Maintaining Quality Assurance Within Software Evolution: Lessons Learned With PFLOTRAN

Jennifer M. Frederick and Glenn E. Hammond

SAND2017-9211-C
What is Software Evolution?

**soft·ware ev·o·lu·tion**
ˈsôf(t)wer/ /ˌevəˈloʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to repeated improvements and updates.
What is Software Evolution?

Standard definition:

The gradual development of code, from a simple to a more complex form, due to repeated improvements and updates.

New domain science:

- New process models
- Increasingly mechanistic process models
- Programming paradigms
- Novel numerical methods
What is Software Evolution?

**Standard definition:**

The gradual development of code, from a simple to a more complex form, due to *repeated* improvements and updates.

**New computational science:**

- Changing third party libraries
- Changing operating systems
- Changing programming language
- New or architecturally changing computer hardware
What is Software Evolution?

**Software Evolution**

ˈsôf(t)wer/ /ˌevəˈloʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to **repeated** improvements and updates.

---

When software is constantly evolving, how can we ensure software quality?
What is Software Quality Assurance?

**software quality assurance**
ˈsôf(t)wer/ ˈkwälədē/ əˈSHoʊrəns/

IEEE standard:

A planned and systematic pattern of all actions necessary **to provide adequate confidence** that software conforms to established technical requirements.

When software is constantly evolving, how can we ensure software quality?
What is Software Quality Assurance?

Software quality assurance
ˈsôf(t)wer/ ˈkwälədē/ əˈSHôərəns/

IEEE standard:

A planned and systematic pattern of all actions necessary to provide adequate confidence that software conforms to established technical requirements.

Software quality includes:

- Correctness
- Reliability
- Efficiency
- Survivability
- Maintainability
- Testability
- Availability
- Portability
What is Software Quality Assurance?

**software quality assurance**

ˈsôf(t)wer/ˈkwälədē/əˈSHōrəns/

IEEE standard:

A planned and systematic pattern of all actions necessary to **provide adequate confidence** that software conforms to established technical requirements.

How does PFLOTRAN ensure software quality under an evolving software framework?
Reactive multiphase flow and transport code for porous media

Open source license (GNU LGPL 2.0)

Object-oriented Fortran 2003/2008
- Pointers to procedures
- Classes (extendable derived types with member procedures)

Founded upon well-known (supported) open source libraries
- MPI, PETSc, HDF5, METIS/ParMETIS/CMAKE

Demonstrated performance
- Maximum # processes: 262,144 (Jaguar supercomputer)
- Maximum problem size: 3.34 billion degrees of freedom
- Scales well to over 10K cores
- Nuclear waste disposal
  - Waste Isolation Pilot Plant (WIPP) in Carlsbad, NM
  - DOE Spent Fuel and Waste Science & Technology Program
  - SKB Forsmark Spent Fuel Nuclear Waste Repository (Sweden, Amphos21)
- Climate: coupled overland/groundwater flow; CLM
  - Next Generation Ecosystem Experiments (NGEE) Arctic
  - DOE Earth System Modeling (ESM) Program
- Biogeochemical transport modeling
- CO₂ sequestration
- Enhanced geothermal energy
- Radioisotope tracers
- Colloid-facilitated transport

Simulation by Emily Stein, SNL

~7800 m
10,892,330 hexes 1024 cores ~2 hours

www.pflotran.org

pa.sandia.gov
Steps taken to minimize impacts of software evolution:

- Open source development
- Software configuration management
- Modular object oriented design
- Automated testing suites
- Online documentation
Open Source Development

- Google
  “Open source is an adjective denoting software for which the original source code is made freely available and may be redistributed and modified.”

- Open source software refers to code that:
  - Is free
  - Is publicly available
  - Can be legally modified
  - Can be legally shared with anyone
Open Source Development

- PFLOTRAN has an open source GNU Lesser General Public License (LGPL).
  - The original or modified source may not be sold for profit.
  - Third-party software linked to or wrapped around PFLOTRAN (e.g. graphical user interfaces [GUIs], pre-/post-processing tools, etc.) may be proprietary.
Benefits of Open Source Software

- Encourages collaboration
  - Development, testing, debugging can be shared

- Transparency exposes implementation details critical to scientific reproducibility, but excluded by journal publications.

- More optimal use of funding
  - Funding pooled across diverse set of projects/budgets.
  - What would have been spent on licensing fees can be redirected toward development.
  - Infinite benefit to those unfunded

- The open source community can drive the code to evolve beyond the original vision (evolution).

- The most fit codes tend to survive (natural selection)
PFLOTRAN employs the Git distributed source control management tool for configuration management.

Git logs all changes to a code repository
- version control

Git allows developers to:
- Clone the base repository
- Modify and test code in a development branch
- Merge changes back into base repository
- Pinpoint problematic changesets (snapshots of code versions)
Software Configuration Management

- The PFLOTRAN source code repository is hosted at Bitbucket.org.
  https://bitbucket.org/pflotran/pflotran

- Bitbucket is a web-based hosting service for software development projects that use Git.

