Introduction to Unit Testing

Jun-Ru Chang

2012/05/03

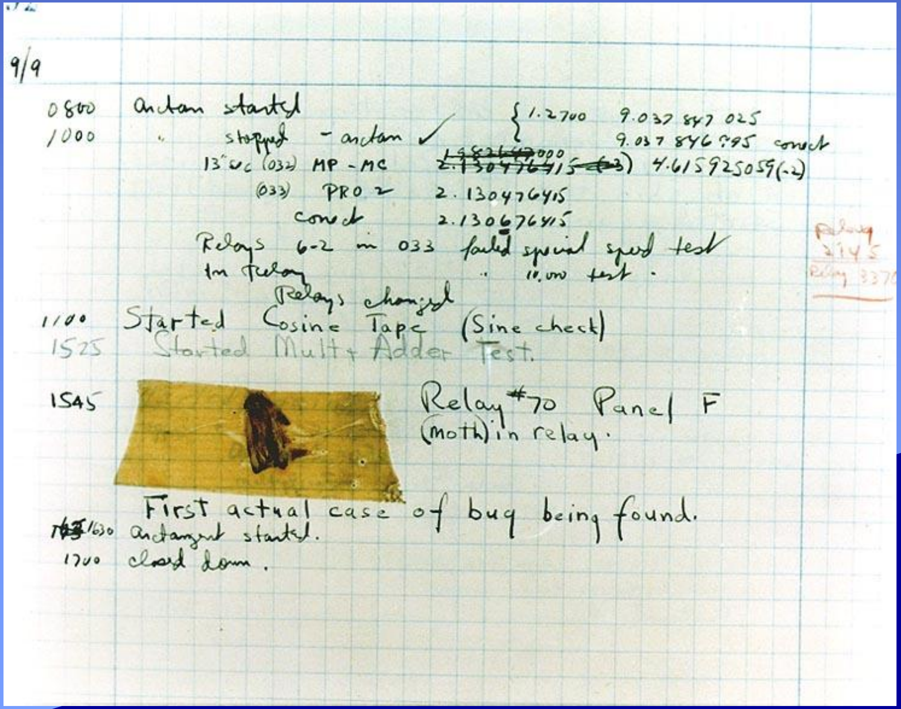
Introduction

- 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.

```
Connection establishing. Please wait ................100%
Connected!
        ****************************************      V4.01
          MAGI SYSTEM REMOTE TERMINAL
             AUTHORIZED UES ONLY
        ****************************************
SECURITY LEVEL:EEE              SECURITY:CHECK
PERSONAL CODE: ------           NAME: ----------
MAGI LOCKED
Device : 0X04 FF71, Apple iPhone....................CHECKED
License : EVA Pilot, Second.C ........................CHECKED
Encryption : TRUE ........................................CHECKED
USERNAME : Souryu Asuka Langray
PASSWORD : **********                              PASSED
Please scan fingerprint blow to gain entry
```

