Introduction to Automated Unit Testing (xUnit)

Brian Nielsen

Arne Skou

{bnielsen | ask}@cs.aau.dk

AALBORG UNIVERSITY DENMARK

BRICS Basic Research in Computer Science

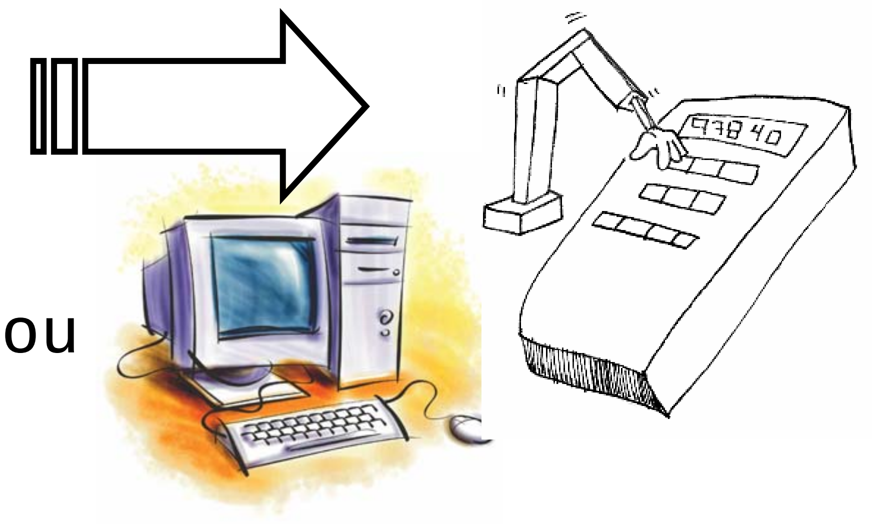
CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Conventional Test Execution

- Ad hoc manner
  - Manual stimulation & observation
  - E.g. adding a function to a module, which runs tests on the module’s functions
  - Uncomenting or deleting test code / drivers / printf /#ifdefs
  - Assert and debug builds
  - Home-brewed test-code and test runners

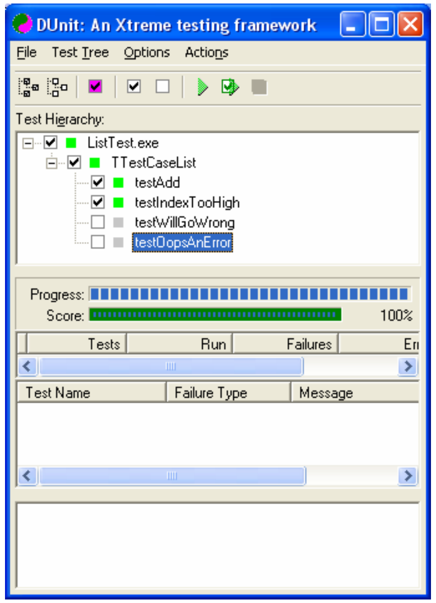
Automated Testing

- "Code that isn't tested doesn't work"
- "Code that isn't regression tested suffers from code rot (breaks eventually)"
- "If it is not automated it is not done!"
  - Boring
  - Repetitive
  - Necessary
  - Error-prone (for humans)
  - Better done by you than your users



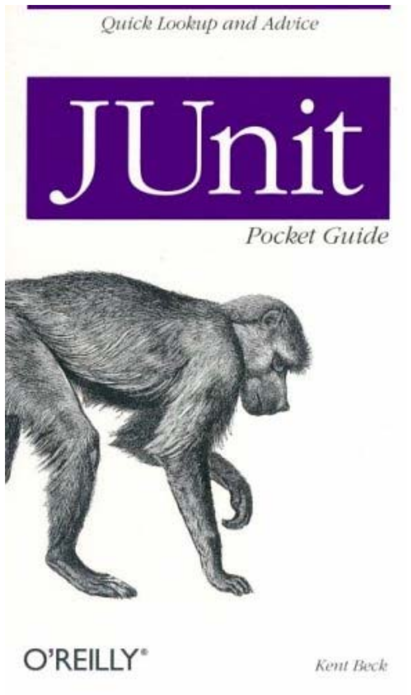
What is a testing framework?

