VERA-Component Software Development Practices

Each code team has their own workflow
# Process Management

## #5717 Provide mechanism to perturb pin by pin enrichment and mass

<table>
<thead>
<tr>
<th>Reported by:</th>
<th>dfy</th>
<th>Owned by:</th>
<th>CASL PnP Project Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>P3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keywords:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer:</td>
<td>dfy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Action Status:</td>
<td>Create a pin by pin 3D distribution by sampling the perturbation distributions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent Ticket:</td>
<td></td>
<td>Supports Milestone:</td>
<td><a href="https://vminfo.casl.gov/trac/casl_milestones/ticket/2582">https://vminfo.casl.gov/trac/casl_milestones/ticket/2582</a></td>
</tr>
<tr>
<td>Effort Est. (hrs):</td>
<td></td>
<td>Effort Act. (hrs):</td>
<td></td>
</tr>
<tr>
<td>Start Date:</td>
<td></td>
<td>End Date:</td>
<td></td>
</tr>
</tbody>
</table>

This story will provide the ability to perturb material densities, fuel element isotopes, IFBA thicknesses, and burnable poison in MPACT.

I worked with Jinan on this class design and we came up with a pretty simple class:

```plaintext
TYPE :: MaterialPerturb

  LOGICAL (SBK) :: init => FALSE.
  CLASS (ReactorModelType), POINTER :: model => NULL()
  CLASS (XMeshType), POINTER :: xmesh => NULL
  TYPE (StringType) :: filename

  CONTAINS
    PROCEDURE, PASS :: init => init_Mate
    PROCEDURE, PASS :: clear => clear_Ma
    PROCEDURE, PASS :: perturb => perturb
    PROCEDURE, PASS, PRIVATE :: expand2Dpi
    PROCEDURE, PASS, PRIVATE :: sampledat
    PROCEDURE, PASS, PRIVATE :: perturb_f
    PROCEDURE, PASS, PRIVATE :: perturb_p
    ENDTYPE MaterialPerturb
```

Another note on this. Initial implementation will assume that the user provides only 3D pin distributions of factors (expand2pin3d and sampledataset methods will not be needed).

Minor comments/questions:
- would rather not instantiate every time (but mostly personal preference)
- we have pin and core... will we also (eventually) have assembly? ...qtr assembly?
- yeah, ensuring we don't change control rods and such should be simple
- overall, this design looks good and seems clear enough to me

I'm adding several private classes here just to

Don't think quarter assembly makes sense... but pin, radial, assembly, radial assembly, axial, and core would be in play... that was the deferred component.
Code Development

- Detailed coding standards
- Inline documentation
VERA Component Code Reviews

Burnup Zernike Polynomials for CTF

Description: Copied the existing Zernike power structure and applied it to radial burnup. Created a new interface to pass this information to CTF. A unit test will be added soon.

CASL Ticket #5754
Edited 2 weeks ago by Walker, Erik D

Request to merge burnup_zernike into master

Pipeline #55954 passed for 3c267a99 on burnup_zernike

Merge request approved; you can approve additionally
Approved by

Merged by Stimpson, Shane G 2 weeks ago
The changes were merged into master with 25642558
The source branch has been removed

Pipeline #55933 passed for 25642558 on master

Stimpson, Shane G @ggs started a discussion on an old version of the diff 2 months ago
Resolved by Stimpson, Shane G 2 weeks ago

MPACT_libs/Factories/src/FeedbackFactory.f90

Stimpson, Shane G @ggs commented 2 months ago
these changes make sense, as well as the indexing ones downstream

BUT to prevent having to make sweeping indexing changes later, it might make sense to have a counter the increments with each additional EV instead of flakier indexing changes.

I know this isn’t something you did, but might as well consider it here.

