Automated Testing of Formulations in the Next Generation Water Resources Modeling Framework

Jessica L. Garrett[1,2], Keith S. Jennings[1,2], Zachary Wills[1,2], Nels J. Frazier[1,2], Shengting Cui[1,2], Ahmad Jan[1,2], Fred L. Ogden[1], Trey C. Flowers[1]

[1]NOAA National Water Center
[2]Lynker, Inc

OWP OFFICE OF WATER PREDICTION

SESSION NUMBER: H095

Needs and Concerns

The Next Generation Water Resources Modeling Framework (NextGen) formulation teams have officially incorporated 12 models and modules into the NextGen build runtime, including those written in C, C++, Python and Fortran. With each model comes a plethora of challenges:

- Are Basic Model Interface (BMI) functions properly defined for the model domain?
- Will this model provide reasonable output given a sample input dataset?
- Are input and output variable names and units correctly assigned?
- Does the model run within NextGen framework?
- Are all state variables sound and functioning as expected?

Github Actions

What it does: Implements software standards for version control and product code updates, all over the browser!

Why it helps:

- Eliminates local environment challenges
- Facilitate the proper use of model code
- Check standalone scripts
- BMI unit testing
- Framework build/runs
- Promote better community involvement &
- Overall quality of collaborative application design

How to implement:

- Start with a workflow YAML file(s)
- Place in `.github/workflows` directory in a repo

How Github Actions & Workflows Aid Model & Module Development

Triggers

Event that will prompt the workflow to run.
Common example includes:

- Push to master branch
- Pull requests
- `workflow_dispatch` = manual button

This workflow has a `workflow_dispatch` event trigger. Run workflow

OS & Versioning

```
strategy:
  matrix:
    os: [ubuntu-latest, macos-latest]
    python-version: [3.7, 3.8, 3.9] #
  fail-fast: false
runs-on: ${{ matrix.os }}
```

Env, Libs & Packages

```
- name: activate conda
  uses: conda-incubator/setup-miniconda@v2
  with:
    activate-environment: bmi_lstm
    environment-file: ./environment.yml
    auto-activate-base: false

- name: install packages
  run: |
    python -m pip install --upgrade pip
    pip install -e .
```

Job 1

```
# RUN MAIN BMI UNIT TEST
- name: Run BMI Unit Test LSTM
  run: python lstm/run_bmi_unit_test.py

# RUN MAIN STANDALONE SCRIPT
- name: Run Standalone LSTM
  run: python -m lstm
```

Job 2

```
# BMI: Time Functions
print("\nTIME INFORMATION\n**********")

# ------------ get_start_time() ------------
try:
                print (" start time: " +
str(bmi.get_start_time()))
                pass_count += 1
except:
                bmi_except('get_start_time()')

# ------------ get_end_time()------------
try:
                print (" end time: " +
str(bmi.get_end_time()))
                pass_count += 1
except:
                bmi_except('get_end_time()')

# ------------ get_current_time() ------------
try:
                print (" current time: " +
str(bmi.get_current_time()))
                pass_count += 1
except:
                bmi_except('get_current_time()')

# ------------ get_time_step()------------
try:
                print (" time step: " +
str(bmi.get_time_step()))
                pass_count += 1
except:
                bmi_except('get_time_step()')

# ------------ get_time_units() ------------
try:
                print (" time units: " +
bmi.get_time_units())
                pass_count += 1
except:
                bmi_except('get_time_units()')
```

BMI Functions:
4 Model Control
5 Model Information
5 Model Time
6 Variable Information
5 Variable Getters & Setters
16 Model Grid

test_unit (macos-latest, 3.9)
succeeded 12 hours ago in 11m 51s

- Set up job
- Checkout the commit
- setup python
- activate conda
- install packages
- Run BMI Unit Test LSTM
- Run Standalone LSTM
- Post activate conda
- Post setup python
- Post Checkout the commit
- Complete job

NextGen Integration Workflow

Example: Build & Run NextGen Coupled with PET & CFE

test_surfacebmi_plus_cfe (ubuntu-latest)
succeeded 16 minutes ago in 4m 24s

- Set up job
- checkout actions in another repo
- Build PET
- Build SLoTH
- Build Ngen
- Cleanup Ngen Build
- Run surfacebmi plus cfebmi
- Checkout the commit
- Build CFE Library for Ngen
- Re-checkout Ngen
- Post checkout action
- Post Build Ngen
- Post Re-checkout Ngen
- Post Checkout the commit
- Complete job

```
# Run ngen with CFE with PET
inputfile="realizations/realization_cfe_pet_ngenCI.json"
./ngen-build/ngen ./data/catchment_data.geojson "cat-27"
./data/nexus_data.geojson "nex-26" $inputfile
```

1. Define triggers and environment(s)
2. Link some pre-exisitng (build) actions from an external source
3. Build necessary libraries, including `ngen`.
4. Gather input data
5. Build and run `ngen`

Note: Many pre & post processes are included in workflow by default (developer is not required to manually include in YAML file).

Future Work

- Incorporate unit testing workflow for all 12 formulations
- Add mass balance checks when appropriate
- More robust error messaging
- Check for memory leaks
- Include more `ngen` build/run workflows

REFERENCES:
Github Actions Docs: https://docs.github.com/en/actions/using-workflows
CFE Workflow Examples: https://github.com/NOAA-OWP/cfe/actions
LSTM (Python) BMI Unit Testing Script: https://github.com/NOAA-OWP/lstm/blob/master/lstm/run_bmi_unit_test.py

Lynker

CONTACT
Website: https://water.noaa.gov
Email: jessica.garrett@noaa.gov

View my poster and other AGU materials