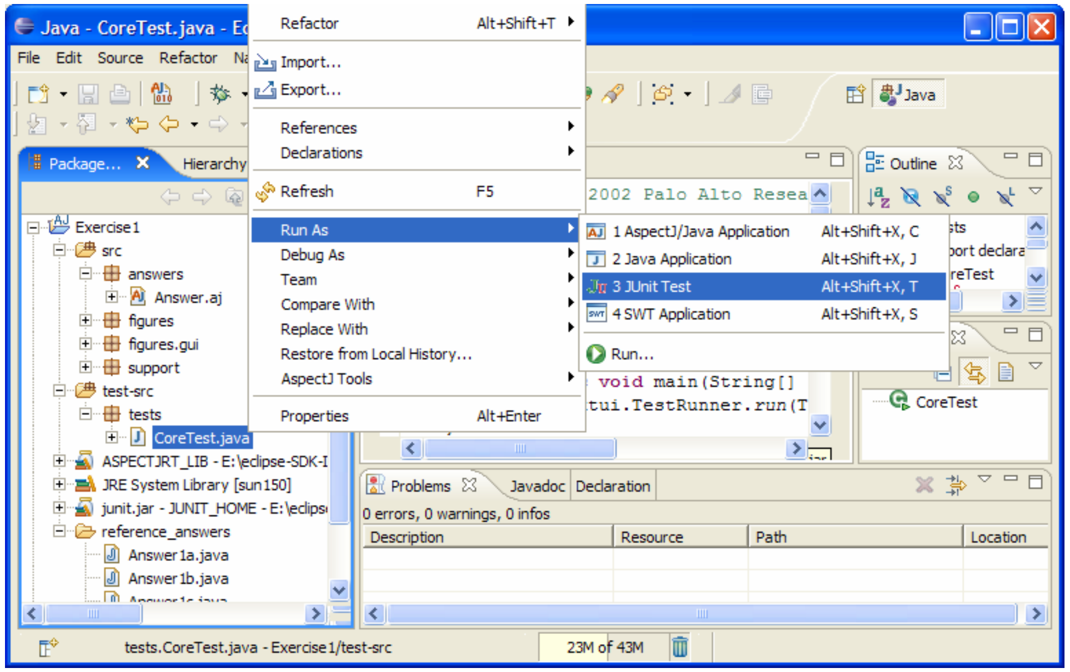
- A test framework is a software tool for writing and running unit-tests
- provides reusable test functionality which:
  - Enables automatic execution for regression tests
  - Is standardized
  - Easy to use
  - GUI-test case browser/runner
  - Test report generation



What is a testing framework

- Programmer Friendly
  - Test cases written in same language as implementation
  - Well integrated in IDE’s

