Dr. Jan Vermaak
RELAP5-3D Code Architect

RELAP5-3D Test System

INL/CON-26-92682

Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

Idaho National Laboratory

Background

- Formal software testing first reference: Kent Beck “Test First Design”, 1999-2002
- First few tests were made for RELAP in 1997 by … George Mesina
- In the 90s the battle of testing ideologies began:
  - TDD: Test Driven Development
  - RDD: Requirements Driven Development

Background 2

- Many perspectives on testing:
  - Robert C. Martin's 3 laws for TDD:
    - Write no production code except to pass a failing test.
    - Write only enough of a test to demonstrate failure.
    - Write only enough production production code to pass the test.
  - David Farley Red-Green-Blue:
    - RED: Write a failing test
    - GREEN: Write code to pass the test
    - BLUE: Refactor and improve quality

Test Driven Development (TDD) is not always liked

- Casey Muratori (coined the term IMGUI – Immediate Mode GUI 2002):
  - Spending too much time developing for tests takes time away from actual use cases
  - Snapshot-testing is useful
  - Acceptance-testing is useful
- ASME NQA-1 does not actually require TDD but rather
  - Requirements Specification
  - Traceability
  - Verification and Validation

What is the RELAP5-3D Testing Philosophy?

CI/CD: Continuous Integration
Continuous Development
PR: Pull Request

1. Regression Testing: Evaluating regression is a high priority.
2. Testing during CI/CD: two phases i) condensed for PRs, ii) full set for CD.
3. Snapshot Testing: Shall be used to evaluate the effect of new/refactored code.
4. Evaluate the performance of key processes.
5. Document test cases and results.
6. Every test shall have an associated requirement.
7. Every requirement shall have at least one associated test.
8. Do not over-sample the parameter space.
9. Do not over-test, evaluate the value added by a test carefully.
10. Prioritize good tests over many tests.