What is Bug

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

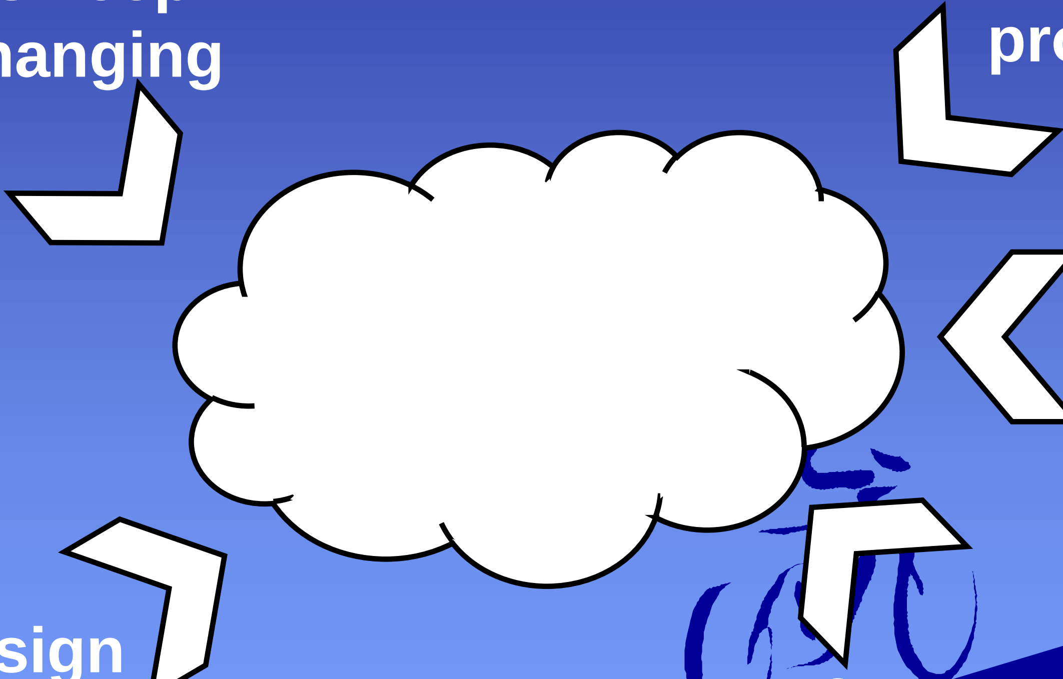


9/9

0800 Arctan started
1000 " stopped - arctan ✓ {1.2700 9.037 847 025
9.037 846 995 correct
13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ (-3)) 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10,000 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antangent started.
1700 closed down.

Relay 2145
Relay 3370

How to Make Bug



Software specifications keep constantly changing

Lack of proper skill set in programmers

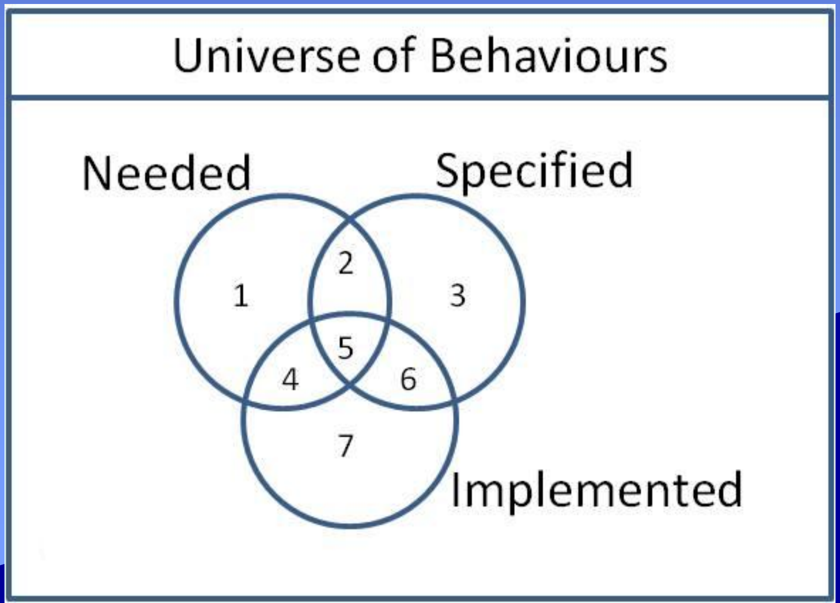
Time pressure

Software design is rushed or changed

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.

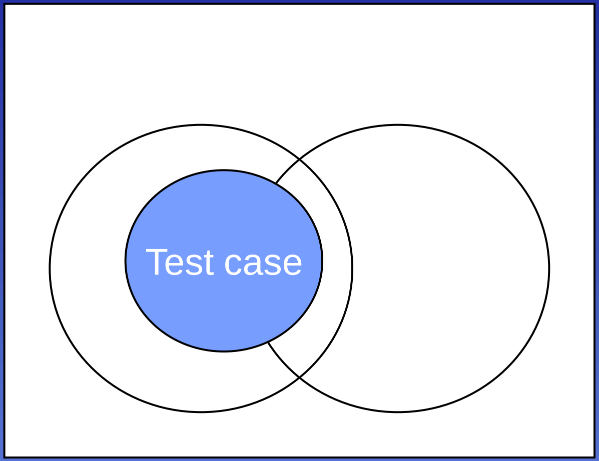
Cost to fix a defect		Time detected				
		Requirements	Architecture	Construction	System test	Post-release
Time introduced	Requirements	1x	3x	5-10x	10x	10-100x
	Architecture	-	1x	10x	15x	25-100x
	Construction	-	-	1x	10x	10-25x

source: McConnell, Steve (2004). Code Complete (2nd ed.). Microsoft Press. pp. 29. ISBN 0-7356-1967-0