- Provides:
  - Git operations (clone, fork, branch, etc.)
  - Wiki for information or documentation
  - Source tree
  - Pull requests
  - Commit logs
  - Issue tracker

critical for software quality assurance (availability)
PFLOTRAN is an open source, state-of-the-art massively parallel subsurface flow and reactive transport code. The code is developed under a GNU LGPL license allowing for third parties to interface proprietary software with the code, however any modifications to the code itself must be documented and remain open source. PFLOTRAN is written in object oriented, freely formatted Fortran 2003. The choice of Fortran over C/C++ was based primarily on the need to enlist and preserve tight collaboration with experienced domain scientists, without which PFLOTRAN’s sophisticated process models would not exist.

PFLOTRAN employs parallelization through domain decomposition using the MPI-based PETSc framework with pflotran tracking the git xssh-0.2.0 branch of PETSc available through Bitbucket.

Please visit our website, pflotran.org, for more information. Looking for the old wiki? The old wiki can still be accessed here.

Travis CI Status

master branch

Installation Instructions

PFLOTRAN can be installed on Linux, Mac, and Windows systems. Click for installation instructions.

Documentation

All PFLOTRAN documentation can be found on the documentation site within pflotran.org. Documentation includes a user's guide, theory guide, as well as input deck card pages.

Contact Information

Please use the mailing lists instead of emailing developers directly. This will allow more developers to answer questions and the answers would be archived.
Software Configuration Management

PFLOTRAN

PFLOTRAN is an open source, state-of-the-art massively parallel subsurface flow and reactive transport code. The code is developed under a GNU GPL license allowing for third parties to interface proprietary software with the code, however any modifications to the code itself must be documented and remain open source. PFLTRAN is written in object oriented, free formatted Fortran 2003. The choice of Fortran over C/C++ was based primarily on the need to enlist and preserve tight collaboration with experienced domain scientists, without which PFLOTRAN’s sophisticated process models would not exist.

PFLOTRAN employs parallelization through domain decomposition using the MPI-based PETSc framework with pflotran tracking the git xsk:0.3.0 branch of PETSc available through Bitbucket.

Please visit our website, pflotran.org, for more information. Looking for the old wiki? The old wiki can still be accessed here.

Travis CI Status

Master branch

Installation Instructions

PFLOTRAN can be installed on Linux, Mac, and Windows systems. Click for installation instructions.

Documentation

PFLOTRAN documentation can be found on the documentation site within pflotran.org. Documentation includes a user’s guide, theory guide, as well as input deck card pages.

Contact Information

Please use the mailing lists instead of emailing developers directly. This will allow more developers to answer questions and the answers would be archived.
Software Configuration Management
Software Configuration Management

Pull requests are reviewed, commented on, and scrutinized by a second pair of eyes.
Software Configuration Management

Unique commit numbers and descriptions.

Automatic builds each time the code has changed.
Modular Object Oriented Design

- Object oriented design involves the organization of data and procedures into a hierarchy of containers (objects).

- The use of objects:
  - Improves data locality (modularity)
  - Eases code refactoring (rewriting)
  - Facilitates extensibility (for adding capability)

- PFLOTRAN uses object oriented modern FORTRAN 2003/2008
Automated Testing Suites

- As open source development fosters a growing community of developers, the code can’t break!

- **Unit tests**
  - Individual routines are executed in isolation.
  - Results are compared with a gold standard to within a tolerance.

- **Regression tests** – focus on changes in simulation results
  - Full simulations are executed.
  - Simulations results are sampled and compared to a gold standard to within a tolerance.

- **Verification tests**
  - Full simulations are executed, for which there is a known solution.
  - Simulation results are sampled and compared to an analytical solution within a tolerance.

```python
if (abs(test_value - gold_standard) > tolerance) report_error()
```
Automated Testing Suites

- As open source development fosters a growing community of developers, *the code can’t break!*.

- **Unit tests**
  - Individual routines are executed in isolation.
  - Results are compared with a gold standard to *within a tolerance*.

- **Regression tests** – focus on changes in simulation results
  - Full simulations are executed.
  - Simulations results are sampled and compared to a gold standard to *within a tolerance*.