Walker, Erik D @ew4 changed this line in version 4 of the diff 2 weeks ago

Walker, Erik D @ew4 commented 2 weeks ago
I added a counter to make things easier in the future

Line-by-line review of changes
VERA Integration

Component Repositories

- Code A Master Branch
- Code B Master Branch

Developers

Sync Script
- Failed
- Passed

VERA Repositories

- Code A Master Branch
- Code B Master Branch

Extensive Nightly Testing
Testing prior to integration with VERA

Full nightly test suite run before integration into VERA
# Continuous and Nightly VERA testing

>10,000 tests run daily

## Nightly

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update Revision</th>
<th>Configure Error</th>
<th>Configure Warn</th>
<th>Build Error</th>
<th>Build Warn</th>
<th>Not Run</th>
<th>Fail</th>
<th>Pass</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>u233.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_DEBUG_DEBUG_SHARED</td>
<td>614023</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>737</td>
<td>0</td>
<td>1</td>
<td>1135</td>
<td>Feb 06, 2019 - 06:06 UTC</td>
</tr>
<tr>
<td>u233.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_DEBUG_STATIC</td>
<td>614023</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>650</td>
<td>0</td>
<td>1</td>
<td>1654</td>
<td>Feb 06, 2019 - 12:18 UTC</td>
</tr>
<tr>
<td>james007.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_DEBUG_HEAVY</td>
<td>614023</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>771</td>
<td>0</td>
<td>0</td>
<td>2108</td>
<td>Feb 06, 2019 - 06:10 UTC</td>
</tr>
<tr>
<td>u233.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_DEBUG_SHARED</td>
<td>614023</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>760</td>
<td>0</td>
<td>0</td>
<td>1673</td>
<td>Feb 06, 2019 - 06:06 UTC</td>
</tr>
<tr>
<td>u233.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_STATIC</td>
<td>614023</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>661</td>
<td>0</td>
<td>0</td>
<td>1655</td>
<td>Feb 06, 2019 - 13:52 UTC</td>
</tr>
</tbody>
</table>

## Continuous

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update Revision</th>
<th>Configure Error</th>
<th>Configure Warn</th>
<th>Build Error</th>
<th>Build Warn</th>
<th>Not Run</th>
<th>Fail</th>
<th>Pass</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>pu239.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_DEBUG_CI</td>
<td>614023</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>708</td>
<td>0</td>
<td>0</td>
<td>1411</td>
<td>Feb 06, 2019 - 06:04 UTC</td>
</tr>
<tr>
<td>pu239.ornl.gov</td>
<td>Linux-GCC-5.4.0-MPI_RELEASE_DEBUG_CI</td>
<td>614023</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>249</td>
<td>0</td>
<td>0</td>
<td>507</td>
<td>Feb 06, 2019 - 03:33 UTC</td>
</tr>
</tbody>
</table>

>10,000 tests run daily
VERA Release Process

VERA Repositories

Code A
Master Branch

Code B
Master Branch

Nightly Testing

Test Pass & Significant new features
- Every 1-2 months

Release Candidate 1

Release Candidate 2

Release Candidate 3

Beta Release (Optional)

Consensus with PHI/AMA Leadership
Every 6 months

VERA Release Through RSICC

Internal Users

Friendly External Users

External Users
VERA Deployment

• VERARun provides a mechanism to use “release” versions of the code on deployed machines

```
[~]$ verarun -l
   Available VERA versions:
       VERA_4.0RC3
       VERA_4.0RC2
       VERA_4.0RC1
       VERA_3.9
       VERA_3.8

[~]$ verarun -v VERA_4.0RC1 p7
```
Software Quality Objectives: The Path Toward NQA-1

• The goal of this work is to produce an NQA-1 certified version of the VERA suite.
• US Utilities are required to use NQA-1 software for safety calculations. Having an NQA-1 version of VERA will add value to the product by making adoption by utilities easier.

• Primary focus on year one is the core simulator and BISON
Organization of VERA for QA Purposes

• For QA purposes, VERA is organized as a collection of software “Products”.
• A product is one or more computer codes (executables) that can be used to perform analysis and meet software requirements. Products may also include utility programs that directly support the product.
• Each Product will have a separate Software Management Plan (SMP) and corresponding QA documentation.
• A Product Software Manager (PSM) will be assigned to each Product.
• Products may contain source code from external sources (e.g. PetSc, MPI, Origen, Futility). The QA of any external source code will be included in the QA of the product. (no CGD)
• Products may also have dependencies on other Products (e.g. MPACT depends on VERAIn, MPACT depends on CTF). The “higher-level code” must include the QA for the coupling.
Working Definitions

• **Product** – VERA component that is delivered to customers and is designed, developed and managed under a Software Management Plan (SMP).