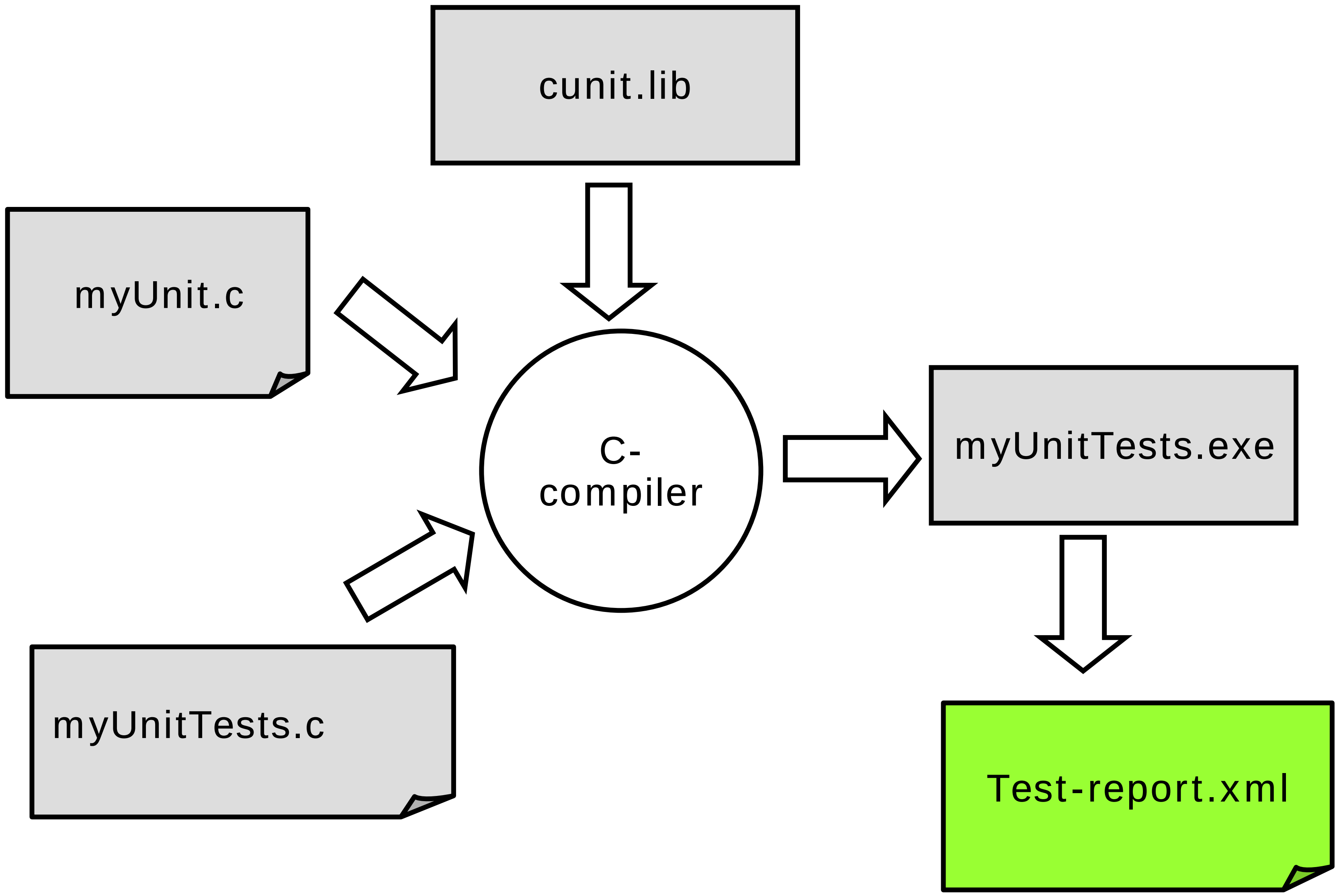



Java - CoreTest.java - Ec
File Edit Source Refactor Na
Package... Hierarchy
Exercise1
src
answers
Answer.aj
figures
figures.gui
support
test-src
tests
CoreTest.java
ASPECTJRT_LIB - E:\eclipse-SDK-I
JRE System Library [sun150]
junit.jar - JUNIT_HOME - E:\eclips
reference_answers
Answer1a.java
Answer1b.java
Refactor Alt+Shift+T
Import...
Export...
References
Declarations
Refresh F5
Run As
Debug As
Team
Compare With
Replace With
Restore from Local History...
AspectJ Tools
Properties Alt+Enter
1 AspectJ/Java Application Alt+Shift+X, C
2 Java Application Alt+Shift+X, J
3 JUnit Test Alt+Shift+X, T
4 SWT Application Alt+Shift+X, S
Run...
Java
Outline
CoreTest
2002 Palo Alto Resea
void main(String[]
tui.TestRunner.run(T
Problems Javadoc Declaration
0 errors, 0 warnings, 0 infos
Description Resource Path Location
tests.CoreTest.java - Exercise1/test-src
23M of 43M

What is xUnit?

