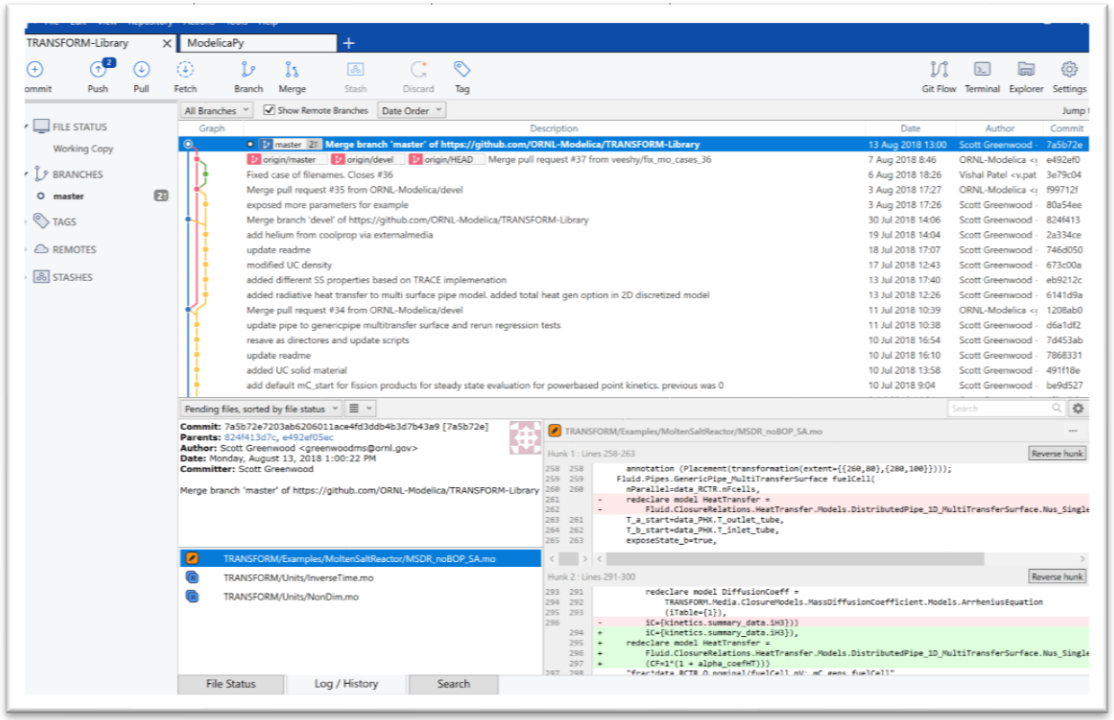
Version Control and Github

- description of version control tools and recommendations

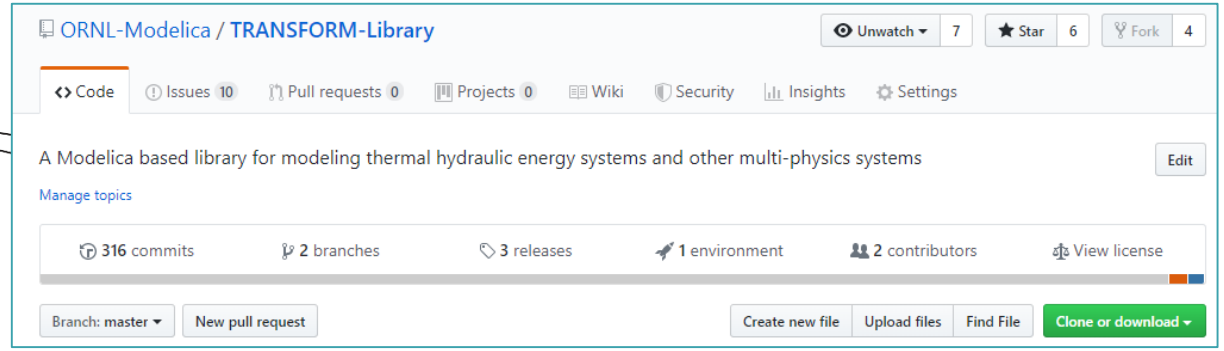
Version Control

- Git is the most popular version control software
  - mecurial is another option
- Can use directly via terminal or via third-party interface
- SourceTree is our preferred third-party interface
  - Intuitive GUI
  - Great for pick up and go
  - Con: Have to create an Atlassian account (email)
    - but I have yet to receive any junkmail
- GitKraken is another option
  - paid and free version