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”

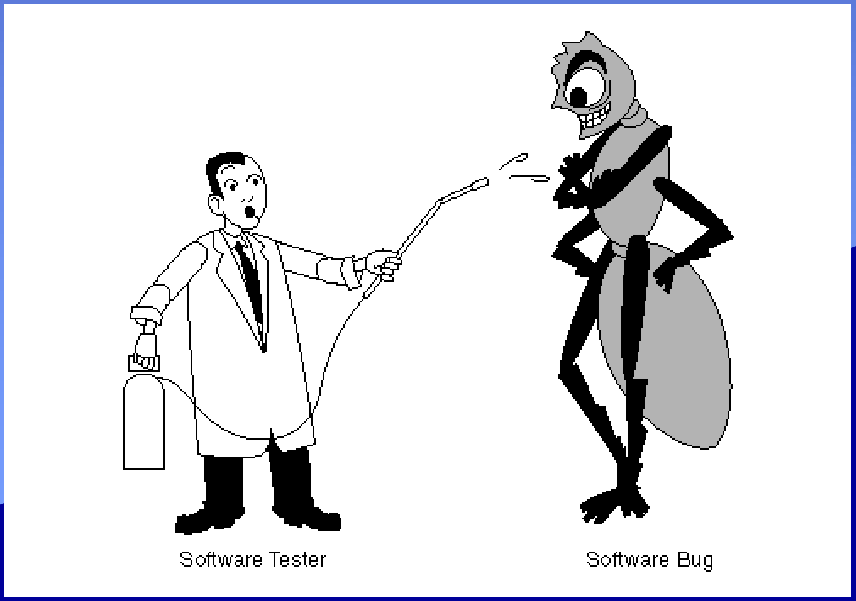
Software testing (cont.)



- 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

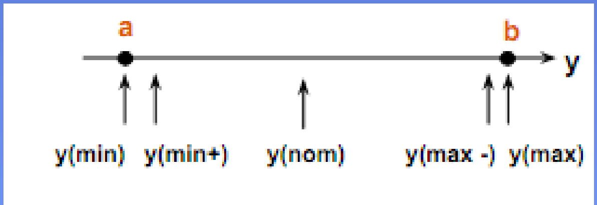
Software testing (cont.)

- Test case
  - “Bugs lurk in corners and congregate at boundaries…”
  - The pesticide paradox

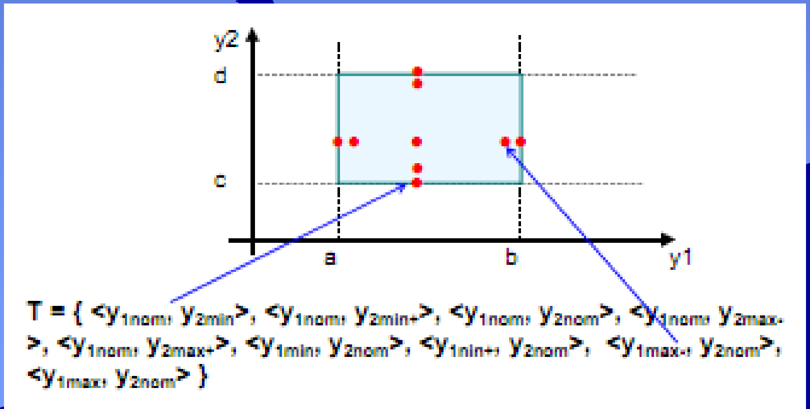


Boundary Value Analysis

- Boundary value analysis
  - Input data
  - Loop iteration
  - Output fields



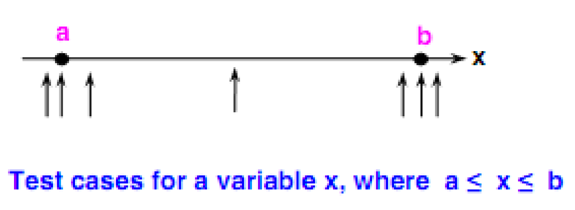
Test cases for a variable y, where a ≤ y ≤ b