- A set of “Frameworks” for programming and automated execution of test-cases
- X stands for programming language
  - Most Famous is J-UNIT for Java
  - But exists for almost all programming languages
  - C-unit, Cpp-Unit, DUnit, JUnit NUnit, ...
- A framework is a collection of classes, procedures, and macros

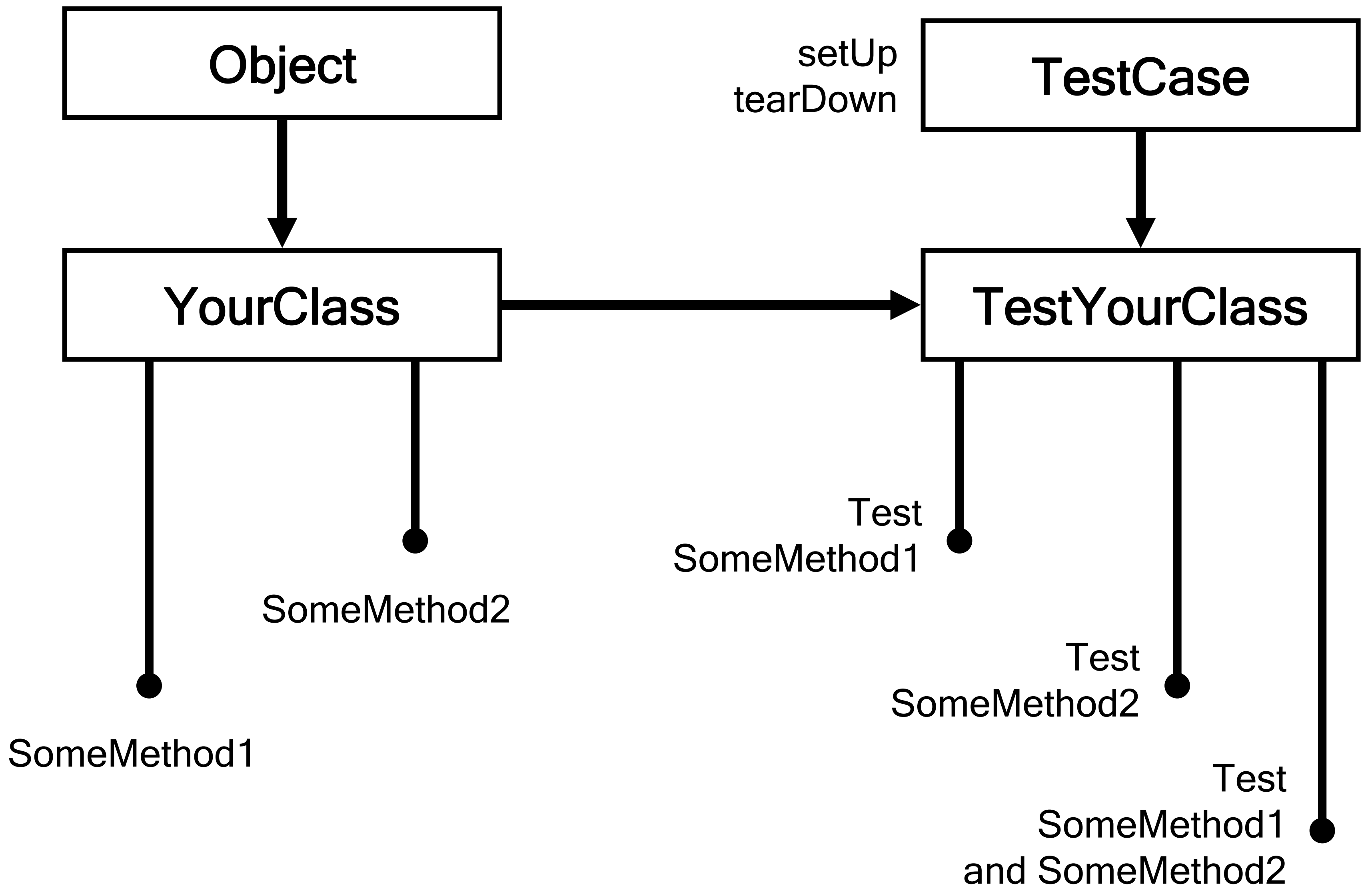
Basic Use of FrameWork



Concepts

- Assertions
  - Boolean expression that compares expected and actual results
  - The basic and smallest building-block
  - General: **ASSERT** (expected, actual)
- Test Case
  - A class that extends “TestCase”s
  - A composition of concrete test procedures
  - May contain several assertions and test for several test objectives