The RELAP5-3D Test System Requirements

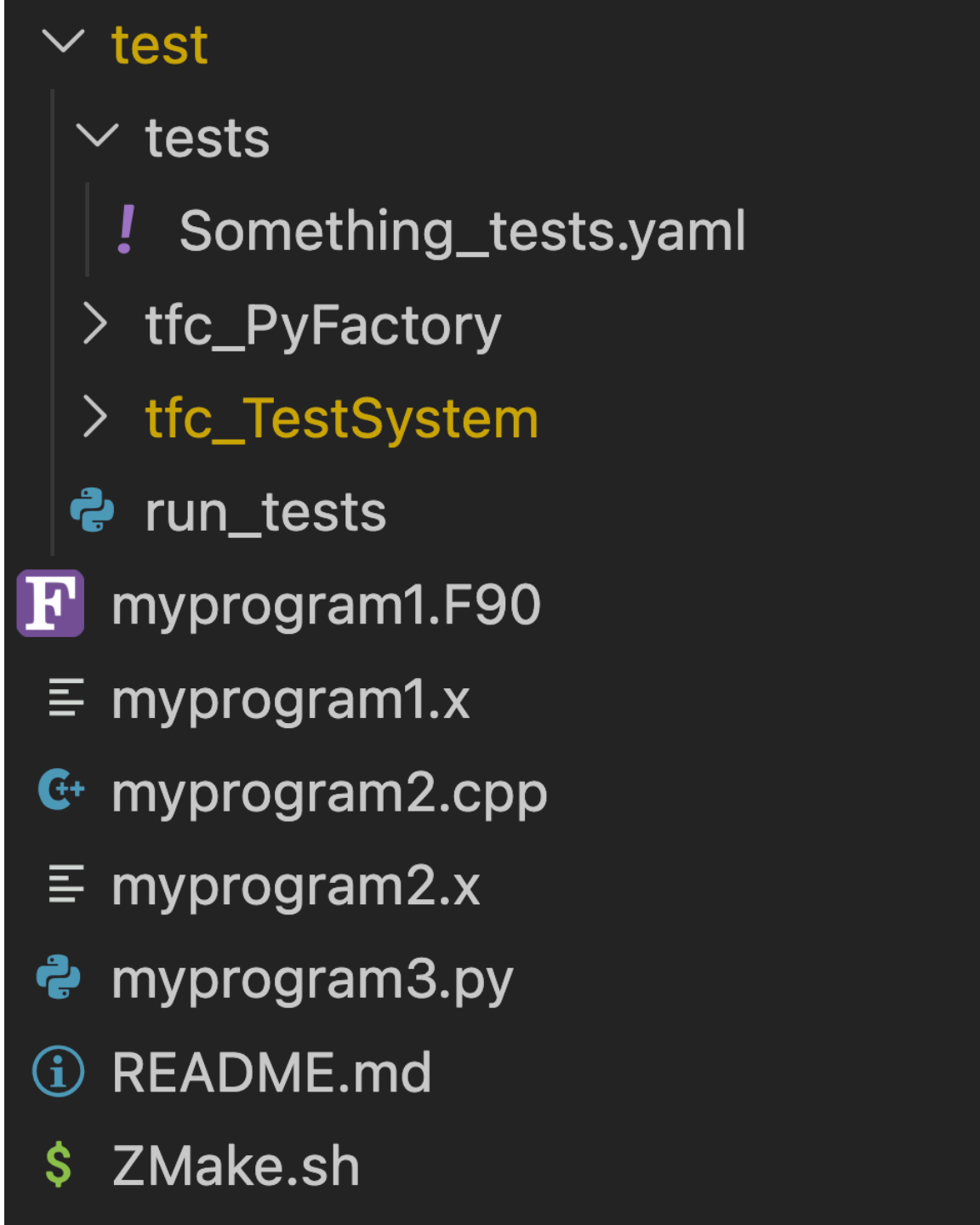
1. Test specifications must be apparent.
2. A test shall execute a command line driven executable.
3. The arguments shall be customizable.
4. A test must have the ability to run several checks.
5. A test must have the ability to link to several requirements.
6. A test must be able to run sub-scripts before running the executable.
7. A test must be able to run sub-scripts after running the executable.
8. A test must be able to run sub-scripts after the checks have executed.
9. A test must be derivable from a template.
10. A test must have the ability to be dependent on another test.

TFCTestSystem

- github.com/tfc-engineering/tfc_TestSystem
- Python script implementing a factory pattern
- Checks are 100% extendible
- Pulled into RELAP5-3D using a git submodule

TFCTestSystem 1

- Create a test directory in your repo
- Clone tfc_PyFactory and tfc_TestSystem into test
- Copy examples/ExampleTestSystemEXE.py to test/run_tests and modify it (chmod u+x etc.)
- Create a test/tests folder and place within it a file *tests.yaml



TFCTestSystem 2

- Example specifications

```
# Tests polynomial root finder with real
# distinct roots
test1:
  executable: $PROJECT_ROOT/myprogram1.x
  args: '-a 1   -b 0   -c "-1"   -j 2'
  checks:
    - type: ExitCodeCheck
    - type: HasStringCheck
      line_key: root1=   2.000 root2=  -1.000
  requirements: [math.1, math.poly]
```