• **External Library** – External source that is included in VERA Products. External libraries do not have a separate QA plan and the QA for these is included in the Product QA. (If an External library does have a QA plan, it should be classified as a Product).

• **Third Party Library (TPL)** – common designation in CASL to represent a library from an external source (MPI, GCC, CMake, etc.). TPL’s will usually be considered as “External Libraries”, but they may also be considered as Products if necessary.

• **Utility** – A separate executable or script that is used to support a product. The QA for the utility is included in the Product QA.

• **Dependency** – A Product may depend on another product. This relationship is a “dependency”. The interface for the dependency must be defined and tested in one of the products.

• **Product Software Manager (PSM)** - Each Product has one individual responsible for the management of the QA for that Product.
VERA Products

- It is important to split QA responsibilities into smaller pieces to manage more effectively across different development groups
- Product list may be slightly modified or added to in the future
- The following details for each product are not complete. Official details will be included in the product software inventories.
- First year focuses on first 3 components

1. VERA-IO
2. CTF
3. MPACT
4. Bison
5. MAMBA
6. Tiamat
7. VERAShift
1. VERA-IO

• Product Software Manager – Kevin Clarno

• Includes:
  – VERAIn – Common input processor used to generate XML files
  – VERAOut – Text based output utility
  – VERARun – Run script to run many executables in parallel environment
  – VERAView – graphical output visualizer

• Includes libraries: none

• Depends on: none
2. CTF

- Product Software Manager – Bob Salko
- Includes:
  - CTF – Standalone subchannel T/H code
  - xml2ctf – input preprocessor
  - driver – standalone CTF driver
- Includes libraries: MPI, PetSc, Futility
- Depends on: VERA-IO and Mamba
3. MPACT

- Product Software Manager – Brendan Kochunas
- Includes:
  - Standalone neutronics executable
  - Executable coupled with CTF
  - Executable coupled with CTF+MAMBA
  - ORIGEN (subset)
  - Post-processing utilities
- Includes libraries: MPI, PetSc, Futility, etc.
- Depends on: VERA-IO, CTF, and MAMBA
4. Bison

- Product Software Manager – Rich Williamson
- Includes:
  - Bison – Standalone Fuel Performance Code
  - xml2bison – input preprocessor (?)
- Includes libraries: Moose, MPI, PetSc, libmesh, etc.
- Depends on: none

Bison QA will be done under INL QA management program
Other Components

• MAMBA
  – Includes:
    • MAMBA - standalone CRUD code
    • Driver utilities
  – Includes libraries: MPI, PetSc, Futility, etc.
  – Depends on: none

• Tiamat
  – Includes:
    • TIAMAT - Standalone automation of fuel performance code runs using HDF5 MPACT output file
    • May include drivers for other fuel performance codes such as FAST, FRAPCON and ENIGMA
  – Includes libraries: Moose, MPI, PetSc, Futility, etc.
  – Depends on: Bison

• VERAShift
  – Includes:
    • Subset of Shift (which also includes Omnibus, Denovo, … treated as library)
  – Includes libraries: MPI, PetSc, etc.
  – Depends on: MPACT
VERA QA Program Plan (QAPP)

- QAPP is the overarching quality plan for VERA
- QAPP defines:
  - Quality Requirements (high-level)
  - Procedures
  - Documentation
  - Training requirements
  - Roles and Responsibilities
- QAPP will define lower Product QA documentation
- QAPP is being independently developed with SLT and ORNL Performance Assurance and Quality organization

Work in Progress
Product QA Requirements

• Each Product will be required to generate the following documentation:
  a) Software Management Plan (SMP)
  b) Theory Manual(s)
  c) User Manuals(s)
  d) Software Test Plan (STP)
  e) Software Test Report (STR)

• This is a preliminary list and details will change based upon the development of the QAPP
Questions?

A DOE Energy Innovation Hub

www.casl.gov