  - E.g all test of a particular function
- Test Suite
  - Collection of related test cases
  - Can be executed automatically in a single command

xUnit



Object
setUp
tearDown
TestCase
YourClass
TestYourClass
SomeMethod2
SomeMethod1
Test SomeMethod1
Test SomeMethod2
Test SomeMethod1 and SomeMethod2

Java Example

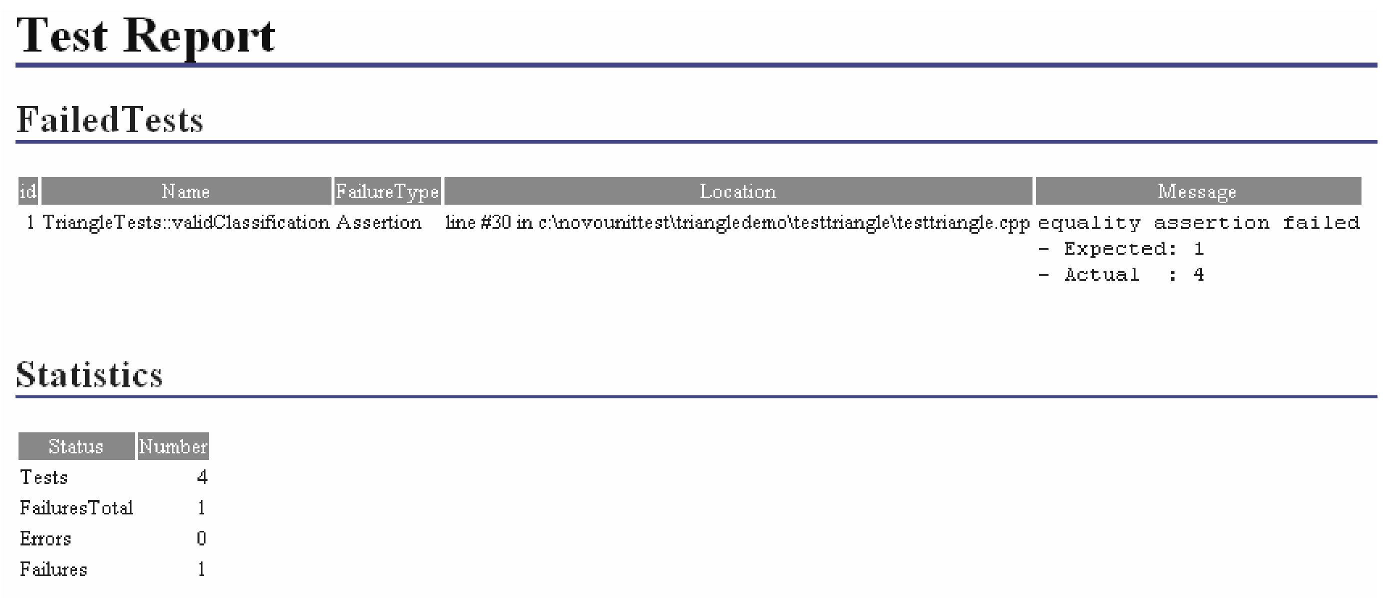
```
class ClassifyTriangle {

    public enum TriangleKind { invalidTriangle, equilateralTriangle,
                               isoscelesTriangle, scaleneTriangle};

    public TriangleKind classifyTriangle(int a, int b, int c) {
           ...
           return kind;
    }

    public String checkTriangle(String[] args) {
           ...
    }
}
```