TFCTestSystem 3

- Example outputs:

```
Running tests with weight class:['short', 'none']
[1]  test/tests/test1  ..................................................  Passed 0.05s
Done executing tests with weight class: ['short', 'none']

Elapsed time                       : 0.07 seconds
Number of initialization warnings : 0
Number of tests run                : 1
Number of tests skipped            : 0
Number of failed tests             : 0
```

```
Running tests with weight class:['short', 'none']
[1]  test/tests/test1  ..................................................  Failed 0.05s
Done executing tests with weight class: ['short', 'none']

Elapsed time                       : 0.07 seconds
Number of initialization warnings : 0
Number of tests run                : 1
Number of tests skipped            : 0
Number of failed tests             : 1

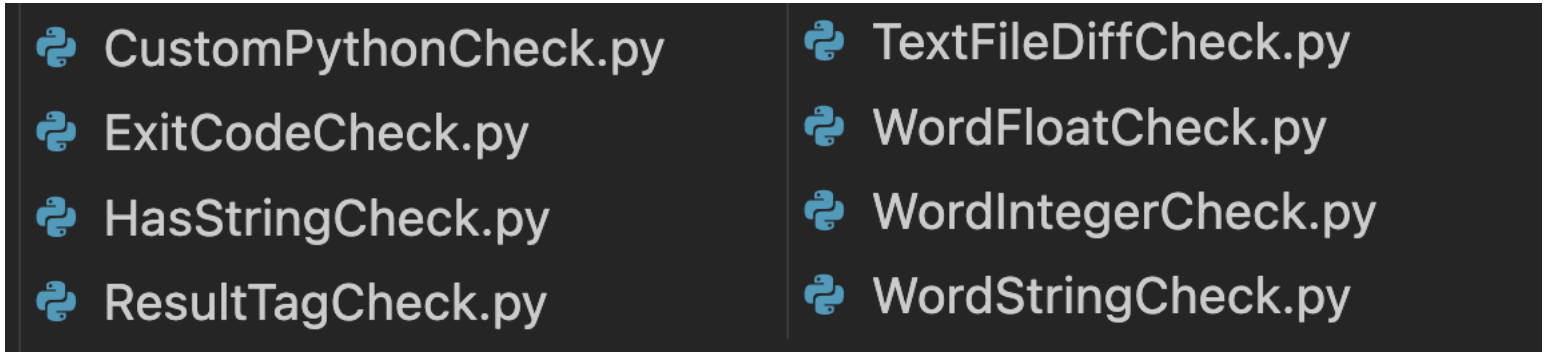
Failure reasons:
test/tests/test1:
<class 'checks.HasStringCheck.HasStringCheck'> Could not find line containing "root1=   2.000 root2=  -1.000".
```

Test Objects

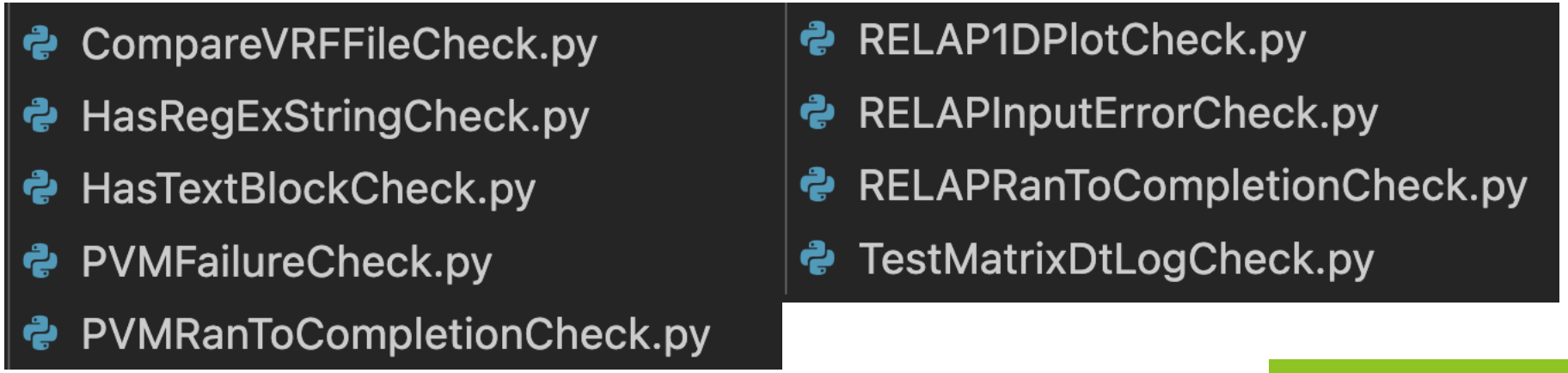
- `args`. Arguments to pass to the test program.
- `checks`. An array of check-inputs (See Check objects below).
- `project_root`. (Semi-Optional, automatically set from TFCTestSystem). The path to the project root directory.
- `disable_mpi`. (Optional, default = True). Flag to suppress running the application via MPI.
- `num_procs`. (Optional, default = 1). The number of mpi processes used.
- `weight_class`. (Optional, default = "short"). The weight class "short"/"intermediate"/"long"
- `outfileprefix`. (Optional, default = ""). Will default to the test name + .out, otherwise outfileprefix+.out.
- `skip`. (Optional, default = ""). If non-empty, will skip with this message.
- `pass_flag`. (Optional, default = ""). If non-empty, will pass with this message.
- `fail_flag`. (Optional, default = ""). If non-empty, will fail with this message.
- `dependencies`. (Optional, default = []). A list of dependent test names before this test can run.
- `prerun_script`. (Optional, default = ""). A shell script to run before the test is executed.
- `precheck_script`. (Optional, default = ""). A shell script to run after the case has been run but before the checks are executed.
- `postrun_script`. (Optional, default = ""). A shell script to run after the test is executed.
- `relative_offset_workdir`. (Optional, default = ""). A relative working directory to be added to the default working directory.
- `debug`. (Optional, default = False). Flag for debug printout. Will print the console output file to the screen for failed tests.
- `executable`. (Optional, default = ""). Executable to use instead of system wide default.
- `copy_test`. (Optional, default = []). This parameter provides a way to run a copy of a given test with modifications made by a script. A list of length 3 should be provided specifying the name extension to be added to the copied file name, the path to the script for creating a copy of the test, and the file location of the file to be copied. The script specified should create a copy of the test input file with any number of modifications. Both the original and the new input decks will be included in the test system.
- `requirements`. (Optional, default = []). An array of strings representing id_tags for requirements that are addressed by this test.

