Introduction to Unit Testing

Jun-Ru Chang

2012/05/03
Introduction

• Software is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it.
What is Bug

- What is Software Testing
  - Software testing is an action which attempt to find bugs either manually or through automation tools.
How to Make Bug

Software specifications keep constantly changing

Software design is rushed or changed

Lack of proper skill set in programmers

Time pressure

Software complexity or poor documentation
How to Make Bug (cont.)

- 2, 3: Unimplemented spec
- 1, 2: Unfulfilled Needs
- 4, 7: Unexpected Behavior
- 6, 7: Undesired Behavior
Cost of Fixing Defects

- The earlier a defect is found, the cheaper it is to fix it.

<table>
<thead>
<tr>
<th>Time introduced</th>
<th>Cost to fix a defect</th>
<th>Requirements</th>
<th>Architecture</th>
<th>Construction</th>
<th>System test</th>
<th>Post-release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td></td>
<td>1x</td>
<td>3x</td>
<td>5-10x</td>
<td>10x</td>
<td>10-100x</td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
<td>-</td>
<td>1x</td>
<td>10x</td>
<td>15x</td>
<td>25-100x</td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td>-</td>
<td>-</td>
<td>1x</td>
<td>10x</td>
<td>10-25x</td>
</tr>
</tbody>
</table>

Software testing

• Dijkstra’s criticism, “Program testing can be used to show the presence of bugs, but never to show their absence”
  – Only as good as the test data selected
  – Compared to “expected output”
Software testing (cont.)

• Methodologies
  – Black box testing
  – White box testing

• Myths about testing
  – Bugs are simple to remove
  – A bug is caused in exactly one module
  – Most bugs will be caught by the compiler
  – Bug fixes always make the program better
    • Imperfect debug
Software testing (cont.)

• Test case
  – “Bugs lurk in corners and congregate at boundaries…”
  – The pesticide paradox
Boundary Value Analysis

- Boundary value analysis
  - Input data
  - Loop iteration
  - Output fields
Boundary Value Analysis (cont.)

- Robustness boundary value analysis
- Worst case boundary value analysis
- Robust worst case boundary value analysis

Test cases for a variable $x$, where $a \leq x \leq b$
Boundary Value Analysis (cont.)

• Hierarchy
  – Boundary value testing: $4n+1$
  – Robustness: $6n+1$
  – Worst case: $5^n$
  – Robust worst case: $7^n$
White-box Testing
Software testing (cont.)

• Complete testing
  – At the end of testing, you know there are no remaining unknown bugs.

IMPOSSIBLE
Can’t test all inputs, timing, and paths
Software testing (cont.)

• When to stop testing?
  – Cost
  – Coverage strategy
Software testing (cont.)

• Type of software testing
  – Unit testing
  – Integration testing
  – Function testing
  – System testing
  – Load testing
  – Stress testing
  – Performance testing
  – Regression testing
  – …etc.
Regression testing

• Seek to uncover new errors in existing functionality after changes have been made to a system, such as functional enhancements, patches or configuration changes.
Extreme Programming
Test-Driven Development (TDD)

• TDD is an evolutionary approach to development which combines **test-first** development where you write a test before you write just enough production code to fulfill that test and **refactoring**.
Unit Testing Framework

• Kent Beck
  – Simple Smalltalk Testing
  – JUnit
• CUnit, NUnit, C++Unit...
  – XUnit
CUnit

• CUnit is a lightweight system for writing, administering, and running unit tests in C. It provides C programmers a basic testing functionality with a flexible variety of user interfaces.
CUnit (cont.)

- CU_initialize_registry(): Initialize the test registry
- CU_add_suite(): Add suite to the test registry
- CU_add_test(): Add tests to the suites
- CU_console_run_tests(): Run tests
- CU_cleanup_registry(): Cleanup the test registry
- CU_ASSERT(int expression)
CUnit (cont.)

if(CUE_SUCCESS != CU_initialize_registry()){
    return CU_get_error();
}

end:

CU_cleanup_registry();
return CU_get_error();
CUnit (cont.)

CU_pSuite addSuite = CU_add_suite("add_1", init_add_1, clean_add_1);

void testadd1(){
    CU_ASSERT( 0 == add(0, 0));
    CU_ASSERT( 2 == add(2, 0));
}

void testadd2(){
    CU_ASSERT(-1 == add(0, -1));
    CU_ASSERT(-2 == add(-1, -2));
}
CUnit (cont.)

if(CU_add_test(addSuite, "correct suite", testadd1) == NULL ||
   CU_add_test(addSuite, "fail suite", testadd2) == NULL)
goto end;

CU_basic_run_tests();
CUnit (cont.)

• Compile & execution
  – gcc add.c tc1.c -lcunit
  – ./a.exe

• Official website:
  http://cunit.sourceforge.net/index.html
$ ./a.exe

CUnit - A unit testing framework for C - Version 2.1-2
http://cunit.sourceforge.net/

Suite add_1, Test fail suite had failures:
1. tc1.c:22 - -2 == add(-1, -2)
clean

Run Summary: Type Total Ran Passed Failed Inactive
suites 1 1 n/a 0 0
tests 2 2 1 1 0
asserts 4 4 3 1 n/a

Elapsed time = 0.000 seconds
Setup

• cygwin
  – http://www.cygwin.com/
  – choose download
    • add ftp://ftp.ntu.edu.tw/cygwin
  – select package
    • gcc: Devel -> gcc-core: C compiler
    • cunit: Libs -> CUnit
  – C:\cygwin\home\USER_NAME
Practice

• Fibonacci Sequence
  – $F(0) = 1$, $F(1) = 1$
  – $F(m) = F(m-1) + F(m-2)$, $m \geq 0$
Practice

• 4 Basic Arithmetic Operations
  – Integer