Java Example

```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
public class ClassifyTriangleTest extends TestCase {
    protected void setUp() { }
    protected void setUp() { }

    public void testEquilateral() {
            ClassifyTriangle c=new ClassifyTriangle();
            assertEquals(equilateralTriangle, c.classifyTriangle(5,5,5));
            //add more tests here
    }
public void testCommandLine() {
            ClassifyTriangle c=new ClassifyTriangle();
            assertEquals("Error Code 40!\n",
                        c.checkTriangle({"-1", "Hello World", "-1"});
    }
    public static void main (String[] args) {
            junit.textui.TestRunner.run(ClassifyTriangleTest.class);
    }
}
```

Test Reports

```
C:\NovoUnitTest\TriangleDemo\cppunitDemo>Debug\cppunitDemo.exe
.F...

c:\novounittest\triangledemo\testtriangle\testtriangle.cpp(30):Assertion
Test name: TriangleTests::validClassification
equality assertion failed
- Expected: 1
- Actual  : 4

Failures !!!
Run: 4   Failure total: 1   Failures: 1   Errors: 0
```



Test Report

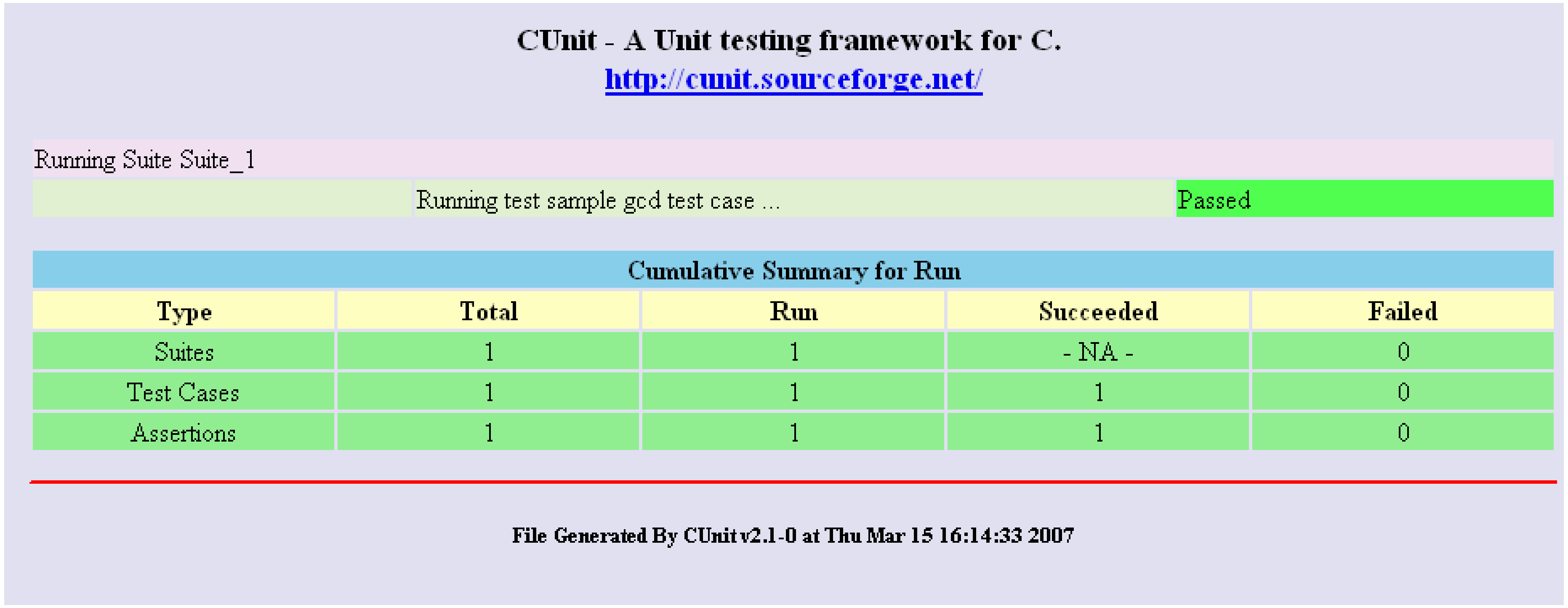
FailedTests

id	Name	FailureType	Location	Message
1	TriangleTests::validClassification	Assertion	line #30 in c:\novounittest\triangledemo\testtriangle\testtriangle.cpp	equality assertion failed - Expected: 1 - Actual : 4

Statistics

Status	Number
Tests	4
FailuresTotal	1
Errors	0
Failures	1

Test Runner XML file



CUnit - A Unit testing framework for C.
http://cunit.sourceforge.net/

Running Suite Suite_1		
	Running test sample gcd test case ...	Passed

Cumulative Summary for Run				
Type	**Total**	**Run**	**Succeeded**	**Failed**
Suites	1	1	- NA -	0
Test Cases	1	1	1	0
Assertions	1	1	1	0

File Generated By CUnit v2.1-0 at Thu Mar 15 16:14:33 2007

Advice: xUnit style

- Test cases exhibits isolation
  - Independent of other tests
  - Execution order irrelevant
- Set up an independent environment
  - setUp / tearDown methods scenario
- Each test case performs a ***distinct logical check***
  - $\Rightarrow$ one or few **asserts** per test method
  - BUT consider amount of test code declarations to be written (when a assert fails the test method is stopped and no further asserts are checked).