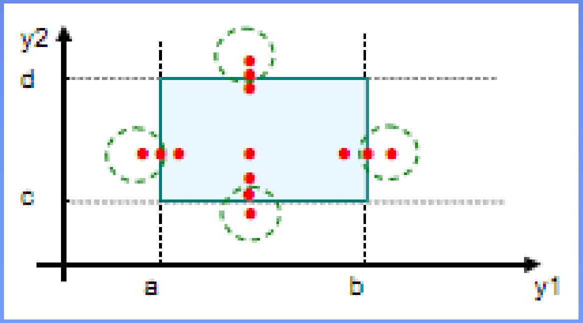
$T = \{ <y_{1nom}, y_{2min}>, <y_{1nom}, y_{2min+}>, <y_{1nom}, y_{2nom}>, <y_{1nom}, y_{2max-}>, <y_{1nom}, y_{2max+}>, <y_{1min}, y_{2nom}>, <y_{1nin+}, y_{2nom}>, <y_{1max-}, y_{2nom}>, <y_{1max}, y_{2nom}> \}$

Boundary Value Analysis (cont.)

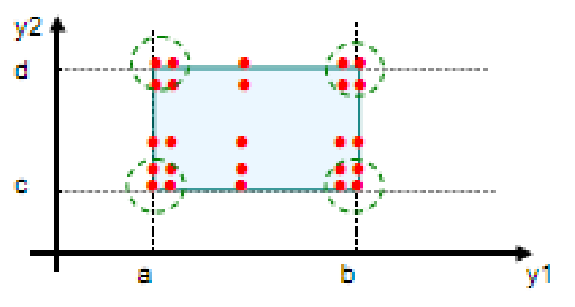
- Robustness boundary value analysis



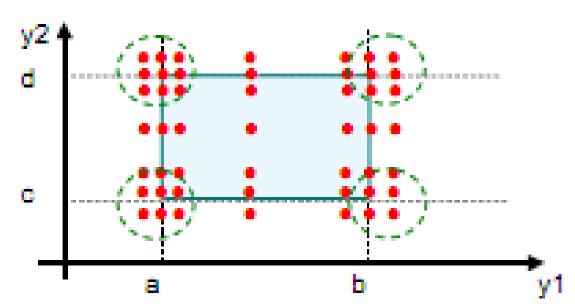
Test cases for a variable x, where a ≤ x ≤ b



- Worst case boundary value analysis



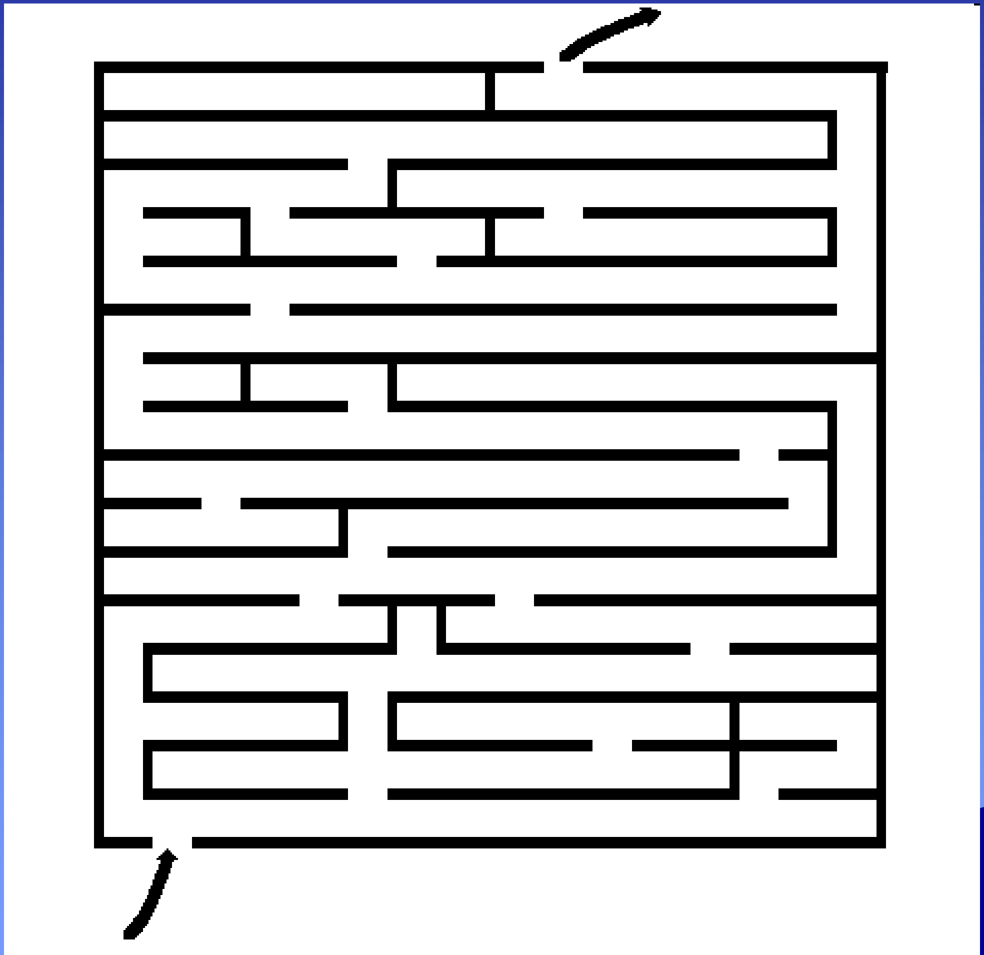
- Robust worst case boundary value analysis



