Unit Testing

Best Practices Workshop, March 25-26 2019, Hampton VA

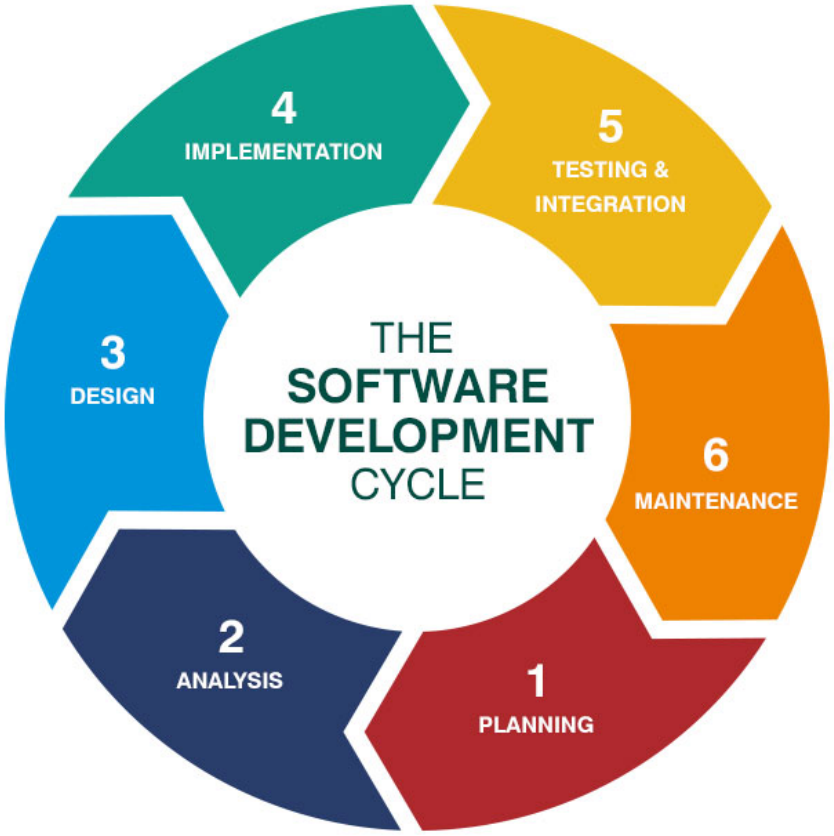
Carlos Cruz
Jules Kouatchou
Brent Smith

NASA GSFC Code 606/610 (ASTG/GMAO)
Greenbelt, Maryland 20771

Outline

- Why is testing important?
- Types of testing
- Unit testing

Testing


4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
THE
SOFTWARE
DEVELOPMENT
CYCLE
6
MAINTENANCE
2
ANALYSIS
1
PLANNING

Why is testing important?

Geoffrey Chang: multidrug transporter proteins

- Some inherited code flipped two columns of data, inverting an electron-density map
- Resulted in an incorrect protein structure
- Resulted in 5 retracted publications: One was cited 364 times
- Many papers and grant applications conflicting with his results were rejected

A scientist's nightmare: software problem leads to five retractions. G. Miller Science. 2006 Dec 22;314(5807):1856-7.

Why is testing important?

The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

The following paragraphs are extracted from the report of the Inquiry Board. An interesting article on the accident and its implications by James Gleick appeared in The New York Times Magazine of 1 December 1996. The CNN article reporting the explosion, from which the above graphics were taken, is also available.

On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.

The failure of the Ariane 501 was caused by the complete loss of guidance and altitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.

The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer.*

*SRI stands for Système de Référence Inertielle or Inertial Reference System.

Software error…due to conversion of floating point to integer

Test Granularity

- Unit tests: test individual components
- Integration tests: test interaction of larger pieces of software
- System tests: test full software system at the user interaction level

Levels of Testing

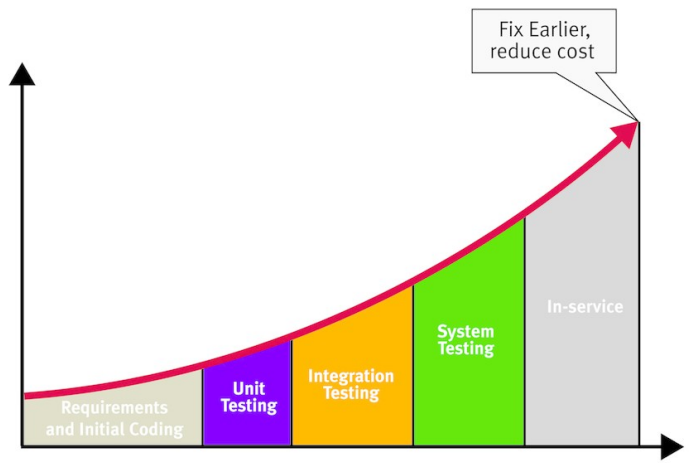
- Verification tests: Does the code implement the intended algorithm correctly?
- Acceptance tests: Assert acceptable functioning for a specific customer
- Regression tests: Compare current observable output to a gold standard
- Performance tests: Focus on the runtime and resource utilization
- Installation tests: Verify that the configure-make-install is working as expected

What is Unit Testing?

The goal of unit testing is to isolate software components at the **smallest testable level** or **unit** and show that each **unit** of the software performs as designed.

Unit Testing Benefits

- Unit testing increases confidence in changing or maintaining code.
- Codes are more reusable.
- The cost of fixing a defect is smaller.
- Debugging is easier.
- Codes are more reliable.

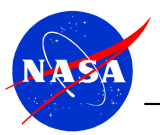


How to unit test?

- Use a version control system.
- Find a tool/framework for your language.
- Isolate the development environment from the test environment.
- Write test cases that are independent of each other.
- Before fixing a defect, write a test that exposes the defect.
- Use test data that is close to that of production.
- Aim at maximizing code coverage.
- Perform unit tests continuously and frequently.

Fortran unit testing frameworks[1]

- FUnit
- Fortran Unit Test Framework (FRUIT)
- FortUnit
- Ftnunit
- pFUnit
- Vegetables
- UnitTest
- Zofu



[1] http://fortranwiki.org/fortran/show/Unit+testing+frameworks

pFUnit

pFUnit is a unit testing framework enabling JUnit-like testing of serial and MPI-parallel software written in Fortran.

Requirements:

- Fortran 2003+
- MPI
- OpenMP
- GNU make
- Python 2.7+
- CMake

Doxygen is used to generate documentation.

Unit testing with Python

Relatively easy to do.

- Use the unittest module.
  - unittest module is built-in!
- Creating test cases is accomplished by subclassing unittest.TestCase.

Demo

How to motivate yourself to write tests

- Tests protect YOU from other people from breaking your work
- You may already have some
- Drivers for generating conference or paper results: Just reduce the problem size
- User submitted bugs: Ask for a file that reproduces the issue
- Examples: Add a pass/fail condition and you have a test

How do I determine what other tests I need?

- Code coverage tools
- Expose parts of the code that aren't being tested
- gcov: standard utility with the GNU compiler collection suite; counts the number of times each statement is executed
- lcov: GUI for gcov

The End

Questions?