Checks

- 8 basic checks (for now)



- RELAP extended checks



Nice features 1

- Highly parallelizable
- Working in offset “temporary directories” is possible
- Templates with substitutable $MACROS

```
TEMPLATE_runtest_only:
  prerun_script: "mkdir temp_$TEST_NAME \n cp $TEST_NAME.i temp_$TEST_NAME/"
  args: "-rgtest -i $TEST_NAME.i -O ../out/$TEST_NAME.o -tpfdir $TPF_LOC"
  relative_offset_workdir: "temp_$TEST_NAME/"
  postrun_script: "rm -r temp_$TEST_NAME"
  checks: [ {type: RELAPRanToCompletionCheck} ]
```

```
3dflowA:
  from_template: TEMPLATE_runtest_only

3dflowB:
  from_template: TEMPLATE_runtest_only
```

Nice features 2

- Executable can be assigned globally, YAML local, and test-specific
- Tests can be assigned to different weight-classes, e.g., short, medium, long
- Tests can be assigned time-limits
- A configuration file can be used for style-type items (and other things)
- Tests and checks know about each other

Nice features 3

- Test result database
- Requirements parsing
- Automatic RTM (Requirements Traceability Matrix)

```
---------------------------------------------------
title: Output Specifications
link: "|page|/software_requirements/SRS_01_Output.html"
---------------------------------------------------

<!--topic console_output-->
## Console Output

Req 1. Upon each invocation of the executable the software version shall be printed
    in the following format

```text
RELAP5-3D/Ver:4.6.1      US Department of Energy
```

@note [[aatl(subroutine)]] and [[aaetit(subroutine)]] must be updated to reflect
the proper version number when a new version of the code is released.

Req 2. The following banner for copyright and export control shall be printed

```text
 ____________________________________________
|   COPYRIGHT      |  EXPORT CONTROLLED     |
|                  |      INFORMATION       |
| Battelle Energy  |  Contains technical    |
|    Alliance      | data whose export is   |
```
```

DEMO

Conclusion

- TFCTestSystem is open for everyone to use
- RELAP specific items are protected in the repo
- Meant as a replacement for MATLAB, Octave, SciLab

Questions

Idaho National Laboratory

Battelle Energy Alliance manages INL for the U.S. Department of Energy's Office of Nuclear Energy. INL is the nation's center for nuclear energy research and development, and also performs research in each of DOE's strategic goal areas: energy, national security, science and the environment.