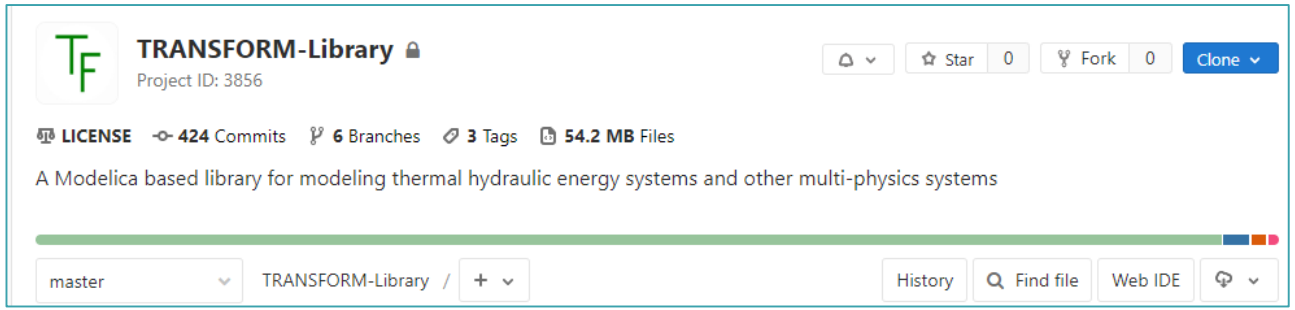




Git Services



- Github
  - #1 service
  - Trivia: Microsoft recently bought Github
- Gitlab
  - Used by ORNL (internal servers)
- Repos can be:
  - Public (free)
  - Private (paid)



GitLab

Very Basic Tutorial

- “stage” – puts changes on a cue to be saved
- “commit” – save the changes locally
- “push” – move saved changes from local to remote repository
- “pull” – move saved changed from remote to local repository
- “branch” – create a copy of the repository
- “merge” – squish changes from different branches together

Unit Test

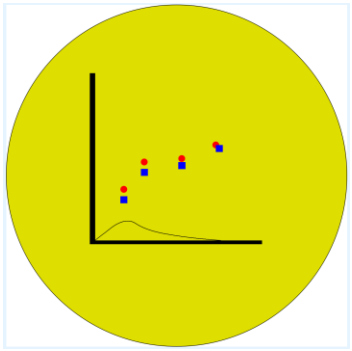
Why, oh why must I Unit Test!

Unit tests are an important part of making sure we don't, unintentionally, ruin our life or others. 🤦‍♀️

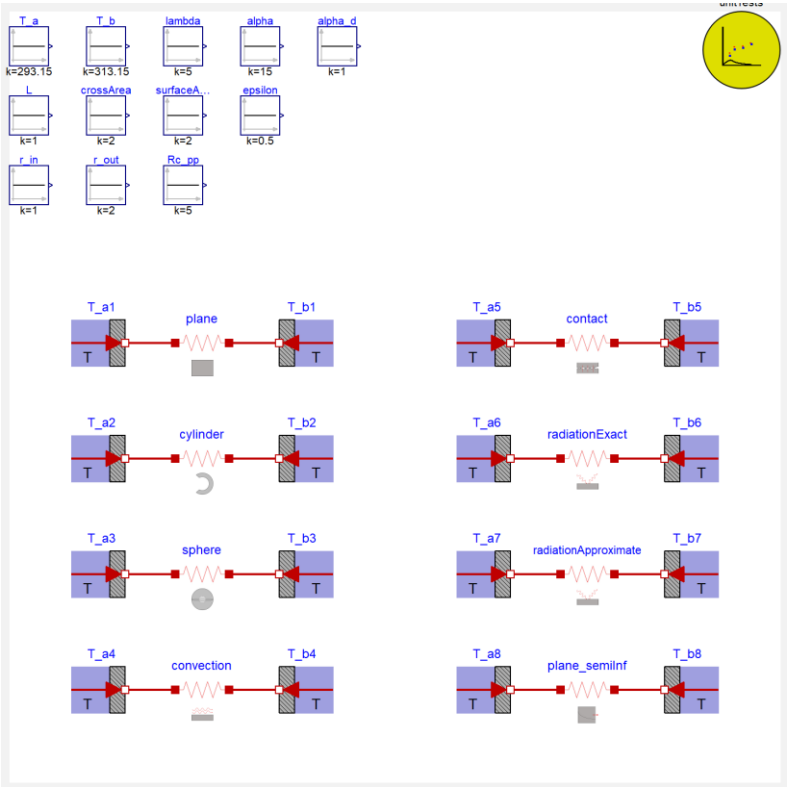
--> Insert funny analogy here 😑 <--

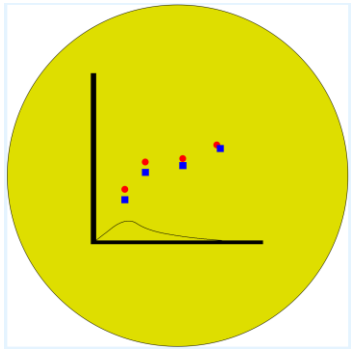
In that spirit, contributors, let us all do our part! ☯️