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.)

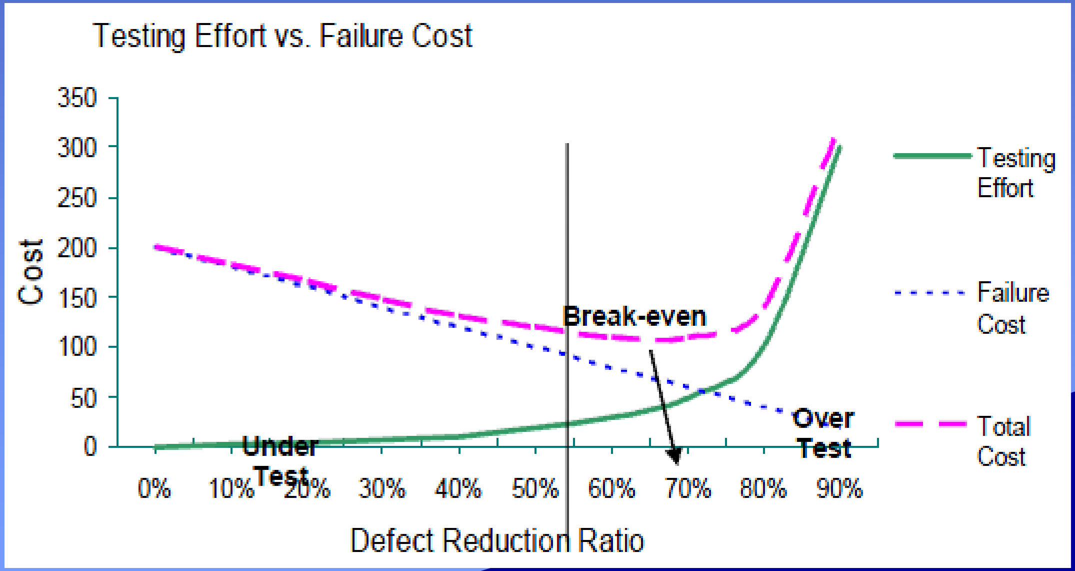
- Complete testing
  - At the e l of test yo ow there are no remaining

IMPOSSIBLE

Can’t test all inputs, timing, and paths

Software testing (cont.)

- When to stop testing?
  - Cost
  - Coverage strategy



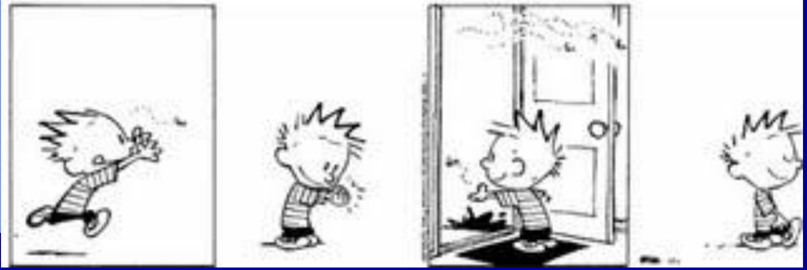
Software testing (cont.)