- Test expected errors and exceptions

Advice: xUnit style

- Make them fast;
  - If slow, developers won’t run them.
    - Smoke test suites
    - Complete test suites
- All developers must know about them;
  - Everyone who touches the code must run the tests.
  - Add to common code-repository
- Make test-code as nice and readable as implementation code
  - Documentation, Maintainability

Advice: Daily Builds

- Regression testing “must” be automated
  - This requires they report pass/fail results in a standardized way
- Daily (***Nightly***) builds and testing
  - Clean & check out latest build tree
  - Run tests
  - Put results on a web page & send mail (if tests fail)

Advice: Version Control

- Keep test code in a separate directory
- Keep both tests-sources and implemenation-source in version control
- Don’t checkin unless version passes all tests

Advice: Application

- Design and program for testability
- Directly applicable to
  - Pure function libraries
  - API
- (With some footwork also user interfaces, network-, web-, and database applications)

Advice: xUNIT principles

- Write test suite for each unit in the program.
- All test can be executed (automatically) at any time.
- For each program modification all tests must be passed before the modification is regarded as complete - regression testing
- Test First – implement later!
- Originally based on “eXtreme Programming” principles:
  - Lightweight software development methodology – by programmers for programmers
- TDD (Test Driven Development) cycle
  1. Write test case, and check it fails
  2. Write the new code
  3. Check that the test passes (and maybe refactor, re-test)

Conclusions

- Code that isn’t tested doesn’t work”
- “Code that isn’t regression tested suffers from code rot (breaks eventually)”
- A unit testing framework enables efficient and effective unit & regression testing
- Use xUNIT to store and maintain all the small tests that you write anyway
- Write tests instead of playing with debugger and printf – tests can be automatically repeated

END

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER