Introduction to Unit Test

Authors: LanBT & ThuyNT

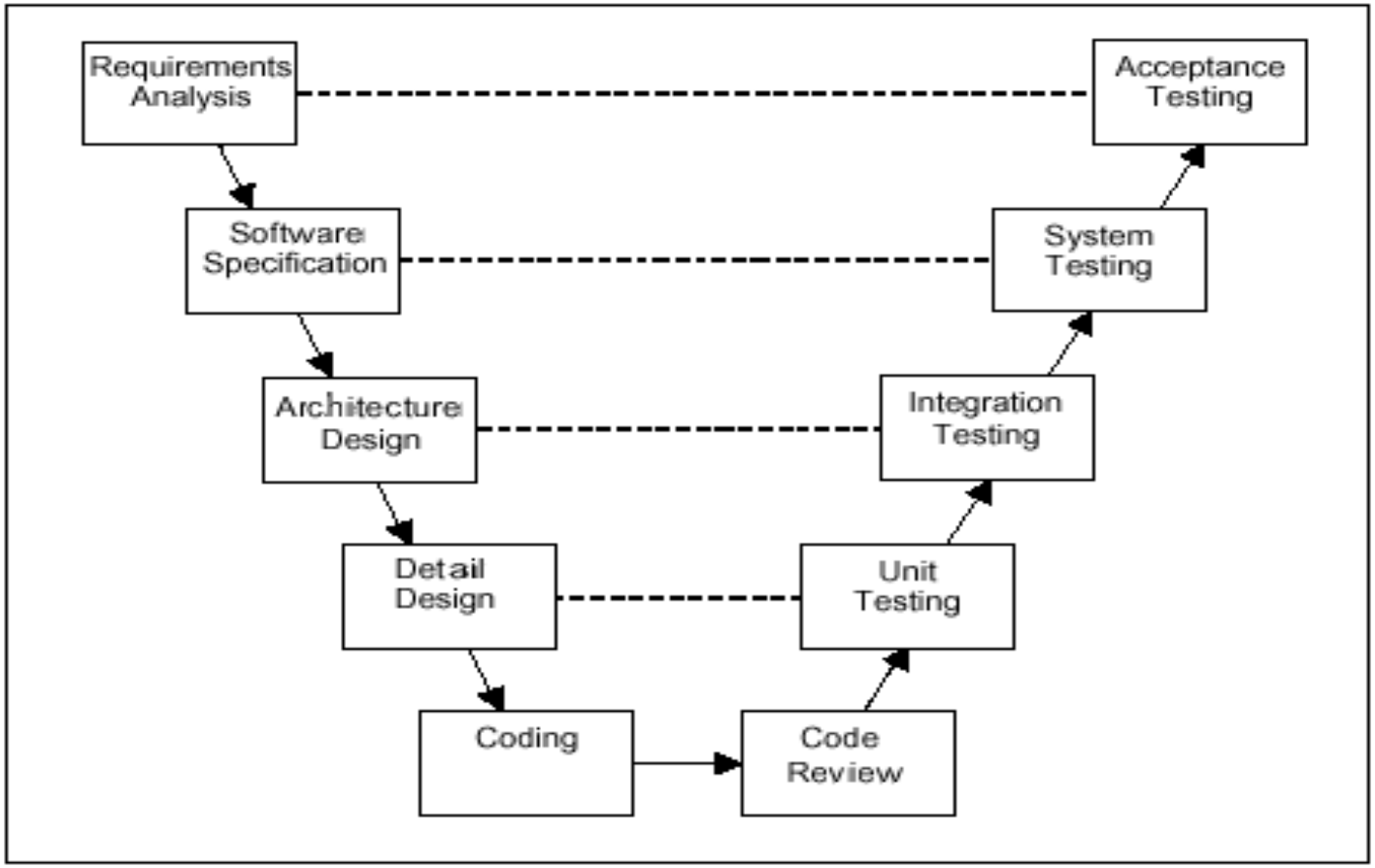
Contents

1. Unit Test Fundamentals: Answer the question of what, why, when doing the Unit Test
2. Methodologies to do Unit Test
3. Unit Testing Tools
   - CUnit/ CPPUnit Framework
   - Parasoft C++ Test 6.7

What is UniTest

- “Unit testing” refers to testing software code at the smallest testable unit (method or function) and base on detail design
- Exception testing
  - Range of feasible input
- Functional testing
  - Conform to specification
  - Black Box Testing
  - White Box testing
- Regression testing
  - Conducted after a change
  - To find new fault

When do Unit Test



Why do Unit Test

- Ensure quality of software unit
- Detect defects and issues early
- Reduce the Quality Effort & Correction Cost

Unit Test - Methodologies

To implement 1 software testing, we use both:

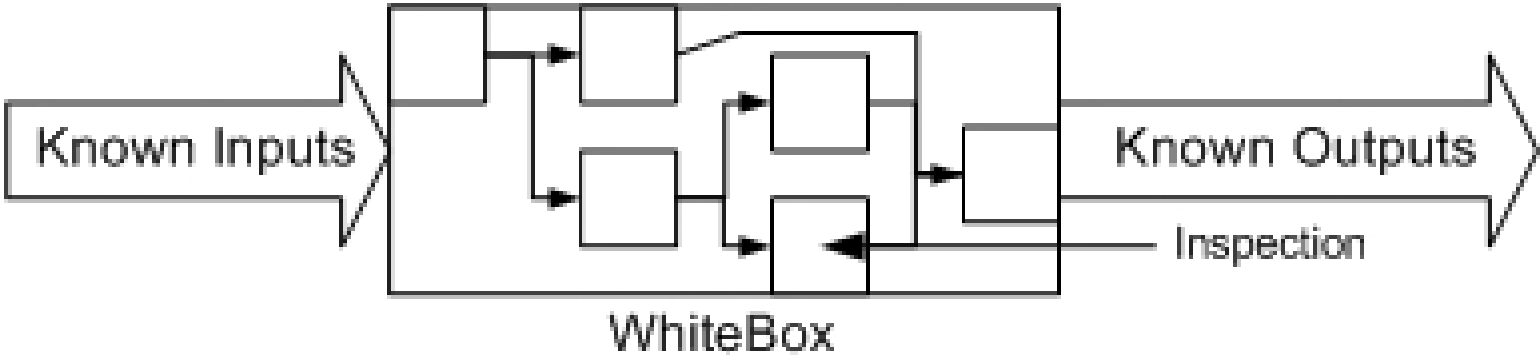
- Black box Test
- White box Test

Back box testing

A Simple Black box Specification

- Based on external behavior of the unit tested
- Specification based testing
- Strategies
  - Equivalence partitioning
  - Boundary-value analysis
  - Combination strategy

White box testing (1)

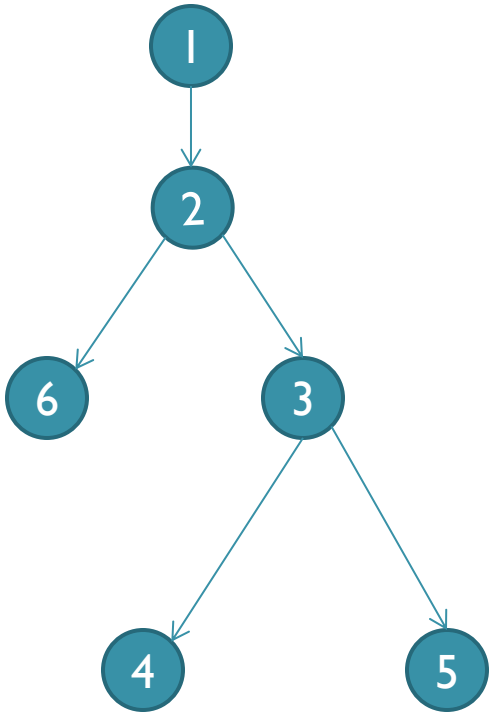


- Based on internal behavior of unit
- Code coverage based testing
- Criteria
  - Statement coverage
  - Decision coverage
  - Path coverage

White box testing (2)

- int Func(int a,int b)

```
1 {
  2 if (a > 0)
      3 if( b > 0)
            4 return(a+b);
        else
            5 return(a-b);
    else
      6 return 0;
  }
```



6 tatements: 1,2,3,4,5,6

4 decisions(branchs):2→6, 2→3, 3→4, 3→5

3 paths: 1 – 2 – 6, 1 – 2 – 3 – 4,1 – 2 – 3 – 5

Ex:Test case: a =1, b = 1 has TC = 4/6, DC = 1/3, PC = 2/4

Cunit Framework

- Cunit is used to test C code *units*
- Using Steps:
  - Write functions for tests
  - Initialize the test registry
  - Add suites to the test registry
  - Add tests to the suites
  - Run tests using an appropriate interface
  - Cleanup the test registry

CPPUnit Framework

- Used to test C/C++ code units
- Using Steps
  - Create your class
  - Create your new testing class deriving from TestFixture class
  - Create the event manager and test controller
  - Add a listener that colllects test result
  - Add a listener that print dots as test run
  - Add the top suite to the test runner
  - Print test in a compiler compatible format

Parasoft C++ test

- Give automated – creating test case testing for C/C++ units (functions, methods)
- Source Unit testing
- Native Unit testing
- Test case Results
- Stub configuration


Q&A