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

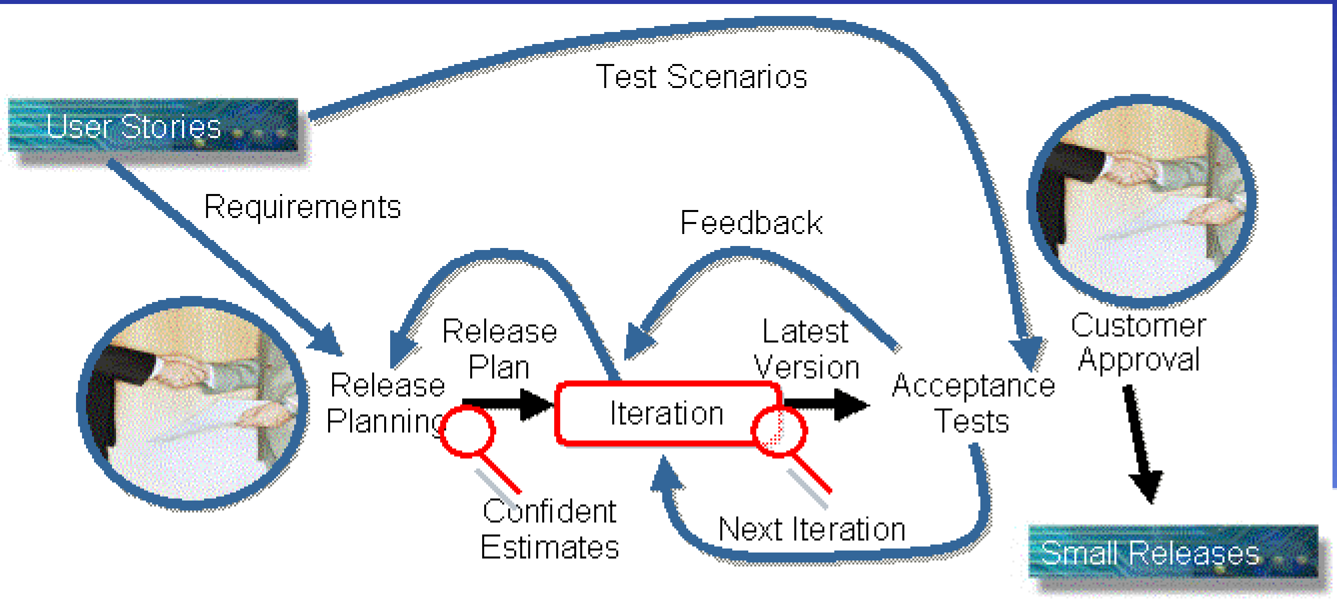
Regression testing

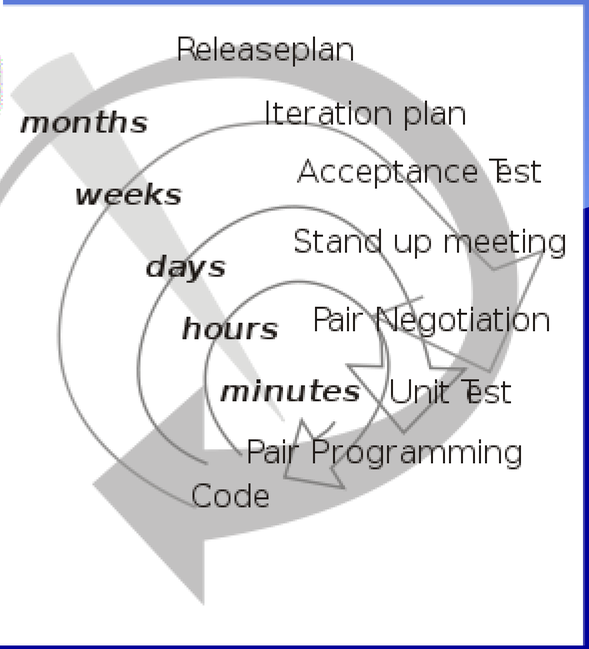
- Seek to uncover new errors in existing functionality after changes have been made to a system, such as functional enhancements, patches or configuration changes.

Regression:
"when you fix one bug, you introduce several newer bugs."