- **Verification tests**
  - Full simulations are executed, for which there is a known solution.
  - Simulation results are sampled and compared to an analytical solution *within a tolerance*.

```python
if (abs(test_value - gold_standard) > tolerance) report_error()
```

*Why a tolerance?*
Accommodates small variations in software and hardware configurations (Linux vs Mac)
Automated Testing Suites: Regression Tests

This is what a successful run of the unit tests and regression tests looks like:
Automated Testing Suites: Regression Tests

- Example regression test failure:
  - Perturb the critical pressure for the water equation of state by 10 billionths of a percent

```diff
diff -r f9f01bbf557a src/pflotran/eos_water.F90
--- a/src/pflotran/eos_water.F90 Thu Jul 28 18:59:00 2016 -0700
+++ b/src/pflotran/eos_water.F90 Fri Jul 29 10:31:57 2016 -0700
@@ -893,6 +893,7 @@
tc1 = H2O_CRITICAL_TEMPERATURE ! K
pc1 = H2O_CRITICAL_PRESSURE ! Pa
+ pc1 = pc1 + 1.d-10*H2O_CRITICAL_PRESSURE ! perturb by 1e-10
vc1 = 0.00317d0 ! m^3/kg
utc1 = one/tc1 ! 1/C
upc1 = one/pc1 ! 1/Pa
```
Automated Testing Suites: Regression Tests

This is what a failed run of the unit tests and regression tests looks like:
Automated Testing Suites: Regression Tests

pflotran-tests-2016-07-29_10-27-50.testlog

543_flow-np8...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/local/bin/mpiexec -np 8 /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix 543_flow-np8
  # 543_flow-np8 : run time : 1.31 seconds
  diff 543_flow-np8.regression.gold 543_flow-np8.regression
  543_flow-np8... passed.

543_hanford_srfcplx_param...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix 543_hanford_srfcplx_param
  # 543_hanford_srfcplx_param : run time : 2.91 seconds
  diff 543_hanford_srfcplx_param.regression.gold 543_hanford_srfcplx_param.regression
  FAIL: LIQUID VELOCITY [m/d]:1 : 1.084136795e-11 > 1e-12 [relative]
  FAIL: LIQUID VELOCITY [m/d]:31 : 7.3779567027e-12 > 1e-12 [relative]
  FAIL: LIQUID VELOCITY [m/d]:31 : 1.76111798338e-12 > 1e-12 [relative]
  FAIL: LIQUID VELOCITY [m/d]:29 : 2.25552127701e-12 > 1e-12 [relative]
  FAIL: LIQUID VELOCITY [m/d]:29 : 1.61796082447e-11 > 1e-12 [relative]
  FAIL: UO3.2H2O SI:Min : 4.37393289458e-12 > 1e-12 [relative]
  FAIL: UO2(PO3)2 SI:Min : 4.34539859641e-12 > 1e-12 [relative]
  FAIL: UO2SO4 SI:Min : 4.32535887832e-12 > 1e-12 [relative]
  FAIL: Torbernite SI:Min : 8.75578249827e-12 > 1e-12 [relative]
  FAIL: (UO2)3(PO4)2.4H2O SI:Min : 1.30878004044e-11 > 1e-12 [relative]
  FAIL: UO2CO3 SI:Min : 4.36306510613e-12 > 1e-12 [relative]
  FAIL: UOF4 SI:Min : 4.34665543516e-12 > 1e-12 [relative]
  FAIL: Saleeite SI:Min : 8.72937379374e-12 > 1e-12 [relative]
  FAIL: Schoepite SI:Min : 4.37393289458e-12 > 1e-12 [relative]
  543_hanford_srfcplx_param... failed.

this test didn't use the change in eos_water.f90, so it passes

this test used the change in eos_water.f90, so it fails
Automated Testing Suites: Verification Tests

- A set of > 50 tests that verify the code against an analytical solution
- Automatic spatial convergence testing is also performed
- The tests can be run each time a major portion of the code changes
Automated Testing Suites: Verification Tests

- 2D Domain (10x10 cells)
- Heat Conduction (steady state solution)
- Dirichlet (scalar) temperature boundary conditions

\[ T = \frac{x}{L}^\circ C \text{ north face} \]
\[ T = \frac{y}{L}^\circ C \text{ east face} \]
\[ T = 0^\circ C \text{ west face} \]
\[ T = 0^\circ C \text{ south face} \]

Analytical (fill) vs. PFLOTRAN (contour) TH Mode 0.00% erro

Temperature [°C]

Distance [m]

Distance [m]

\[ \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \]

\[ T(x, y) = T_0 \frac{x y}{L L} \]
Automated Testing Suites: Verification Tests

- 3D Domain (10x10x10 cells)
- Fluid flow/pressure field (steady state solution)
- Dirichlet (scalar) pressure boundary conditions

\[ \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = 0 \]

**Analytical solution**

\[ p(x, y, z) = p_0 \left( \frac{x}{L} + \frac{y}{L} + \frac{z}{L} \right) \]
PFLOTRAN’s Online Documentation

- We use a documentation generator program (Sphinx) to generate both the website and the PDF versions of the documentation:

  documentation.pflotran.org
We use a documentation generator program (Sphinx) to generate both the website and the PDF versions of the documentation:
Conclusion

Although software tends to have a limited lifespan, careful design and planning in the development of a code can significantly lengthen the duration of a software application’s viable existence.

PFLOTRAN attempts to minimize the impact of software evolution through:

- Open source development
- Software configuration management
- Modular object oriented design
- Automated testing
- Online documentation