Unit Tests



- TRANSFORM has >300 regression tests and counting
- Unit tests provide a means to ensure the library
  - Has working examples for users to explore the library
  - Remains functional as changes are made
  - Has code coverage on its testing
- The unit test model is in any example that has been added to the regression system
  - Examples with the unit test model and example icon can be auto-added to the regression system





To Play with Existing Tests

To see what tests are already part of the system you can run (from your IDE) all the regression tests. *This can be helpful in finding issues if the python regression system fails.* To do so:

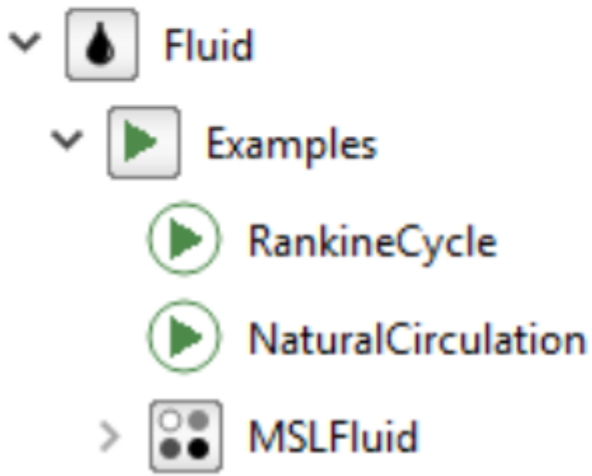
1. Copy `runAll_Dymola.mos` from `/TRANSFORM-Library/` to your favorite place to run models. Recommended for that to be a directory such as `/Documents/Dymola/`.
2. Open up your IDE and execute the `runAll_Dymola.mos` script.

TADA!

Unit Test Setup --Modelica--

Unit tests are automagically added to the regression system from a python script if 3 simple things are followed.

1. The model to be added must be put in a package labeled `Examples` complete with the icon extension `TRANSFORM.Icons.ExamplesPackage`.



2. The model to be added must also extend the icon `TRANSFORM.Icons.Example` at the highest level.

- **Note:** Currently a model's unit test in an `Examples` package can not have the same name as any other model in any other `Examples` package, regardless of path. I know, I know... super lame. This is a current "feature" of the buildingspy regression system we are piggybacking  . Fortunately, when you run the regression test system you'll get a friendly reminder letting you know you need to change it.

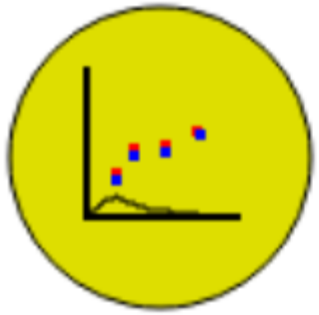
```
model RankineCycle
  extends TRANSFORM.Icons.Example;

  package Medium = Modelica.Media.
  parameter SI.MassFlowRate m flow
```

3. The unit test model `TRANSFORM.Utilities.ErrorAnalysis.UnitTests` must be placed on the model with the default name `unitTests`.

   - **Note:** The variable(s) set to `unitTest.x` will be added to the regression system. `unitTest.x_reference` provides a place to put the expected result if desired to make a comparison from the IDE. However, values other than `x` from `UnitTests` are not saved to the regression test reference result file.

unitTests



```
TRANSFORM.Utilities.ErrorAnalysis.UnitTests unitTests(
  n=1,
  x={pump.medium.p})
  ∃ ;
```

Cleaning up a Library

4. Lots of extraneous files get put to a Modelica library during development. Before running the python scripts to add files to the regression system you need to clean up files, either manually or using a directory saving approach (*recommended*) described below.

    - (From IDE) Right-click `TRANSFORM` and check "Store as one file".
    - (From IDE) Save TRANSFORM to a temporary location (e.g, Desktop).
    - Move the directory `/TRANSFORM-Library/TRANSFORM/Resources` up a level to `/TRANSFORM-Library/`. This file will not be retained in the single file version of the library.
    - Delete the directory `/TRANSFORM-Library/TRANSFORM/`
    - (From IDE) Right-click `TRANSFORM` and uncheck "Store as one file". If prompted, select "Directores - No Questions".
    - (From IDE) Save TRANSFORM back to the original location (i.e., `/TRANSFORM-Library/`).
    - Now is a good time to check Git to make sure the process performed as expected. You should now have a clean library.
    - Commit changes to Git.

Add Unit Tests to Regression System --Python--

1. Run `createUnitScrips.py` from its location (i.e., `TRANSFORM-Library\TRANSFORM\Resources\python` as the working directory)
   - This deletes the entire `/Resources/Scripts/` directory and rebuilds it from scratch

2. Check that the expected `.mos` files were added to the `/Resources/Scripts/` directory.
3. Run `pip install buildingspy` from the terminal if buildingspy is not already installed
4. Run regression system from its location (i.e., `TRANSFORM-Library\` as the working directory). Indicate `y` when prompted to add the results to testing system. These files are saved in `/TRANSFORM-Library/TRANSFORM/Resources/ReferenceResults/`.
   - Linux: `regtestsLinux.py`
   - Windows: `regtestsWin_customBuildPy.py`
   - **Note:** By default running this file will go through all tests. To simulate a single test use the `TestSinglePackage` function.

5. If everything looks good, stage and push your changes to your repository and create a merge request.

That's it! Treat yourself 🍨

Troubleshooting

1. If having issues with buildingspy regression test on Windows system replace buildingspy file `regressiontest.py` with the one from ModelicaPy (`pip install modelicapy` or download zip file from Github/Gitlab)
   - `regressiontest.py` in buildingspy is normally located at `C:\Users\USERNAME\AppData\Local\Continuum\anaconda2\Lib\site-packages\buildingspy\development`
   - Once replaced, delete the old `regressiontest.pyc` file if present.
   - The new `regressiontest.py` file from ModelicaPy may need to be compiled before it can be used. If so, change to the `buildingspy\development` path and run the following in python (i.e., IPython) to get a new `regressiontest.pyc` file.
     - import py_compile
     - py_compile.compile('regressiontest.py')

2. It is highly recommended to simulate the `runAll*.mos` file from Dymola to ensure all unit tests are able to simulate. The current regression system has little to no inspection ability in the event a model fails to simulate.

Modelica and Python

- .mat results, run/simulate model
- simulate example(?) and read/plot results

Modelica and Python

- Variety of tools exist to interact with Modelica depending on the tool
- We'll use Python and the "buildingspy" library from LBNL
- To install: pip install buildingspy
- To read in .mat file:

```
In [ ]: from buildingspy.io.outputfile import Reader
        r = Reader('NAMEOFMATFILE.mat','dymola')
```

- To read variable:

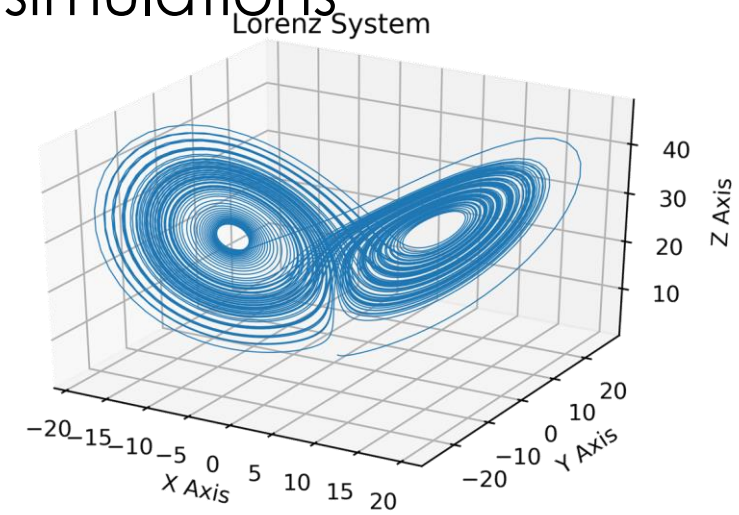
```
In [ ]: time, variable = r.values('ModelicaVariablePathAndName')
```

Hands-On Example 3: Results Analysis with Python

- Install python (Recommend via Anaconda)
- Install buildingspy
- “cd” to the directory of your .mat file
- From previous slide make a script/jupyter notebook follow steps
- View the result
  - For example: **print(variable)** or **plt.plot(time,variable)**

Other Modelica/Python Tools

- Variety of Modelica/Python tools can be found online
  - fmpy, pyfmi, buildingspy, etc.
- ModelicaPy is one created for use with TRANSFORM
- Has tool to auto clean and categorize parameter and variable data
- Has tools for parametric sweeps and run simulations
- Example: Lorenz model with plot
  - Scripts are located in TRANSFORM/Resources/python



Modelica and Matlab

- .mat results, run/simulate model
- simulate example(?) and read/plot results

Example

- Dymola provides various files to interact with result files
- We will look at an example in the training Resources folder.

Thank you.

Scott Greenwood

greenwoodms@ornl.gov