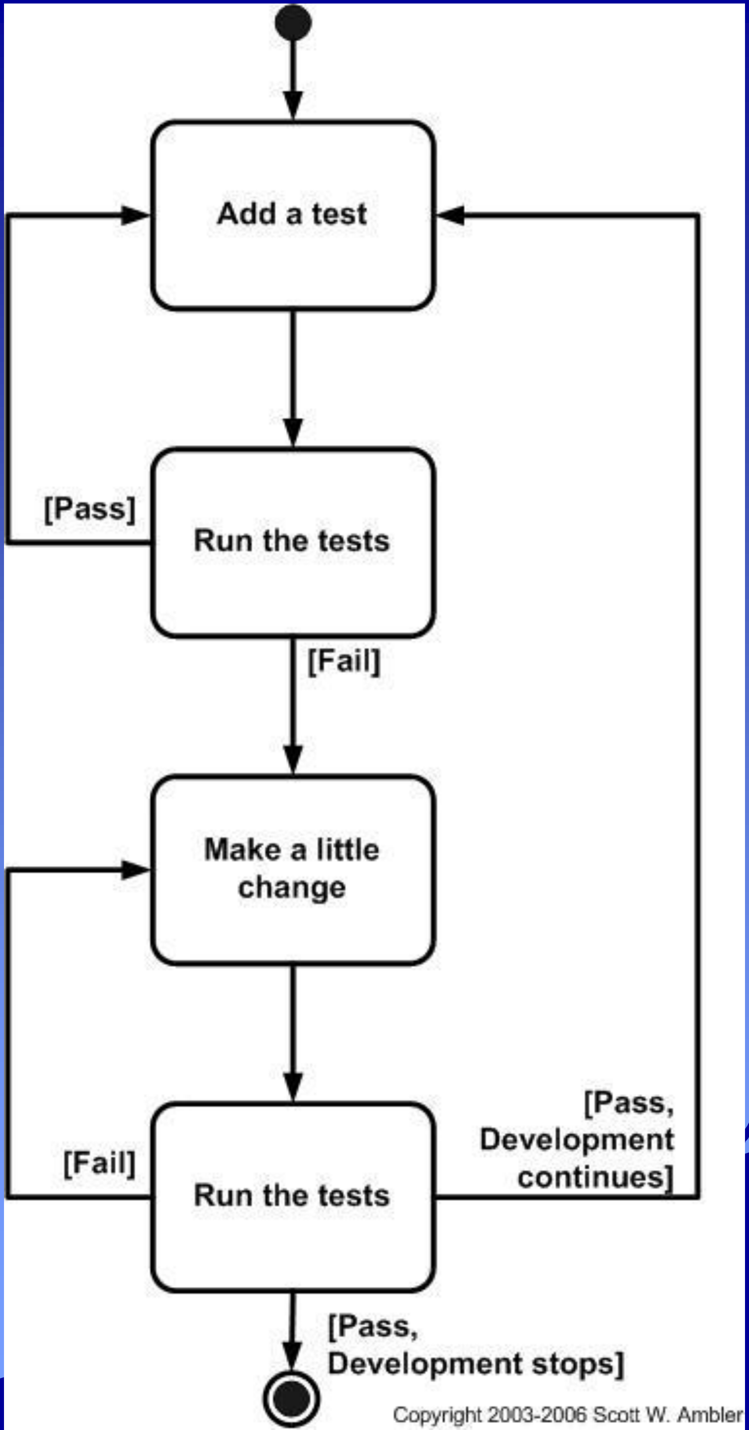
Extreme Programming





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**.



Copyright 2003-2006 Scott W. Ambler

Unit Testing Framework

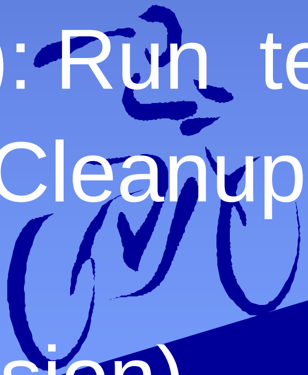
- Kent Beck
  - Simple Smalltalk Testing
  - JUnit
- CUnit, NUnit, C++Unit...
  - XUnit

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

tc1.c

add.c

add.h

CUnit (cont.)

```
$ ./a.exe


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

init
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
```

execute init_add_1

issue code

execute clean_add_1

execution time

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
  - $F(0) = 1$, $F(1) =1$
  - $F(m) = F(m-1) + F(m-2)$, $m>=0$

Practice

- 4 Basic Arithmetic Operations
  - Integer